Zen and the Art of Concurrency Control

An Exploration of TM Safety Property Space with Early Release in Mind

Konrad Siek and Paweł T. Wojciechowski

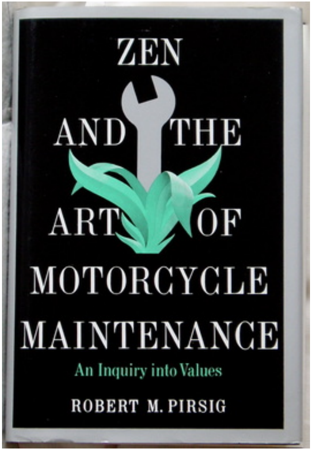
Poznań University of Technology

{konrad.siek,pawel.t.wojciechowski}@cs.put.edu.pl

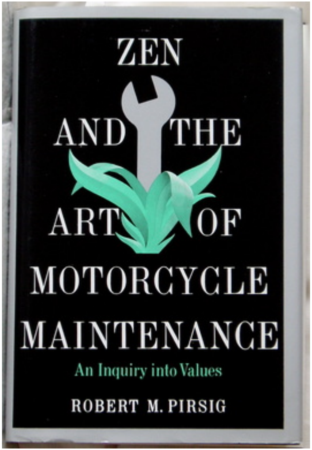
14 VII 2014

Distributed Systems Group

http://dsg.cs.put.poznan.pl

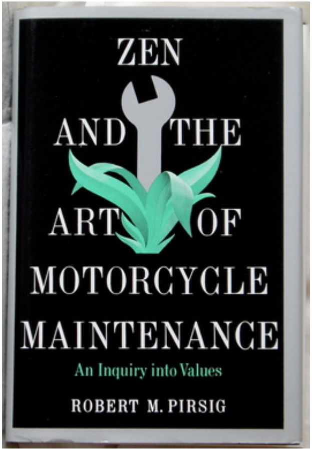


[source: wikipedia]



[source: wikipedia]

Quality is not absolutely applicable
$\rightarrow$depends on the situation



[source: wikipedia]

Quality is not absolutely applicable

$\rightarrow$depends on the situation

quality = TM safety
situation = high contention

High contention

$T_1 \llbracket\ r(x)0, w(x)1\ \rrbracket$

$T_2\ \llbracket\ r(x)0, w(x)1 \circlearrowleft \llbracket\ r(x)1, w(x)2\ \rrbracket$

$T_3\ \ \llbracket\ r(x)0, w(x)1 \circlearrowleft \llbracket\ r(x)1, w(x)2 \circlearrowleft \llbracket\ r(x)2, w(x)3\ \rrbracket$

$T_4\ \ \ \llbracket\ r(x)0, w(x)1 \circlearrowleft \llbracket\ r(x)1, w(x)2 \circlearrowleft \llbracket\ r(x)2, w(x)3 \circlearrowleft \llbracket\ r(x)3, w(x)4\ \rrbracket$

High contention

$$
\begin{array}{ll}
T_1 & \llbracket\ r(x)0, w(x)1\ \rrbracket \\
T_2 & \llbracket\ r(x)0, w(x)1 \circlearrowleft \llbracket\ r(x)1, w(x)2\ \rrbracket \\
T_3 & \llbracket\ r(x)0, w(x)1 \circlearrowleft \llbracket\ r(x)1, w(x)2 \circlearrowleft \llbracket\ r(x)2, w(x)3\ \rrbracket \\
T_4 & \llbracket\ r(x)0, w(x)1 \circlearrowleft \llbracket\ r(x)1, w(x)2 \circlearrowleft \llbracket\ r(x)2, w(x)3 \circlearrowleft \llbracket\ r(x)3, w(x)4\ \rrbracket
\end{array}
$$

$$
\begin{array}{lllll}
T_1 & \llbracket\ r(x)0, w(x)1\ \rrbracket & & & \\
T_2 & & \searrow \llbracket\ r(x)1, w(x)2\ \rrbracket & & \\
T_3 & & & \searrow \llbracket\ r(x)2, w(x)3\ \rrbracket & \\
T_4 & & & & \searrow \llbracket\ r(x)3, w(x)4\ \rrbracket
\end{array}
$$

