Efficient Black-box Checking of Snapshot Isolation in Databases

(Conference VLDB'2024)

Hengfeng Wei

hfwei@nju.edu.cn

August 16, 2023

Transaction and Isolation Level

A transaction is a *group* of operations that is executed atomically.

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$\quad x_2 \leftarrow x_2 - 100$
$\quad \mathsf{W}(acct_2, x_2)$

$acct_1 = acct_2 = 60$

Transaction and Isolation Level

A transaction is a *group* of operations that is executed atomically.

$$\boxed{\begin{array}{l} x_1 \leftarrow \mathsf{R}(acct_1) \\ x_2 \leftarrow \mathsf{R}(acct_2) \\ \textbf{if } x_1 + x_2 > 100 \\ \quad x_1 \leftarrow x_1 - 100 \\ \quad \mathsf{W}(acct_1, x_1) \end{array}} \qquad \boxed{\begin{array}{l} x_1 \leftarrow \mathsf{R}(acct_1) \\ x_2 \leftarrow \mathsf{R}(acct_2) \\ \textbf{if } x_1 + x_2 > 100 \\ \quad x_2 \leftarrow x_2 - 100 \\ \quad \mathsf{W}(acct_2, x_2) \end{array}} \qquad \boxed{\begin{array}{l} x_1 \leftarrow \mathsf{R}(acct_1) \\ x_2 \leftarrow \mathsf{R}(acct_2) \end{array}}$$

$acct_1 = acct_2 = 60$

Transaction and Isolation Level

A transaction is a *group* of operations that is executed atomically.

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$\quad x_1 \leftarrow x_1 - 100$
$\quad \mathsf{W}(acct_1, x_1)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$\quad x_2 \leftarrow x_2 - 100$
$\quad \mathsf{W}(acct_2, x_2)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$

Isolation Level (aka Transactional Consistency Model)

$acct_1 = acct_2 = 60$

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

Serializability (SER)

All transactions appear to execute in some total order.

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$x_1 \leftarrow x_1 - 100$
$\mathsf{W}(acct_1, x_1)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$x_2 \leftarrow x_2 - 100$
$\mathsf{W}(acct_2, x_2)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$

Serializability (SER)

$acct_1 = acct_2 = 60$

Serializability (SER)

All transactions appear to execute in some total order.

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$x_1 \leftarrow x_1 - 100$
$\mathsf{W}(acct_1, x_1)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$x_2 \leftarrow x_2 - 100$
$\mathsf{W}(acct_2, x_2)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$

Serializability (SER)

$acct_1 = acct_2 = 60$

$\mathsf{R}(acct_1, 60)$
$\mathsf{R}(acct_2, 60)$
$\mathsf{W}(acct_1, -40)$

Serializability (SER)

All transactions appear to execute in some total order.

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$x_1 \leftarrow x_1 - 100$
$\mathsf{W}(acct_1, x_1)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$x_2 \leftarrow x_2 - 100$
$\mathsf{W}(acct_2, x_2)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$

Serializability (SER)

$acct_1 = acct_2 = 60$

$\mathsf{R}(acct_1, 60)$
$\mathsf{R}(acct_2, 60)$
$\mathsf{W}(acct_1, -40)$

$\mathsf{R}(acct_1, 60)$
$\mathsf{R}(acct_2, 60)$

Serializability (SER)

All transactions appear to execute in some total order.

$x_1 \leftarrow \mathsf{R}(acct_1)$ $x_2 \leftarrow \mathsf{R}(acct_2)$ **if** $x_1 + x_2 > 100$ $x_1 \leftarrow x_1 - 100$ $\mathsf{W}(acct_1, x_1)$	$x_1 \leftarrow \mathsf{R}(acct_1)$ $x_2 \leftarrow \mathsf{R}(acct_2)$ **if** $x_1 + x_2 > 100$ $x_2 \leftarrow x_2 - 100$ $\mathsf{W}(acct_2, x_2)$	$x_1 \leftarrow \mathsf{R}(acct_1)$ $x_2 \leftarrow \mathsf{R}(acct_2)$

Serializability (SER)

$acct_1 = acct_2 = 60$

$\mathsf{R}(acct_1, 60)$ $\mathsf{R}(acct_2, 60)$ $\mathsf{W}(acct_1, -40)$	$\mathsf{R}(acct_1, 60)$ $\mathsf{R}(acct_2, 60)$	$\mathsf{R}(acct_1, 60)$ $\mathsf{R}(acct_2, 60)$

Serializability (SER)

All transactions appear to execute in some total order.

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$x_1 \leftarrow x_1 - 100$
$\mathsf{W}(acct_1, x_1)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$x_2 \leftarrow x_2 - 100$
$\mathsf{W}(acct_2, x_2)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$

Serializability (SER)

$acct_1 = acct_2 = 60$

$\mathsf{R}(acct_1, 60)$
$\mathsf{R}(acct_2, 60)$
$\mathsf{W}(acct_1, -40)$

$\mathsf{R}(acct_1, 60)$
$\mathsf{R}(acct_2, 60)$

$\mathsf{R}(acct_1, 60)$
$\mathsf{R}(acct_2, 60)$

too expensive, especially for distributed transactions

Snapshot Isolation (SI)

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$x_1 \leftarrow x_1 - 100$
$\mathsf{W}(acct_1, x_1)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$x_2 \leftarrow x_2 - 100$
$\mathsf{W}(acct_2, x_2)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$

Snapshot Isolation (SI)

$acct_1 = acct_2 = 60$

Snapshot Isolation (SI)

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$x_1 \leftarrow x_1 - 100$
$\mathsf{W}(acct_1, x_1)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$x_2 \leftarrow x_2 - 100$
$\mathsf{W}(acct_2, x_2)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$

Snapshot Isolation (SI)

$acct_1 = acct_2 = 60$

$\mathsf{R}(acct_1, 60)$
$\mathsf{R}(acct_2, 60)$
$\mathsf{W}(acct_1, -40)$

Snapshot Isolation (SI)

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$x_1 \leftarrow x_1 - 100$
$\mathsf{W}(acct_1, x_1)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$x_2 \leftarrow x_2 - 100$
$\mathsf{W}(acct_2, x_2)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$

Snapshot Isolation (SI)

$acct_1 = acct_2 = 60$

$\mathsf{R}(acct_1, 60)$
$\mathsf{R}(acct_2, 60)$
$\mathsf{W}(acct_1, -40)$

$\mathsf{R}(acct_1, 60)$
$\mathsf{R}(acct_2, 60)$
$\mathsf{W}(acct_2, -40)$

Snapshot Isolation (SI)

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$x_1 \leftarrow x_1 - 100$
$\mathsf{W}(acct_1, x_1)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$x_2 \leftarrow x_2 - 100$
$\mathsf{W}(acct_2, x_2)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$

Snapshot Isolation (SI)

$acct_1 = acct_2 = 60$

$\mathsf{R}(acct_1, 60)$
$\mathsf{R}(acct_2, 60)$
$\mathsf{W}(acct_1, -40)$

$\mathsf{R}(acct_1, 60)$
$\mathsf{R}(acct_2, 60)$
$\mathsf{W}(acct_2, -40)$

$\mathsf{R}(acct_1, -40)$
$\mathsf{R}(acct_2, -40)$

Snapshot Isolation (SI)

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$x_1 \leftarrow x_1 - 100$
$\mathsf{W}(acct_1, x_1)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$x_2 \leftarrow x_2 - 100$
$\mathsf{W}(acct_2, x_2)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$

Snapshot Isolation (SI)

$acct_1 = acct_2 = 60$

WRITE SKEW

$\mathsf{R}(acct_1, 60)$
$\mathsf{R}(acct_2, 60)$
$\mathsf{W}(acct_1, -40)$

$\mathsf{R}(acct_1, 60)$
$\mathsf{R}(acct_2, 60)$
$\mathsf{W}(acct_2, -40)$

$\mathsf{R}(acct_1, -40)$
$\mathsf{R}(acct_2, -40)$

Snapshot Isolation (SI)

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$x_1 \leftarrow x_1 - 100$
$\mathsf{W}(acct_1, x_1)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$
if $x_1 + x_2 > 100$
$x_2 \leftarrow x_2 - 100$
$\mathsf{W}(acct_2, x_2)$

$x_1 \leftarrow \mathsf{R}(acct_1)$
$x_2 \leftarrow \mathsf{R}(acct_2)$

Snapshot Isolation (SI)

$acct_1 = acct_2 = 60$

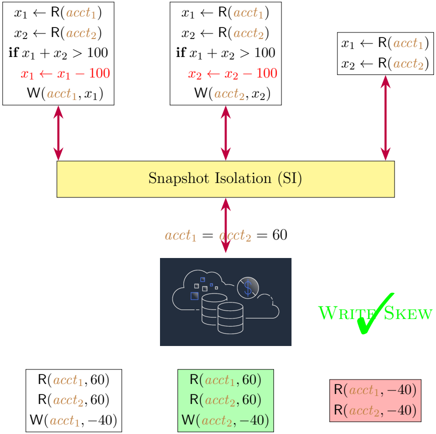
WRITE SKEW ✓

$\mathsf{R}(acct_1, 60)$
$\mathsf{R}(acct_2, 60)$
$\mathsf{W}(acct_1, -40)$

$\mathsf{R}(acct_1, 60)$
$\mathsf{R}(acct_2, 60)$
$\mathsf{W}(acct_2, -40)$

$\mathsf{R}(acct_1, -40)$
$\mathsf{R}(acct_2, -40)$

Snapshot Isolation (SI)



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

Snapshot Isolation (SI)

$x \leftarrow \mathsf{R}(acct)$ $x \leftarrow x + 50$ $\mathsf{W}(acct, x)$	$x \leftarrow \mathsf{R}(acct)$ $x \leftarrow x + 25$ $\mathsf{W}(acct, x)$	$\mathsf{R}(acct)$

Snapshot Isolation (SI)

$acct = 0$

Snapshot Isolation (SI)

$x \leftarrow \mathsf{R}(\mathit{acct})$
$x \leftarrow x + 50$
$\mathsf{W}(\mathit{acct}, x)$

$x \leftarrow \mathsf{R}(\mathit{acct})$
$x \leftarrow x + 25$
$\mathsf{W}(\mathit{acct}, x)$

$\mathsf{R}(\mathit{acct})$

Snapshot Isolation (SI)

$\mathit{acct} = 0$

$\mathsf{R}(\mathit{acct}, 0)$
$\mathsf{W}(\mathit{acct}, 50)$

Snapshot Isolation (SI)

$x \leftarrow \mathsf{R}(acct)$
$x \leftarrow x + 50$
$\mathsf{W}(acct, x)$

$x \leftarrow \mathsf{R}(acct)$
$x \leftarrow x + 25$
$\mathsf{W}(acct, x)$

$\mathsf{R}(acct)$

Snapshot Isolation (SI)

$acct = 0$

$\mathsf{R}(acct, 0)$
$\mathsf{W}(acct, 50)$

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

Snapshot Isolation (SI)

