How Hard is Weak Memory Testing?

S. Chakraborty, S. Krishna, U. Mathur and Andreas Pavlogiannis
Concurrency is Everywhere

Concurrency is a ubiquitous computing paradigm
Concurrency is Everywhere

Concurrency is a ubiquitous computing paradigm

Verification:

- Is my program correct in a given concurrent setting?
- What behaviors are possible in a concurrent environment?
Concurrency is Everywhere

Concurrency is a ubiquitous computing paradigm

Verification:

- Is my program correct in a given concurrent setting?
- What behaviors are possible in a concurrent environment?

Not all concurrency is the same
## Store Buffer under x86-TSO

### Store Buffer

\[
x = 0, \quad y = 0
\]

\[
w(y, 1); \quad w(x, 1);
\]

\[
a := r(x, 0) \quad \parallel \quad b := r(y, 0)
\]

Not sequentially consistent
Store Buffer under x86-TSO

\[
x = 0, y = 0
\]

\[
w(y, 1);
\]
\[
a := r(x, 0)
\]

\[
w(x, 1);
\]
\[
b := r(y, 0)
\]
Store Buffer under x86-TSO

Store Buffer

\[
\begin{align*}
x &= 0, y &= 0 \\
w(y, 1); &\quad w(x, 1); \\
a := r(x, 0); &\quad b := r(y, 0)
\end{align*}
\]
Store Buffer under x86-TSO

<table>
<thead>
<tr>
<th>Store Buffer</th>
<th>x = 0, y = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>w(y, 1);</td>
<td>w(x, 1);</td>
</tr>
<tr>
<td>a := r(x, 0)</td>
<td>b := r(y, 0)</td>
</tr>
</tbody>
</table>

Behavior possible under x86-TSO

Not sequentially consistent

\[ a = 0 \quad b = 0 \]

\[ x = 0, y = 0 \quad x = 1 \]

\[ y = 1 \]
Store Buffer under x86-TSO

Store Buffer

\[ x = 0, y = 0 \]

\[ w(y, 1); \quad w(x, 1); \]
\[ a := r(x, 0) \quad b := r(y, 0) \]

Not sequentially consistent

\[ a = 0 \quad b = 0 \]
\[ y = 1 \quad x = 1 \]
\[ x = 0, y = 0 \]

\[ \]
Store Buffer under x86-TSO

\[
\begin{align*}
x &= 0, \quad y = 0 \\
w(y, 1); & \quad \| \quad w(x, 1); \\
a := r(x, 0) & \quad \| \quad b := r(y, 0)
\end{align*}
\]
Store Buffer under x86-TSO

Store Buffer

\[
\begin{align*}
  x & = 0, y = 0 \\
  w(y, 1); & \quad w(x, 1); \quad a := r(x, 0) \quad b := r(y, 0)
\end{align*}
\]

- Behavior possible under x86-TSO
Memory Models

- Formal models of concurrency
- Specifications of all possible communication patterns
  - Buffers/caching
  - Out-of-order execution
  - Speculation
  - Cache coherence protocols
  - Compiler optimizations
  - Message delays
  - ...
- In all cases: weak data consistency
## Memory Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC</td>
<td>Standard Sequential Consistency</td>
</tr>
<tr>
<td>TSO</td>
<td>x86-Total Store Order</td>
</tr>
<tr>
<td>PSO</td>
<td>Sparc-Partial Store Order</td>
</tr>
<tr>
<td>RA</td>
<td>The release-acquire semantics of C11</td>
</tr>
<tr>
<td>Relaxed</td>
<td>The relaxed fragment of C11</td>
</tr>
<tr>
<td>Relaxed-Acyclic</td>
<td>Relaxed + (po (\cup) rf)-acyclicity</td>
</tr>
<tr>
<td>CC</td>
<td>Causal consistency</td>
</tr>
<tr>
<td>CCv</td>
<td>Causal convergence</td>
</tr>
<tr>
<td>CM</td>
<td>Causal memory</td>
</tr>
</tbody>
</table>
Program executions are represented as execution graphs.

Executions on Weak Memory

An execution is a tuple $X = (E, po, rf, mo)$, where:

- $E$ is a set of events
- $po$ is the program order over $E$, total on each thread
- $rf$ is a reads-from relation on $W \times R$
  - $(w, r) \in rf$ means $r$ reads the value written by $w$
- $mo_x$ is a total modification order over all writes $w(x)$
  - $mo = \bigcup_x mo_x$
A memory model $\mathcal{M}$ defines a set of axioms that every execution must satisfy.

**Consistency**

If an execution $X = (E, \text{po}, rf, \text{mo})$ satisfies all axioms of $\mathcal{M}$, we say that $X$ is **consistent** in $\mathcal{M}$, written as $X \models \mathcal{M}$.
Consistent Executions

A memory model \( \mathcal{M} \) defines a set of axioms that every execution must satisfy.

### Consistency

If an execution \( X = (E, po, rf, mo) \) satisfies all axioms of \( \mathcal{M} \), we say that \( X \) is consistent in \( \mathcal{M} \), written as \( X \vDash \mathcal{M} \).

