Safety of Transactions in Transactional Memory: TMS is Necessary and Sufficient

Hagit Attiya, Technion
Sandeep Hans, Technion
Alexey Gotsman, IMDEA
Noam Rinetzky, Tel-Aviv University
TM Consistency Conditions

Opacity

- Validity of all transactions (including aborted ones) is checked together

Transactional Memory Specification (TMS1/2)

- In TMS1, validity of each response is checked against a coherent subset of the transactions
  - May even include aborted transactions

Virtual World Consistency

- [Imbs & Raynal 09]
Comparing TM Consistency Conditions

What is the “right” consistency condition?

Does the TM consistency condition allows to program with a simpler (i.e., atomic) TM in mind?

• If local variables are rolled back after a transaction aborts, \( TMS(1) \) is sufficient and necessary for programming with an atomic TM in mind.

• If local variables are not rolled back on an abort (e.g., ScalaTM), the stronger Opacity condition is necessary [Attiya, Gotsman, H, Rinetzky 13].
Observational Refinement

[He, Hoare, Sanders 86]

• What is guaranteed for client programs, when an implementation is replaced with a simpler one?

Client Program

$TM_C$
Interactions of a Program using TM

- **Local actions**: access only the local variables
- **Global actions**: interact with other client programs
- **Interface actions**: interact with TM

Client Program

```plaintext
x:=0; y:=0; z:=0;
result := abort;
while(result == abort){
    result := atomic{
        x = X.read();
        y = Y.read();
        z = 42 / (x-y);
        Z.write(z);
    }
} 
g := z;
```

TM
**Histories**

**History**: Finite sequence of interface actions

**Well-formed**: Threads are sequential

---

**Transactional Memory (TM)**: set of histories

- well-formed, prefix-closed

```plaintext
x:=0; y:=0; z:=0;
result := abort;
while(result == abort){
    result := atomic{
        x = X.read();
        y = Y.read();
        z = 42 / (x-y);
        Z.write(z);
    }
} 
g := z;
```
Trace Equivalence

**Trace:** includes also local and global actions

Two traces are **observationally equivalent** $\tau \sim \tau'$ if threads have the same sequence of local values, except for local values inside aborted transactions.

**TM**$_C$ **observationally refines** **TM**$_A$ if every trace $\tau$ with history in **TM**$_C$ has a trace $\tau' \sim \tau$ with history in **TM**$_A$.
Why Observational Refinement?

Prove properties for $\text{TM}_A$ and deduce the same for $\text{TM}_C$

$\text{TM}_C$ observationally refines $\text{TM}_A$ if every trace $\tau$ with history in $\text{TM}_C$ has a trace $\tau' \sim \tau$ with history in $\text{TM}_A$
- transaction of $\phi$ is included
- **some visible** transactions are included
- for every included transaction, **exactly** all past committed transactions are included
- **Commit** included aborted transactions (by replacing abort with commit)
- **Commit** included commit-pending transactions
- **Remove** all other transactions
H’TMSPast(H₁φ)
- H’ is a subsequence of H
- H’ contains transaction of φ and some visible transactions in H
- for every included transaction T in H’, exactly all past committed transactions are included

H’c ∈ cTMSPast(H₁φ)
- commit all commit-pending transactions
- replace aborted actions by committed actions
Definition of TMS

H^c \sqsubseteq S

- S preserves the **per-thread** and **real-time** order of H^c

H \sqsubseteq_{tms} TMA

- all committed transactions have a serialization
- for every response action \( \phi \), there is a complete past \( H^c \) and a history \( S \in TMA \) such that \( H^c \sqsubseteq S \)

TM^C is TMS \( \iff \) for every \( H \in TM^C \), \( H \sqsubseteq_{tms} TMA_{\text{ATOMIC}} \)
Main Result

\[ \text{TMC} \equiv_{tms} \text{TMA} \iff \text{TMC} \text{ observationally refines } \text{TMA} \]

- no nesting of atomic blocks
- no access to global variables in atomic blocks
Every trace $\tau$ observed when running with $\text{TM}_C$ has an equivalent trace $\tau'$ observed when running with $\text{TM}_A$

- Consider a trace $\tau$ whose history $H$ is in $\text{TM}_C$
- $\text{TM}_C \subseteq_{\text{tms}} \text{TM}_A \Rightarrow H^C \in \text{cTMSPast}(H)$ and $H^C \cong S \in \text{TM}_A$
- From $\tau$ and $S$, get a trace $\tau' \sim \tau$ of $\text{TM}_A$ whose history is $S$
Let $X$ be the beginning of the last included transaction.

- For every thread $t$, take the trace until the latest of:
  - The last non-transactional action before $X$
  - The last transactional action of $t$ in $H'$
What’s Next?

- **Extend** the results to handle nesting and access to global variables in atomic blocks

- **Weaker** observations are preserved by VWC?

- **Stronger** observations (e.g., global accesses in a transaction) are preserved by deferred update opacity or TMS2?