

Cache Coherence (Continued)

Miguel Gomez-Garica

6.823 Fall 2022

Adapted from prior course offerings

Cache Coherence

» Two *necessary* conditions:

1. Write propagation: Writes **eventually** become visible to other processors
2. Write serialization: All processors observe writes to one location appear to happen in a consistent order

» MSI protocol provides a *sufficient* condition via single-writer multi-reader policy

- Only one cache may have write permission at any given point in time
- Multiple caches can have read-only permission at a given point in time

Write-back caches: MSI

» Three stable states per cache-line

- Invalid (I): Cache does not have a copy
- Shared (S): Cache has read-only copy; clean
- Modified (M): Cache has only copy; writable; (potentially) dirty

» Processor-initiated actions:

- Read: needs to upgrade permission to S
- Write: needs to upgrade permission to M
- Evict: relinquish permissions (caused by access to a different cache line)

MSI directory states

- » Uncached (Un): No cache has a valid copy
- » Shared (Sh): One or more caches in S state. Must track sharers.
- » Exclusive (Ex): One of the caches in M state. Must track owner.
- » Does the directory need transient states?
 - Yes on downgrades/invalidations, to guarantee serialization

MSI Protocol: Caches (1/3)

Transitions initiated by processor accesses:

M

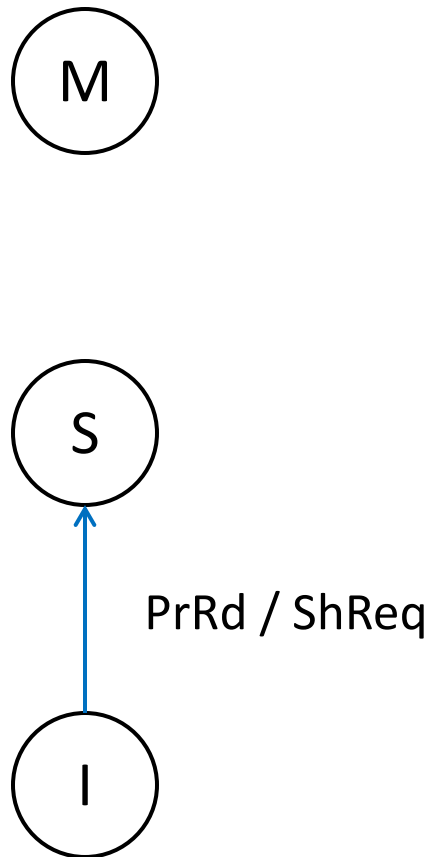
S

I

Actions
Processor Read (PrRd)
Processor Write (PrWr)
Shared Request (ShReq)
Exclusive Request (ExReq)

MSI Protocol: Caches (1/3)

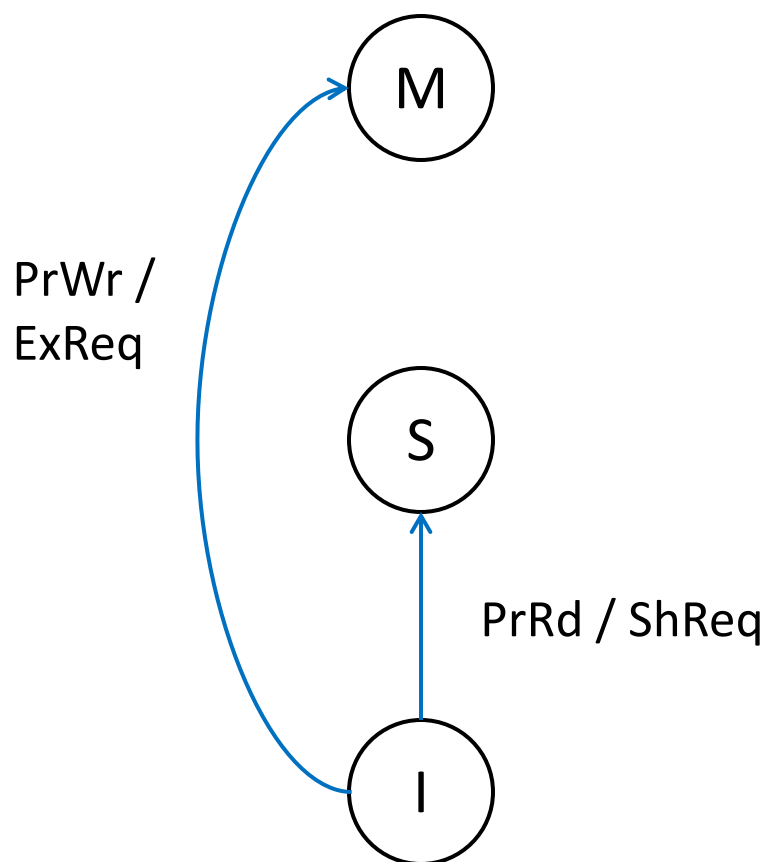
Transitions initiated by processor accesses:



Actions
Processor Read (PrRd)
Processor Write (PrWr)
Shared Request (ShReq)
Exclusive Request (ExReq)

MSI Protocol: Caches (1/3)

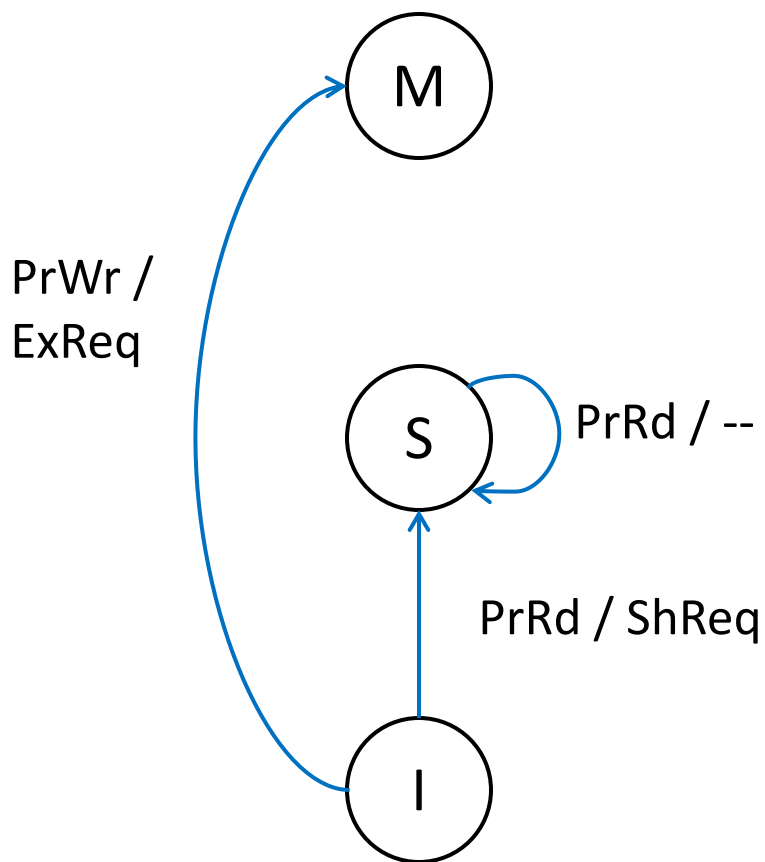
Transitions initiated by processor accesses:



Actions
Processor Read (PrRd)
Processor Write (PrWr)
Shared Request (ShReq)
Exclusive Request (ExReq)

MSI Protocol: Caches (1/3)

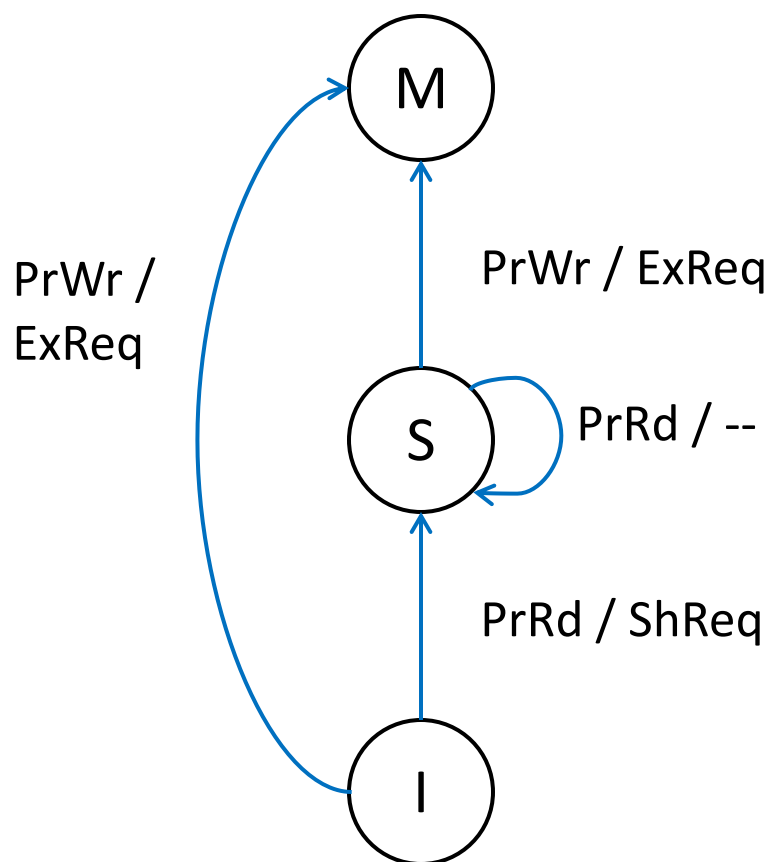
Transitions initiated by processor accesses:



Actions
Processor Read (PrRd)
Processor Write (PrWr)
Shared Request (ShReq)
Exclusive Request (ExReq)

MSI Protocol: Caches (1/3)

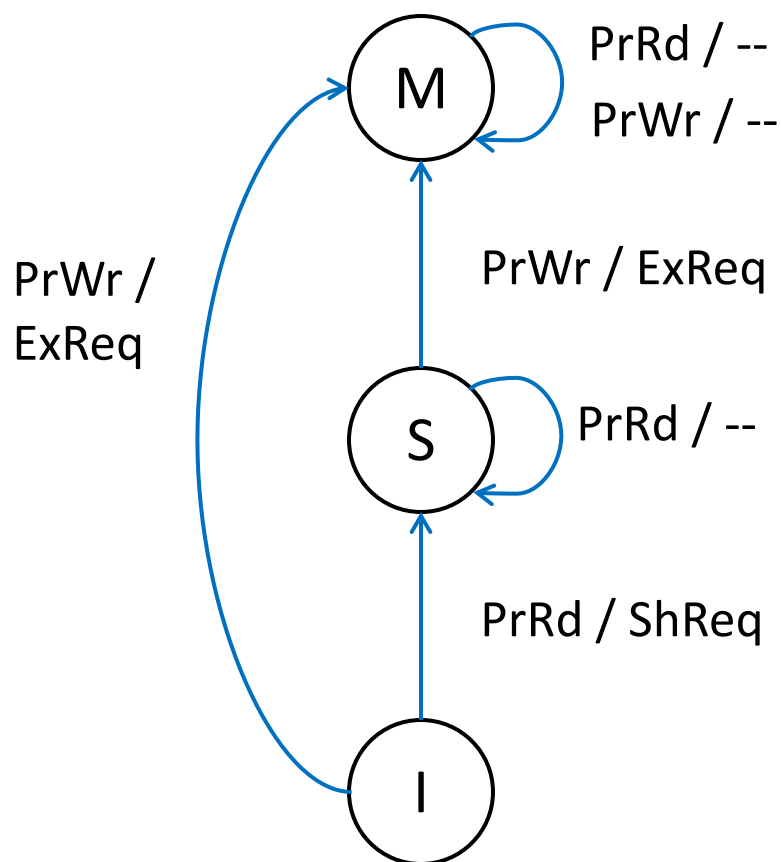
Transitions initiated by processor accesses:



Actions
Processor Read (PrRd)
Processor Write (PrWr)
Shared Request (ShReq)
Exclusive Request (ExReq)

MSI Protocol: Caches (1/3)

Transitions initiated by processor accesses:



Actions
Processor Read (PrRd)
Processor Write (PrWr)
Shared Request (ShReq)
Exclusive Request (ExReq)

MSI Protocol: Caches (2/3)

Transitions initiated by directory requests:

M

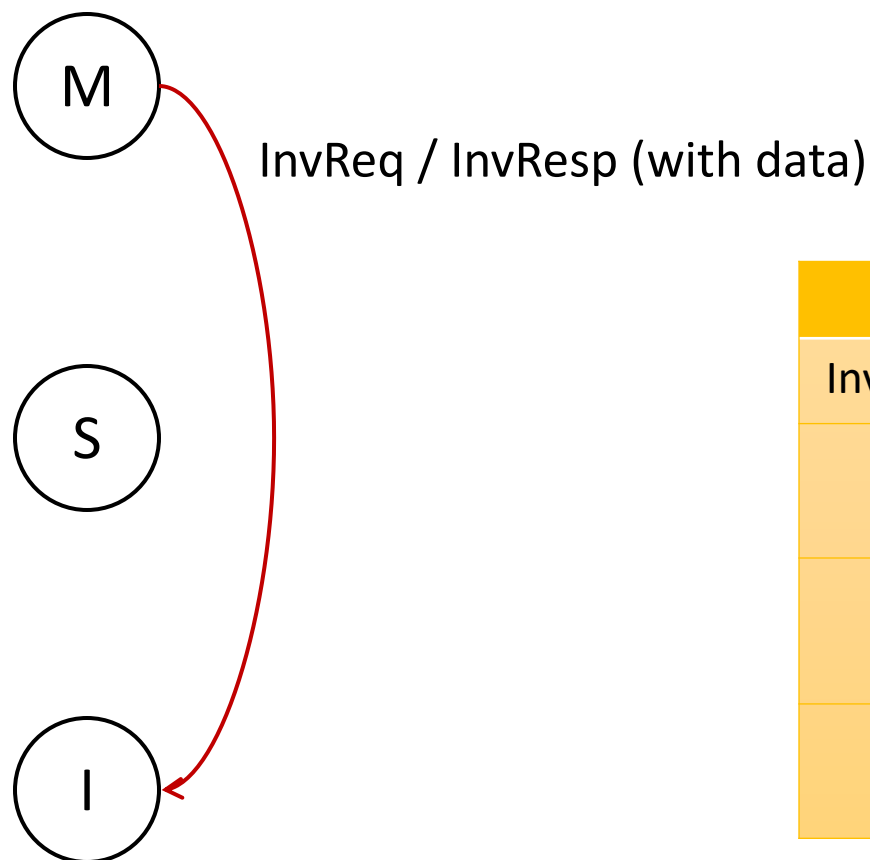
S

I

Actions
Invalidation Request (InvReq)
Downgrade Request (DownReq)
Invalidation Response (InvResp)
Downgrade Response (DownResp)

MSI Protocol: Caches (2/3)

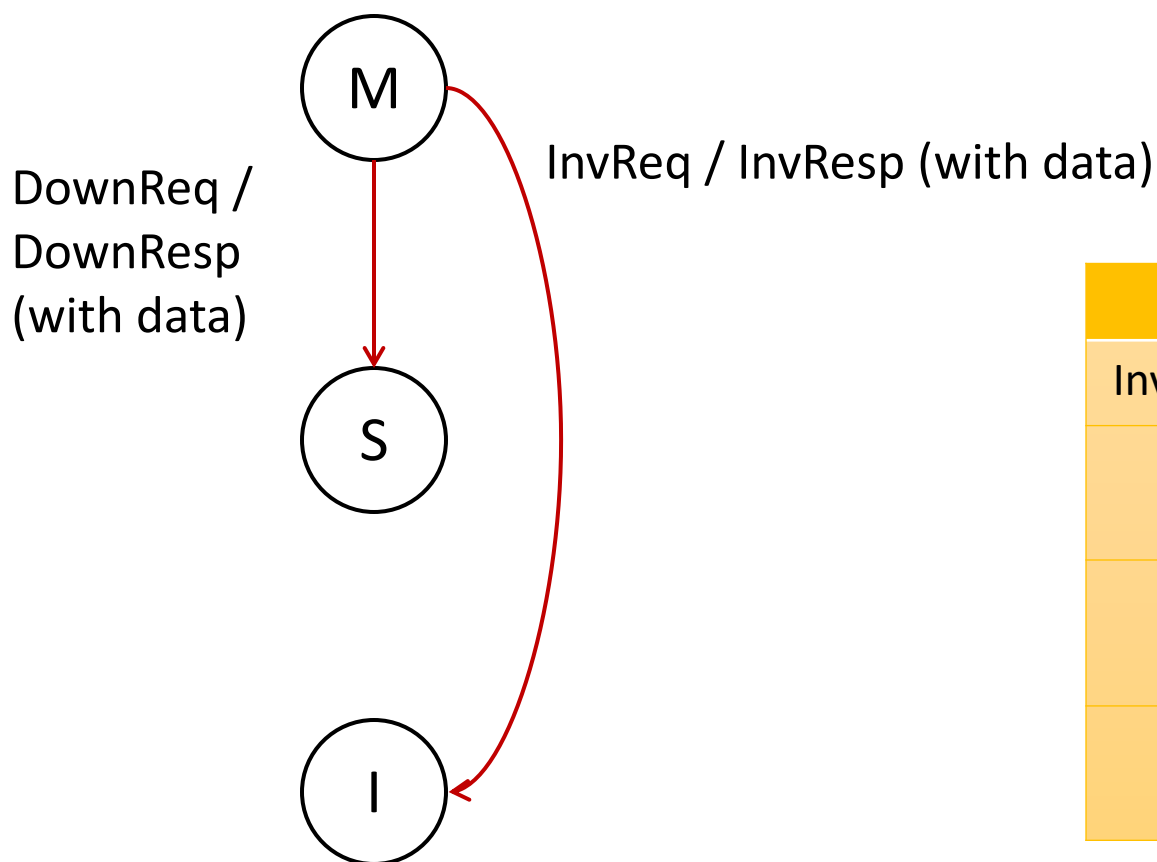
Transitions initiated by directory requests:



Actions
Invalidation Request (InvReq)
Downgrade Request (DownReq)
Invalidation Response (InvResp)
Downgrade Response (DownResp)

MSI Protocol: Caches (2/3)