Early release

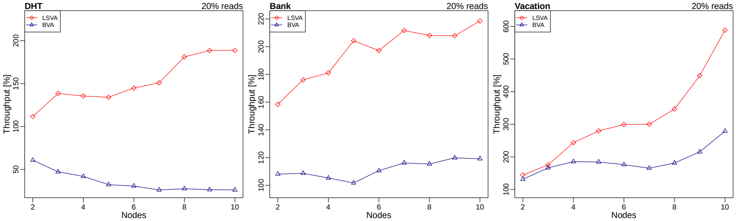
$$
\begin{array}{ll}
T_1 \llbracket\ \mathrm{r}(x)0,\ \mathrm{w}(x)1,\ \mathrm{r}(y)0,\ \mathrm{w}(y)1\ \rrbracket & \\
T_2 & \searrow \llbracket\ \mathrm{r}(x)1,\ \mathrm{w}(x)2,\ \mathrm{r}(y)1,\ \mathrm{w}(y)2\ \rrbracket
\end{array}
$$

Early release

$$\begin{array}{ll} T_1 & [\![\ \mathrm{r}(x)0,\ \mathrm{w}(x)1,\ \mathrm{r}(y)0,\ \mathrm{w}(y)1\ ]\!] \\ T_2 & \qquad\qquad\qquad [\![\ \mathrm{r}(x)1,\ \mathrm{w}(x)2,\ \mathrm{r}(y)1,\ \mathrm{w}(y)2\ ]\!] \end{array}$$

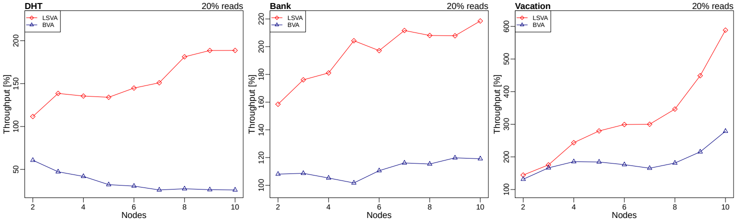
Early release

$$T_1 \llbracket\ \mathrm{r}(x)0,\ \mathrm{w}(x)1,\ \mathrm{r}(y)0,\ \mathrm{w}(y)1\ \rrbracket$$
$$T_2 \qquad\qquad\qquad\quad \llbracket\ \searrow\mathrm{r}(x)1,\ \mathrm{w}(x)2,\ \searrow\mathrm{r}(y)1,\ \mathrm{w}(y)2\ \rrbracket$$



Early release

$$
\begin{array}{ll}
T_1 & [\![\ \mathrm{r}(x)0,\ \mathrm{w}(x)1,\ \mathrm{r}(y)0,\ \mathrm{w}(y)1 \ ]\!] \\
T_2 & \qquad\qquad\qquad [\![\ \mathrm{r}(x)1,\ \mathrm{w}(x)2,\ \mathrm{r}(y)1,\ \mathrm{w}(y)2 \ ]\!]
\end{array}
$$



- M. Herlihy, V. Luchango, M. Moir, I. W. N. Scherer. *Software Transactional Memory for Dynamic-sized Data Structures*. PODC'03.
- H. E. Ramadan, I. Roy, M. Herlihy, E. Witchel. *Committing conflicting transactions in an STM*. PPoPP'09.
- P. Felber, V. Gramoli, R. Guerraoui. *Elastic Transactions*. DISC'09.
- A. Bieniusa, A. Middelkoop, P. Thiemann. *Brief Announcement: Actions in the Twilight—Concurrent Irrevocable Transactions and Inconsistency Repair*. PODC'10.
- K. Siek, P. T. Wojciechowski. *Brief Announcement: Relaxing Opacity in Pessimistic Transactional Memory*. DISC'14. (TBR)

Which TM safety properties can be used for early release?

Early release

Definition

Transaction T_i releases x early in H iff there is some prefix H' of H, such that T_i **is live** in H' and there exists T_j in H' such that there is a non-local **read operation** op_j **in** $H'|T_j$ reading v from x and a **preceding write operation** op_i **in** $H'|T_i$ writing x to v.

Example:

$$\begin{array}{ll} T_1 & [\![\ \mathrm{r}(x)0,\ \mathrm{w}(x)1,\ \mathrm{r}(y)0,\ \mathrm{w}(y)1\ ]\!] \\ T_2 & \qquad\qquad [\![\ \searrow\mathrm{r}(x)1,\ \mathrm{w}(x)2,\ \searrow\mathrm{r}(y)1,\ \mathrm{w}(y)2\ ]\!] \end{array}$$

Serializability

Definition

History H is serializable iff there exists some linear extension (sequential witness history) $\hat{S}$ such that $\hat{S}$ only contains legal transactions.

