Volition: Scalable and Precise Sequential Consistency Violation Detection

Xuehai Qian, Benjamin Sahelices
Josep Torrellas, Depei Qian
University of Illinois

University of Illinois

http://iacoma.cs.uiuc.edu/

i-acoma group

Sequential Consistency (SC)

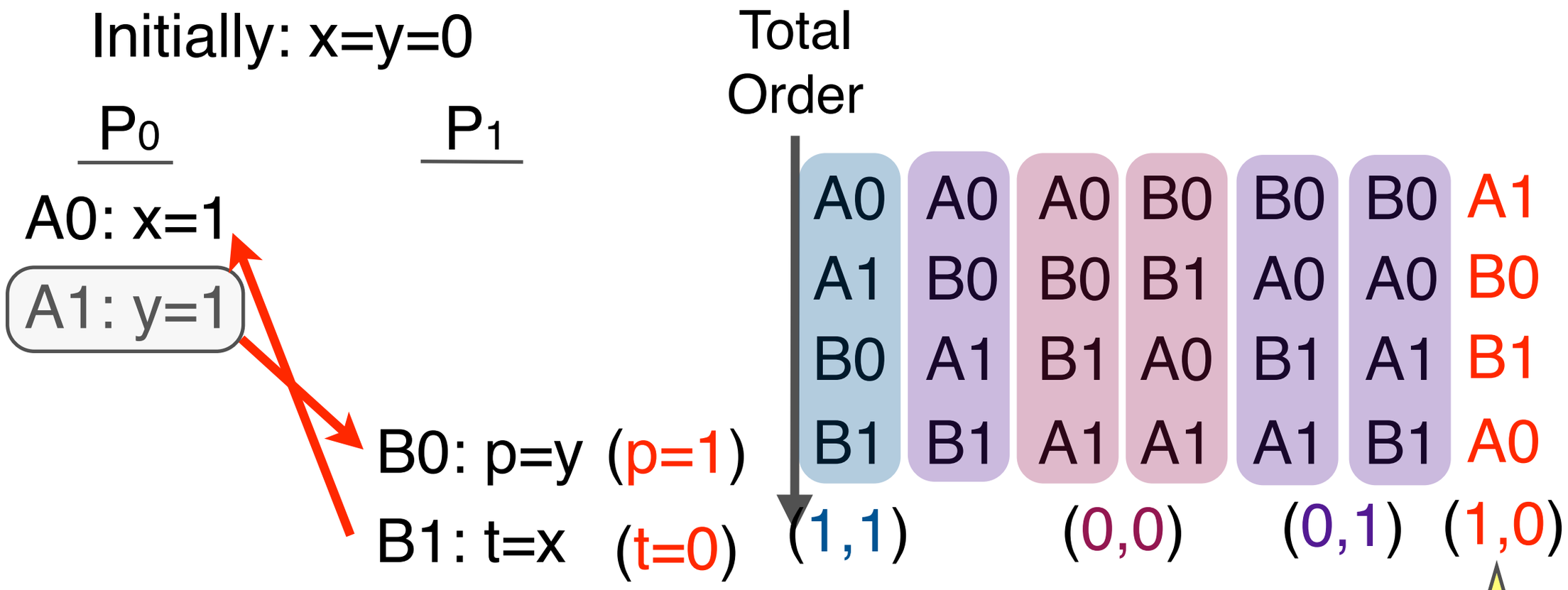
P_0	P_1
A0: x=1	B0: p=y
A1: y=1	B1: t=x

Total Order

A0	A0	A0	B0	B0	B0
A1	B0	B0	B1	A0	A0
B0	A1	B1	A0	B1	A1
B1	B1	A1	A1	A1	B1

- In SC, memory accesses:
  - Appear atomic
  - Have a total global order
  - For each thread, follow program order

Sequential Consistency Violation(SCV)



- SCV: accesses reorder that does not conform to SC
- Machines support relaxed memory models, not SC
- Machines may induce SC violations (SCV)

SCV can cause very unintuitive results (bugs)

Why Detecting SCVs is Important?