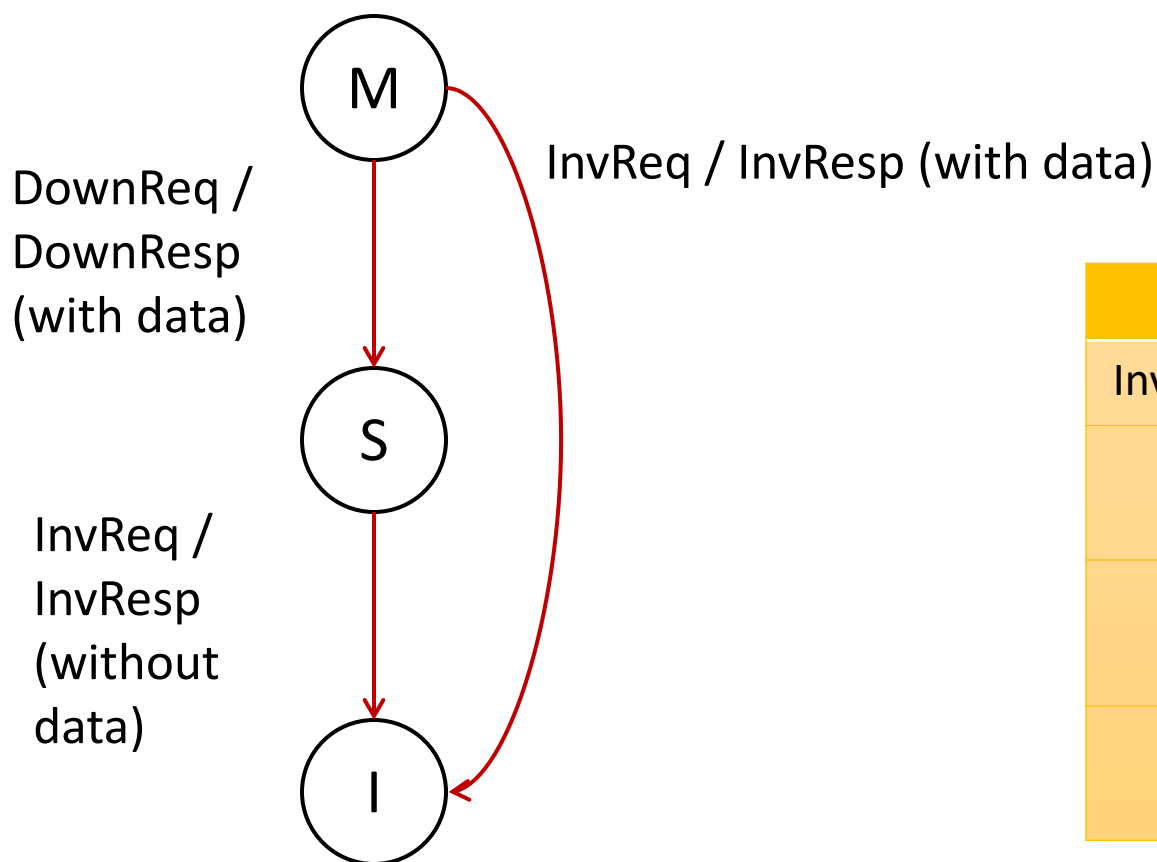
Transitions initiated by directory requests:



Actions
Invalidation Request (InvReq)
Downgrade Request (DownReq)
Invalidation Response (InvResp)
Downgrade Response (DownResp)

MSI Protocol: Caches (2/3)

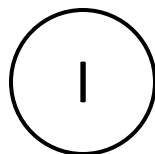
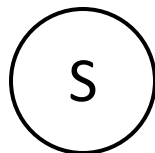
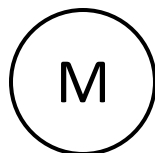
Transitions initiated by directory requests:



Actions
Invalidation Request (InvReq)
Downgrade Request (DownReq)
Invalidation Response (InvResp)
Downgrade Response (DownResp)

MSI Protocol: Caches (3/3)

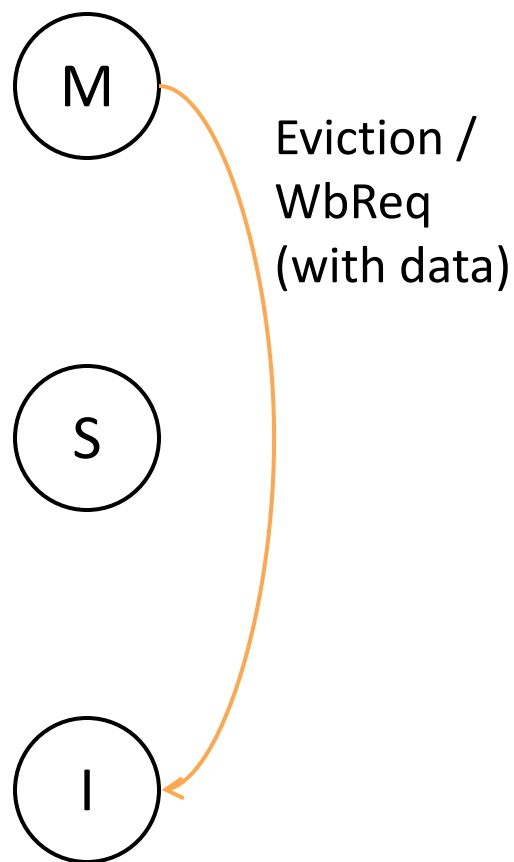
Transitions initiated by evictions:



Actions
Writeback Request (WbReq)

MSI Protocol: Caches (3/3)