$x \leftarrow \mathsf{R}(acct)$
$x \leftarrow x + 50$
$\mathsf{W}(acct, x)$

$x \leftarrow \mathsf{R}(acct)$
$x \leftarrow x + 25$
$\mathsf{W}(acct, x)$

$\mathsf{R}(acct)$

Snapshot Isolation (SI)

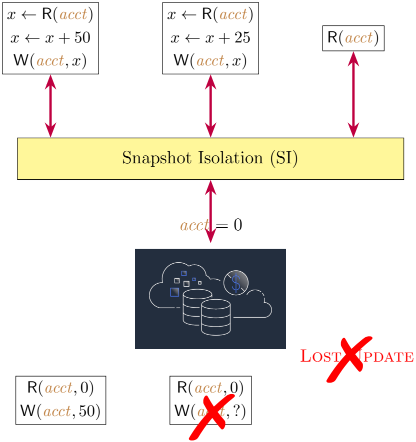
$acct = 0$

$\mathsf{R}(acct, 0)$
$\mathsf{W}(acct, 50)$

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

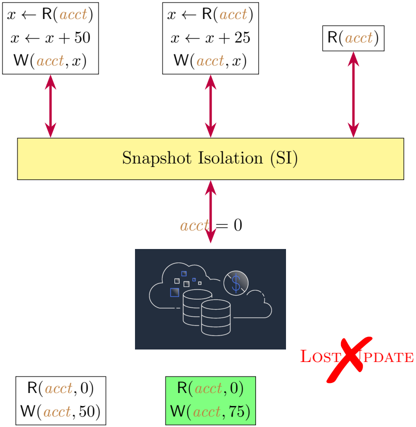
Snapshot Write: Concurrent transactions cannot write to the same key. One of them must be aborted.

Snapshot Isolation (SI)



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

Snapshot Isolation (SI)



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

Snapshot Isolation (SI)

Snapshot Isolation (SI)

Snapshot Isolation (SI)

Snapshot Isolation (SI)

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

Snapshot Isolation (SI)

Snapshot Isolation (SI)

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

$\mathsf{R}(x, post)$
$\mathsf{W}(y, comment)$

Snapshot Isolation (SI)

Snapshot Isolation (SI)

W(x, *post*)

R(x, *post*)
W(y, *comment*)

R(x, *empty*)
R(y, *comment*)

Snapshot Isolation (SI)

Snapshot Isolation (SI)

CAUSALITY VIOLATION

$\mathsf{W}(x, post)$	$\mathsf{R}(x, post)$ $\mathsf{W}(y, comment)$	$\mathsf{R}(x, empty)$ $\mathsf{R}(y, comment)$

Snapshot Isolation (SI)

Snapshot Isolation (SI)

CAUSALITY VIOLATION

W(x, *post*)

R(x, *post*)
W(y, *comment*)

R(x, *empty*)
R(y, *comment*) ✗

Databases and Snapshot Isolation

database logos

Many databases claim to support SI.

Databases and Snapshot Isolation

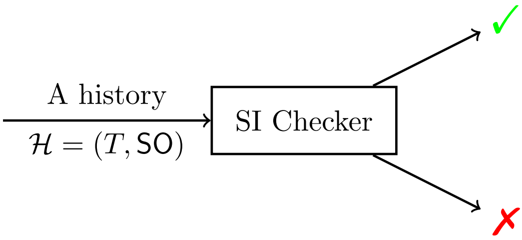
+papers

Databases may fail to provide SI as they claim.

The SI Checking Problem

Definition (The SI Checking Problem)

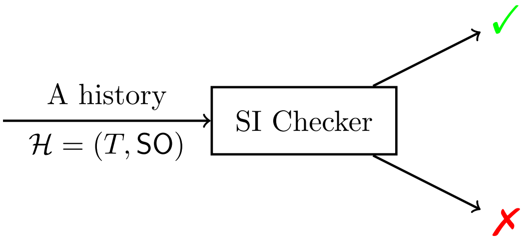
The SI checking problem is the decision problem of determing whether a given history $\mathcal{H} = (T, \mathsf{SO})$ satisfies SI?



The SI Checking Problem

Definition (The SI Checking Problem)

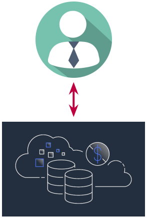
The SI checking problem is the decision problem of determing whether a given history $\mathcal{H} = (T, \mathsf{SO})$ satisfies SI?



SO : *session order* among the set T of transactions

The SI Checking Problem

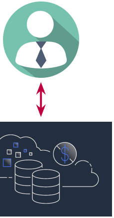
Black-box checking: do not rely on database internals



The histories are collected from database logs.

The SI Checking Problem

Black-box checking: do not rely on database internals

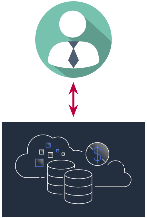


$\mathsf{W}(x, post)$	$\mathsf{R}(x, post)$ $\mathsf{W}(y, comment)$	$\mathsf{R}(x, empty)$ $\mathsf{R}(y, comment)$	Lost Update ✗

The histories are collected from database logs.

The SI Checking Problem

Black-box checking: do not rely on database internals



W(x, *post*)	R(x, *post*) W(y, *comment*)	R(x, *empty*) R(y, *comment*)

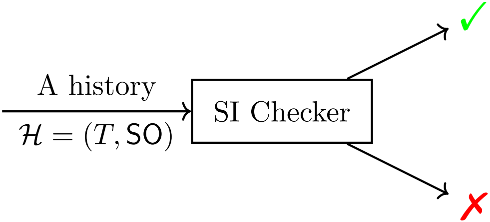
LOST UPDATE

R($acct_1$, 60) R($acct_2$, 60) W($acct_1$, −40)	R($acct_1$, 60) R($acct_2$, 60) W($acct_2$, −40)	R($acct_1$, −40) R($acct_2$, −40)

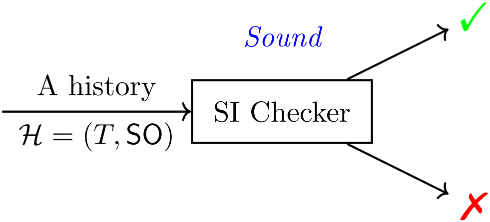
WRITE SKEW

The histories are collected from database logs.

The SI Checking Problem

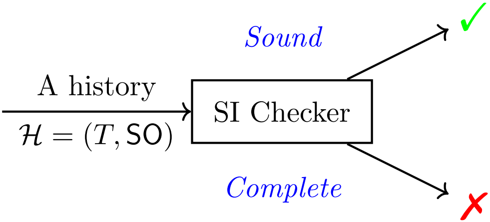


The SI Checking Problem



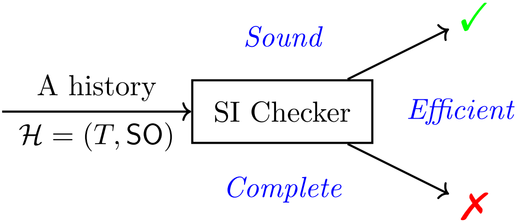
Sound: If the checker says ✗, then the history does *not* satisfy SI.

The SI Checking Problem



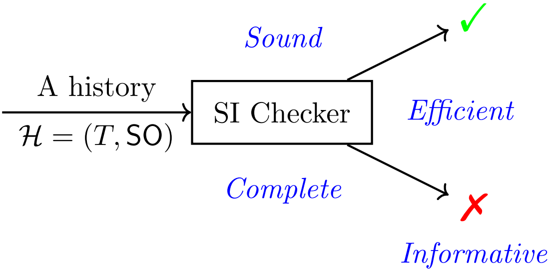
Complete: If the checker says ✓, then the history *satisfies* SI.

The SI Checking Problem



Efficient: The checker should *scale* up to large workloads.

The SI Checking Problem

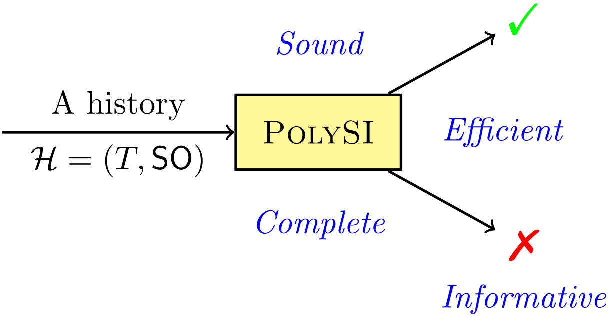


Informative: The checker should provide understandable *counterexamples* if it says ✗.

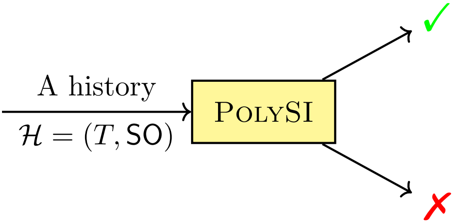
The SI Checking Problem

related-work

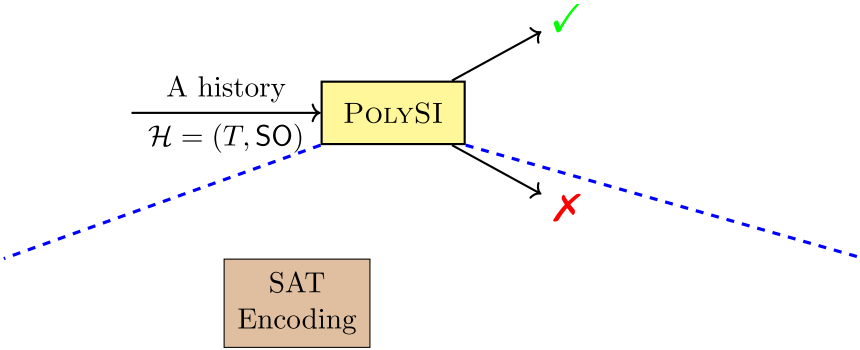
Contribution: the PolySI Checker



Contribution: the PolySI Checker

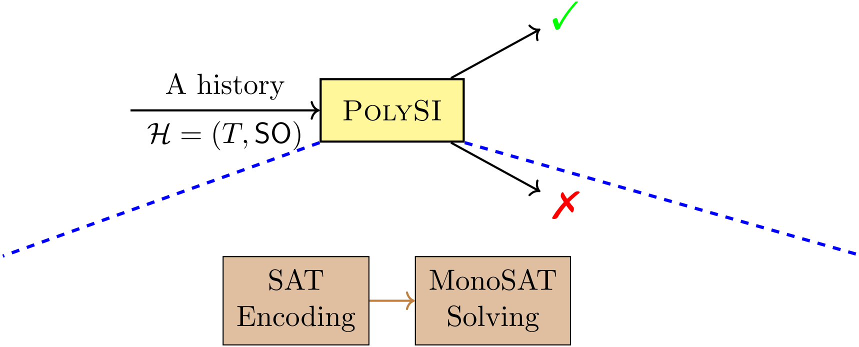


Contribution: the PolySI Checker



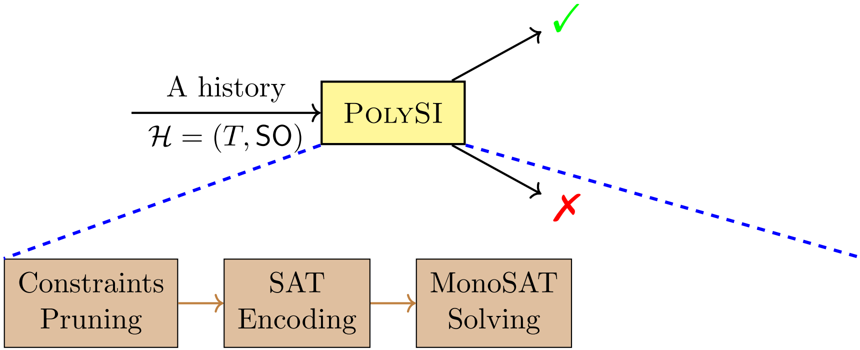
Sound & Complete: polygraph-based characterization of SI

