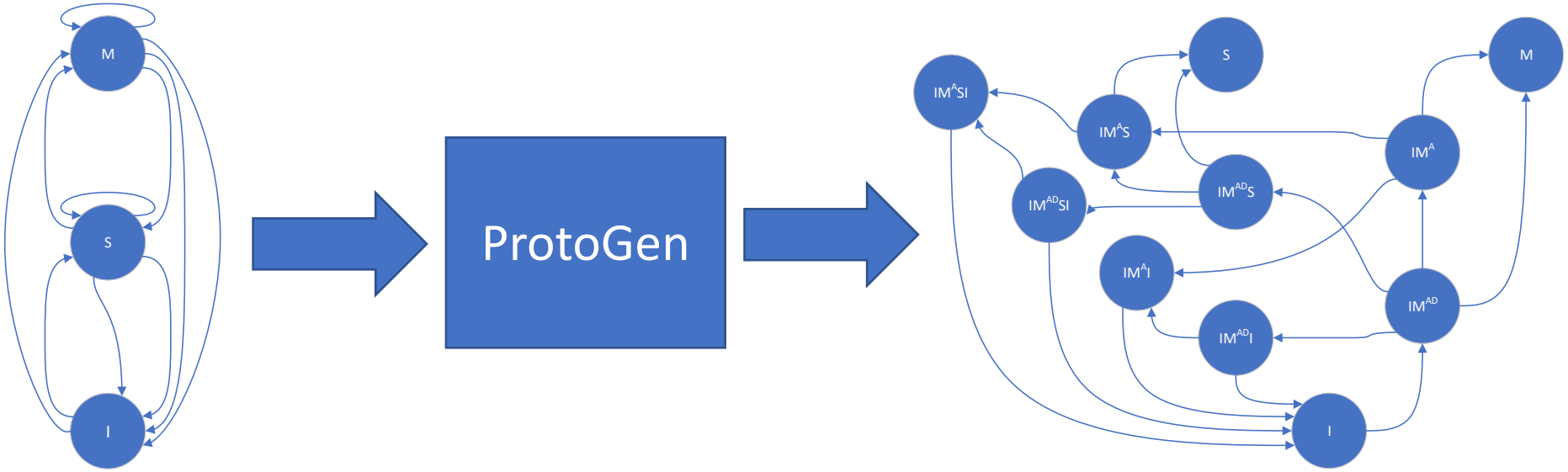
THE UNIVERSITY of EDINBURGH
informatics

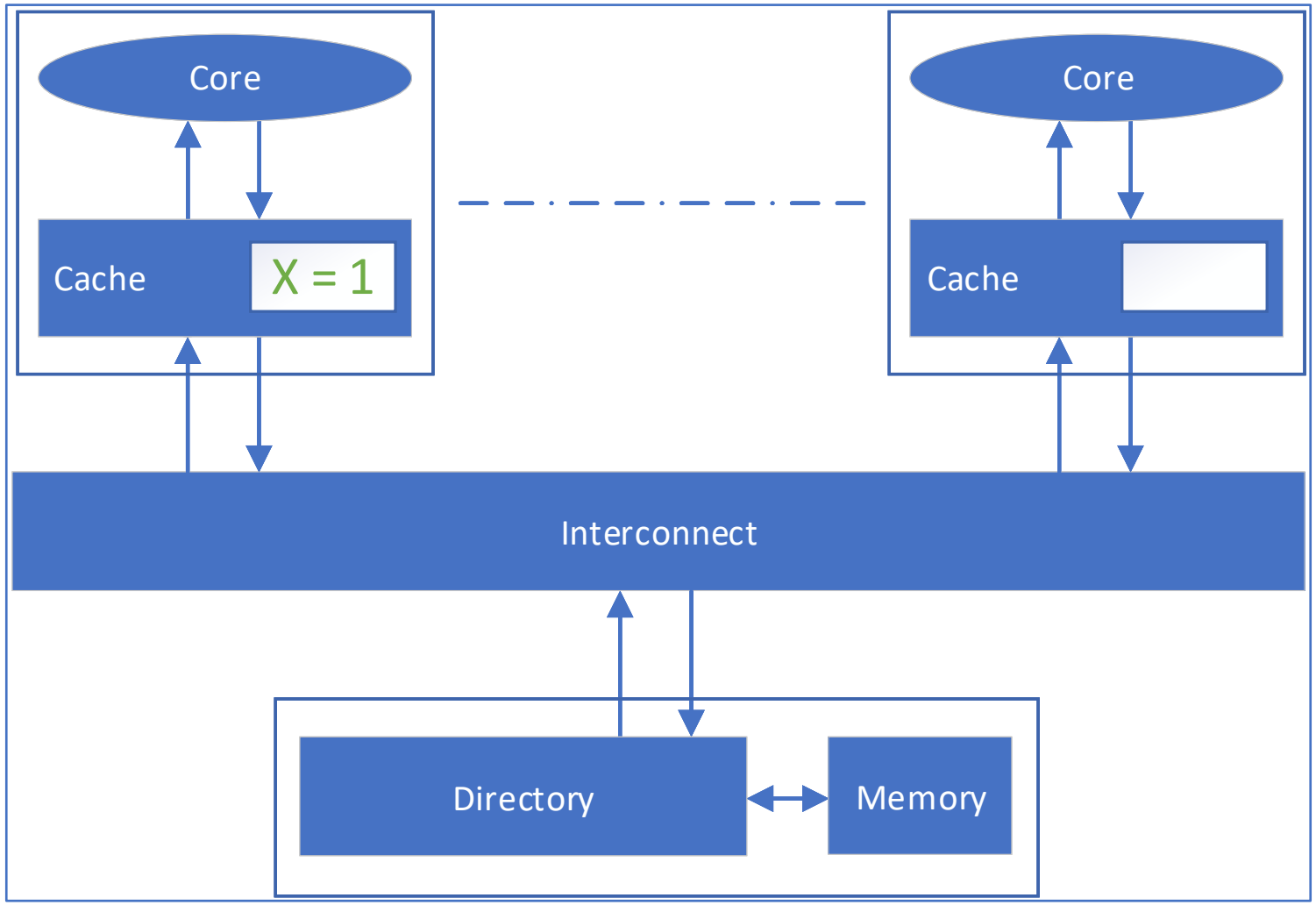
Duke UNIVERSITY | ELECTRICAL & COMPUTER ENGINEERING

Cache Coherence Protocols Are ~~Hard~~ Easy

Nicolai Oswald, **Vijay Nagarajan**, Daniel J. Sorin



Directory Cache Coherence



"Cache coherence protocols are notoriously difficult to design and veri
Memory Systems, 2004]

"... directory-based cach
ar

"Sophis
difficult

"... designing and verifying a new hardware coherence protocol is difficult"
[Spandex: A Flexible Interface for Efficient Heterogeneous Coherence - ISCA 2018]

"The coherence problem is difficult, because it requires coordinating events across nodes"
[IEEE Concurrency 2000]

sly

lt to design and implement correctly" [ASPLOS 2017]

"Cache coherence protocols for distributed shared memory multiprocessors are notoriously difficult to design" [ICFS 1996]

Bugs in the Wild

No.	Errata Description
1	
51	
52	
57	
58	
60	
61	
62	
63	
64	
65	
66	
68	
69	Multiprocessor Coherency Problem with Hardware Prefetch Mechanism
70	Microcode Patch Loading in 64-bit Mode Fails To Use EDX

63 TLB Flush Filter Causes Coherency Problem in Multiprocessor Systems

Description

If the TLB flush filter is enabled in a multiprocessor co
between the page tables in memory and the translations
the possible use of stale translations even after software

Potential Effect on System

Unpredictable system failure.

Suggested Workaround

In MP systems, disable the TLB flush filter by setting HWCR.FFDIS (bit 6 of MSR 0xC001_0015).