Memory models may be ordered in terms of the behaviors they allow, i.e., the executions they admit.

### Weak(er) Memory Models

Given two memory models \( \mathcal{M}_1, \mathcal{M}_2 \), we say that \( \mathcal{M}_2 \) is weaker than \( \mathcal{M}_1 \), written \( \mathcal{M}_1 \sqsubseteq \mathcal{M}_2 \), if for every execution \( X \), we have

\[
X \vDash \mathcal{M}_1 \Rightarrow X \vDash \mathcal{M}_2
\]

Eg, \( SC \sqsubseteq TSO \sqsubseteq RA \sqsubseteq \{CC, Relaxed\} \)
Examples of Consistency

(a) Total Store Order (TSO)

(b) Causal Convergence (CCv)

(c) Release/Acquire (RA)

(d) Causal Memory (CM)

(e) Causal Consistency (CC)

(f) Relaxed (Relaxed)
Examples of Consistency

(a) Total Store Order (TSO)
Examples of Consistency

(a) Total Store Order (TSO)    (b) Causal Convergence (CCv)

\[ w(x) \quad w(y) \]
\[ w(x) \quad w(y) \]
\[ r(y) \quad r(x) \]

\[ w(x) \quad w(y) \quad w(z) \]
\[ r(y) \quad r(x) \quad r(z) \]
Examples of Consistency

(a) Total Store Order (TSO)  (b) Causal Convergence (CCv)  (c) Release/Acquire (RA)
Examples of Consistency

(a) Total Store Order (TSO)  
(b) Causal Convergence (CCv)  
(c) Release/Acquire (RA)

(d) Causal Memory (CM)
Examples of Consistency

(a) Total Store Order (TSO)

(b) Causal Convergence (CCv)

(c) Release/Acquire (RA)

(d) Causal Memory (CM)

(e) Causal Consistency (CC)

(f) Relaxed (Relaxed)
Testing Weak Memories

• Aligning a model to an implementation is hard, litmus tests

Is my implementation correct?

Implementation → Test → Memory Model
Testing Weak Memories

- Aligning a model to an implementation is hard, litmus tests

Is my model correct?

Implementation → Test → Memory Model
Testing Weak Memories

- Aligning a model to an implementation is hard, litmus tests

  Is my model correct?

  Implementation → Test → Memory Model

- Also, model checking, dynamic analyses
Testing Weak Memories

- Aligning a model to an implementation is hard, litmus tests

- Also, model checking, dynamic analyses

- Is the **observed behavior** of the program in alignment with the model?
- Observed behavior is thread-local
  - No rf, no mo
Weak Memory Testing, Formally

<table>
<thead>
<tr>
<th>The Testing Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Given an abstract execution $\bar{X} = (E, po)$ and a memory model $M$ is there a reads from relation $rf$ and a modification order $mo$ such that $X = (E, po, rf, mo) \models M$?</td>
</tr>
</tbody>
</table>
The Testing Problem

Given an abstract execution $\overline{X} = (E, po)$ and a memory model $M$ is there a reads from relation $rf$ and a modification order $mo$ such that $X = (E, po, rf, mo) \models M$?
Weak Memory Testing, Formally

The Testing Problem

Given an abstract execution $\overline{X} = (E, po)$ and a memory model $\mathcal{M}$ is there a reads from relation $rf$ and a modification order $mo$ such that $X = (E, po, rf, mo) \models \mathcal{M}$?
How Fast can we Test?

$n$ events, $k$ threads, $d$ memory locations

**NP-complete** for $k = 3$
**NP-complete** for $d = 1$

Sequential Consistency

**P** for $k, d = O(1)$

Weak Memory
How Fast can we Test?

$n$ events, $k$ threads, $d$ memory locations

- **Sequential Consistency**
  - NP-complete for $k = 3$
  - NP-complete for $d = 1$

**Weak Memory**

- NP-complete for $k = O(1)$
- NP-complete for $d = O(1)$

**P** for $k, d = O(1)$
How Fast can we Test?

$n$ events, $k$ threads, $d$ memory locations

**NP-complete** for $k = 3$
**NP-complete** for $d = 1$

Sequential Consistency

**P** for $k, d = O(1)$

**NP-complete** for $k = O(1)$
**NP-complete** for $d = O(1)$

Weak Memory

???
The Hardness of Weak Memory Testing

Theorem (Hardness of bounded testing)

Testing is \textbf{NP-hard} for any memory model among

- CCv
- RA
- CM
- CC
- Relaxed-Acyclic

\textit{even for abstract executions with bounded}

- \textit{threads}
- \textit{memory locations}
- \textit{values read/written}
The Hardness of Weak Memory Testing

**Theorem (Hardness of bounded testing)**

*Testing is NP-hard for any memory model among*

- CCv
- RA
- CM
- CC
- Relaxed-Acyclic

*even for abstract executions with bounded*

- *threads*
- *memory locations*
- *values read/written*
The Proof is a Bit Involved . . .