Contribution: the PolySI Checker



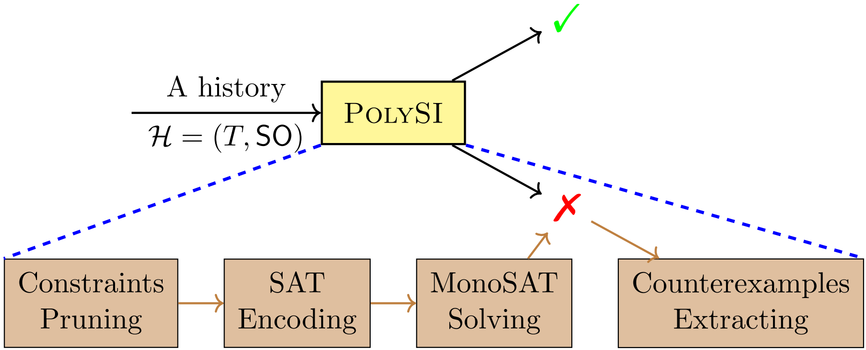
Efficient: utilizing MonoSAT solver optimized for graph problems

Contribution: the PolySI Checker



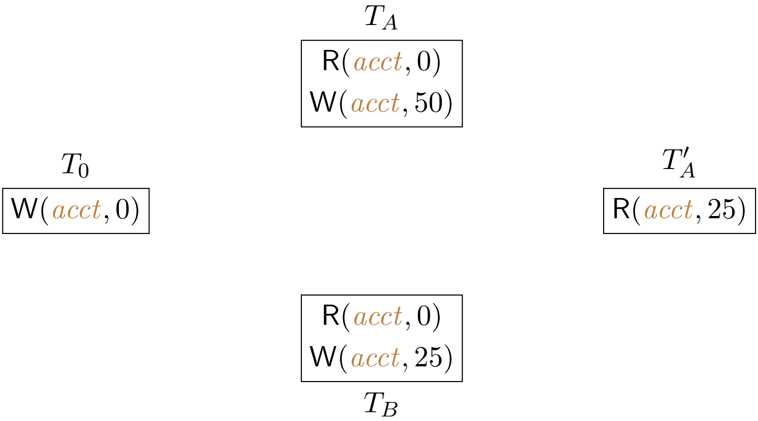
Efficient: domain-specific pruning before encoding

Contribution: the PolySI Checker

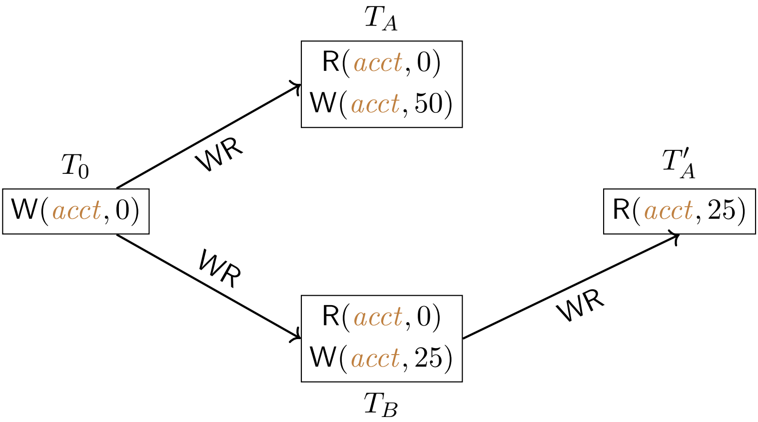


Informative: extract counterexamples from the unsatisifiable core

Dependency Graph-based Characterization of SI

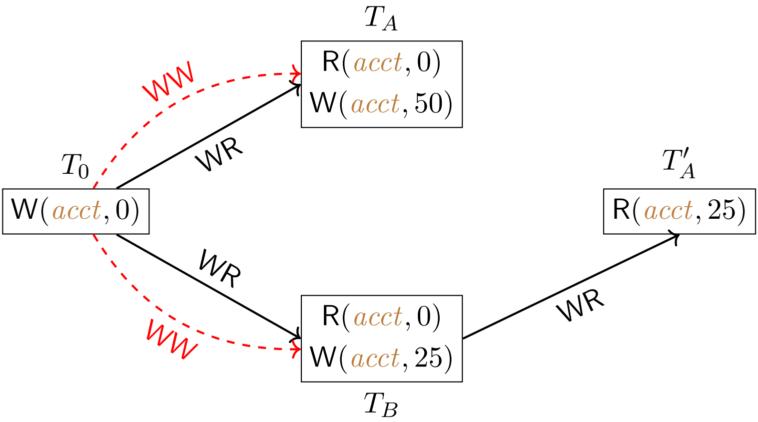


Dependency Graph-based Characterization of SI



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

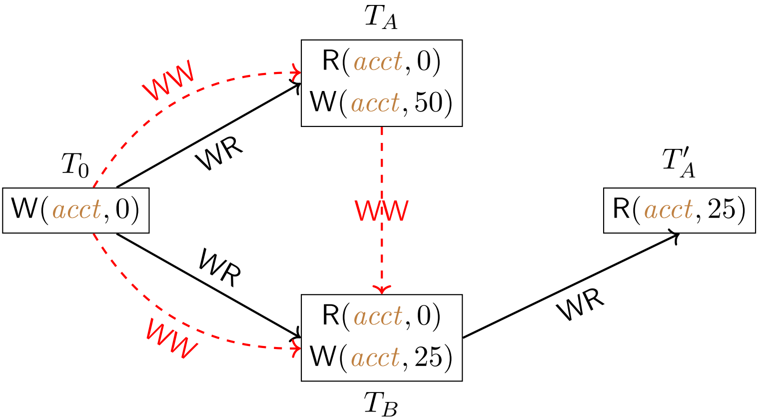
Dependency Graph-based Characterization of SI



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

Dependency Graph-based Characterization of SI

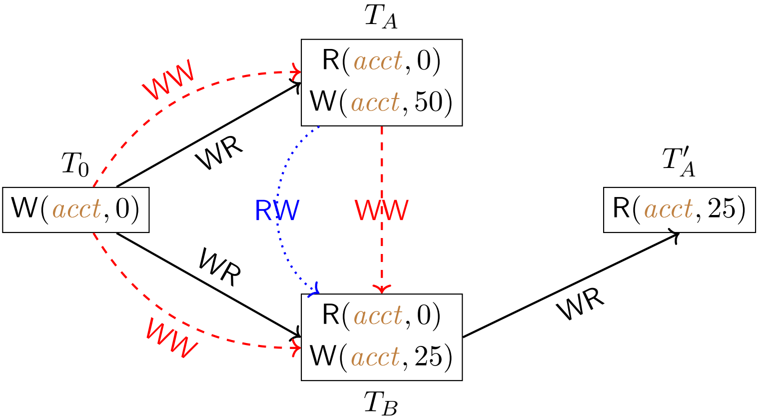
Suppose that $T_A \xrightarrow{\mathsf{WW}} T_B$



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

Dependency Graph-based Characterization of SI

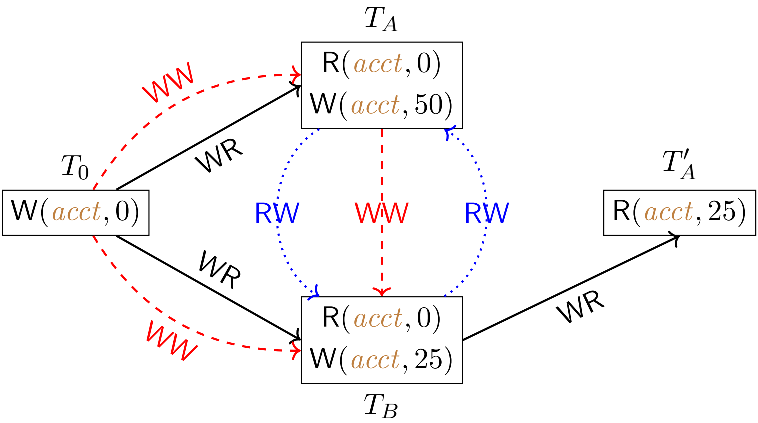
$$T_0 \xrightarrow{\mathsf{WR}} T_A \wedge T_0 \xrightarrow{\mathsf{WW}} T_B \implies T_A \xrightarrow{\mathsf{RW}} T_B$$



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

Dependency Graph-based Characterization of SI

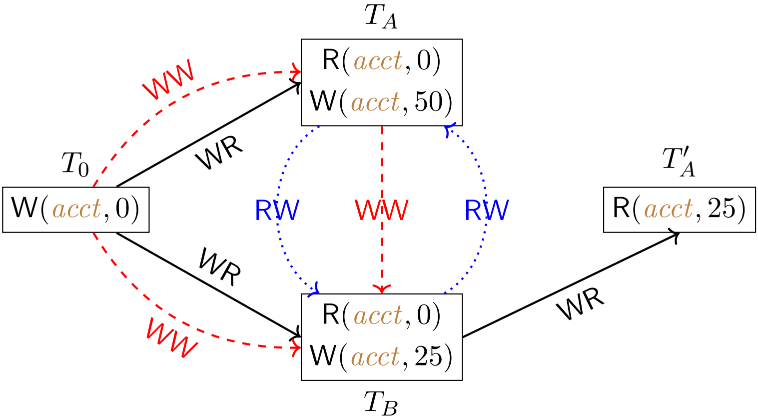
$$T_0 \xrightarrow{\mathsf{WR}} T_B \land T_0 \xrightarrow{\mathsf{WW}} T_A \implies T_A \xrightarrow{\mathsf{RW}} T_A$$