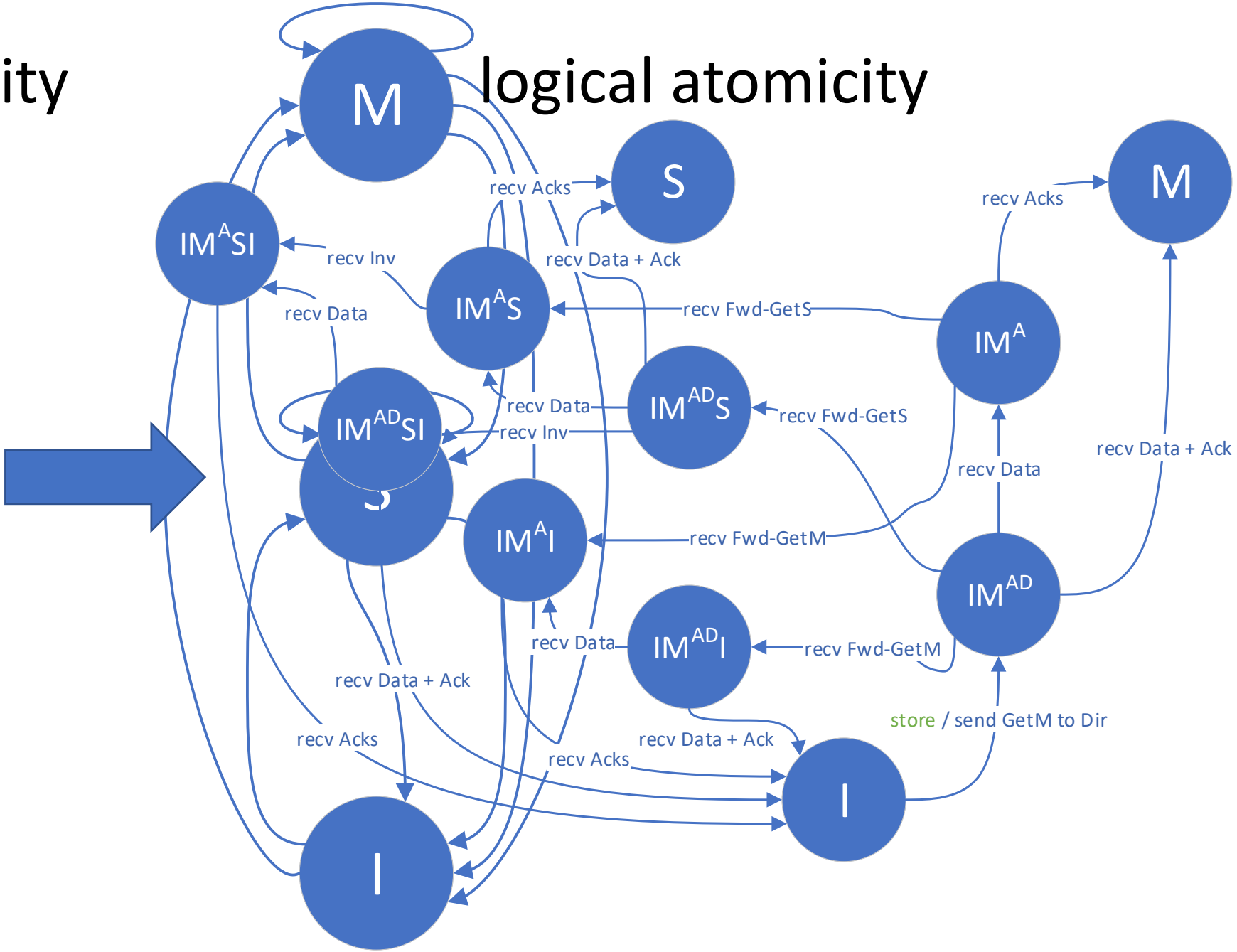
Fix Planned

Yes

From AnandTech: "… coherency was broken and manually disabled on the Galaxy S 4. The implications are serious from a power consumption (and performance) standpoint."