Serializability

Definition

History H is serializable iff there exists some linear extension (sequential witness history) $\hat{S}$ such that $\hat{S}$ only contains legal transactions.

Example:

$$
\begin{array}{ll}
T_1 & [\![\ \mathrm{r}(x)0,\ \mathrm{w}(x)1,\ \mathrm{r}(y)0,\ \mathrm{w}(y)1\ ]\!] \\
T_2 & \qquad\qquad\qquad\quad [\![\ \searrow \mathrm{r}(x)1,\ \mathrm{w}(x)2, \searrow \mathrm{r}(y)1,\ \mathrm{w}(y)2\ ]\!]
\end{array}
$$

Serializability

Definition

History H is serializable iff there exists some linear extension (sequential witness history) $\hat{S}$ such that $\hat{S}$ only contains legal transactions.

Example:

$$
\begin{array}{ll}
T_1 & [\![\ \mathrm{r}(x)0,\ \mathrm{w}(x)1,\ \mathrm{r}(y)0,\ \mathrm{w}(y)1\ ]\!] \\
T_2 & \qquad\qquad\qquad\qquad [\![\ \searrow \mathrm{r}(x)1,\ \mathrm{w}(x)2,\ \searrow \mathrm{r}(y)1,\ \mathrm{w}(y)2\ ]\!]
\end{array}
$$

$\hat{S} = \langle T_1, T_2 \rangle$

Serializability

Definition

History H is serializable iff there exists some linear extension (sequential witness history) $\hat{S}$ such that $\hat{S}$ only contains legal transactions.

Example:

$$\begin{array}{lll} T_1 [\![\ \mathrm{r}(x)0,\ \mathrm{w}(x)1,\ \mathrm{r}(y)0,\ \mathrm{w}(y)1 \ ]\!] & & \\ T_2 & [\![\ \mathrm{r}(x)1,\ \mathrm{w}(x)2,\ \mathrm{r}(y)1,\ \mathrm{w}(y)2 \ ]\!] & \end{array}$$

$\hat{S} = \langle T_1, T_2 \rangle$

A serializable history can contain early release.

Opacity

Components of opacity:

- Serializability
- Real-time order
- Consistency

Opacity

Components of opacity:

- Serializability
- Real-time order
- Consistency

Definition

Non-local op_r in T_i $(i \neq 0)$ is consistent if there is a preceding non-local write operation writing v to x in $H|T_k$ $(T_k \neq T_i)$ where T_k is committed or commit-pending.

Opacity

Components of opacity:

- Serializability
- Real-time order
- Consistency

Definition

Non-local op_r in T_i $(i \neq 0)$ is consistent if there is a preceding non-local write operation writing v to x in $H|T_k$ $(T_k \neq T_i)$ where T_k is **committed or commit-pending**.

Opacity

Components of opacity:

- Serializability
- Real-time order
- Consistency

Definition

Non-local op_r in T_i $(i \neq 0)$ is consistent if there is a preceding non-local write operation writing v to x in $H|T_k$ $(T_k \neq T_i)$ where T_k is **committed or commit-pending**.

Live transaction $\neq$ committed or commit-pending.

Opacity

Components of opacity:

- Serializability
- Real-time order
- Consistency

Definition

Non-local op_r in T_i $(i \neq 0)$ is consistent if there is a preceding non-local write operation writing v to x in $H|T_k$ $(T_k \neq T_i)$ where T_k is **committed or commit-pending**.

Live transaction $\neq$ committed or commit-pending.

An opaque history cannot contain early release.

Elastic opacity

Definition

History H is elastic opaque iff there exists a cutting function f_C that replaces each elastic transaction T_i in H with its consistent well-formed cut C_t, such that $f_C(H)$ is opaque.

Elastic opacity

Definition

History H is elastic opaque iff there exists a cutting function f_C that replaces each elastic transaction T_i in H with its consistent well-formed cut C_t, such that $f_C(H)$ is opaque.