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

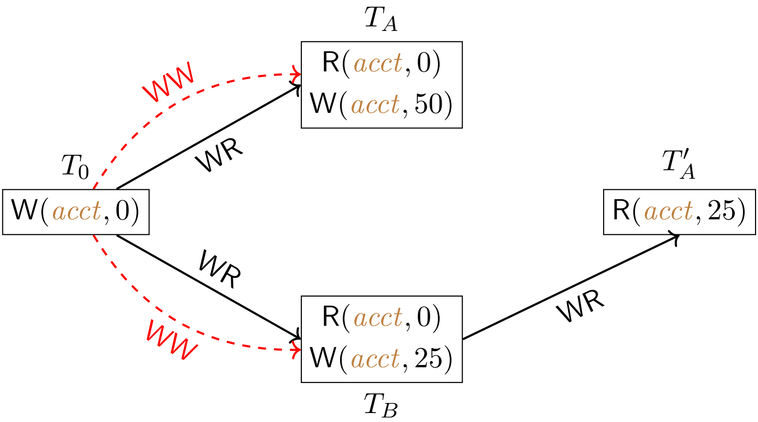
Dependency Graph-based Characterization of SI

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



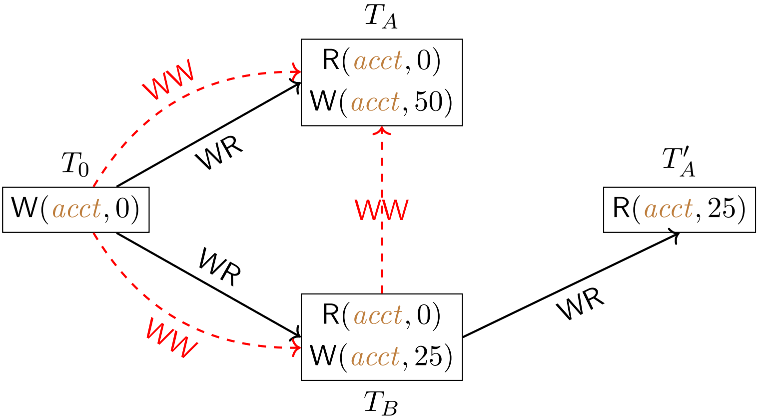
undesired cycle: $T_A \xrightarrow{\mathsf{WW}} T_B \xrightarrow{\mathsf{RW}} T_A$

Dependency Graph-based Characterization of SI



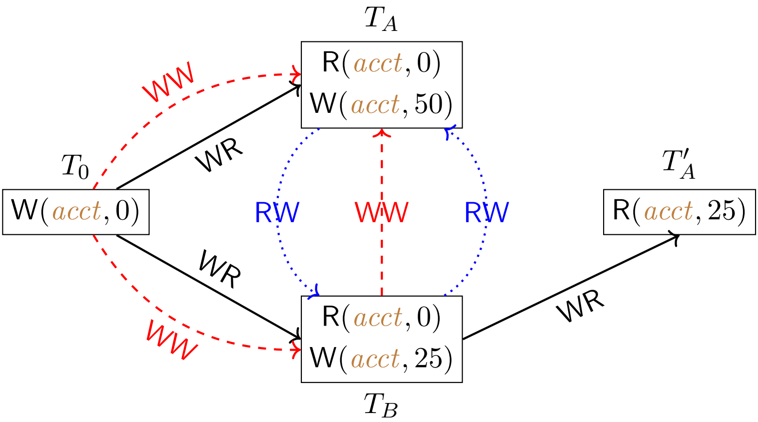
Dependency Graph-based Characterization of SI

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



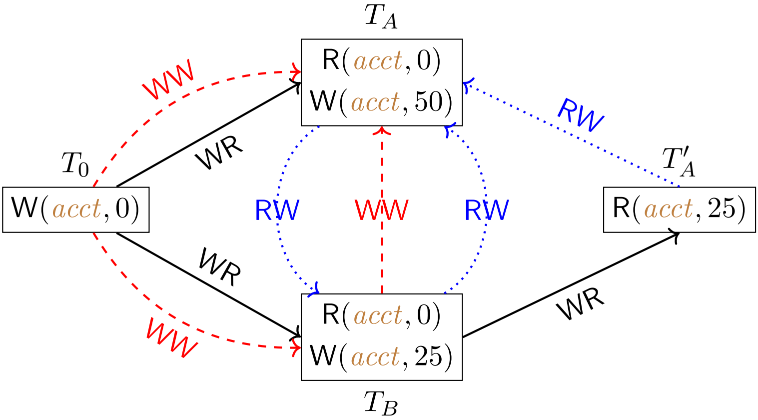
Dependency Graph-based Characterization of SI

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



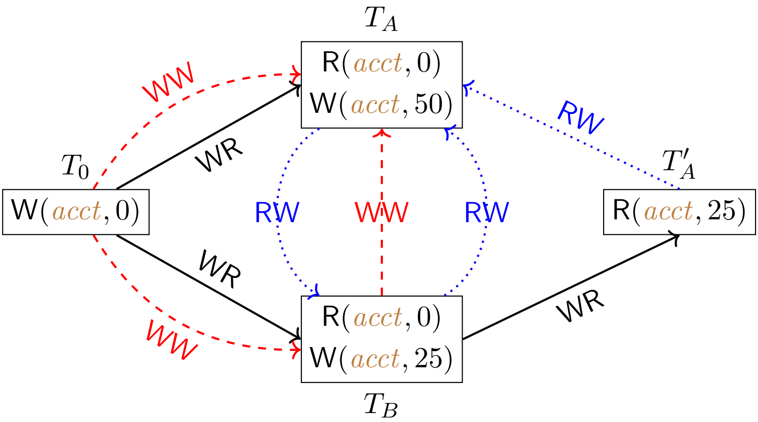
Dependency Graph-based Characterization of SI

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



Dependency Graph-based Characterization of SI

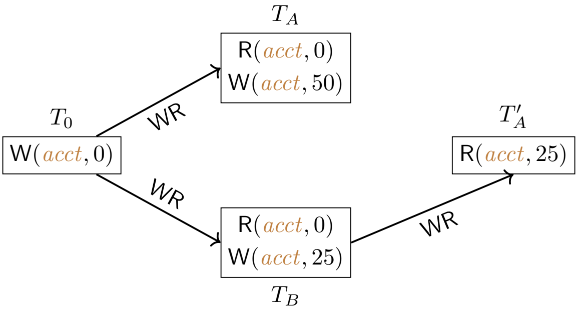
$\boxed{\textit{Suppose that } T_B \xrightarrow{\mathsf{WW}} T_A}$



undesired cycle: $T_B \xrightarrow{\mathsf{WW}} T_A \xrightarrow{\mathsf{RW}} T_B$

Dependency Graph-based Characterization of SI

We have considered both bases $T_A \xrightarrow{\mathsf{WW}} T_B$ and $T_B \xrightarrow{\mathsf{WW}} T_A$.



Either case leads to an undesired cycle.

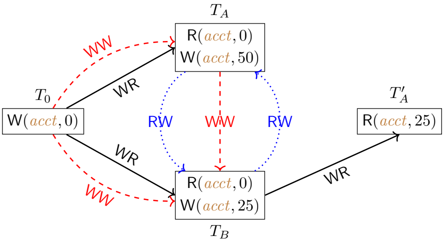
Therefore, it does not satisfy SI.

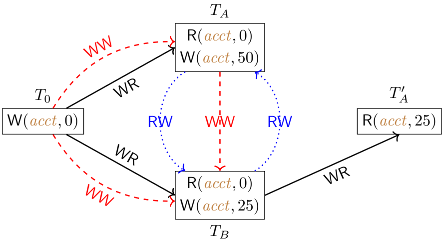
Dependency Graph-based Characterization of SI

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

Informally, a history satisfies SI if only if there exists a dependency graph for it that contains only cycles (if any) with at least two adjacent RW *edges.*

Dependency Graph-based Characterization of SI





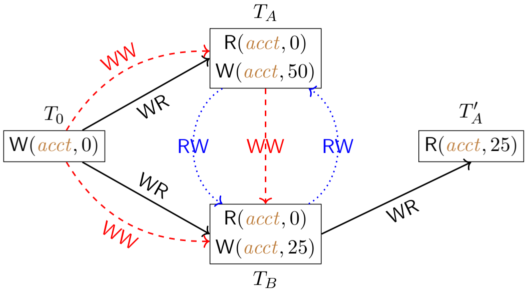
Every possible dependency graph contains an undesired  cycle.

Dependency Graph-based Characterization of SI

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

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

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

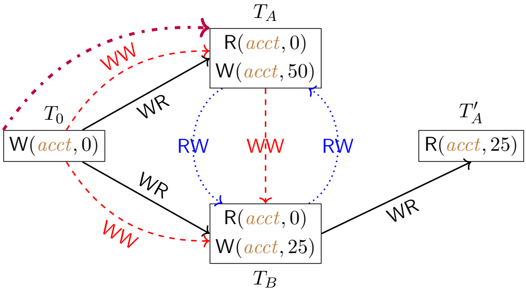


Dependency Graph-based Characterization of SI

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

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

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

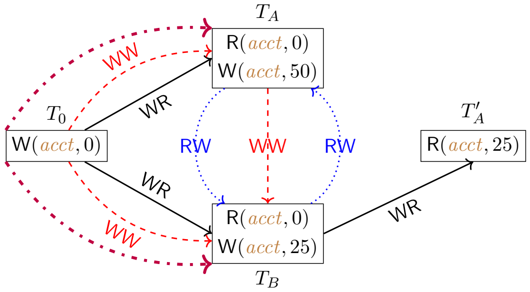


Dependency Graph-based Characterization of SI

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

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

$$\mathcal{H} \models \text{SI} \iff \mathcal{H} \models \text{INT} \land \exists\, \mathsf{WR}, \mathsf{WW}, \mathsf{RW}.\ \mathcal{G} = (\mathcal{H}, \mathsf{WR}, \mathsf{WW}, \mathsf{RW}) \land (((\mathsf{SO}_{\mathcal{G}} \cup \mathsf{WR}_{\mathcal{G}} \cup \mathsf{WW}_{\mathcal{G}}) \,;\, \mathsf{RW}_{\mathcal{G}}?) \ \textit{ is acyclic}).$$

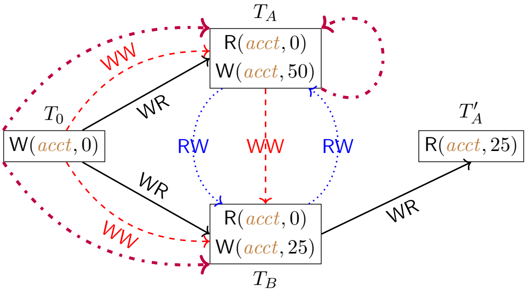


Dependency Graph-based Characterization of SI

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

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

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

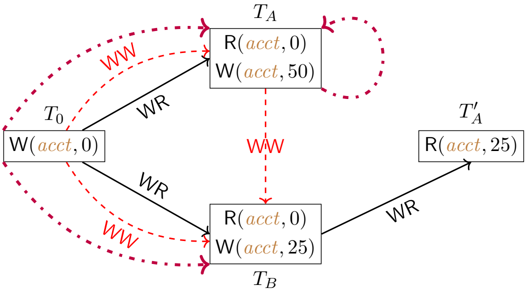


Dependency Graph-based Characterization of SI

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

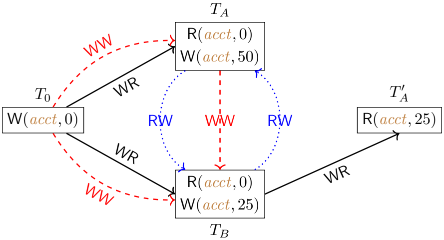
For a history $\mathcal{H} = (T, \mathsf{SO})$,