- Programmers assume SC
  - SCV is almost always a bug: unexpected interleaving
  - Single-stepping debuggers cannot reproduce the bug
- Causes portability problems (e.g Intel TBB)
  - Code may not work across machines
- Lock-free data structures sometimes explicitly use races but rely on SC
  - Traditional data race detectors won't work

Contribution

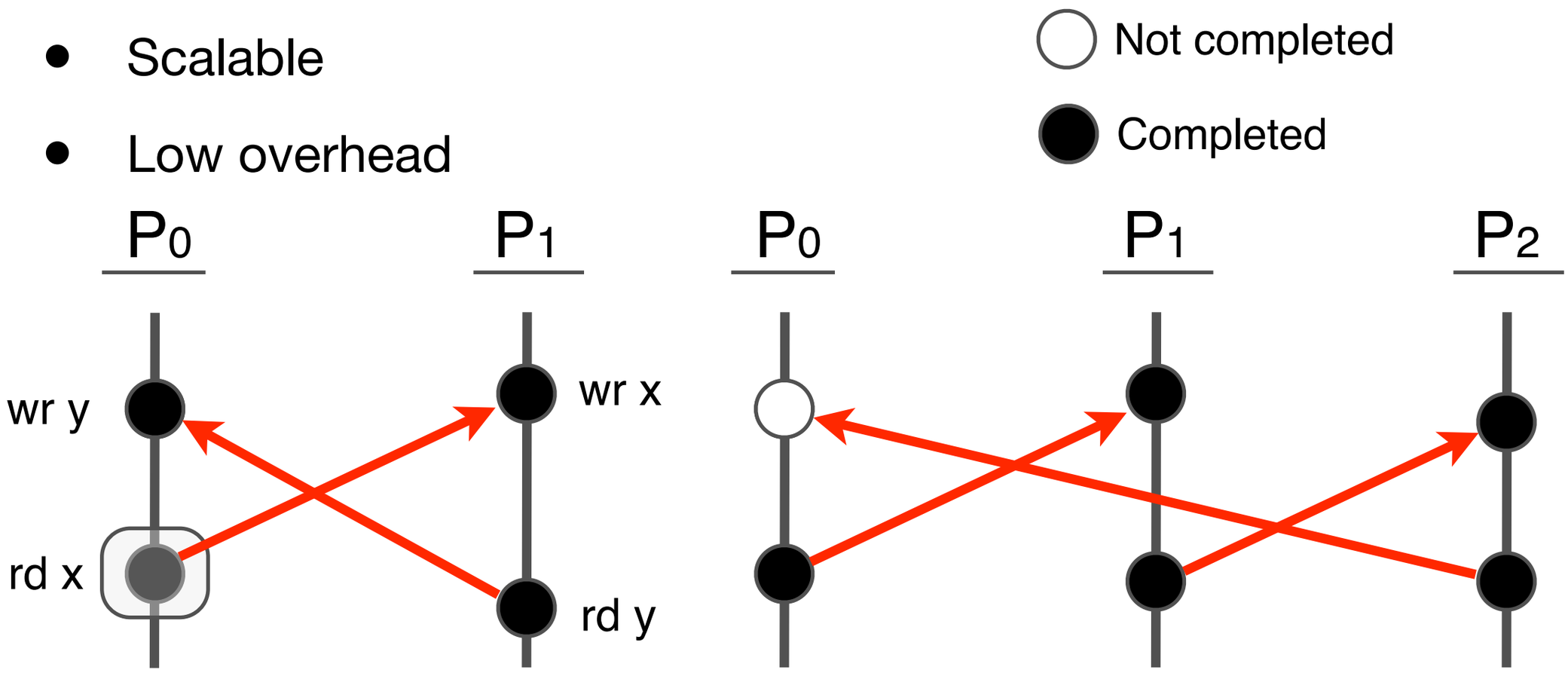
- Volition: first hardware scheme that detects SCVs in a relaxed-consistency machine, such that
  - It is precise
  - It is scalable
  - Works for an arbitrary number of participating processors
- Leverages coherence protocol transactions to detect cycles in memory access orders across threads
- Incurs negligible overhead and is decoupled from the cache coherence protocol