Example:

$$\left.\begin{array}{ll} T_1 & [\![\ \mathrm{r}(y)0, \mathrm{w}(x)1, \qquad\qquad \mathrm{r}(x)1, \mathrm{r}(y)0\ ]\!] \\ T_2 & \qquad\qquad\qquad [\![\ \searrow \mathrm{r}(x)1\ ]\!] \end{array}\right\} H$$

Elastic opacity

Definition

History H is elastic opaque iff there exists a cutting function f_C that replaces each elastic transaction T_i in H with its consistent well-formed cut C_t, such that $f_C(H)$ is opaque.

Example:

$$\left.\begin{array}{llll}
T_1 & \llbracket\ \mathrm{r}(y)0, \mathrm{w}(x)1, & & \mathrm{r}(x)1, \mathrm{r}(y)0\ \rrbracket \\
T_2 & & \llbracket\ \searrow \mathrm{r}(x)1\ \rrbracket &
\end{array}\right\} H$$

$$\left.\begin{array}{llll}
T_1' & \llbracket\ \mathrm{r}(y)0, \mathrm{w}(x)1\ \rrbracket & & \\
T_2 & & \llbracket\ \searrow \mathrm{r}(x)1\ \rrbracket & \\
T_1'' & & & \llbracket\ \mathrm{r}(x)1, \mathrm{r}(y)0\ \rrbracket
\end{array}\right\} f_C(H)$$

Elastic opacity

Definition

History H is elastic opaque iff there exists a cutting function f_C that replaces each elastic transaction T_i in H with its consistent well-formed cut C_t, such that $f_C(H)$ is opaque.

Example:

$$\left.\begin{array}{llll}
T_1 & [\![\ \mathrm{r}(y)0, \mathrm{w}(x)1, & & \mathrm{r}(x)1, \mathrm{r}(y)0\ ]\!] \\
T_2 & & [\![\ \searrow \mathrm{r}(x)1\ ]\!] &
\end{array}\right\} H$$

$$\left.\begin{array}{llll}
T_1' & [\![\ \mathrm{r}(y)0, \mathrm{w}(x)1\ ]\!] & & \\
T_2 & & [\![\ \searrow \mathrm{r}(x)1\ ]\!] & \\
T_1'' & & & [\![\ \mathrm{r}(x)1, \mathrm{r}(y)0\ ]\!]
\end{array}\right\} f_C(H)$$

An elastic opaque history can contain early release.

Elastic opacity

Definition

History H is elastic opaque iff there exists a cutting function f_C that replaces each elastic transaction T_i in H with its consistent well-formed cut C_t, such that $f_C(H)$ is opaque.

Example:

$$\left.\begin{array}{llll} T_1 & [\![\ \mathrm{r}(y)0, \mathrm{w}(x)1, & & \mathrm{r}(x)1, \mathrm{r}(y)0\ ]\!] \\ T_2 & & [\![\ \searrow \mathrm{r}(x)1\ ]\!] & \end{array}\right\} H$$

$$\left.\begin{array}{llll} T_1' & [\![\ \mathrm{r}(y)0, \mathrm{w}(x)1\ ]\!] & & \\ T_2 & & [\![\ \searrow \mathrm{r}(x)1\ ]\!] & \\ T_1'' & & & [\![\ \mathrm{r}(x)1, \mathrm{r}(y)0\ ]\!] \end{array}\right\} f_C(H)$$

An elastic opaque history can contain early release. However...

Elastic opacity

Well-formed cut:

- A subhistory cannot start with a write (unless it is the first subhistory of a cut).
- If there are two writes in a transaction, they are within the same subhistory.
- A subhistory cannot be shorter than two operations (unless the transaction contains only one operation).

Elastic opacity

Well-formed cut:

- A subhistory cannot start with a write (unless it is the first subhistory of a cut).
- If there are two writes in a transaction, they are within the same subhistory.
- A subhistory cannot be shorter than two operations (unless the transaction contains only one operation).

$$\left.\begin{array}{llll} T_1 & [\![\ \mathrm{r}(y)0, \mathrm{w}(x)1, & & \mathrm{r}(x)1, \mathrm{r}(y)0\ ]\!] \\ T_2 & & [\![\ \searrow \mathrm{r}(x)1\ ]\!] & \end{array}\right\} H$$

$$\left.\begin{array}{llll} T_1' & [\![\ \mathrm{r}(y)0, \mathrm{w}(x)1\ ]\!] & & \\ T_2 & & [\![\ \searrow \mathrm{r}(x)1\ ]\!] & \\ T_1'' & & & [\![\ \mathrm{r}(x)1, \mathrm{r}(y)0\ ]\!] \end{array}\right\} f_C(H)$$