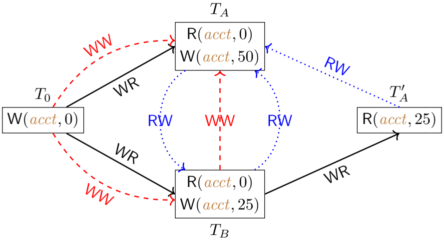
$$\mathcal{H} \models \mathrm{SI} \iff \mathcal{H} \models \textsc{Int} \wedge$$
$$\exists\, \mathsf{WR}, \mathsf{WW}, \mathsf{RW}.\ \mathcal{G} = (\mathcal{H}, \mathsf{WR}, \mathsf{WW}, \mathsf{RW}) \wedge$$
$$(((\mathsf{SO}_{\mathcal{G}} \cup \mathsf{WR}_{\mathcal{G}} \cup \mathsf{WW}_{\mathcal{G}})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



Dependency Graph-based Characterization of SI

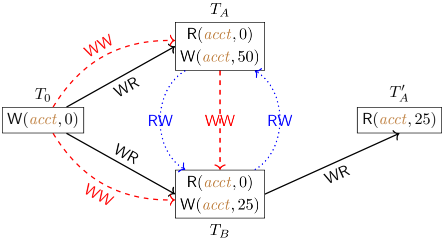
$\mathcal{Q}$: How to capture and resolve all possible WW dependencies?





Dependency Graph-based Characterization of SI

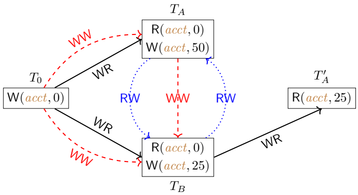
$\mathcal{Q}$: How to capture and resolve all possible WW dependencies?

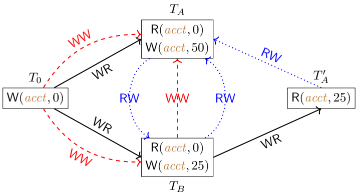


$\mathcal{A}$: encode them into SAT formulas based on (generalized) polygraphs and solve them using SAT solvers.

Polygraphs: A Family of Dependency Graphs

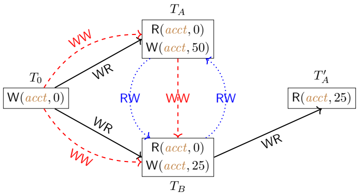
Consider the two cases of WW dependencies between T_A and T_B.

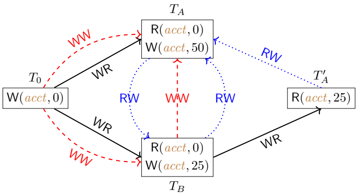


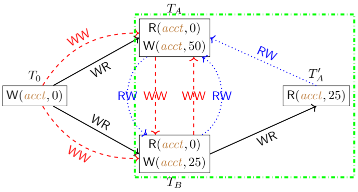


Polygraphs: A Family of Dependency Graphs

Consider the two cases of WW dependencies between T_A and T_B.



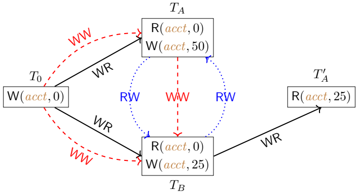


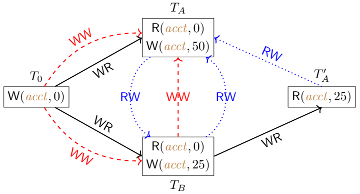


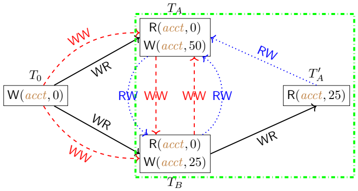
generalized polygraph:

Polygraphs: A Family of Dependency Graphs

Consider the two cases of WW dependencies between T_A and T_B.



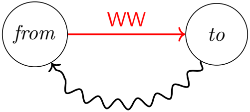


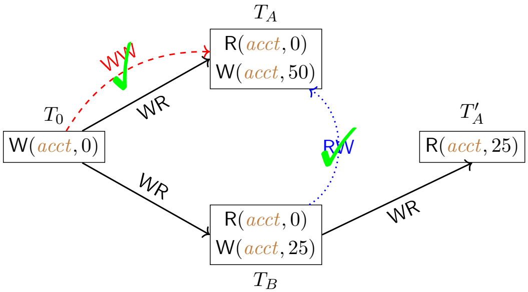


generalized polygraph:

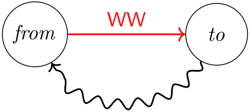
$$\langle either \triangleq \{T_A \xrightarrow{\text{WW}} T_B\}, or \triangleq \{T_B \xrightarrow{\text{WW}} T_A, T'_A \xrightarrow{\text{RW}} T_A\}\rangle$$

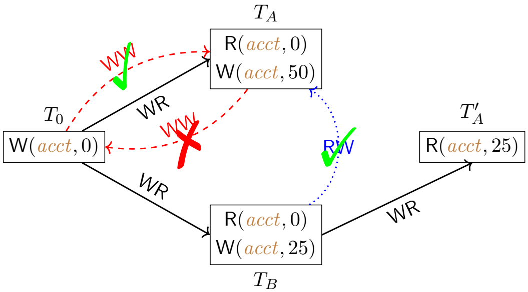
PolySI: Pruning before Encoding (the WW case)





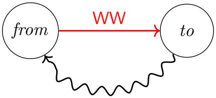
PolySI: Pruning before Encoding (the WW case)

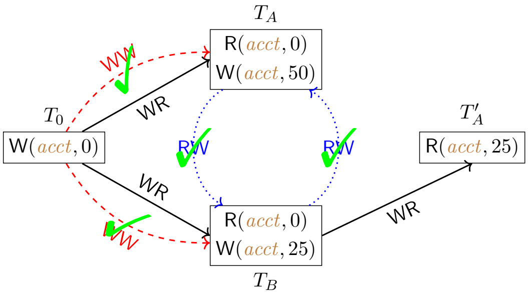




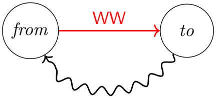
$T_A \xrightarrow{\mathsf{WW}} T_0$ can be pruned due to the $T_A \xrightarrow{\mathsf{WW}} T_0 \xrightarrow{\mathsf{WR}} T_A$ cycle.

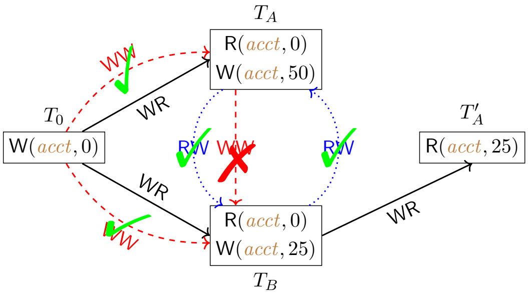
PolySI: Pruning before Encoding (the WW case)





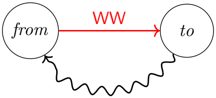
PolySI: Pruning before Encoding (the WW case)

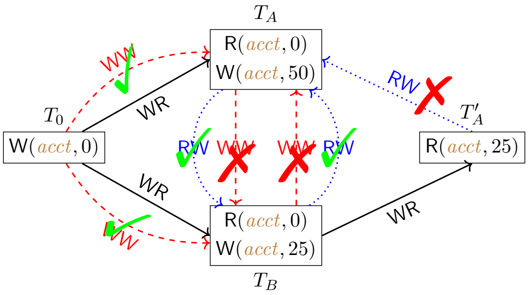




$T_A \xrightarrow{\mathsf{WW}} T_B$ is pruned due to the $T_A \xrightarrow{\mathsf{WW}} T_B \xrightarrow{\mathsf{RW}} T_A$ cycle.

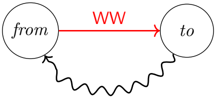
PolySI: Pruning before Encoding (the WW case)

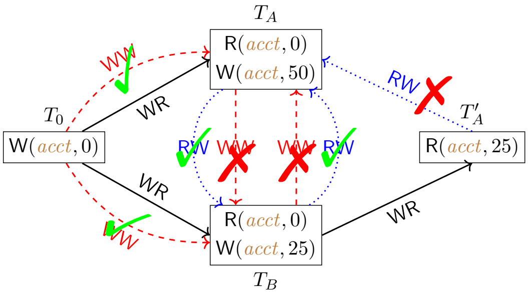




$$T_A \xrightarrow{\text{WW}} T_B \text{ is pruned due to the } T_A \xrightarrow{\text{WW}} T_B \xrightarrow{\text{RW}} T_A \text{ cycle.}$$
$$T_B \xrightarrow{\text{WW}} T_A \text{ is pruned due to the } T_B \xrightarrow{\text{WW}} T_A \xrightarrow{\text{RW}} T_B \text{ cycle.}$$

PolySI: Pruning before Encoding (the WW case)





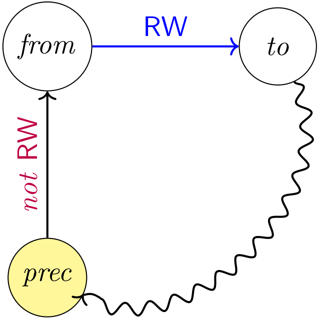
$T_A \xrightarrow{\text{WW}} T_B$ is pruned due to the $T_A \xrightarrow{\text{WW}} T_B \xrightarrow{\text{RW}} T_A$ cycle.

$T_B \xrightarrow{\text{WW}} T_A$ is pruned due to the $T_B \xrightarrow{\text{WW}} T_A \xrightarrow{\text{RW}} T_B$ cycle.

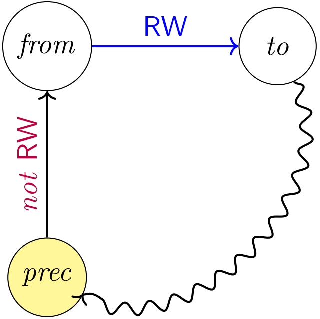
Therefore, we are sure that the history does *not* satisfy SI.