Outline

- Motivation
- Volition Mechanisms
- Evaluation

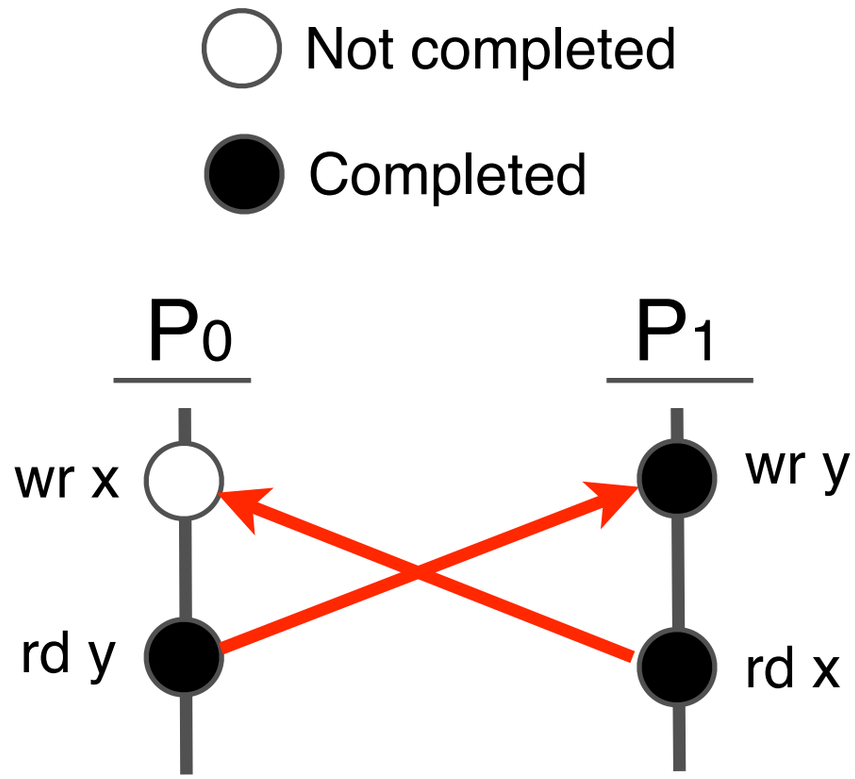
Understanding SCV

- Two or more data races overlap and form a cycle
- Problem: detecting such cycles without affecting the execution timing
- Design Goals:
  - Precise
  - Scalable
  - Low overhead



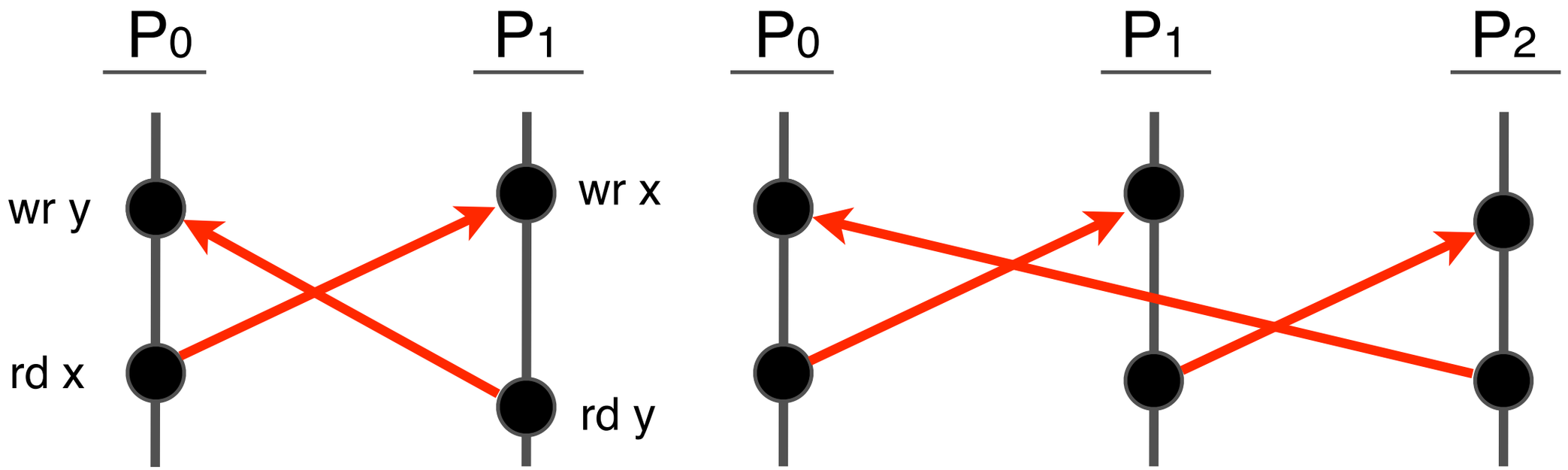
Definition: Active Accesses

- Active Accesses are the accesses that may participate in SCV
- Active Access:
  - Either itself or older local access is not completed OR
  - It is the destination of a dependence from an active access
- Volition provides metadata hardware:
  - For all accesses: Sequence Number (SN)
  - For active accesses: current state (address, completion status, ..)



