A Consistency Checker for Memory Subsystem Traces

Matthew Naylor, Simon Moore, Alan Mujumdar

Email: *matthew.naylor@cl.cam.ac.uk*

Problem

Verify that the **memory subsystem** in a shared-memory multiprocessor implements a well-defined consistency model.

This is a prerequisite for the **correct execution** of concurrent programs on such architectures.

Our approach

Black-box specification-based testing:

1. Feed auto-generated requests to mem subsystem
2. Record a **trace** of all requests and responses
3. Check that trace satisfies consistency model

Attractions of black-box approach

Generic: can be applied to a wide range of implementations and coherence protocols.

Easy to apply: no modifications are required to the design under test.

Drawback of black-box approach

Checking traces is an **NP-complete** problem *[Gibbons and Korach, 1994]*.

Corollary: larger traces involving more cores are more likely to contain bugs yet less likely to be checkable in reasonable time.

State of the art

TSOtool *[Manovit, 2006]* is a conformance checker for the TSO consistency model.

It can handle **large traces**, on the order of millions of memory operations and hundreds of cores.

Achieved through powerful inference rules and careful algorithm design.

BUT...

Many modern memory subsystem implementations are **more relaxed than TSO**.

And TSOtool is a *“proprietary product of Sun Microsystems”*.

Example: Limitations of TSO

Thread 0	Thread 1
`*data := 1`	`*flag == 1`
`*flag := 1`	`*data == 0`

Forbidden under TSO, but observable if:

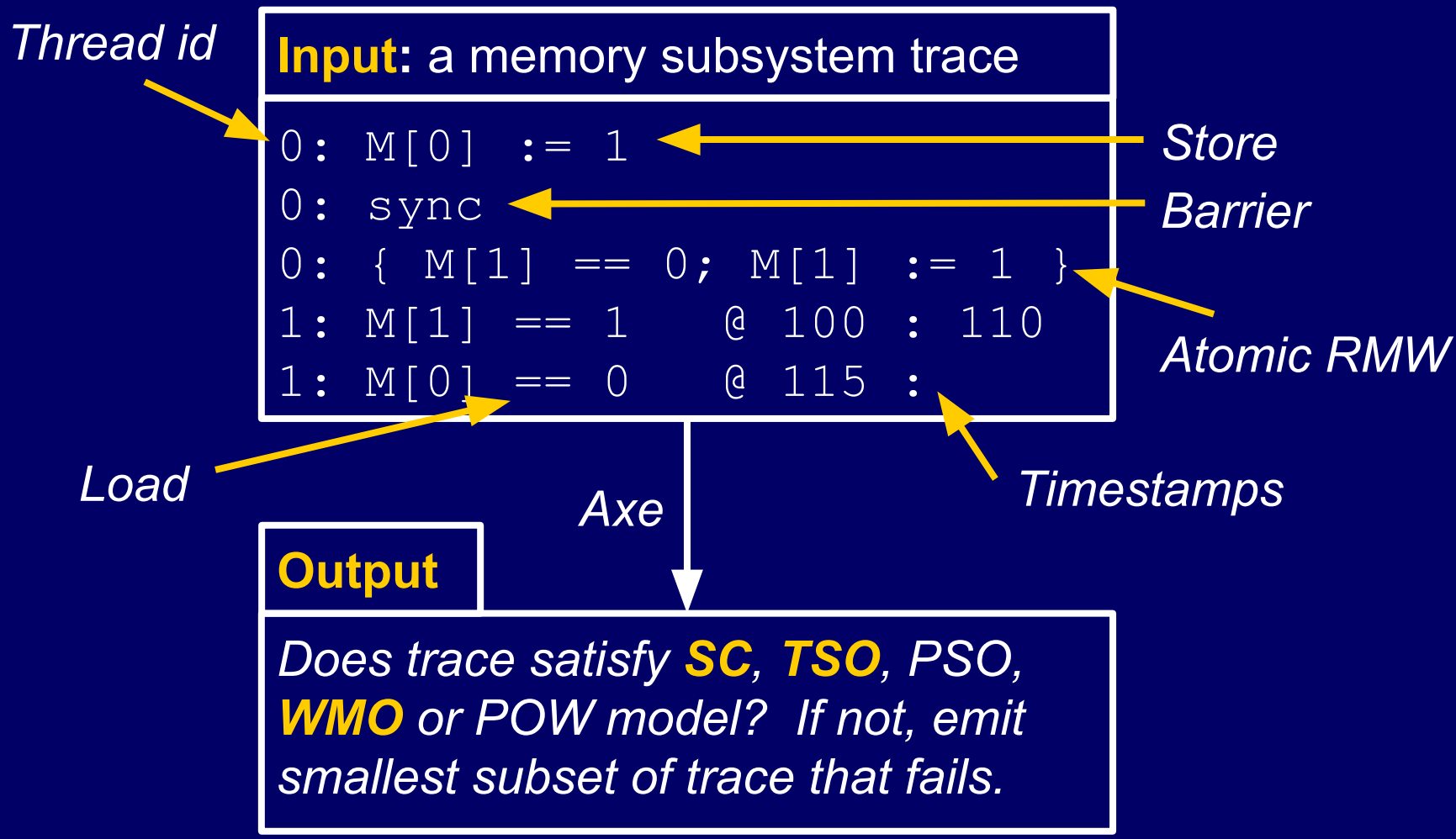
- L1 cache is **non-blocking**, e.g. Rocket Chip, where first load is a miss & second is a hit.
- Or, coherence protocol is **lazy**, e.g. BERI, where second load is a stale hit.