physical atomicity

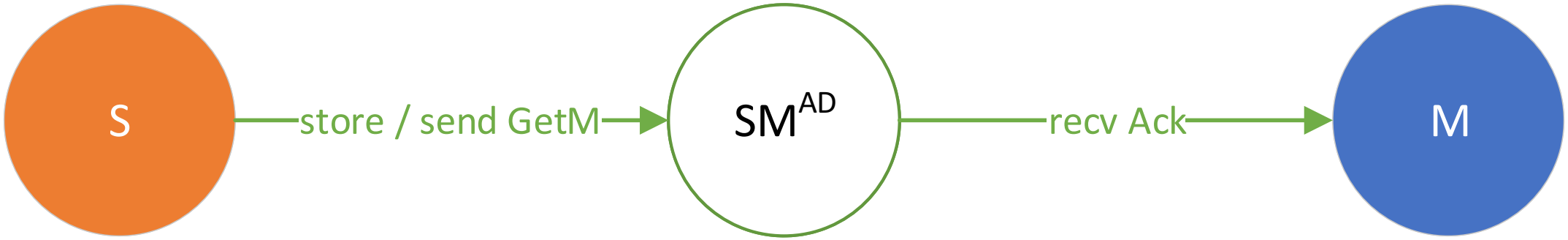
logical atomicity



THE UNIVERSITY of EDINBURGH
informatics

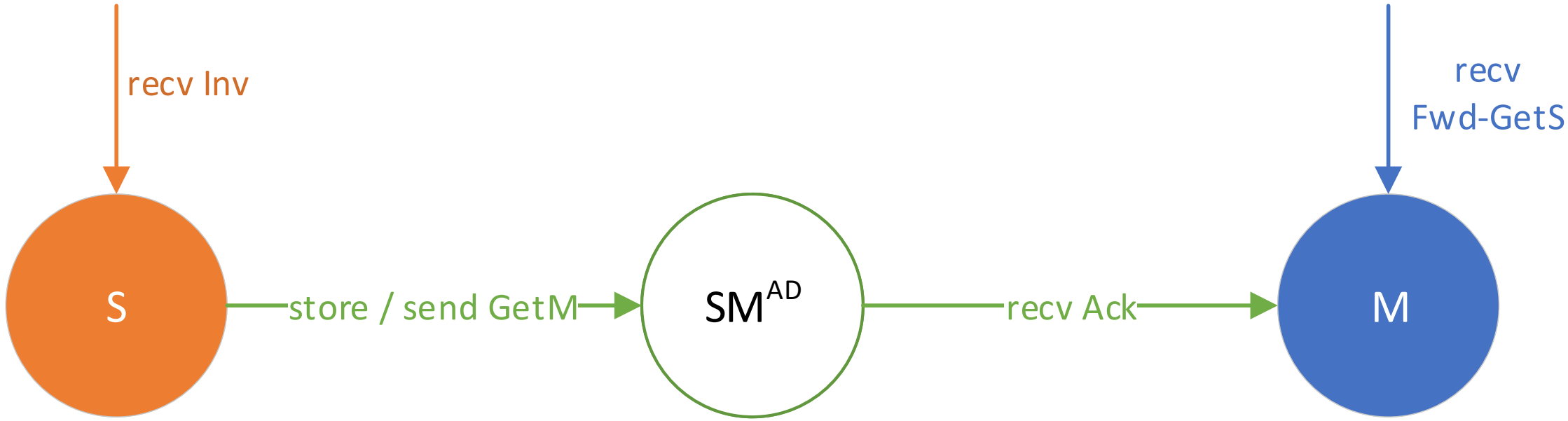
Atomic S to M Transition

Transient States

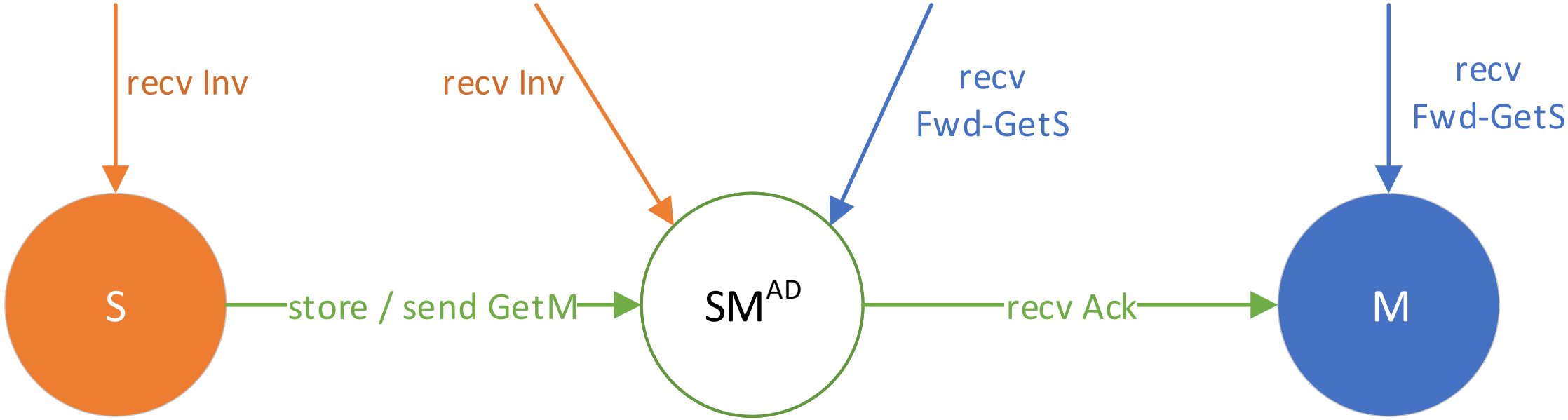


non-atomic transaction

Concurrent Transactions



Concurrent Transactions



non-atomic transactions + concurrency = complexity

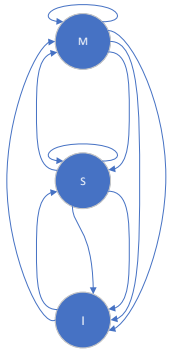
To Summarize…

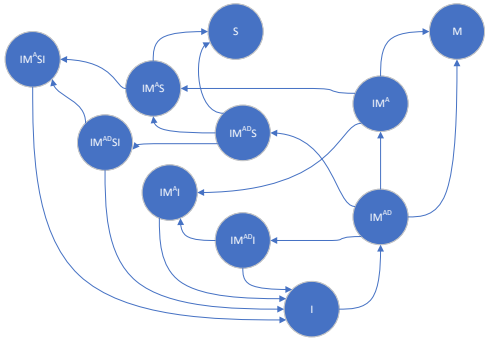
- Stable state protocols assume physically atomic transactions
- Need to support concurrency for performance
- Transient states required to provide logically atomic transactions

Thus…

- Stable state protocol is a *sequential* specification
- The final protocol is a *non-blocking concurrent* implementation
- Transient states are *synchronization* operations