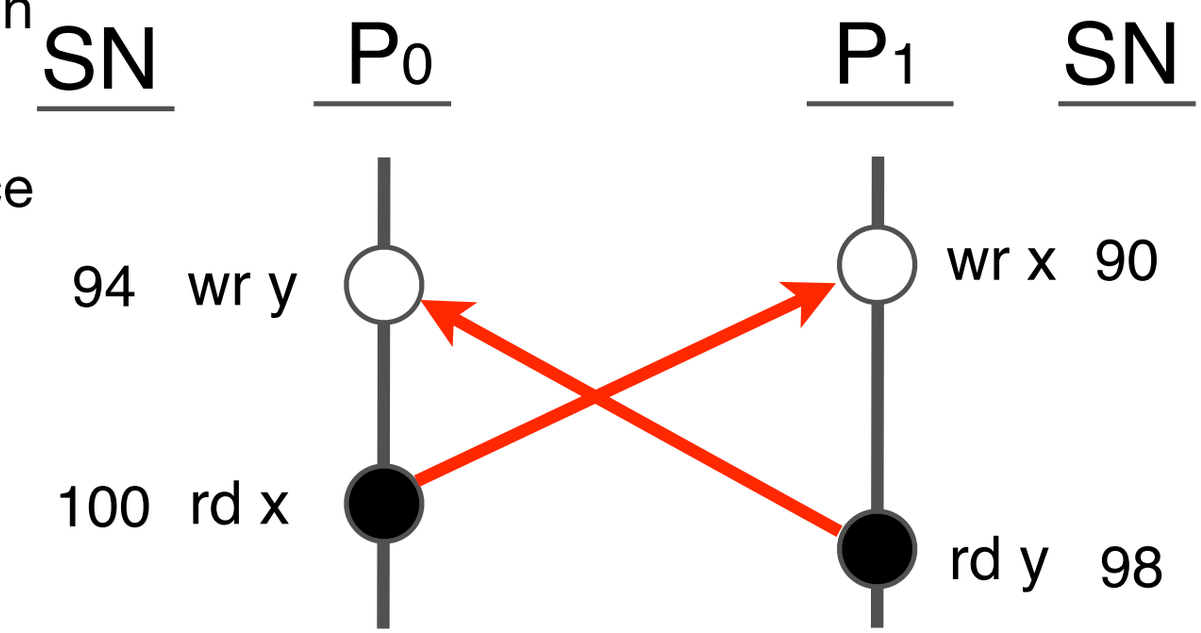
Definition: Active Data Race (AR)

- Data race where the source (and therefore destination) are active
- SCVs are composed of two or more ARs forming a cycle
- Volition detects ARs and cycles in hardware dynamically



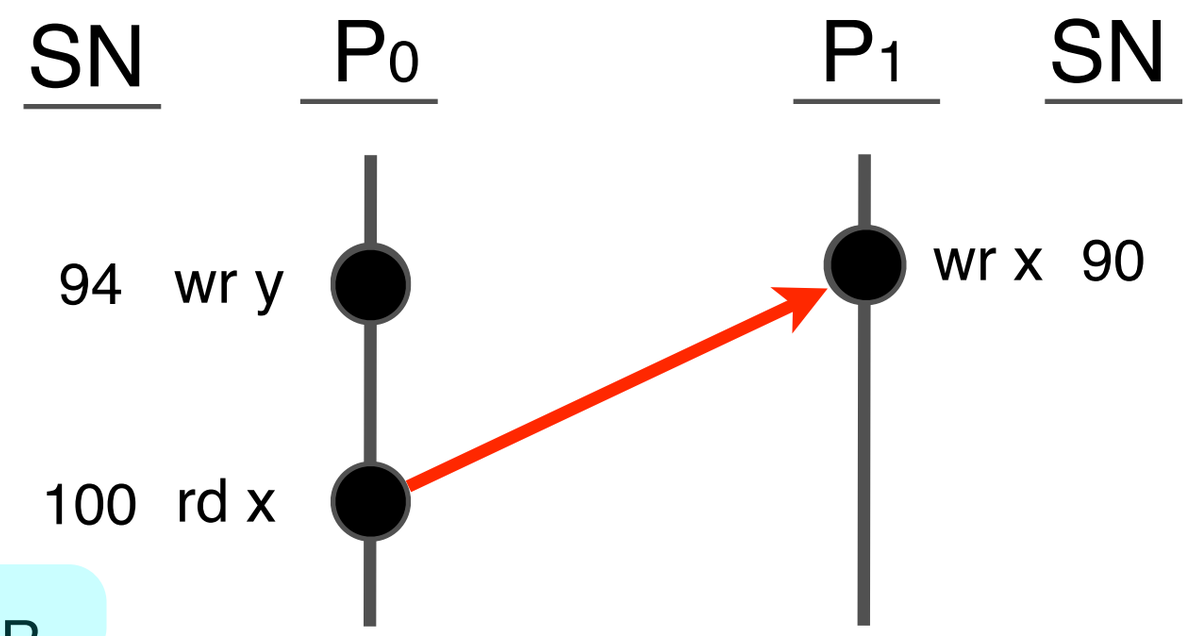
How Does Volition Detect an SCV?