Elastic opacity

Well-formed cut:

- A subhistory cannot start with a write (unless it is the first subhistory of a cut).
- If there are two writes in a transaction, they are within the same subhistory.
- A subhistory cannot be shorter than two operations (unless the transaction contains only one operation).

$$\left.\begin{array}{ll} T_1 & [\![\ \mathrm{r}(y)0, \mathrm{w}(x)1, \qquad\qquad\qquad \mathrm{r}(x)1\ ]\!] \\ T_2 & \qquad\qquad\qquad [\![\ \searrow \mathrm{r}(x)1\ ]\!] \end{array}\right\} H$$

$$\left.\begin{array}{ll} T_1' & [\![\ \mathrm{r}(y)0, \mathrm{w}(x)1\ ]\!] \\ T_2 & \qquad\qquad\qquad [\![\ \searrow \mathrm{r}(x)1\ ]\!] \\ T_1'' & \qquad\qquad\qquad\qquad\qquad\qquad [\![\ \mathrm{r}(x)1\ ]\!] \end{array}\right\} f_C(H)$$

TMS1 & TMS2

Definition (TMS1—Valid Response)

For operation op to return in some subhistory $H|T_i$, there must exist some set of transactions S that follow real-time order and justify the legality of op, and for any $T_j \in S$ it is true that,

- if T_j precedes T_i in real-time order then T_j is committed, or
- T_j is committed or commit-pending otherwise.

TMS1 & TMS2

Definition (TMS1—Valid Response)

For operation op to return in some subhistory $H|T_i$, there must exist some set of transactions S that follow real-time order and justify the legality of op, and for any $T_j \in S$ it is true that,

- if T_j precedes T_i in real-time order then T_j is **committed**, or
- T_j is **committed or commit-pending** otherwise.

TMS1 & TMS2

Definition (TMS1—Valid Response)

For operation op to return in some subhistory $H|T_i$, there must exist some set of transactions S that follow real-time order and justify the legality of op, and for any $T_j \in S$ it is true that,

- if T_j precedes T_i in real-time order then T_j is **committed**, or
- T_j is **committed or commit-pending** otherwise.

Live transaction $\neq$ committed or commit-pending.

TMS1 & TMS2

Definition (TMS1—Valid Response)

For operation op to return in some subhistory $H|T_i$, there must exist some set of transactions S that follow real-time order and justify the legality of op, and for any $T_j \in S$ it is true that,

- if T_j precedes T_i in real-time order then T_j is **committed**, or
- T_j is **committed or commit-pending** otherwise.

Live transaction $\neq$ committed or commit-pending.

A TMS1 history cannot contain early release.

TMS1 & TMS2

Definition (TMS1—Valid Response)

For operation op to return in some subhistory $H|T_i$, there must exist some set of transactions S that follow real-time order and justify the legality of op, and for any $T_j \in S$ it is true that,

- if T_j precedes T_i in real-time order then T_j is **committed**, or
- T_j is **committed or commit-pending** otherwise.

Live transaction $\neq$ committed or commit-pending.

A TMS1 history cannot contain early release.

A TMS2 history cannot contain early release (TMS2 $\subset$ TMS1).

Virtual world consitency

Definition

History H is VWC iff all committed transactions are strict serializable, and for all aborted transactions there exists a linear extension of its causal past that is legal.

Virtual world consitency

Definition

History H is VWC iff all committed transactions are **strict serializable**, and for all aborted transactions there exists a linear extension of its causal past that is legal.

Virtual world consitency

Definition

History H is VWC iff all committed transactions are **strict serializable**, and for all aborted transactions there exists a linear extension of its causal past that is legal.

Example:

$$\begin{array}{ll} T_i & [\![\ \mathrm{r}(x)0, \mathrm{w}(x)1, \mathrm{r}(y)0\ ]\!] \\ T_j & \qquad\qquad\qquad [\![\ \searrow \mathrm{r}(x)1\ ]\!] \end{array}$$

Virtual world consitency

Definition

History H is VWC iff all committed transactions are **strict serializable**, and for all aborted transactions there exists a linear extension of its causal past that is legal.