PolySI: Pruning before Encoding (the RW case)

PolySI: Pruning before Encoding (the RW case)



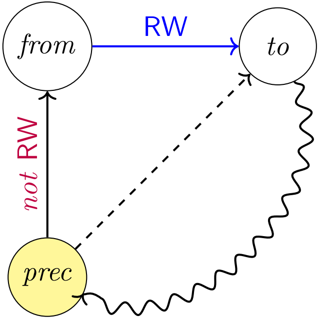
PolySI: Pruning before Encoding (the RW case)



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

Informally, a history satisfies SI if only if there exists a dependency graph for it that contains only cycles (if any) with at least two adjacent RW *edges.*

PolySI: Pruning before Encoding (the RW case)



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

Informally, a history satisfies SI if only if *there exists a dependency graph for it that contains only cycles (if any) with at least two adjacent* RW *edges.*

PolySI: An Illustrating Example of "Long Fork"

T_0 $\boxed{\mathsf{W}(x, 0)\ \mathsf{W}(y, 0)}$

PolySI: An Illustrating Example of "Long Fork"

T_1

$\boxed{\mathsf{W}(x, 1)}$

T_0 $\boxed{\mathsf{W}(x, 0)\ \mathsf{W}(y, 0)}$

PolySI: An Illustrating Example of "Long Fork"

T_1

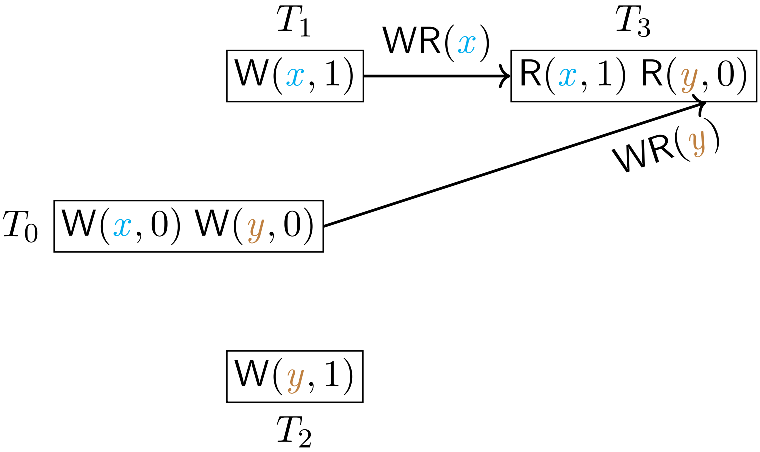
$\boxed{\mathsf{W}(x, 1)}$

T_0 $\boxed{\mathsf{W}(x, 0)\ \mathsf{W}(y, 0)}$

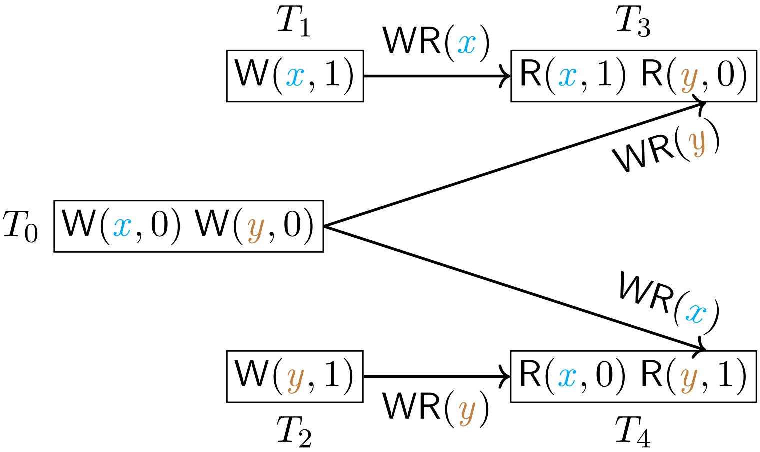
$\boxed{\mathsf{W}(y, 1)}$

T_2

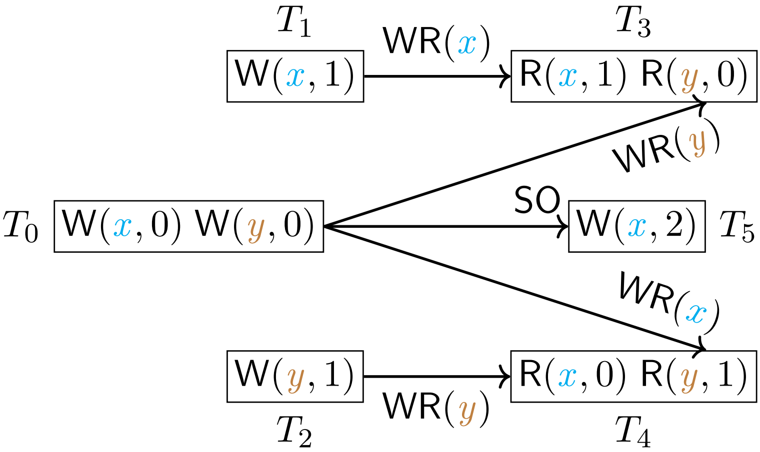
PolySI: An Illustrating Example of "Long Fork"



PolySI: An Illustrating Example of “Long Fork”

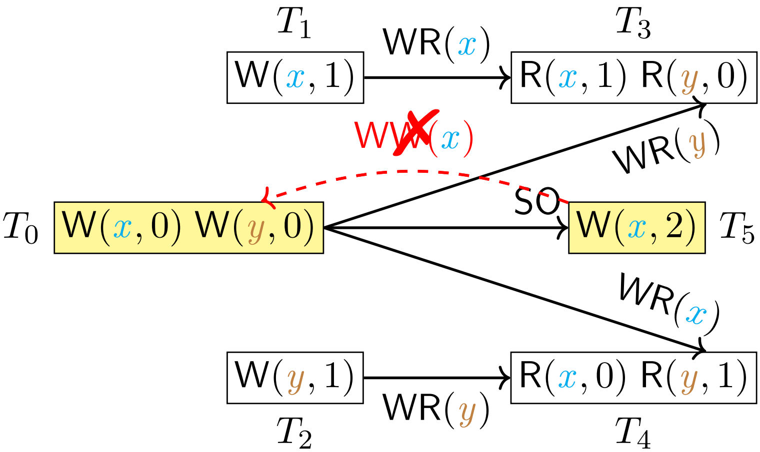


PolySI: An Illustrating Example of "Long Fork"



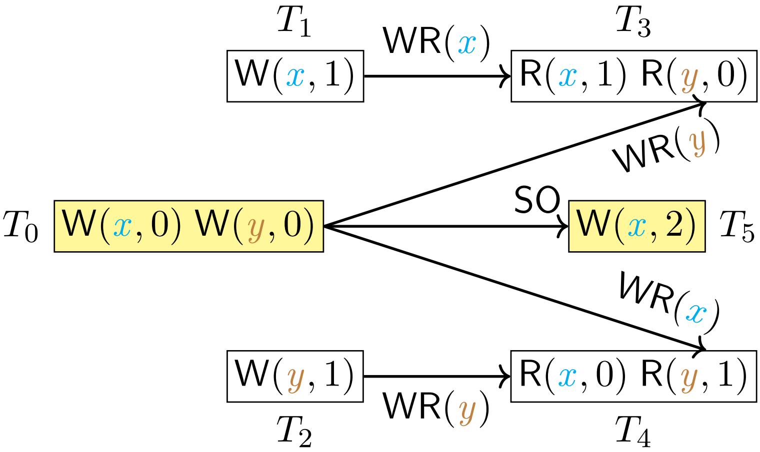
order between T_0, T_1, and T_5 (on x) and between T_0 and T_2 (on y)

PolySI: An Illustrating Example of "Long Fork"

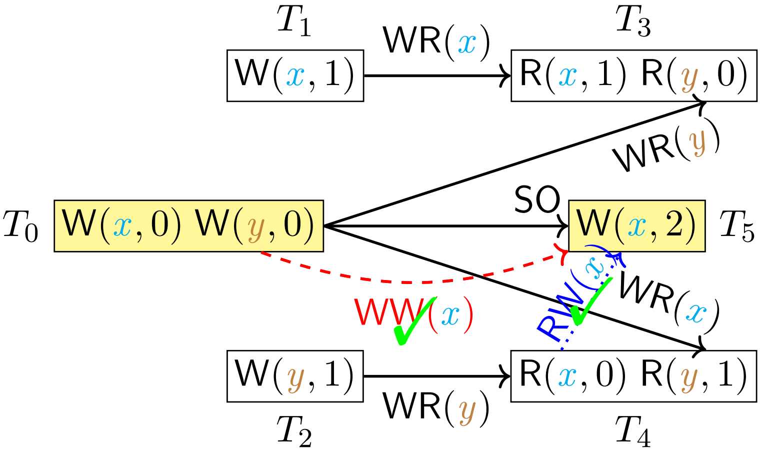


The $T_5 \xrightarrow{\mathsf{WW}(x)} T_0$ case is pruned due to $T_0 \xrightarrow{\mathsf{SO}} T_5 \xrightarrow{\mathsf{WW}(x)} T_0$.

PolySI: An Illustrating Example of "Long Fork"

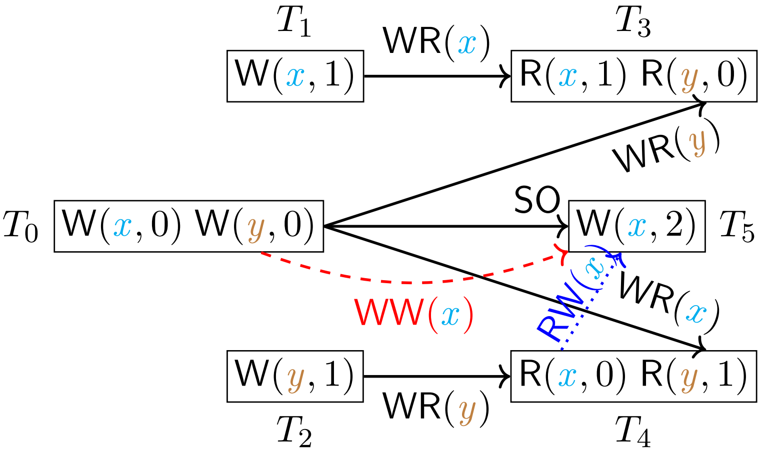


PolySI: An Illustrating Example of "Long Fork"

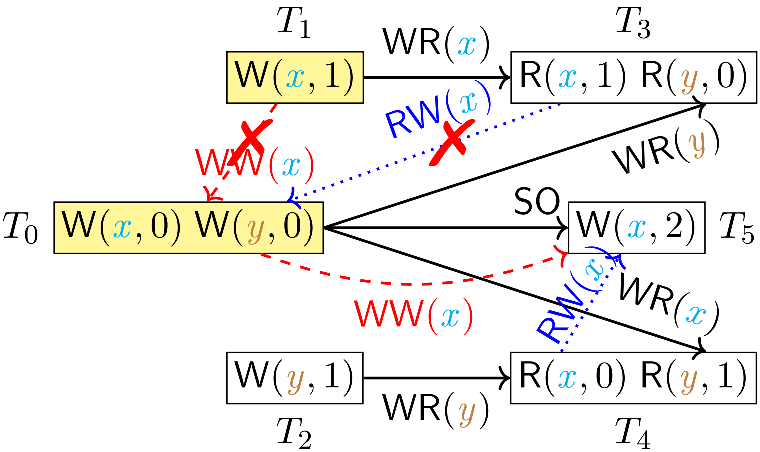


The $T_0 \xrightarrow{\mathsf{WW}(x)} T_5$ case becomes known.