Our main contributions

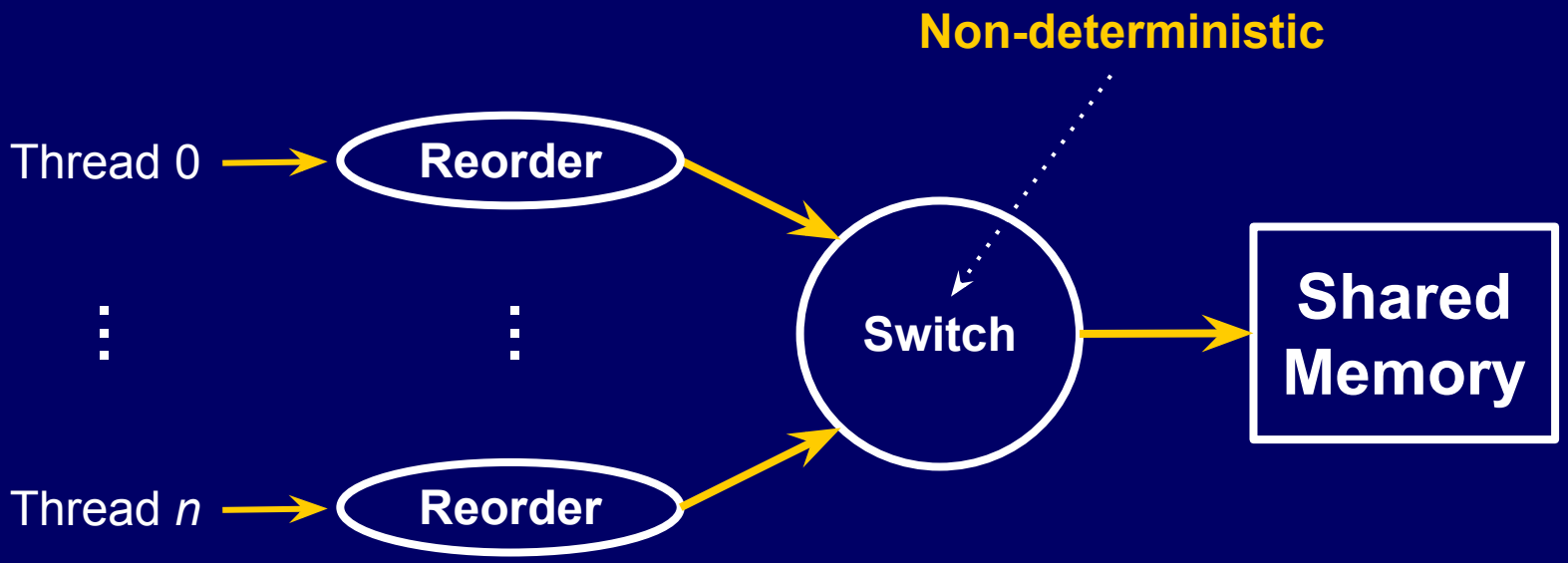
- **Generalisation** of TSOtool's algorithm to support a wider range of consistency models.
- An open-source checker for memory subsystem traces called **Axe**.
- **Experiences** of applying Axe to open-source SoCs BERI and Rocket Chip.

Part II: Axe Consistency Checker

What is Axe?


Thread id
Input: a memory subsystem trace
0: M[0] := 1
0: sync
0: { M[1] == 0; M[1] := 1 }
1: M[1] == 1 @ 100 : 110
1: M[0] == 0 @ 115 :
Store
Barrier
Atomic RMW
Load
Timestamps
Axe
Output
Does trace satisfy SC, TSO, PSO, WMO or POW model? If not, emit smallest subset of trace that fails.

SPARC models



- **SC** prohibits reordering.
- **TSO** can reorder S → L, simulating store buffer.
- **WMO** can additionally reorder S → S, L → L, and L → S (provided addresses differ).

Algorithm demo

Thread 0	Thread 1	Thread 2
`M[0] := 1`	`M[0] == 1`	`M[0] := 2`
	`M[0] == 2`	`M[0] := 3`