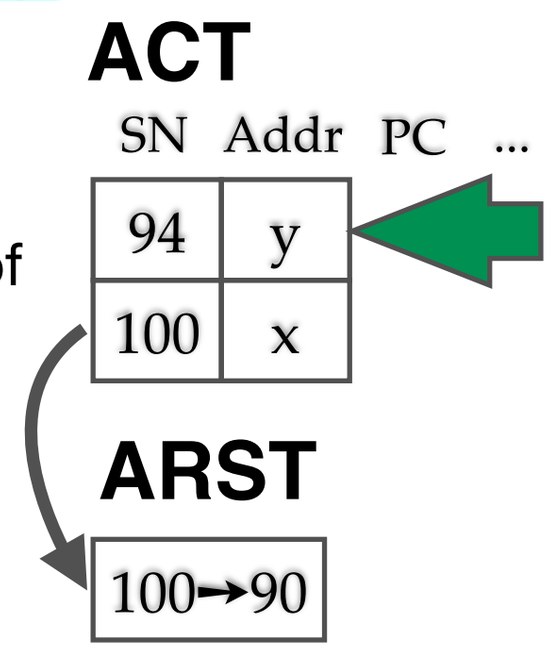
- All active accesses in a processor: recorded in an ACtive Table (ACT)
- If active access is the source of a dependence
  - Record AR in AR Source Table (ARST)
  - Respond with a bit in the coherence response
- Destination:
  - Record the AR in the AR Destination Table (ARDT)
- If source becomes inactive
  - Remove local ARST
  - Send Expire msg to destination, remove ARDT
- SCV: when a processor finds two ARs forming a cycle (i.e. **SNsrc > SNdst**)

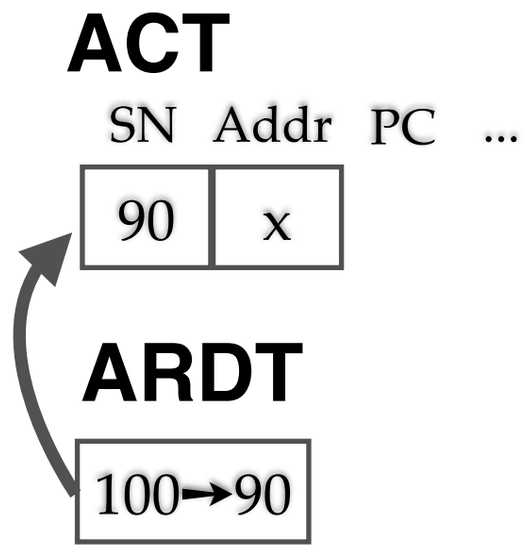


Active Table (ACT)

- FIFO table of all active accesses
- When to insert an ACT entry?
  - Access is issued (program order)
- When to delete an ACT entry?
  - Access is at the head of ACT and completed, AND
  - Access is not the destination of an AR
  - HW keeps deleting entries until the conditions don't hold
- After deleting an entry: if it is the source of an AR
  - Remove the related entry in ARST
  - Send Expire msg
- Remove ARDT entry when expire is received