PolySI: An Illustrating Example of "Long Fork"

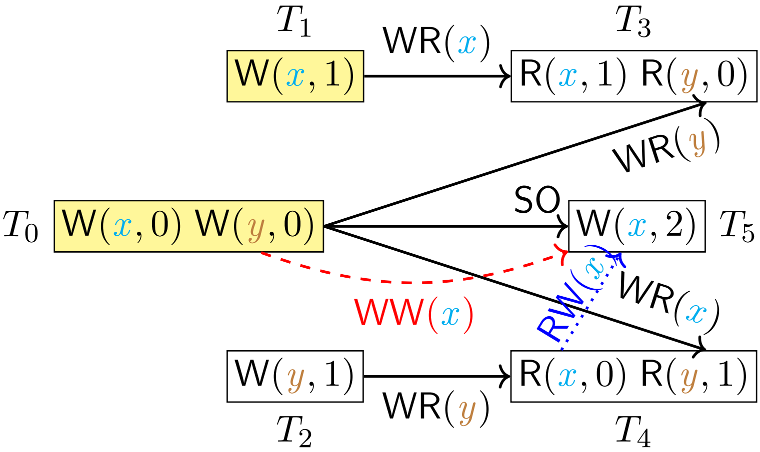


PolySI: An Illustrating Example of “Long Fork”

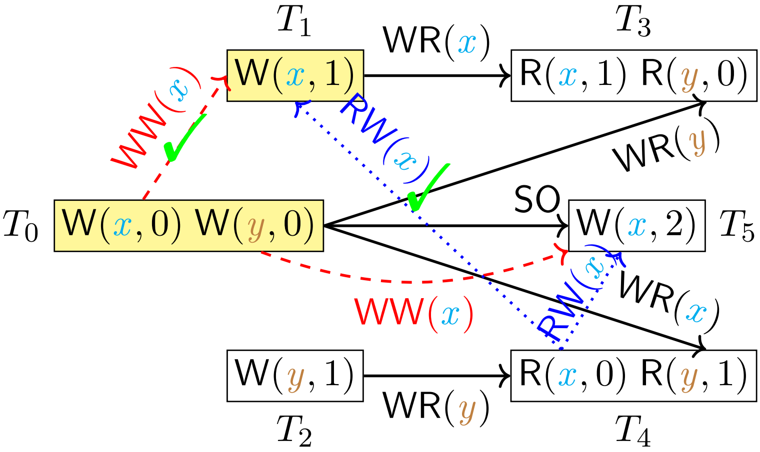


The $T_1 \xrightarrow{\text{WW}(x)} T_0$ case is pruned due to $T_3 \xrightarrow{\text{RW}(x)} T_0 \xrightarrow{\text{WR}(y)} T_3$.

PolySI: An Illustrating Example of "Long Fork"

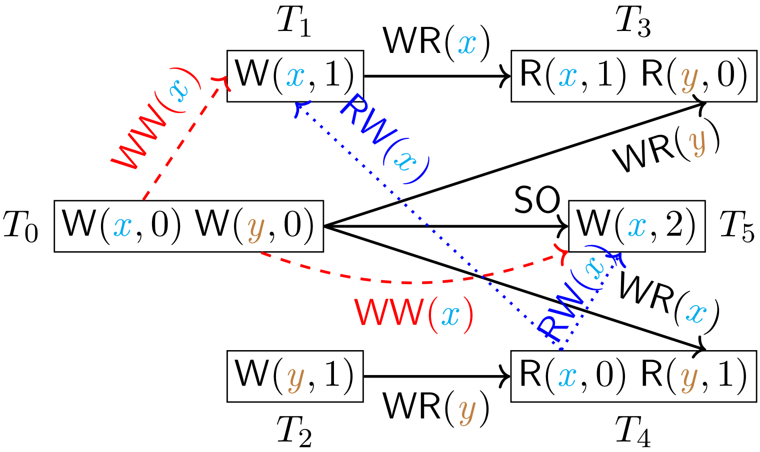


PolySI: An Illustrating Example of “Long Fork”

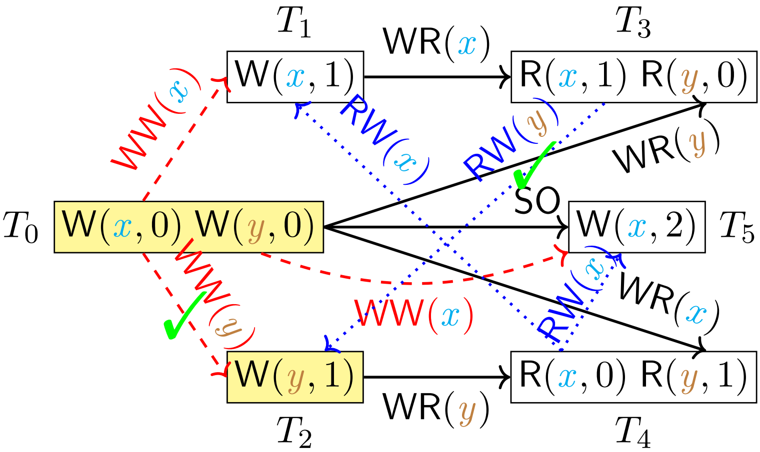


The $T_0 \xrightarrow{\mathsf{WW}(x)} T_1$ case becomes known.

PolySI: An Illustrating Example of "Long Fork"



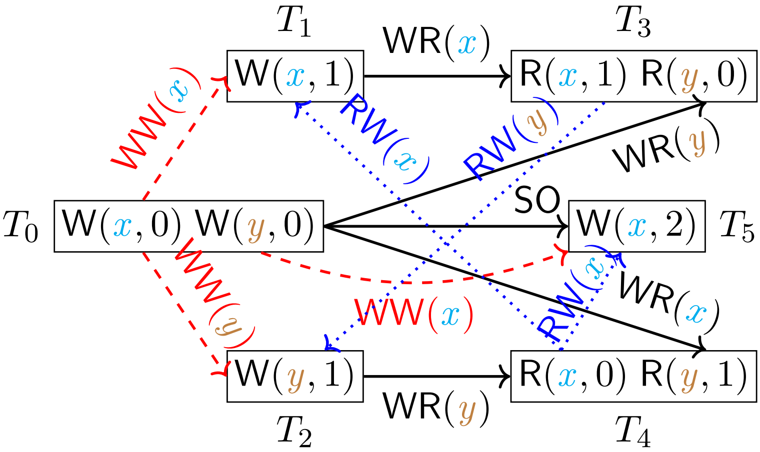
PolySI: An Illustrating Example of "Long Fork"



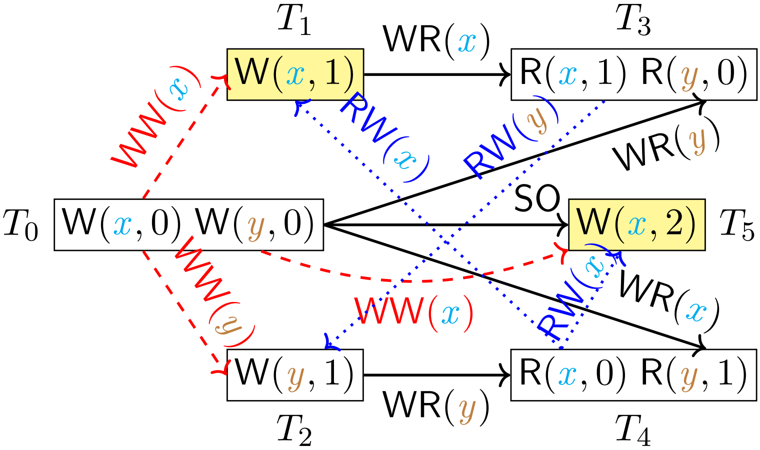
The $T_2 \xrightarrow{\mathsf{WW}(y)} T_0$ case is pruned,

while the $T_0 \xrightarrow{\mathsf{WW}(y)} T_2$ case becomes known.

PolySI: An Illustrating Example of "Long Fork"

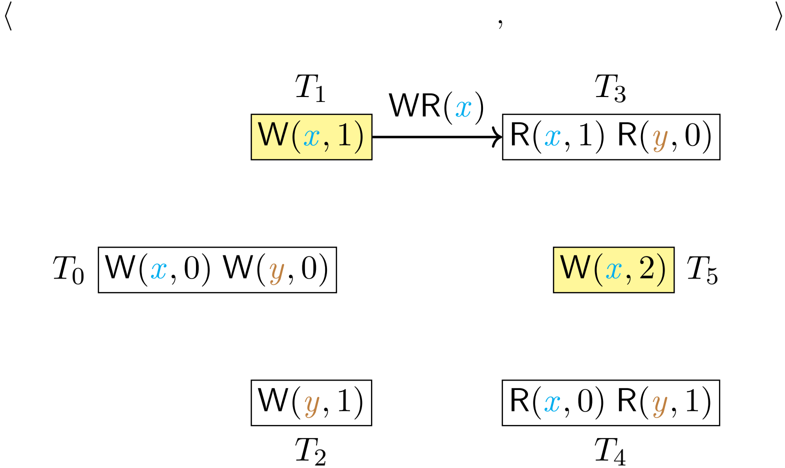


PolySI: An Illustrating Example of “Long Fork”



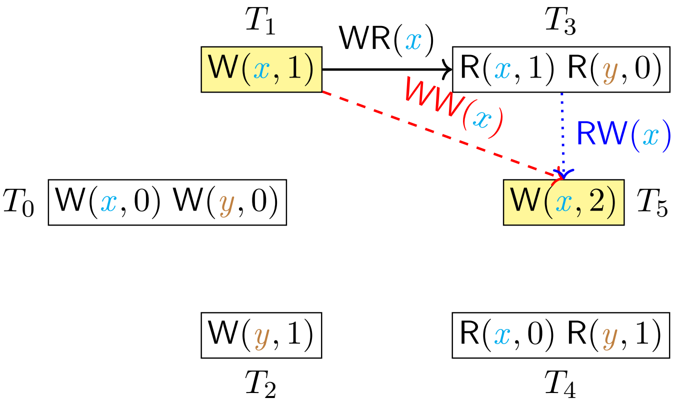
The order between T_1 and T_5 is still uncertain after pruning.