Add thread-local edges

Thread 0

```
M[0] := 1
```

Thread 1

```
M[0] == 1
```

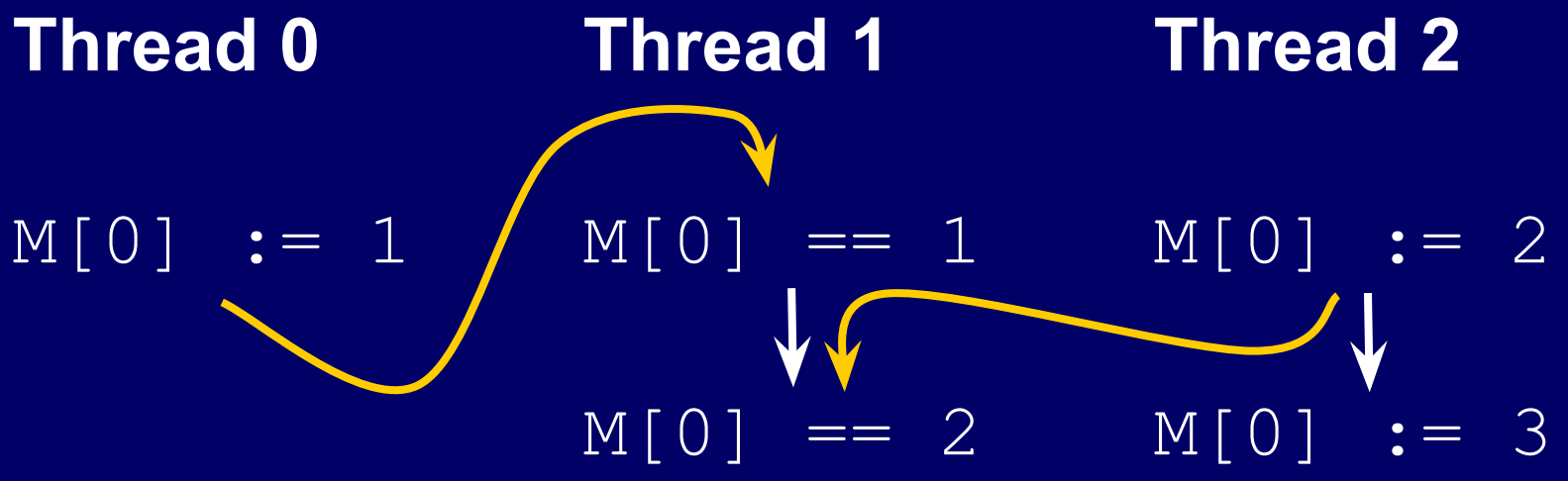
```
M[0] == 2
```

Thread 2

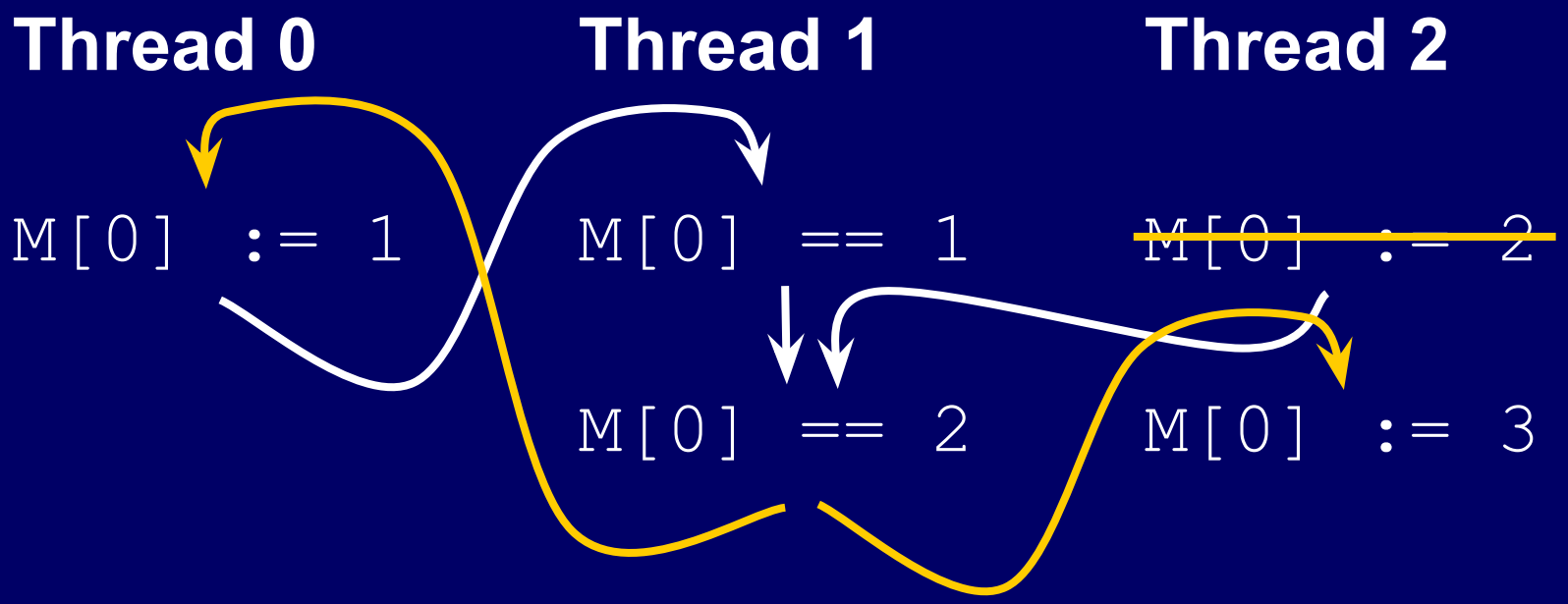
```
M[0] := 2
```

```
M[0] := 3