Example:

$$\begin{array}{lll} T_i & [\![\ \mathrm{r}(x)0, \mathrm{w}(x)1, \mathrm{r}(y)0\ ]\!] & \\ T_j & & [\![\ \searrow \mathrm{r}(x)1\ ]\!] \end{array}$$

$\hat{S} = \langle T_1, T_2 \rangle$

Virtual world consitency

Definition

History H is VWC iff all committed transactions are **strict serializable**, and for all aborted transactions there exists a linear extension of its causal past that is legal.

Example:

$$\begin{array}{lll} T_i & [\![\ \mathrm{r}(x)0, \mathrm{w}(x)1, \mathrm{r}(y)0\ ]\!] & \\ T_j & & [\![\ \searrow \mathrm{r}(x)1\ ]\!] \end{array}$$

$\hat{S} = \langle T_1, T_2 \rangle$

A VWC history can contain early release.

Virtual world consitency

If T_i releases early in H, then T_i cannot abort.

Virtual world consitency

If T_i releases early in H, then T_i cannot abort.

$$
\begin{array}{lll}
T_i & [\![\ \mathrm{r}(x)0, \mathrm{w}(x)1, \mathrm{r}(y)0 & \circlearrowleft \\
T_j & & [\![\ \searrow \mathrm{r}(x)1\ ]\!]
\end{array}
$$

Virtual world consitency

If T_i releases early in H, then T_i cannot abort.

$$\begin{array}{ll} T_i & [\![\ \mathrm{r}(x)0, \mathrm{w}(x)1, \mathrm{r}(y)0 \ \circlearrowleft \\ T_j & \qquad\qquad\qquad\quad [\![\ \searrow \mathrm{r}(x)1 \ ]\!] \end{array}$$

- If T_j eventually commits, then the sequential witness history $\hat{S} = \langle T_i, T_j \rangle$ is illegal.

Virtual world consitency

If T_i releases early in H, then T_i cannot abort.

$$
\begin{array}{lll}
T_i & \llbracket\ \mathrm{r}(x)0, \mathrm{w}(x)1, \mathrm{r}(y)0\ \circlearrowleft & \\
T_j & & \llbracket\ \mathrm{r}(x)1\ \circlearrowleft
\end{array}
$$

- If T_j eventually commits, then the sequential witness history $\hat{S} = \langle T_i, T_j \rangle$ is illegal.

Virtual world consitency

If T_i releases early in H, then T_i cannot abort.

$$\begin{array}{ll} T_i & \llbracket\ \mathrm{r}(x)0, \mathrm{w}(x)1, \mathrm{r}(y)0\ \circlearrowleft \\ T_j & \qquad\qquad\qquad\quad \llbracket\ \mathrm{r}(x)1\ \circlearrowleft \end{array}$$

- If T_j eventually commits, then the sequential witness history $\hat{S} = \langle T_i, T_j \rangle$ is illegal.
- If T_j eventually aborts, its causal past $C(T_j) = \langle T_i, T_j \rangle$ contains two aborted transactions, so it is illegal.

- Serializability
    - supports early release
    - very basic

- Serializability
  - supports early release
  - very basic
- Opacity—no early release
- TMS1 & TMS2—no early release

- Serializability
  - supports early release
  - very basic
- Opacity—no early release
- TMS1 & TMS2—no early release
- Virtual world consistency
  - supports early release
  - transactions cannot abort

- Serializability
  - supports early release
  - very basic
- Opacity—no early release
- TMS1 & TMS2—no early release
- Virtual world consistency
  - supports early release
  - transactions cannot abort
- Elastic opacity
  - supports early release
  - unintuitive cutting rules

Database properties

- Recoverability
- Avoiding Cascading Aborts
- Strictness
- Rigorousness

Database properties

- Recoverability

  History H is recoverable iff for any $T_i, T_j \in H$ s.t. T_j reads from T_i, T_i commits in H before T_j.

- Avoiding Cascading Aborts
- Strictness
- Rigorousness

Database properties

- Recoverability ✓

  History H is recoverable iff for any $T_i, T_j \in H$ s.t. T_j reads from T_i, T_i commits in H before T_j.
- Avoiding Cascading Aborts
- Strictness
- Rigorousness

Database properties

- Recoverability ✓

  History H is recoverable iff for any $T_i, T_j \in H$ s.t. T_j reads from T_i, T_i commits in H before T_j.

- Avoiding Cascading Aborts

  History H Avoids Cascading Aborts iff for any $T_i, T_j \in H$ s.t. T_j reads from T_i, T_i commits before the read.