PolySI: An Illustrating Example of “Long Fork”



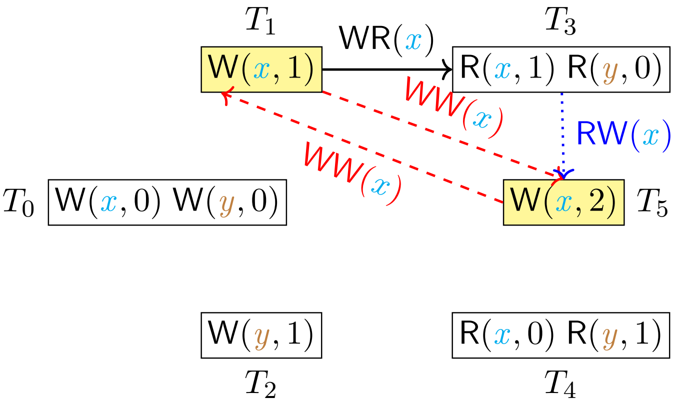
PolySI: An Illustrating Example of "Long Fork"

$$\langle either = \{T_1 \xrightarrow{\mathsf{WW}(x)} T_5, T_3 \xrightarrow{\mathsf{RW}(x)} T_5\}, \qquad \rangle$$



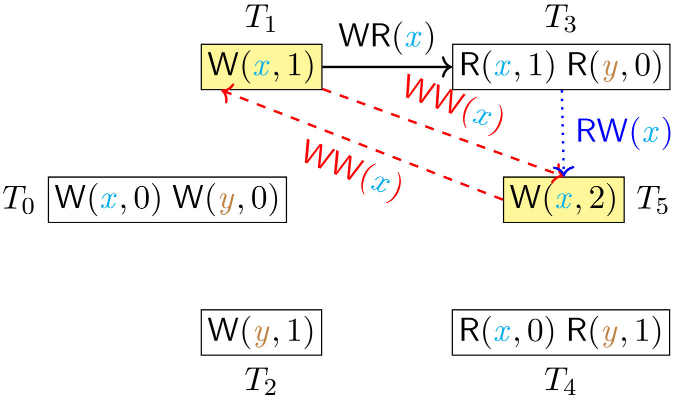
PolySI: An Illustrating Example of "Long Fork"

$$\langle either = \{T_1 \xrightarrow{\mathsf{WW}(x)} T_5, T_3 \xrightarrow{\mathsf{RW}(x)} T_5\}, or = \{T_5 \xrightarrow{\mathsf{WW}(x)} T_1\}\rangle$$



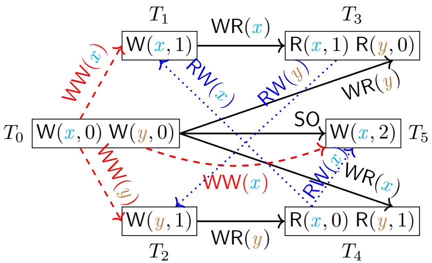
PolySI: An Illustrating Example of "Long Fork"

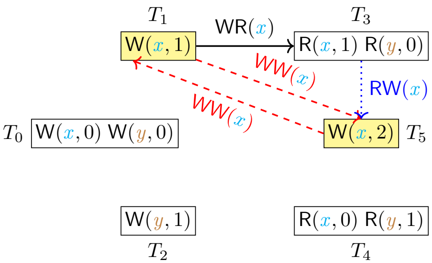
$$\langle either = \{T_1 \xrightarrow{\mathsf{WW}(x)} T_5, T_3 \xrightarrow{\mathsf{RW}(x)} T_5\}, or = \{T_5 \xrightarrow{\mathsf{WW}(x)} T_1\}\rangle$$



$$(\mathsf{BV}_{1,5} \land \mathsf{BV}_{3,5} \land \neg\mathsf{BV}_{5,1}) \lor (\mathsf{BV}_{5,1} \land \neg\mathsf{BV}_{1,5} \land \neg\mathsf{BV}_{3,5})$$

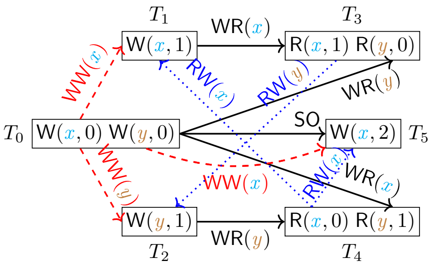
PolySI: An Illustrating Example of "Long Fork"

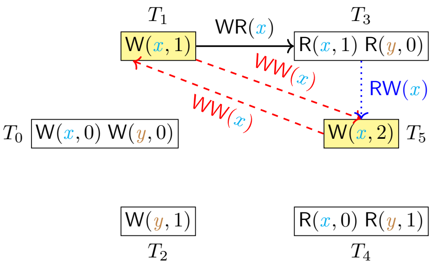




PolySI: An Illustrating Example of "Long Fork"

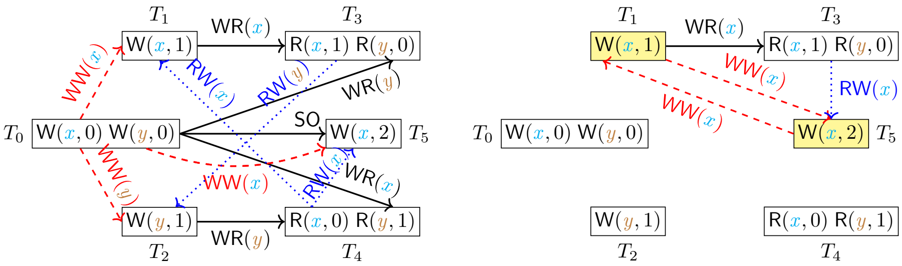
$\boxed{((\mathsf{SO}_{\mathcal{G}} \cup \mathsf{WR}_{\mathcal{G}} \cup \mathsf{WW}_{\mathcal{G}})\ ;\ \mathsf{RW}_{\mathcal{G}}?)}$ *is acyclic.*





PolySI: An Illustrating Example of "Long Fork"

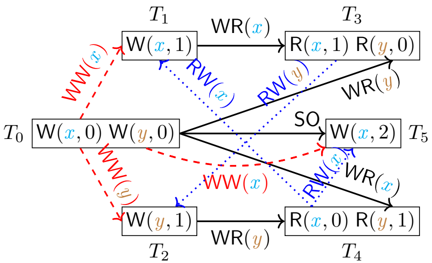
$\boxed{((\mathsf{SO}_{\mathcal{G}} \cup \mathsf{WR}_{\mathcal{G}} \cup \mathsf{WW}_{\mathcal{G}})\ ;\ \mathsf{RW}_{\mathcal{G}}?)}$ *is acyclic.*

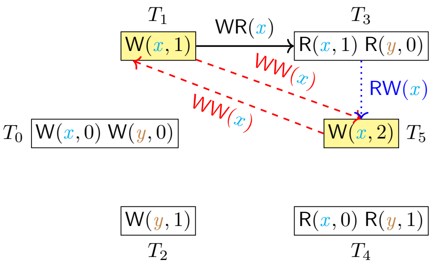


We need to encode the "composition (;)" of dependency edges.

PolySI: An Illustrating Example of "Long Fork"

$\boxed{((\mathsf{SO}_{\mathcal{G}} \cup \mathsf{WR}_{\mathcal{G}} \cup \mathsf{WW}_{\mathcal{G}})\ ;\ \mathsf{RW}_{\mathcal{G}}?)}$ *is acyclic.*



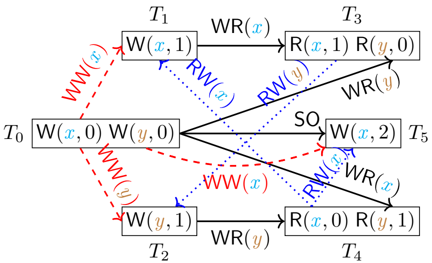


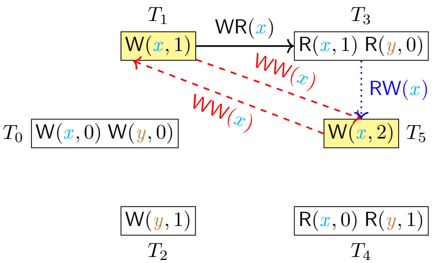
We need to encode the "composition (;)" of dependency edges.

$$T_1 \xrightarrow{\mathsf{WR}} T_3 \xrightarrow{\mathsf{RW}} T_2 : \ \mathsf{BV}^{I}_{1,2} = \mathsf{BV}_{1,3} \wedge \mathsf{BV}_{3,2} \quad (I \text{ for the induced graph})$$

PolySI: An Illustrating Example of "Long Fork"

$\boxed{((\mathsf{SO}_{\mathcal{G}} \cup \mathsf{WR}_{\mathcal{G}} \cup \mathsf{WW}_{\mathcal{G}})\ ;\ \mathsf{RW}_{\mathcal{G}}?)}$ *is acyclic.*





We need to encode the "composition (;)" of dependency edges.

$$T_1 \xrightarrow{\mathsf{WR}} T_3 \xrightarrow{\mathsf{RW}} T_2 :\ \mathsf{BV}^{I}_{1,2} = \mathsf{BV}_{1,3} \wedge \mathsf{BV}_{3,2} \quad (I \text{ for the induced graph})$$

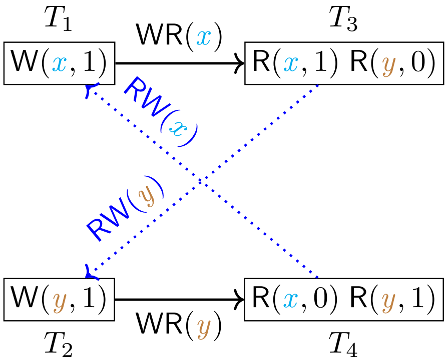
$$T_1 \xrightarrow{\mathsf{WR}} T_3 \xrightarrow{\mathsf{RW}} T_5 :\ \mathsf{BV}^{I}_{1,5} = \mathsf{BV}_{1,3} \wedge \mathsf{BV}_{3,5} \quad (I \text{ for the induced graph})$$

PolySI: An Illustrating Example of “Long Fork”

Feed the SAT formula into the MonoSAT solver [**MonoSAT:AAAI2015**] optimized for *cycle detection*

Assert that the induced graph I is acyclic.

PolySI: An Illustrating Example of "Long Fork"



The undesired cycle for "long fork" found by MonoSAT.

Experimental Evaluation

(1) *Effective:* Can PolySI find SI violations in production databases?

(2) *Informative:* Can PolySI provide understandable counterexamples for SI violations?

(3) *Efficient:* How efficient is PolySI? Is it scalable?

Workloads, Benchmarks, and Setup

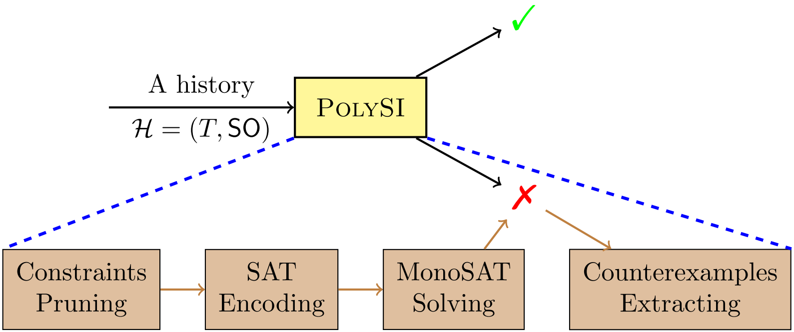
Finding SI Violations

Understanding Violations

Performance

Scalability

Conclusion



A history
$\mathcal{H} = (T, \mathsf{SO})$
PolySI
✓
✗
Constraints Pruning
SAT Encoding
MonoSAT Solving
Counterexamples Extracting

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.