```

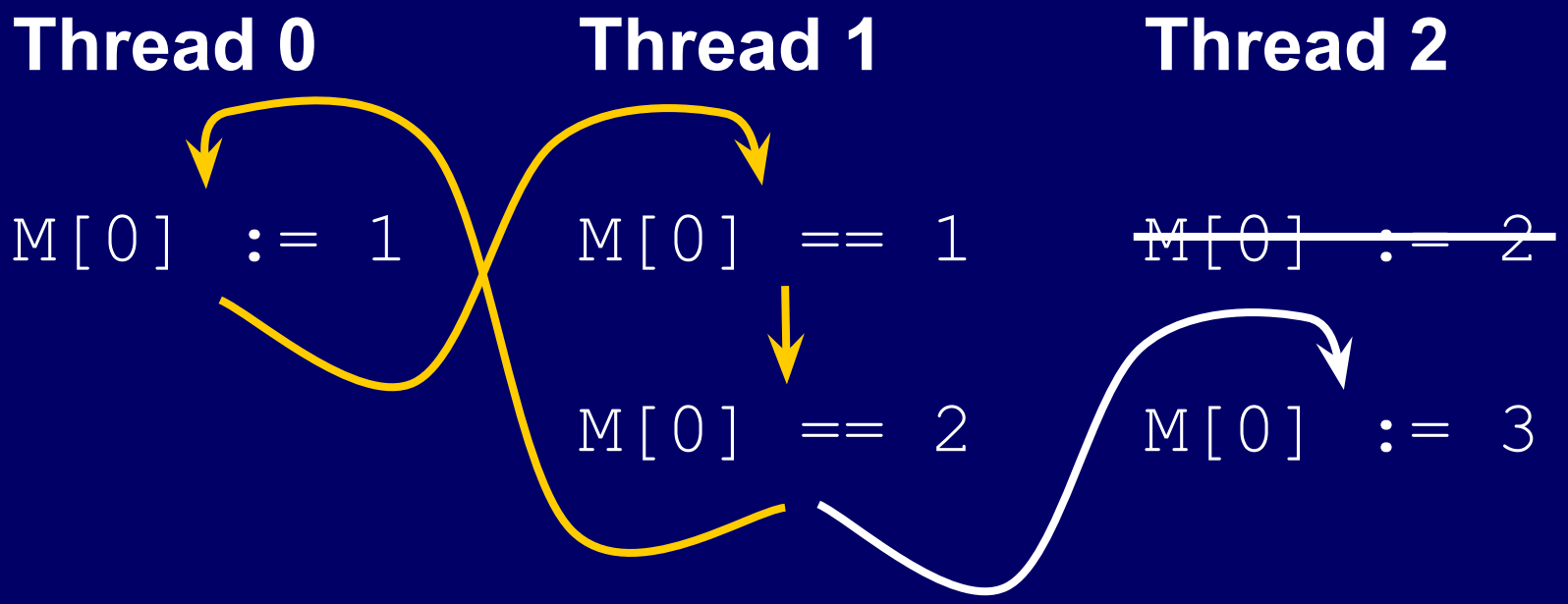
Add reads-from edges



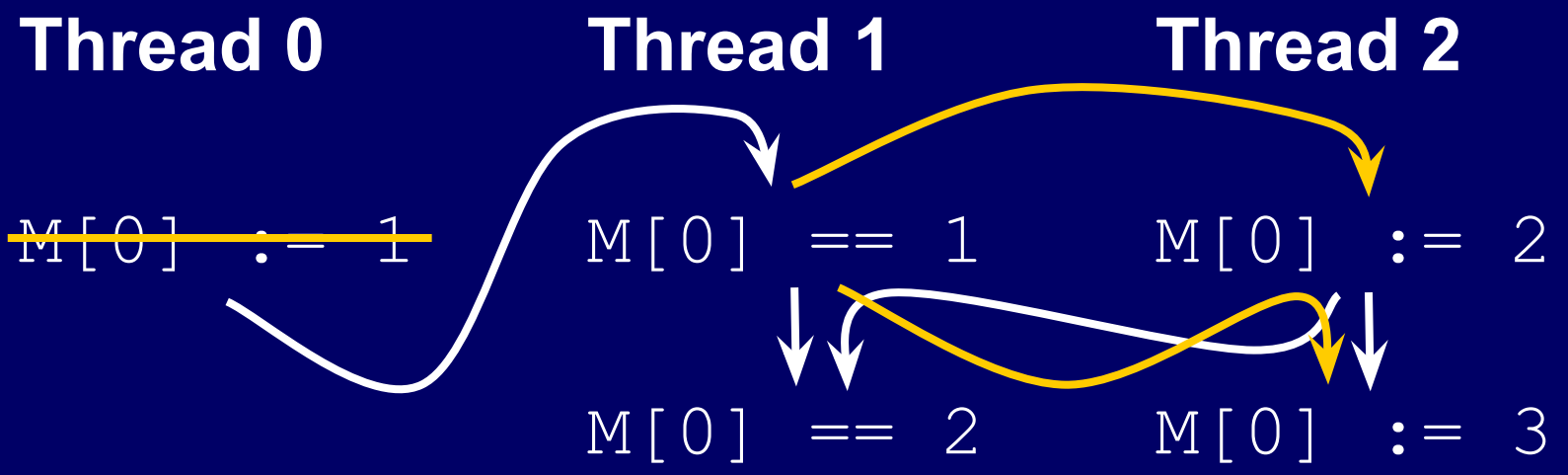
Delete a root, add reads-before edges



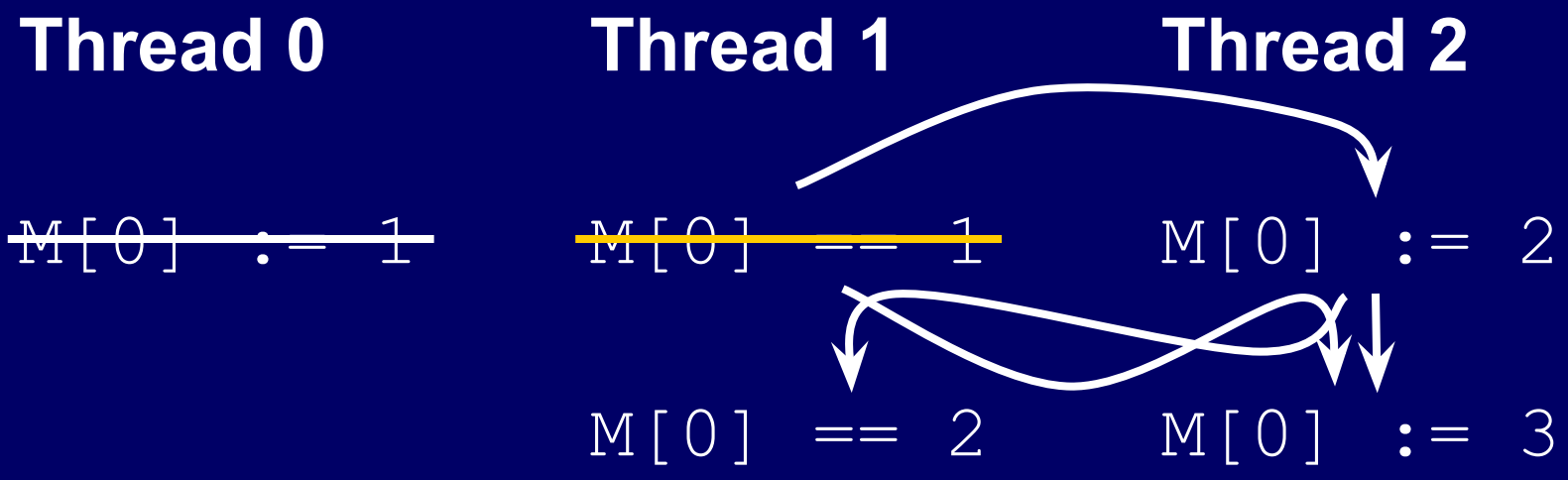
Violation: cycle detected!