- Strictness
- Rigorousness

Database properties

- Recoverability ✓

  History H is recoverable iff for any $T_i, T_j \in H$ s.t. T_j reads from T_i, T_i commits in H before T_j.

- Avoiding Cascading Aborts $\approx$

  History H Avoids Cascading Aborts iff for any $T_i, T_j \in H$ s.t. T_j reads from T_i, T_i commits before the read.

- Strictness
- Rigorousness

Database properties

- Recoverability ✓

  History H is recoverable iff for any $T_i, T_j \in H$ s.t. T_j reads from T_i, T_i commits in H before T_j.

- Avoiding Cascading Aborts ≈

  History H Avoids Cascading Aborts iff for any $T_i, T_j \in H$ s.t. T_j reads from T_i, T_i commits before the read.

- Strictness

  History H is strict iff for any $T_i, T_j \in H$ and given any operation $op_i = \mathrm{r}(x)v$ or $\mathrm{w}(x)v'$ in $H|T_i$, and any operation $op_j = \mathrm{w}(x)v$ in $H|T_j$, if op_i follows op_j, then T_j commits or aborts before op_i.

- Rigorousness

Database properties

- Recoverability ✓

  History H is recoverable iff for any $T_i, T_j \in H$ s.t. T_j reads from T_i, T_i commits in H before T_j.

- Avoiding Cascading Aborts ≈

  History H Avoids Cascading Aborts iff for any $T_i, T_j \in H$ s.t. T_j reads from T_i, T_i commits before the read.

- Strictness ✗

  History H is strict iff for any $T_i, T_j \in H$ and given any operation $op_i = \mathrm{r}(x)v$ or $\mathrm{w}(x)v'$ in $H|T_i$, and any operation $op_j = \mathrm{w}(x)v$ in $H|T_j$, if op_i follows op_j, then T_j commits or aborts before op_i.

- Rigorousness

Database properties

- Recoverability ✓

  History H is recoverable iff for any $T_i, T_j \in H$ s.t. T_j reads from T_i, T_i commits in H before T_j.

- Avoiding Cascading Aborts $\approx$

  History H Avoids Cascading Aborts iff for any $T_i, T_j \in H$ s.t. T_j reads from T_i, T_i commits before the read.

- Strictness ✗

  History H is strict iff for any $T_i, T_j \in H$ and given any operation $op_i = \mathrm{r}(x)v$ or $\mathrm{w}(x)v'$ in $H|T_i$, and any operation $op_j = \mathrm{w}(x)v$ in $H|T_j$, if op_i follows op_j, then T_j commits or aborts before op_i.

- Rigorousness

  History H is rigorous if it is strict and for any $T_i, T_j \in H$ such that T_j writes to variable x, i.e., $op_j = \mathrm{w}(x)v \in H|T_j$ after T_i reads x, then T_i commits or aborts before op_j.

Database properties

- Recoverability ✓

  History H is recoverable iff for any $T_i, T_j \in H$ s.t. T_j reads from T_i, T_i commits in H before T_j.

- Avoiding Cascading Aborts ≈

  History H Avoids Cascading Aborts iff for any $T_i, T_j \in H$ s.t. T_j reads from T_i, T_i commits before the read.

- Strictness ✗

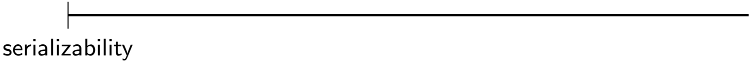
  History H is strict iff for any $T_i, T_j \in H$ and given any operation $op_i = \mathrm{r}(x)v$ or $\mathrm{w}(x)v'$ in $H|T_i$, and any operation $op_j = \mathrm{w}(x)v$ in $H|T_j$, if op_i follows op_j, then T_j commits or aborts before op_i.

- Rigorousness ✗

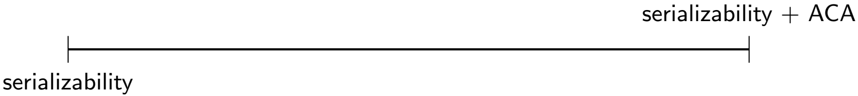
  History H is rigorous if it is strict and for any $T_i, T_j \in H$ such that T_j writes to variable x, i.e., $op_j = \mathrm{w}(x)v \in H|T_j$ after T_i reads x, then T_i commits or aborts before op_j.