Insight



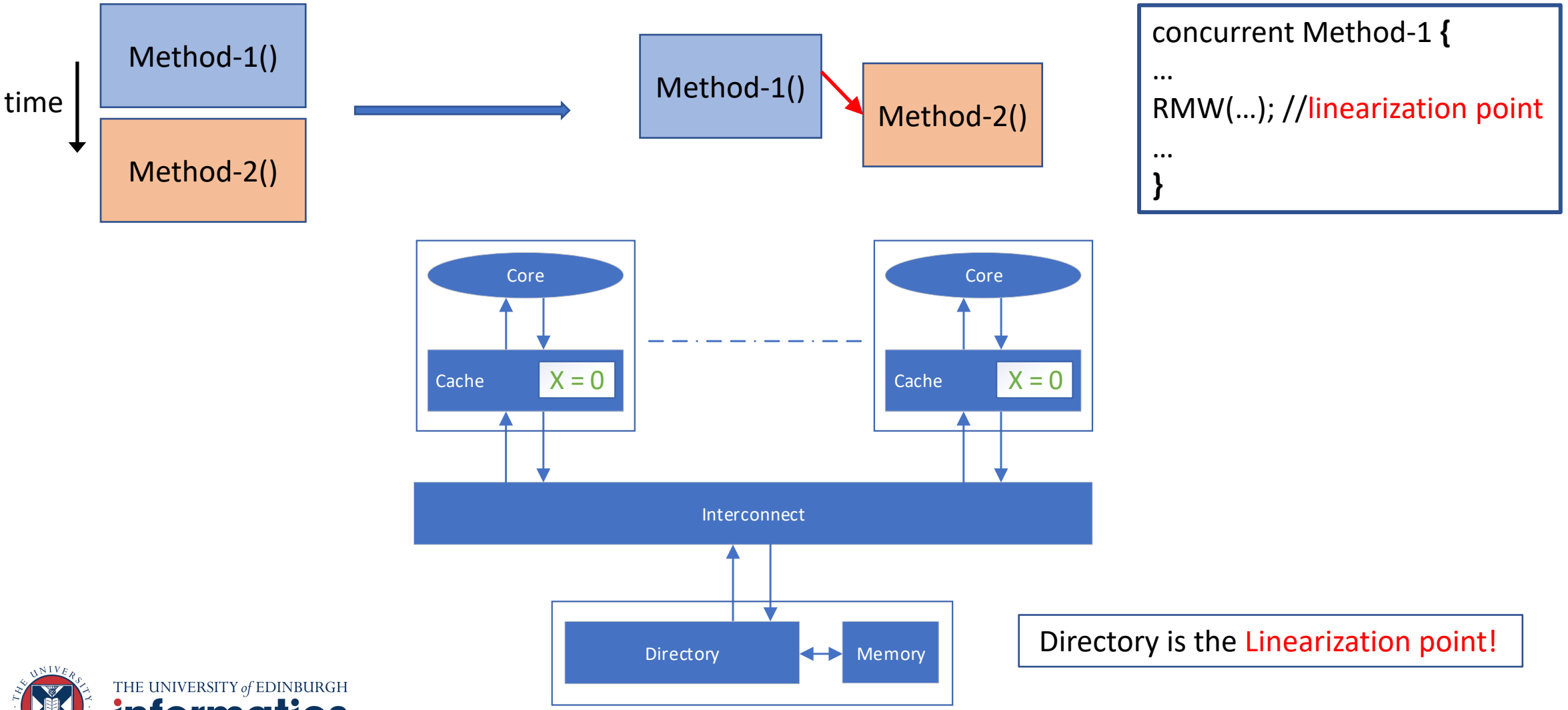


Sequential object {
…
…
…
}

Non-blocking concurrent {
…
…
…
}

No wonder cache coherence protocols are Hard!

Insight


time
Method-1()
Method-2()
Method-1()
Method-2()
concurrent Method-1 {
…
RMW(…); //linearization point
…
}
Core
Cache
X = 0
Core
Cache
X = 0
Interconnect
Directory
Memory
Directory is the Linearization point!

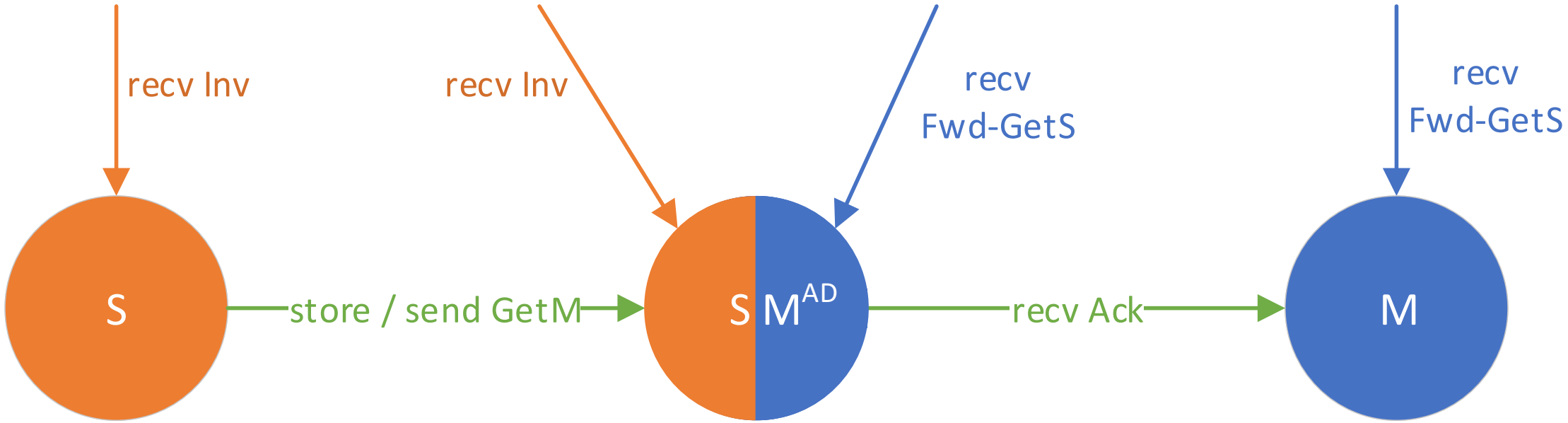
Demystifying Transient States

How do transient states provide logical atomicity?