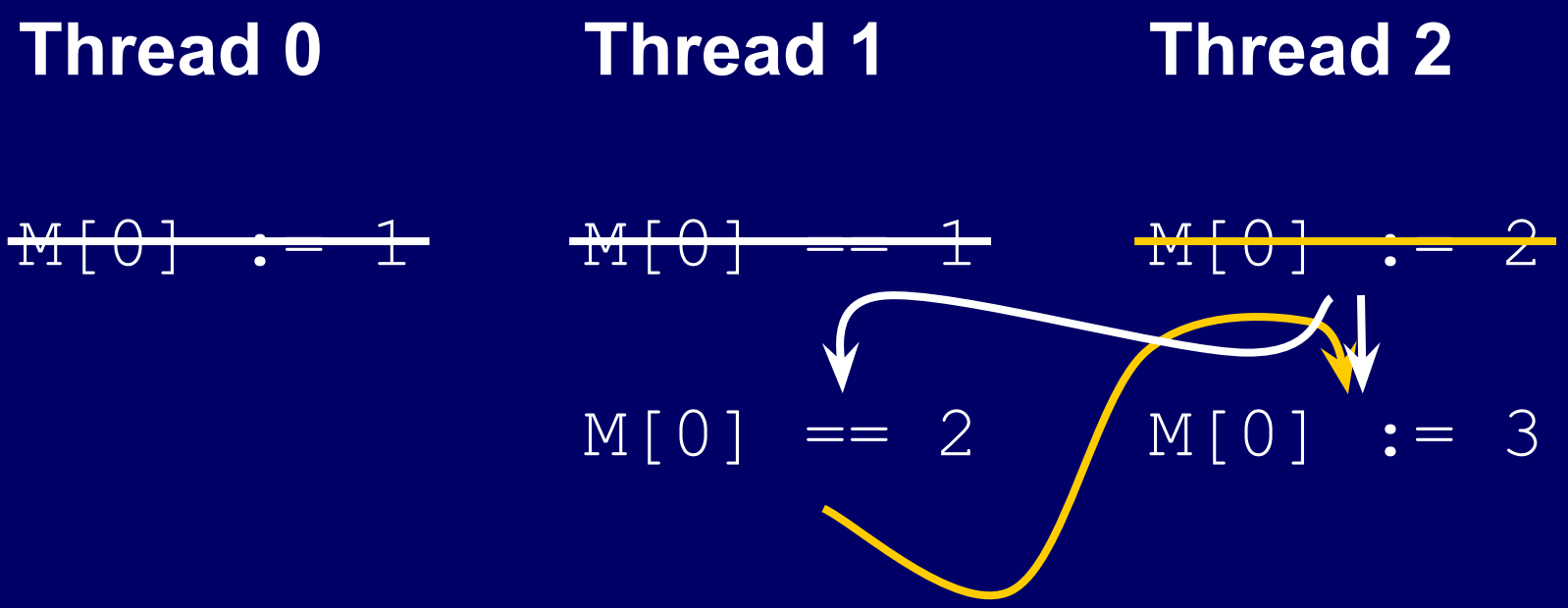
Backtrack, delete a root, add reads-before edges