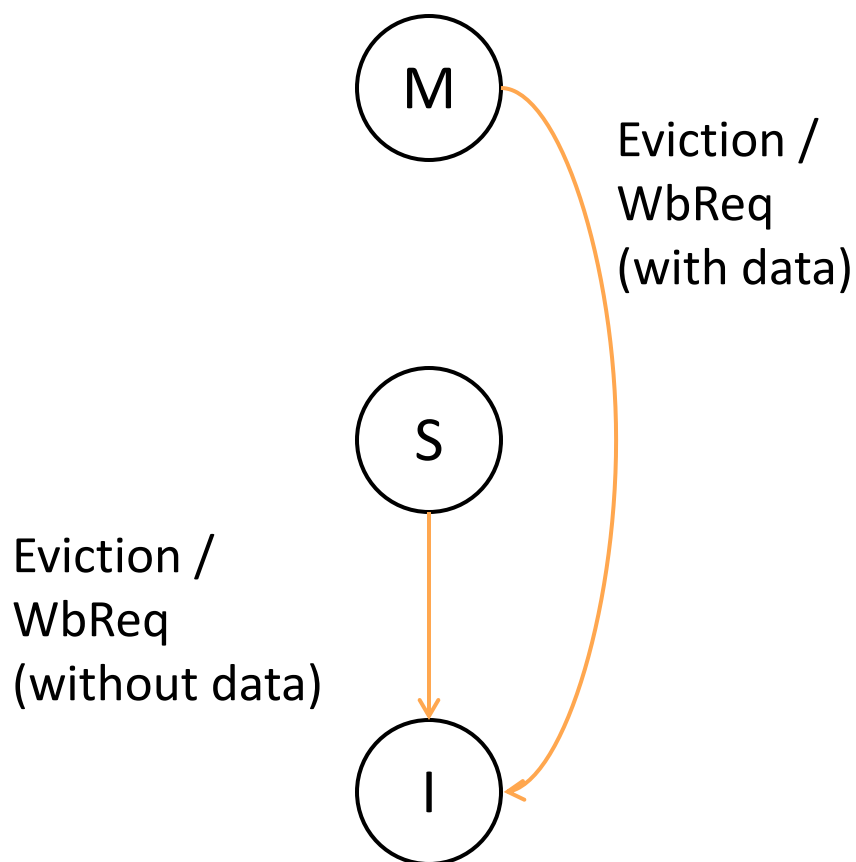
Transitions initiated by evictions:



Actions
Writeback Request (WbReq)

MSI Protocol: Caches (3/3)

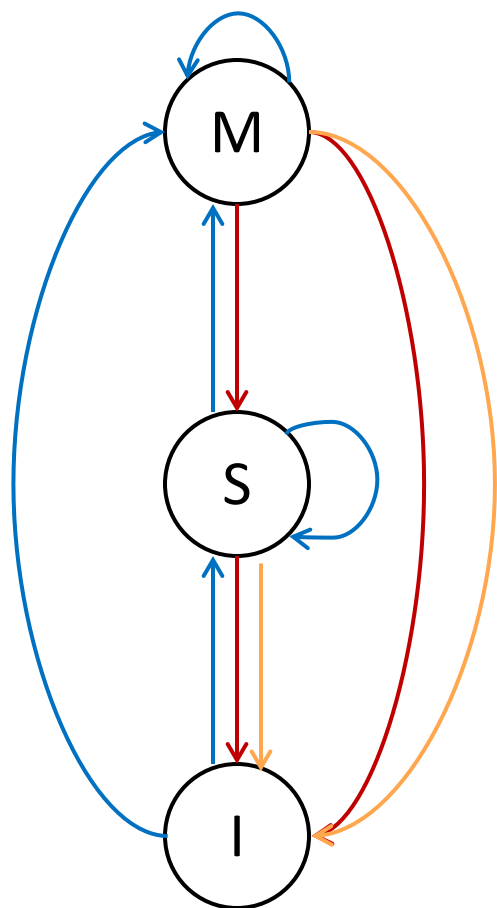
Transitions initiated by evictions:



Actions
Writeback Request (WbReq)

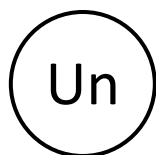
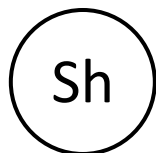
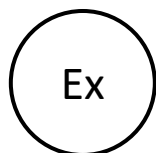
MSI Protocol: Caches

- Transitions initiated by processor accesses
- Transitions initiated by directory requests
- Transitions initiated by evictions



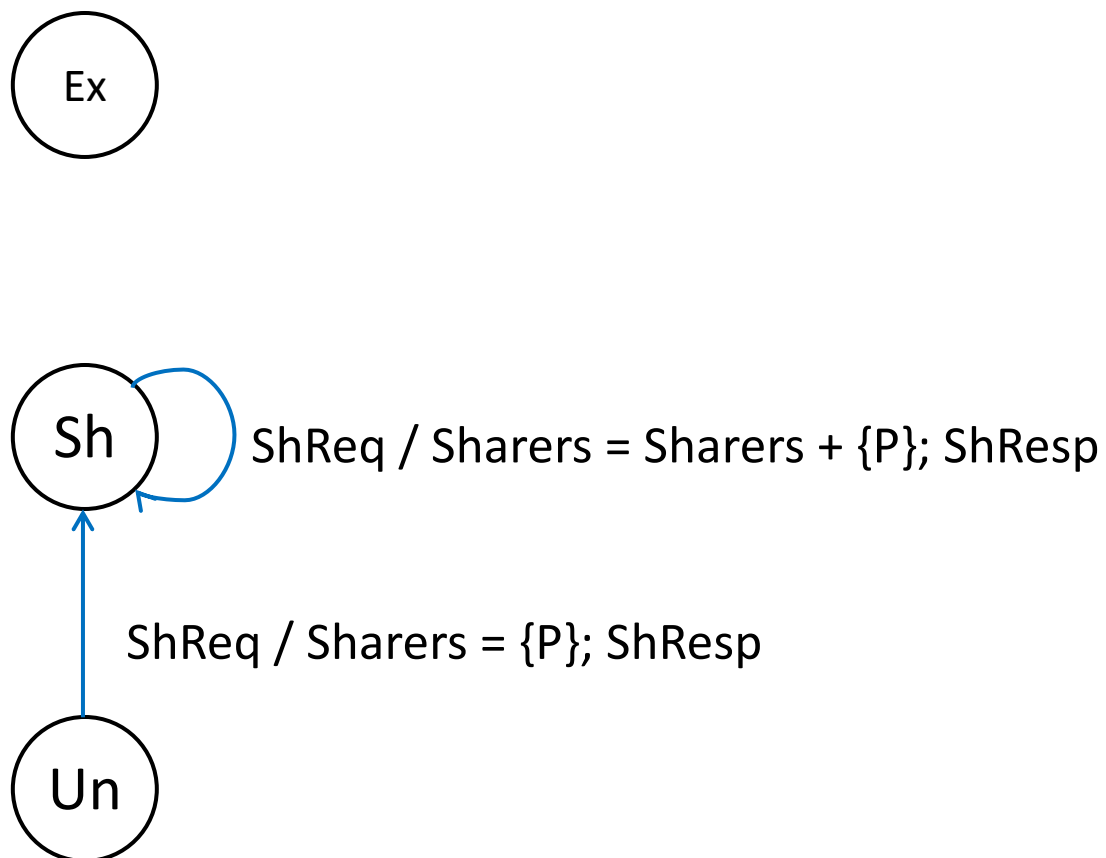
MSI Protocol: Directory (1/2)