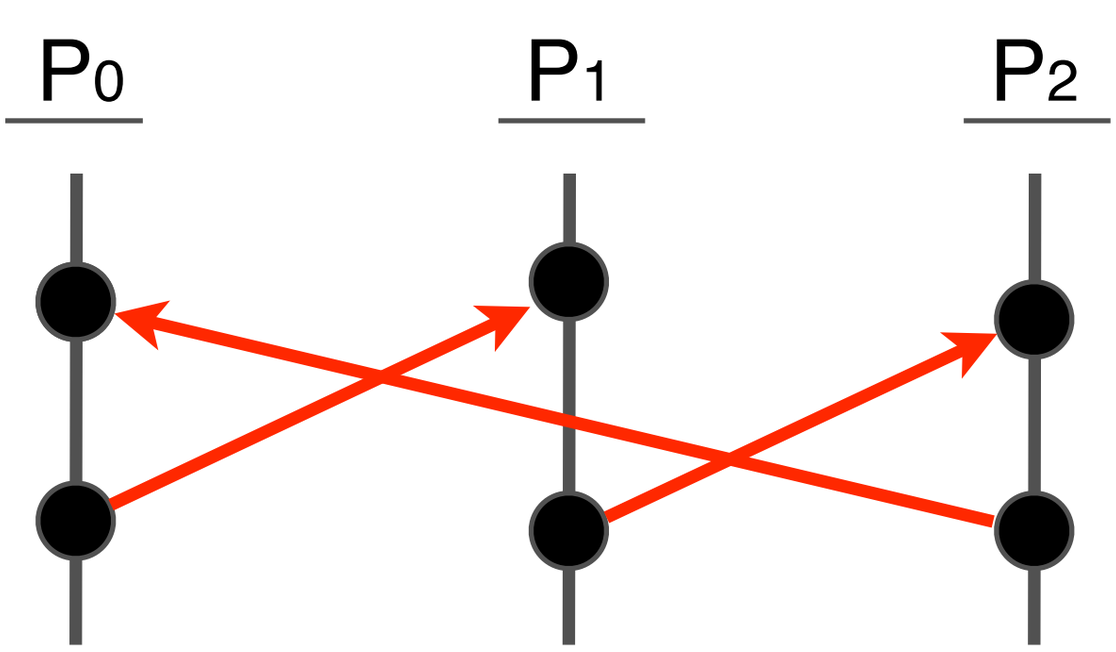


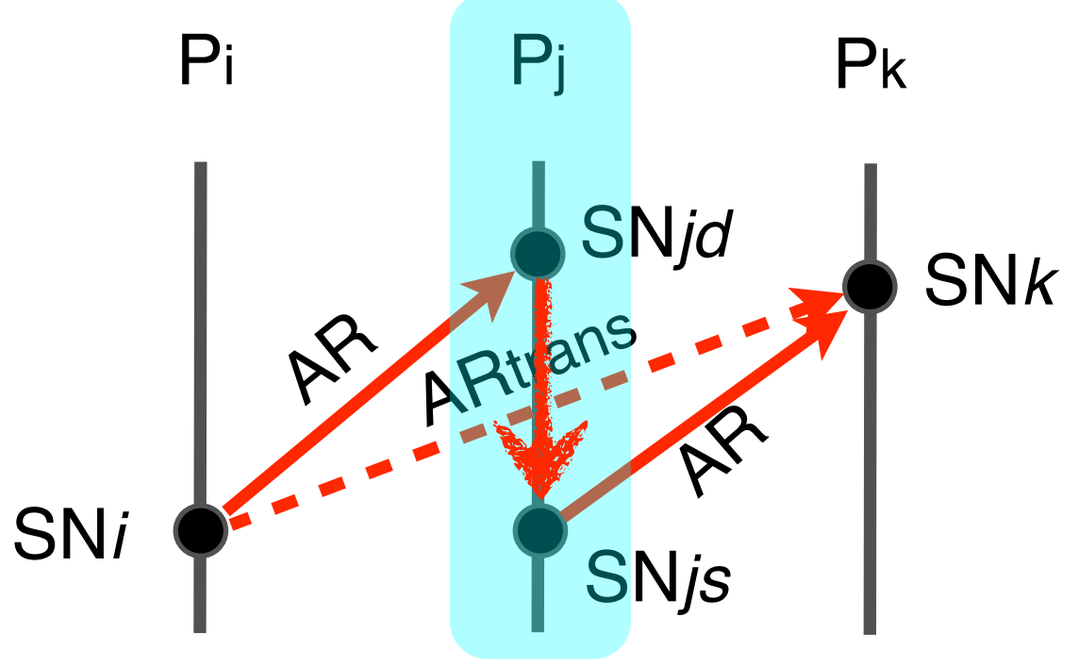




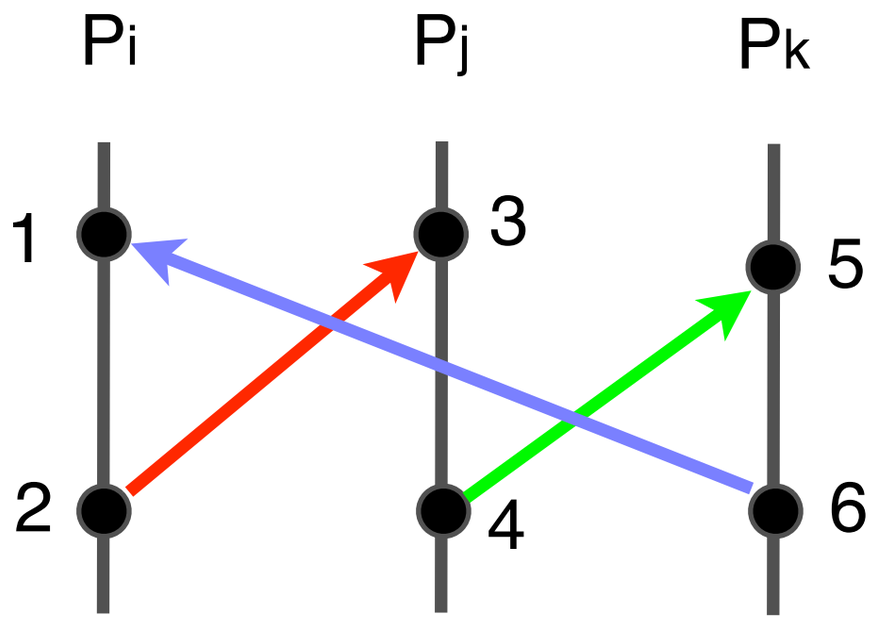
Finding Cycles with any # of Processors

- AR Propagation: Volition creates a transitive dependence out of two dependences
  - ARtrans: from the src of one AR to the dst of the other AR
- Detected by the processor in the middle
  - Only if program order connects these two ARs