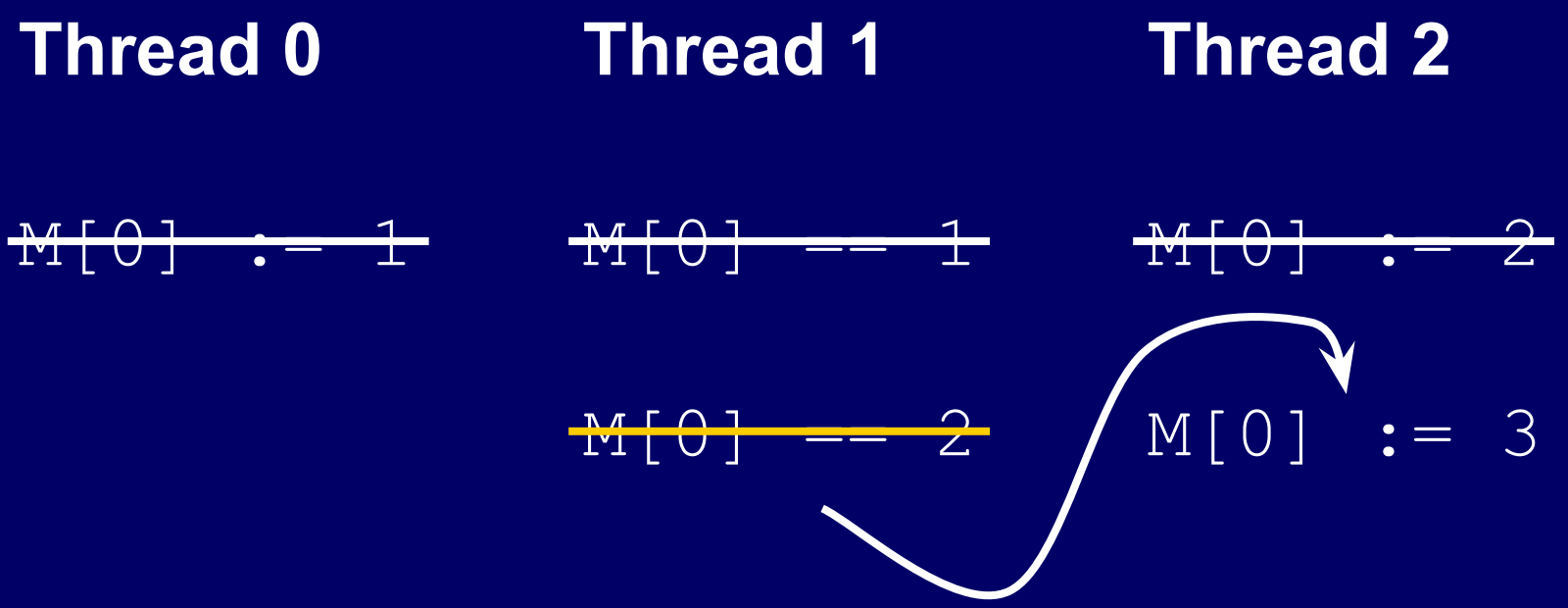
Delete a root



Delete root, add reads-before edges



Delete root



Delete root

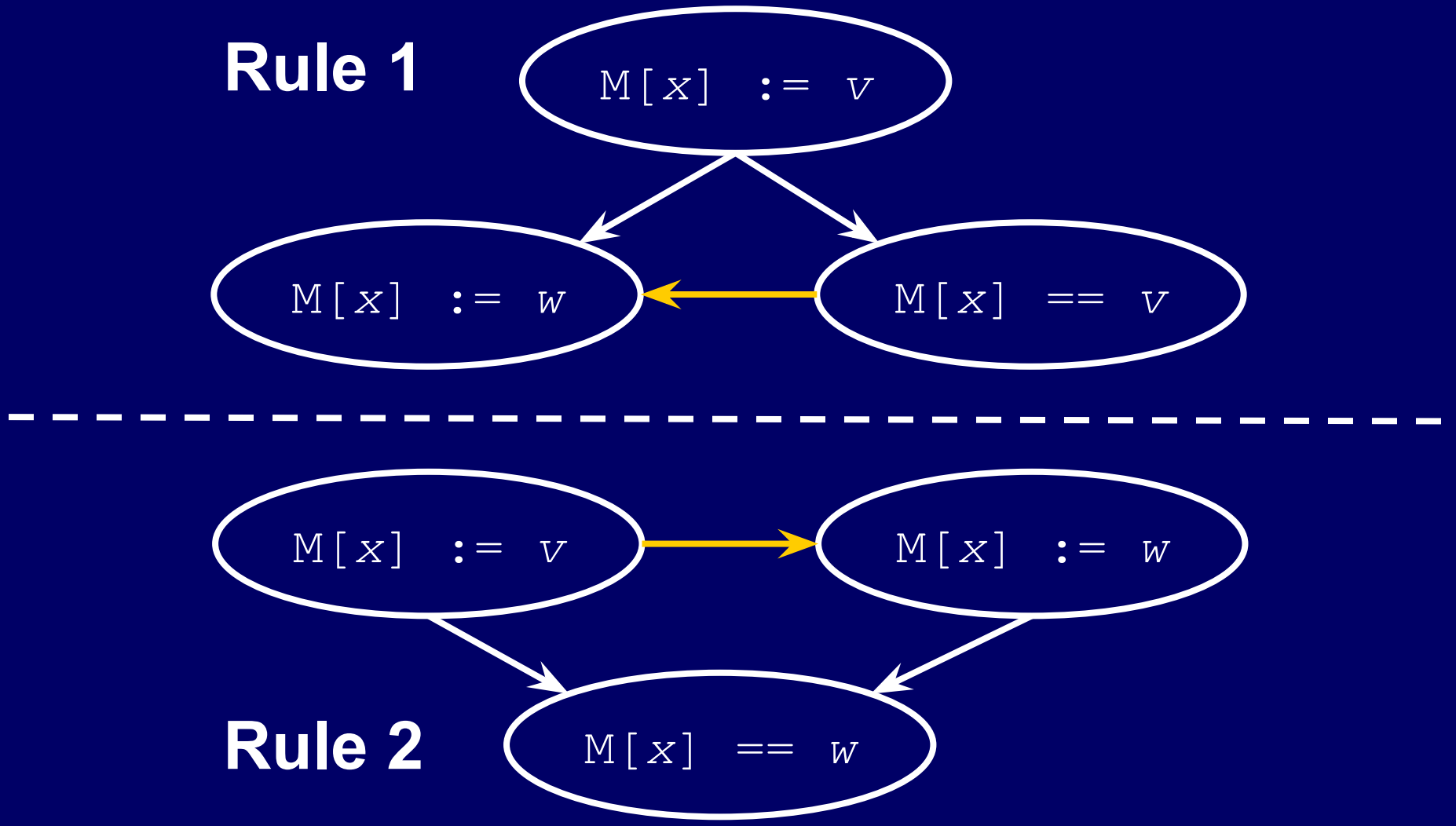
Thread 0	Thread 1	Thread 2
M[0] := 1	M[0] == 1	M[0] := 2
	M[0] == 2	M[0] := 3

Empty graph -- trace is valid!