Transitions initiated by data requests:



MSI Protocol: Directory (1/2)

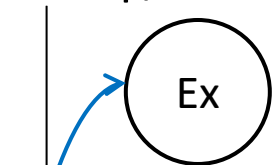
Transitions initiated by data requests:



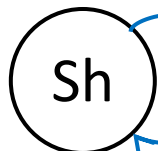
MSI Protocol: Directory (1/2)

Transitions initiated by data requests:

ExReq / Sharers = {P}; ExResp



Ex

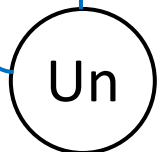


Sh

ShReq / Sharers = Sharers + {P}; ShResp

ShReq / Sharers = {P}; ShResp

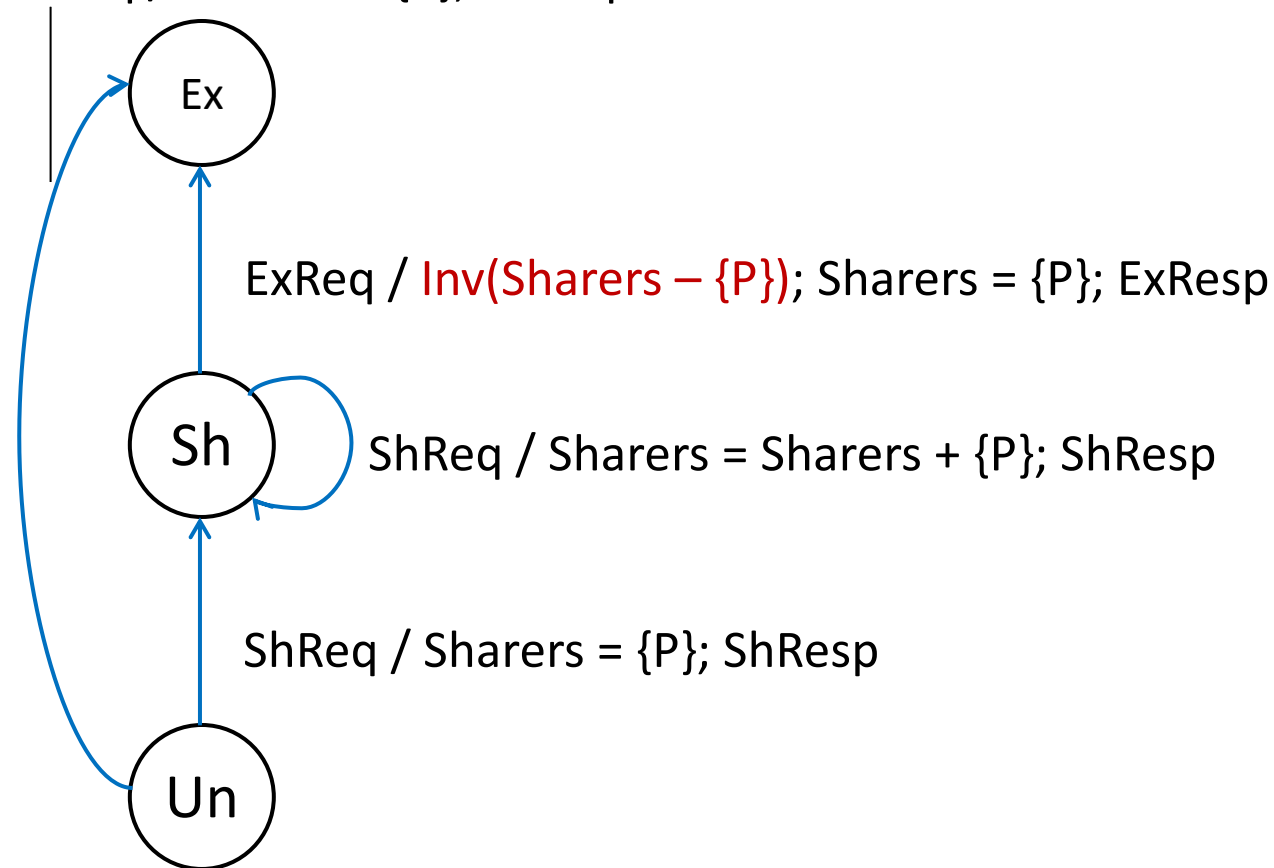
Un



MSI Protocol: Directory (1/2)

Transitions initiated by data requests:

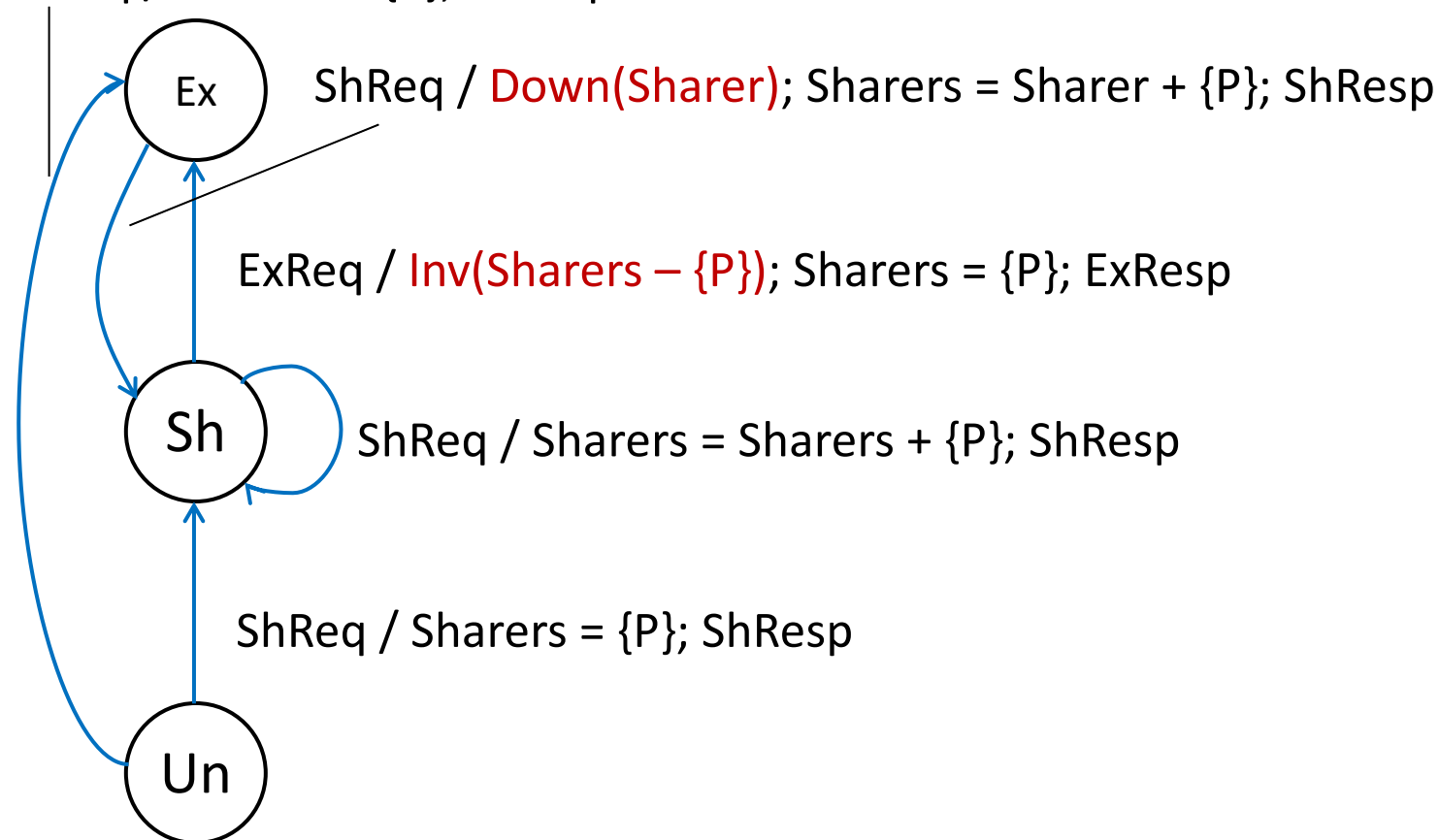
ExReq / Sharers = {P}; ExResp



MSI Protocol: Directory (1/2)

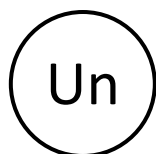
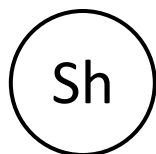
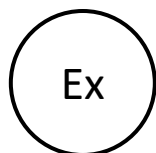
Transitions initiated by data requests:

ExReq / Sharers = {P}; ExResp



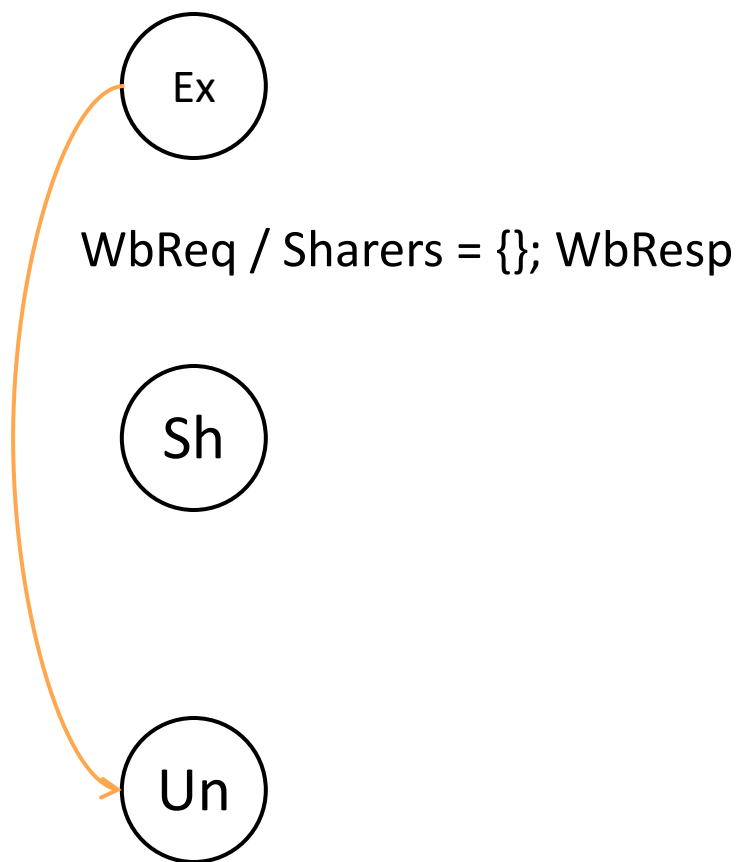
MSI Protocol: Directory (2/2)

Transitions initiated by writeback requests:



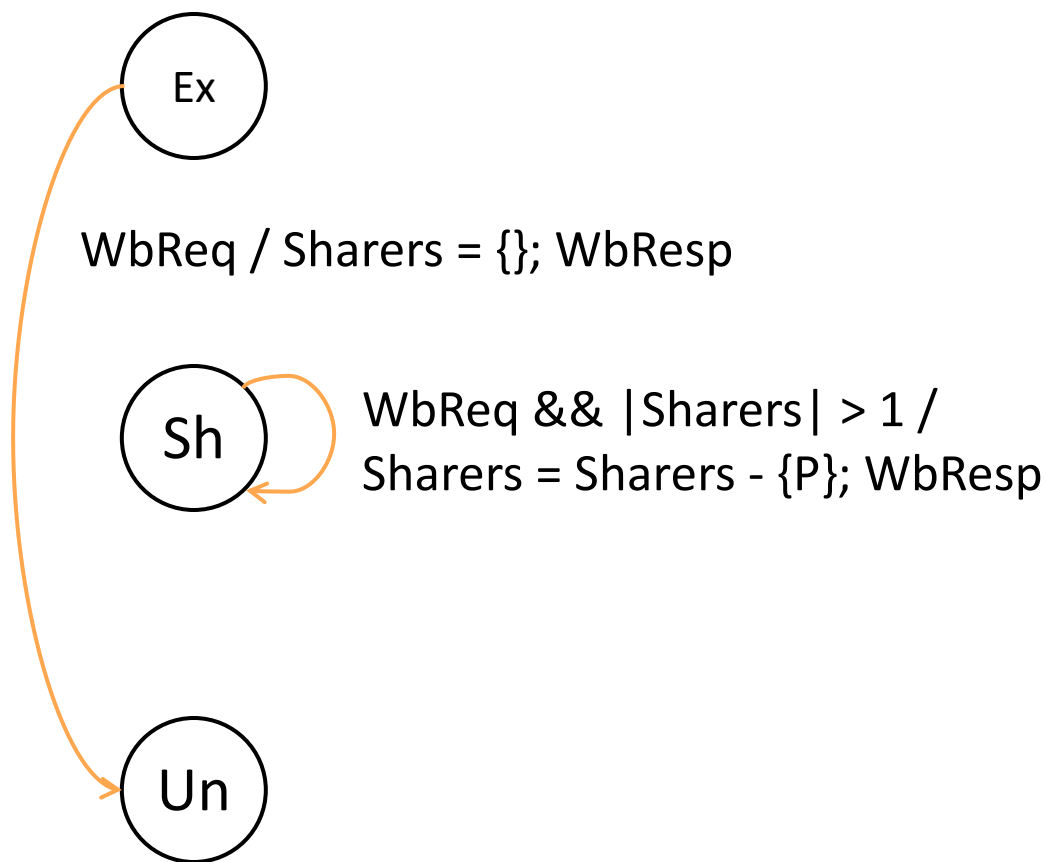
MSI Protocol: Directory (2/2)