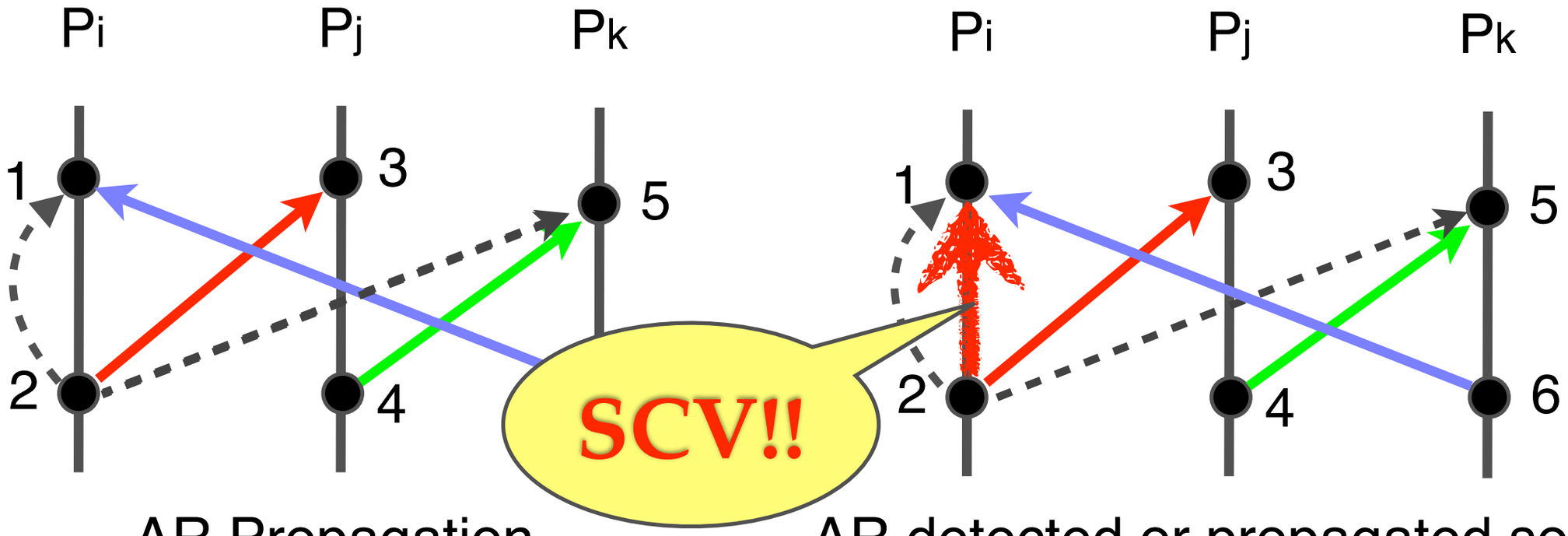




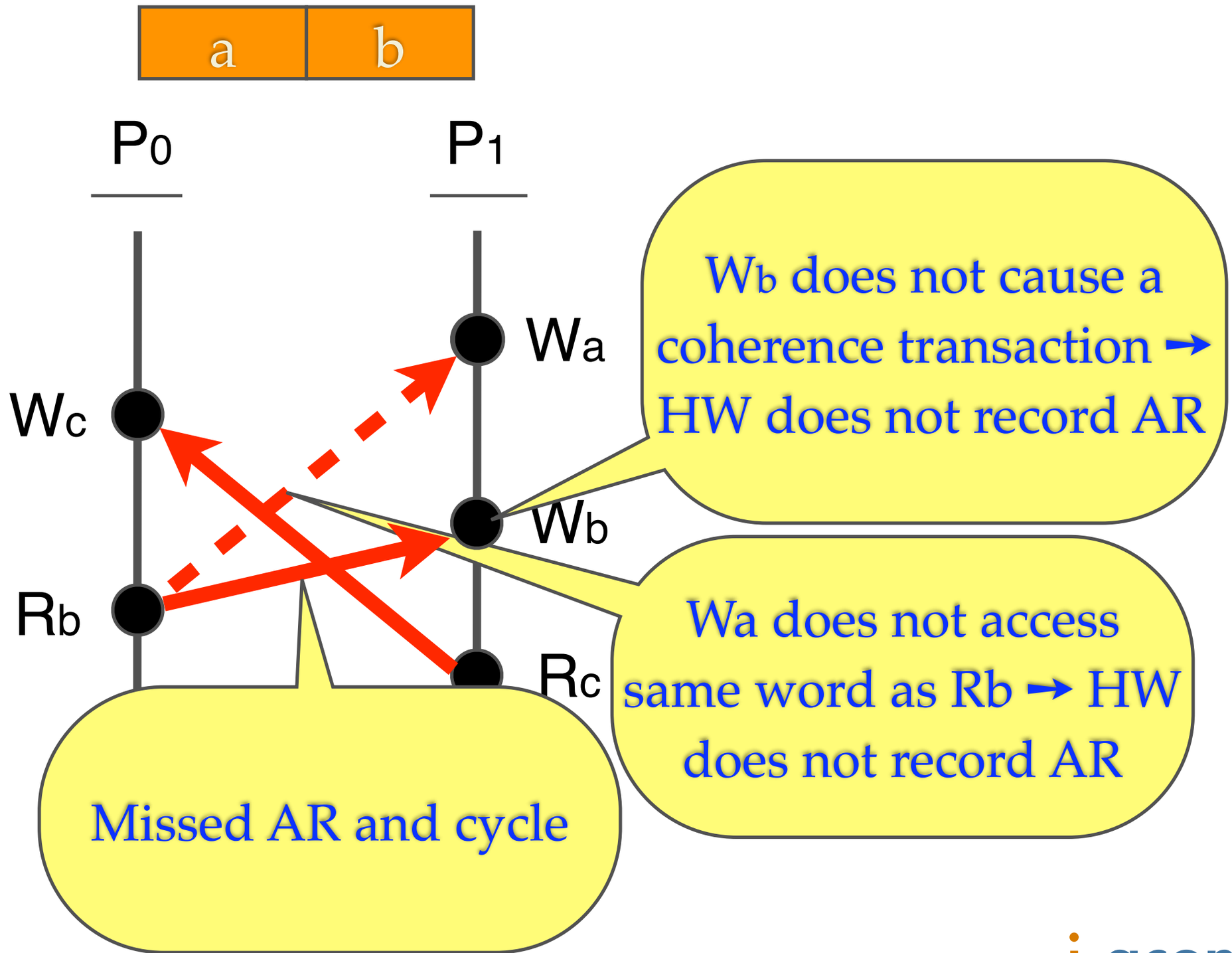
SCV Detection with Arbitrary # of Processors



- In each processor: Volition HW keeps generating transitive dependences
- When a processor detects a **local dependence from younger to older access** → SCV



Multi-word Cache Line Issue



Support for Multi-word Cache Line

- For lines that are being actively shared
  - Augment line with Summary of Active Information (SAI)
    - Contains current information on system-wide active accesses to any of the words
    - Indicates: when future access, do we need to record an AR?

  - When a word is accessed and AR-recording is necessary:
    - If a cache coherence transaction occurs ➙ reuse the transaction to record the AR
    - Else ➙ create a “metadata access” transaction to record the AR
  - SAI info cleared when an access becomes inactive

Summary of Evaluation

- Volition is able to detect SCVs in codes in which we removed fences
- Support for multi-word cache line metadata is needed to avoid missing SCVs
- Using data races as proxies for SCVs is insufficient

Conclusion

- Volition is the first scalable and precise SCV detector
  - Works with directory protocol
  - Can detect SCVs involving an arbitrary number of processors
  - Incurs negligible execution time and traffic overhead
  - Needs modest-sized hardware structures

Volition: Scalable and Precise Sequential Consistency Violation Detection

Xuehai Qian, Benjamin Sahelices
Josep Torrellas, Depei Qian
University of Illinois

University of Illinois

http://iacoma.cs.uiuc.edu/

i-acoma group