Serializability+ spectrum

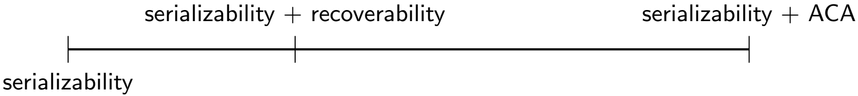
Serializability+ spectrum



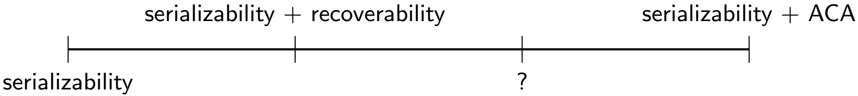
Serializability+ spectrum



Serializability+ spectrum



Serializability+ spectrum



Last-use

Definition (Commit-pending–equivalence)

A live transaction T_i in H is commit-pending–equivalent with respect to x iff it is finished executing all of its operations on x.

Last-use

Definition (Commit-pending–equivalence)

A live transaction T_i in H is commit-pending–equivalent with respect to x iff it is finished executing all of its operations on x.

```
atomic{
    int v = read(x);
    if (v < 0)
        write(x,-v); // commit-pending--equivalent wrt x
    int u = read(y);
    write(y, u + 1); // commit-pending--equivalent wrt y
}
```

Last-use

Definition (Commit-pending–equivalence)

A live transaction T_i in H is commit-pending–equivalent with respect to x iff it is finished executing all of its operations on x.

```
atomic{
     int v = read(x);
     if (v < 0)
         write(x,-v); // commit-pending--equivalent wrt x
     int u = read(y);
     write(y, u + 1); // commit-pending--equivalent wrt y
}
```

Definition (Early release after last use)

Transaction T_i releases x after last use in H iff T_i releases x early in H and T_i is commit-pending equivalent wrt x.

Last-use consistency

Definition

- Let $\mathbb{T}_{er}^{H}$ be a subset of all transactions in history H that release some variable early.
- Let $\mathbb{T}_{lu}^{H}$ be a subset of all transactions in history H that release some variable early only after last-use.

History H satisfies last-use consistency if $\mathbb{T}_{lu}^{H} = \mathbb{T}_{er}^{H}$.

Inconsistent views

Last-use consistency precludes overwriting:

$$\begin{array}{ll} T_i & [\![\ \mathrm{w}(x)0, \quad \mathrm{w}(x)1 \ ]\!] \\ T_j & \qquad [\![\ \searrow \mathrm{r}(x)0 \ \searrow \circlearrowleft T'_j \ [\![\ \mathrm{r}(x)1, \mathrm{w}(x)2 \ ]\!] \end{array}$$

Inconsistent views

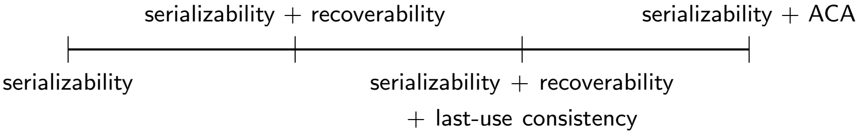
Last-use consistency precludes overwriting:

$$
\begin{array}{ll}
T_i & \llbracket\ \mathrm{w}(x)0,\quad \mathrm{w}(x)1\ \rrbracket \\
T_j & \qquad\quad \llbracket\ \searrow \mathrm{r}(x)0\ \ \searrow \circlearrowleft T'_j\ \llbracket\ \mathrm{r}(x)1, \mathrm{w}(x)2\ \rrbracket
\end{array}
$$

Allowed inconsistent view:

$$
\begin{array}{ll}
T_i & \llbracket\ \mathrm{w}(x)0,\quad \mathrm{w}(x)1 \qquad\quad \circlearrowleft \\
T_j & \qquad\qquad\qquad\qquad \llbracket\ \searrow \mathrm{r}(x)0\ \ \searrow \circlearrowleft T'_j \llbracket\ \mathrm{r}(x)1, \mathrm{w}(x)2\ \rrbracket
\end{array}
$$

Serializability+ spectrum



Conclusions

- Current safety properties not enough for TM with early release
- Spectrum of database consistency properties
- Last-use consistency

Conclusions

- Current safety properties not enough for TM with early release
- Spectrum of database consistency properties
- Last-use consistency
- Future work: last-use opacity

?