Transitions initiated by writeback requests:



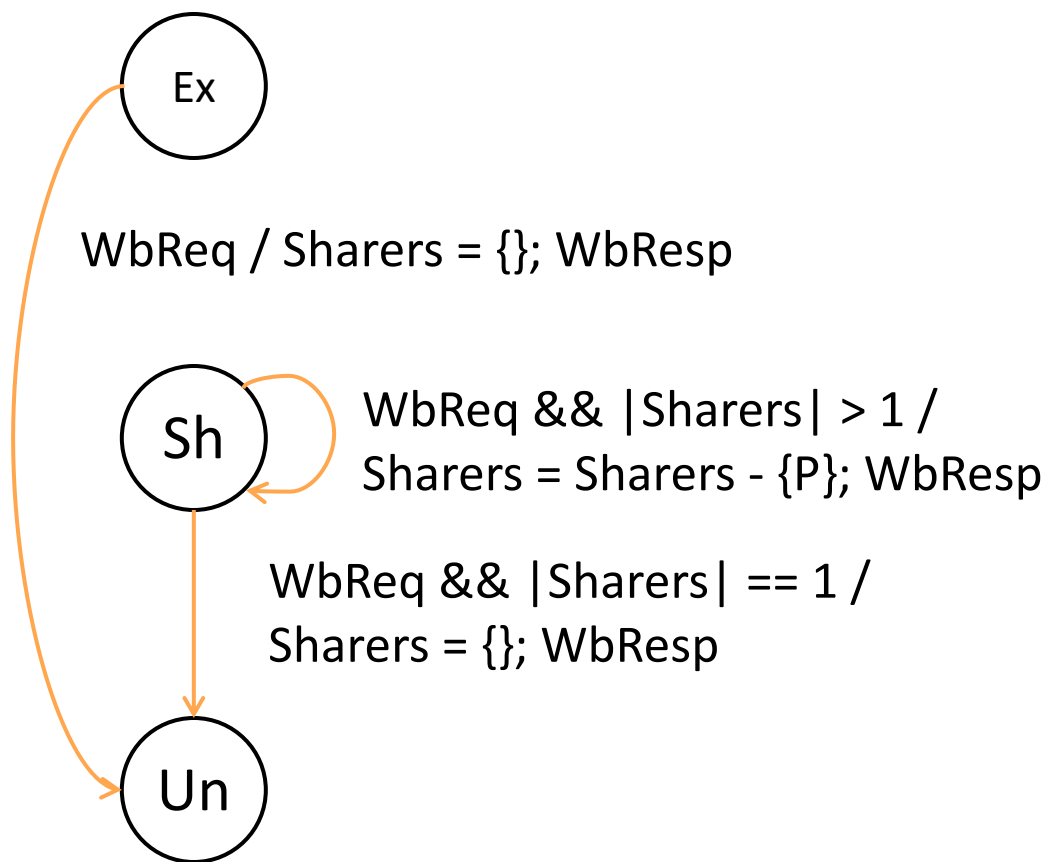
MSI Protocol: Directory (2/2)

Transitions initiated by writeback requests:



MSI Protocol: Directory (2/2)

Transitions initiated by writeback requests:



Optimizations

» Problem: Frequent read-upgrade sequences

- private read-modify-write
- Requires two bus transactions even for private blocks

» Solution: Add Exclusive (E) state

- E: Only one copy, writable, and clean
- Core silently updates to M upon a write to indicate dirty line.

Optimizations

» Problem: Writeback to memory upon M->S downgrade

- Sometimes wastes bandwidth e.g. producer-consumer scenarios
- S implicitly assumes line is clean, allowing silent evictions.

» Solution: Add Owner (O) state

- O: Multiple copies, read-only, and dirty. Also responsible for writing back the data
- Core enters O upon a downgrade.

Lab Task: MSI Coherence Protocol

» Implement with Murphi description language

- Rules: Define transitions between states
- Invariants and asserts: Capture protocol correctness

» Murphi verifier

- Explores reachable states until it finds:
 - A violation of an invariant or assertion, or
 - A state with no possible transitions (deadlock), or
 - It has explored all reachable states and found no errors.
- Exploits symmetry to reduce redundant states

Races

- » Occur when there are multiple messages/requests in flight concerning a single cache line.
- » Try to minimize the opportunity for races by waiting for previous messages before sending new ones.
- » Multiple processors may concurrently initiate conflicting requests.
 - L13-25 shows one case
- » If network may deliver messages out of order, the protocol must handle this. For example:
 - The directory has two messages in flight to one private cache.
 - One processor/cache has two messages in flight to the directory.
- » 3-hop protocol may require you to add more handling for additional races.

Tips

- » Feel free to add to or rename states and messages.
- » Get a 4-hop protocol working first, before attempting 3-hop.
- » Get your protocol working with ProcCount set to 2 before handling the 3-processor case.
- » Write more of assertions and/or invariants.
 - Add assertions/invariants about your transient states.