Reduction from monotone 1-in-3 SAT $\phi = C_1 \land C_2 \land \cdots \land C_m$

- No negations: $C_i = x_i^1 \lor x_i^2 \lor x_i^3$
- For each clause, exactly one variable must be true
The Proof is a Bit Involved . . .

Reduction from monotone 1-in-3 SAT $\phi = C_1 \land C_2 \land \cdots \land C_m$

- No negations: $C_i = x_i^1 \lor x_i^2 \lor x_i^3$
- For each clause, exactly one variable must be true

A Copy Gadget:
How General is this Hardness?

- Hardness proofs are difficult
- Do they generalize?
How General is this Hardness?

- Hardness proofs are difficult
- Do they generalize?

**Theorem**

For any memory model $\mathcal{M}$ with

- $\text{CCv}\subseteq\mathcal{M}\subseteq\text{CC}$, or
- $\text{CM}\subseteq\mathcal{M}\subseteq\text{CC}$

testing bounded executions is NP-hard.
How General is this Hardness?

- Hardness proofs are difficult
- Do they generalize?

**Theorem**

For any memory model $\mathcal{M}$ with

- $CCv \subseteq \mathcal{M} \subseteq CC$, or
- $CM \subseteq \mathcal{M} \subseteq CC$

*testing bounded executions is NP-hard.*

**However!** Bounded testing is in $P$ for some weak memory
How Hard is Weak Memory Testing? **Very Hard**
How Hard is Weak Memory Testing? Very Hard

Thank you!
Questions?
Why Should this be Hard?

![Diagram showing causal consistency]

Not Causally Consistent

Causally Consistent
Why Should this be Hard?

Not Causally Consistent

Causally Consistent; but not SC
Reads-From (RF) Testing

<table>
<thead>
<tr>
<th>RF-Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Given an abstract execution ( \bar{X} = (E, po, rf) ) and a memory model ( \mathcal{M} ) is there a modification order ( mo ) such that ( X = (E, po, rf, mo) \models \mathcal{M} )?</td>
</tr>
</tbody>
</table>
Reads-From (RF) Testing

**RF-Testing**

Given an abstract execution $\bar{X} = (E, po, rf)$ and a memory model $\mathcal{M}$ is there a modification order $mo$ such that $X = (E, po, rf, mo) \models \mathcal{M}$?
Given an abstract execution $\bar{X} = (E, po, rf)$ and a memory model $\mathcal{M}$ is there a modification order $mo$ such that $X = (E, po, rf, mo) \models \mathcal{M}$?
## How Hard is RF-Testing?

<table>
<thead>
<tr>
<th></th>
<th>RF-Testing</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC</td>
<td>NP-complete</td>
<td>P for $k, d = O(1)$</td>
</tr>
<tr>
<td>TSO</td>
<td>NP-complete</td>
<td>P for $k, d = O(1)$</td>
</tr>
<tr>
<td>PSO</td>
<td>NP-complete</td>
<td>P for $k, d = O(1)$</td>
</tr>
<tr>
<td>RA</td>
<td>$O(n \cdot k)$</td>
<td>NP-complete for $k, d = O(1)$</td>
</tr>
<tr>
<td>CC</td>
<td>$O(n \cdot k)$</td>
<td>NP-complete for $k, d = O(1)$</td>
</tr>
<tr>
<td>CCv</td>
<td>$O(n \cdot k)$</td>
<td>NP-complete for $k, d = O(1)$</td>
</tr>
<tr>
<td>CM</td>
<td>$O(n \cdot k)$</td>
<td>NP-complete for $k, d = O(1)$</td>
</tr>
<tr>
<td>Relaxed-Acyclic</td>
<td>$O(n)$</td>
<td>NP-complete for $k, d = O(1)$</td>
</tr>
</tbody>
</table>
### How Hard is RF-Testing?

<table>
<thead>
<tr>
<th></th>
<th>RF-Testing</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC</td>
<td>NP-complete</td>
<td>P for $k, d = O(1)$</td>
</tr>
<tr>
<td>TSO</td>
<td>NP-complete</td>
<td>P for $k, d = O(1)$</td>
</tr>
<tr>
<td>PSO</td>
<td>NP-complete</td>
<td>P for $k, d = O(1)$</td>
</tr>
<tr>
<td>RA</td>
<td>$O(n \cdot k)$</td>
<td>NP-complete for $k, d = O(1)$</td>
</tr>
<tr>
<td>CC</td>
<td>$O(n \cdot k)$</td>
<td>NP-complete for $k, d = O(1)$</td>
</tr>
<tr>
<td>CCv</td>
<td>$O(n \cdot k)$</td>
<td>NP-complete for $k, d = O(1)$</td>
</tr>
<tr>
<td>CM</td>
<td>$O(n \cdot k)$</td>
<td>NP-complete for $k, d = O(1)$</td>
</tr>
<tr>
<td>Relaxed-Acyclic</td>
<td>$O(n)$</td>
<td>NP-complete for $k, d = O(1)$</td>
</tr>
</tbody>
</table>

Multi-copy atomicity?