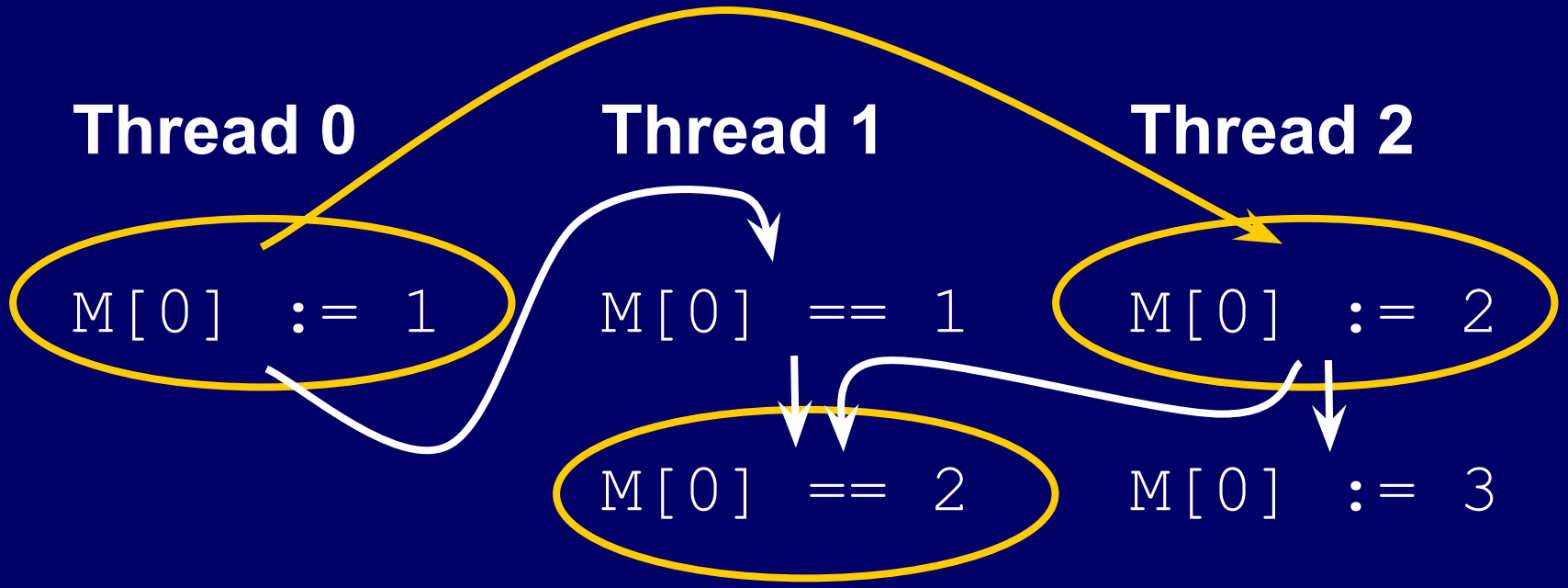
Lesson

- Easy to encounter **backtracking** behaviour during topological sort.
- Routine backtracking is catastrophic for checking even small traces.
- In response, TSOtool uses **inference rules**.

TSOtool's inference rules



Apply Rule 2



Picking a root is now deterministic

Efficient graph representation

- During checking, **adding an edge** to the graph is a very common operation.
- **Problem**: need to quickly determine whether any added edge introduces a cycle.
- Sounds like maintenance of an $O(N^3)$ transitive closure, disastrous for large N.

SC graph representation

- Under **SC**, operations on the same thread are **totally ordered**.
- For each node, we need only maintain the nearest successor on each thread.
- Complexity: ***O(N*T)***

TSO graph representation

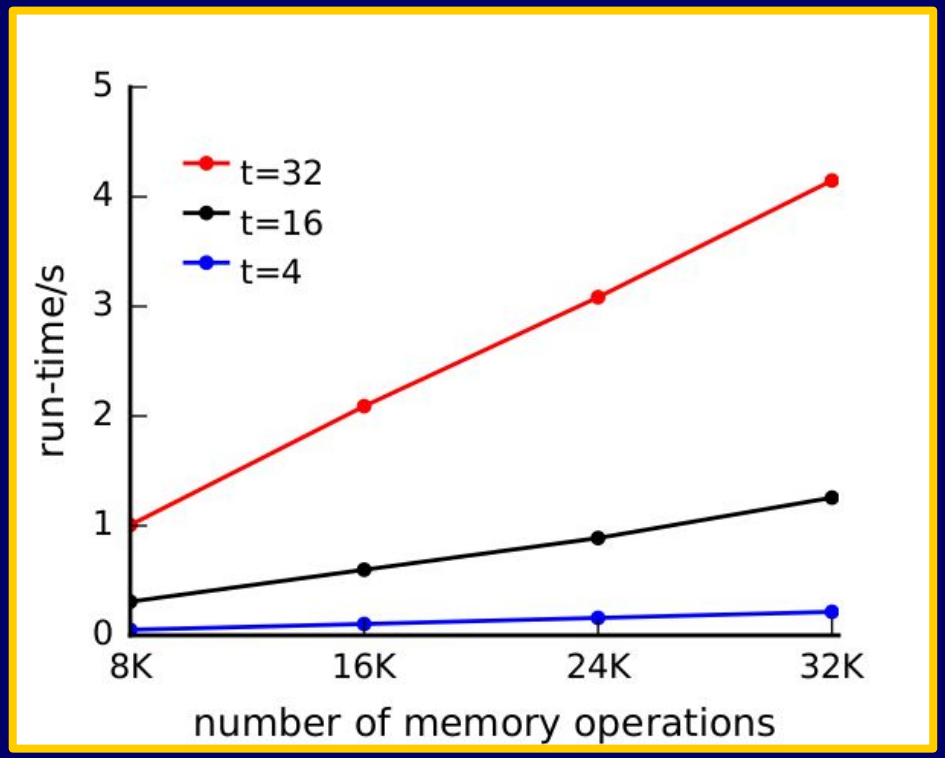
- Under **TSO**, loads on the same thread are **totally ordered**. Likewise for stores.
- For each node, we need only maintain the nearest load & store successor on each thread.
- Complexity: ***O(2*N*T)***

WMO graph representation

- Under **WMO**, loads from same address on same thread are **totally ordered**. Likewise for stores.
- For each node, maintain the nearest load & store successor on each thread for each address.
- Complexity: ***O(2*N*T*A)***
- Still much better than $O(N^3)$: *T* and *A* are small.

Axe performance evaluation (WMO)

Averaged over a range of traces (576 in total):



Checking time grows **linearly** with trace size.

Trace shrinking

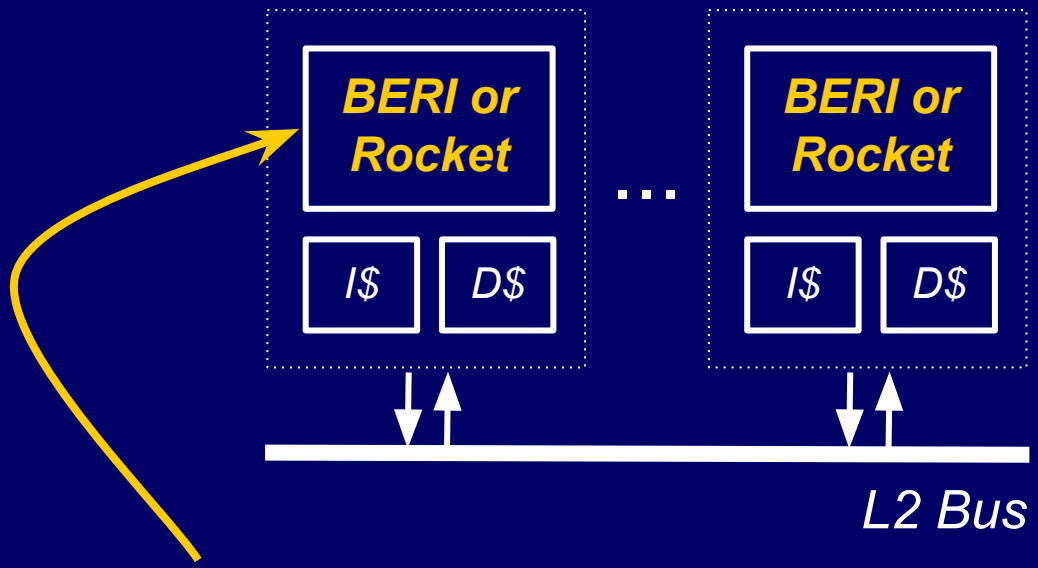
Problem: It's hard to determine **why** a large trace is invalid just by staring at it.

Solution: A trace shrinking procedure.

Given a trace that violates a model, it searches for the **smallest subset** of the trace that still violates the model.

Part III: Applications

Trace generation

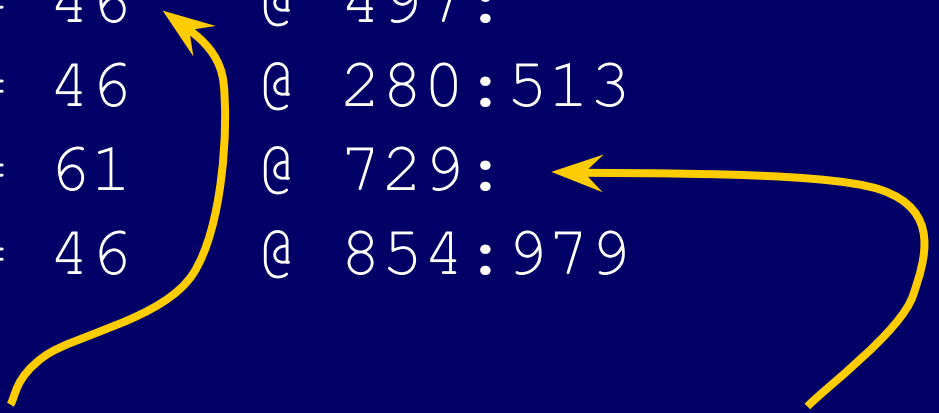


We replaced the core with a random traffic generator that logs all requests & responses, yielding a **random trace**.

Rocket Chip coherence bug

260-element counterexample, after shrinking:

```
0: M[2]  := 46    @ 497:
1: M[2]  == 46    @ 280:513
1: M[2]  := 61    @ 729:
1: M[2]  == 46    @ 854:979
```



Only write of 46 in trace

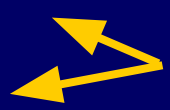
Write of 61 **dropped**

Identified as “race condition” by Rocket Chip devs.

Rocket Chip atomics bug

After shrinking:

```
1: M[3] := 31
0: { M[3] == 31; M[3] := 178 }
0: { M[3] == 178; M[3] := 198 }
1: { M[3] == 178; M[3] := 59 }
```



Not atomic

Bug occurs when a store-conditional is issued before a load-reserve response is received.

BERI barrier bug

After shrinking:

```
1: M[39028] := 76
1: M[39028] := 79        # Set data
1: sync
1: M[2761] := 83         # Set flag
0: M[2761] == 83         # See flag
0: sync
0: M[39028] == 76        # See stale data
```

This bug only observable after generating **cancelled** loads and stores in traffic generator.

Summary & conclusions

- We have **generalised** a state-of-the-art checker to a wider range of consistency models through our **open-source** tool Axe.
- This enabled us to test BERI & Rocket Chip, uncovering several serious bugs, **concisely reported** using our trace shrinking procedure.
- Time complexity now dependent on number of distinct addresses in trace, but **still performs well**.