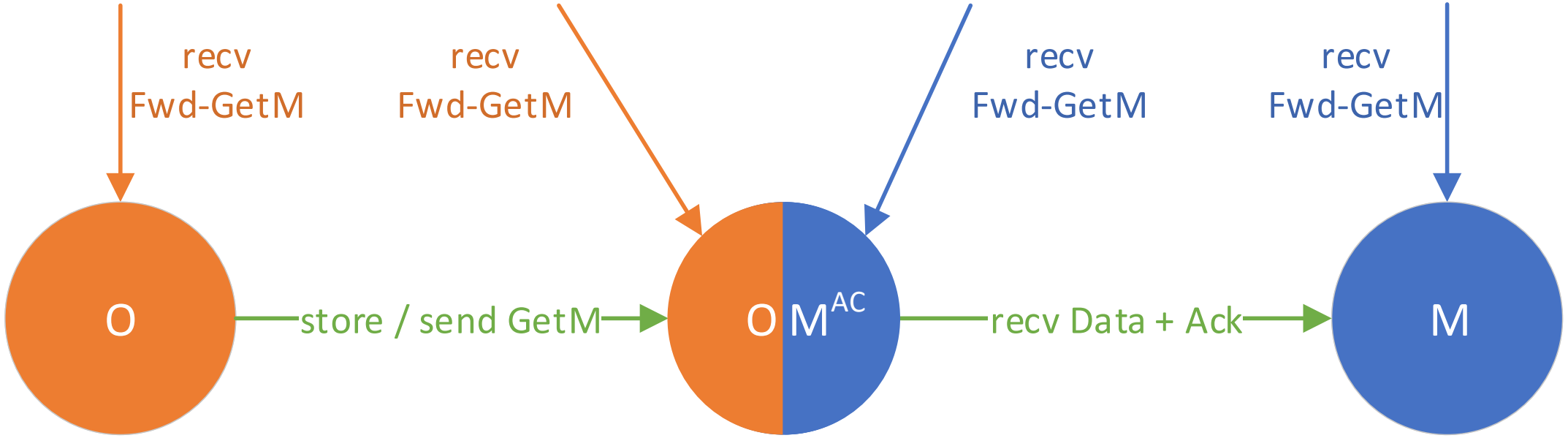
- Convey directory serialization order to caches
- Transient states ensure that caches obey this order

ProtoGen automates by leveraging this insight!

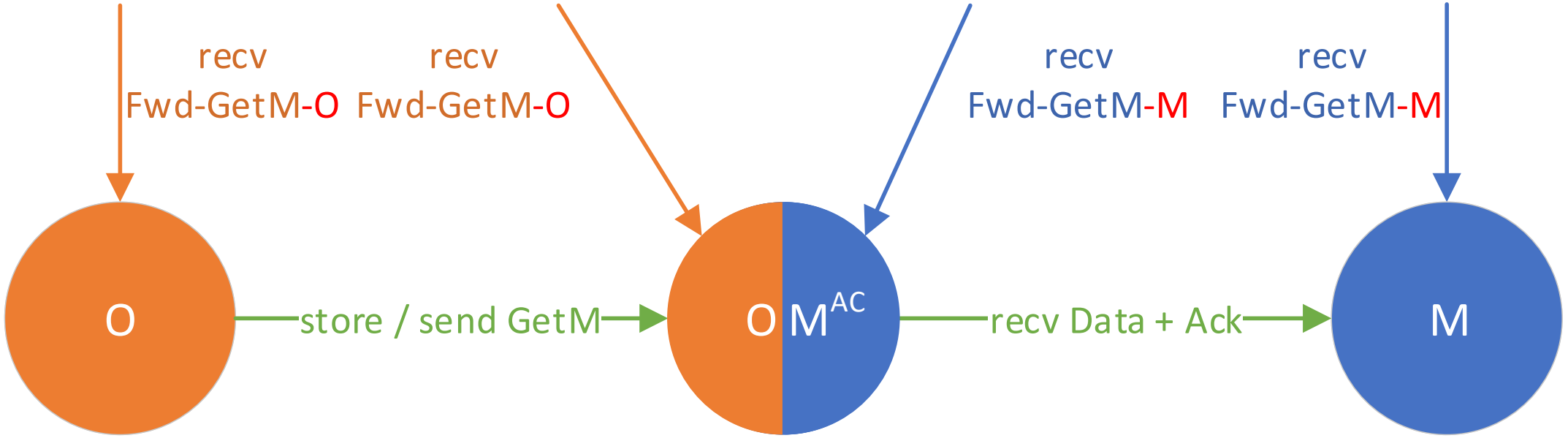
How does cache infer serialization order?



How to resolve name conflicts?



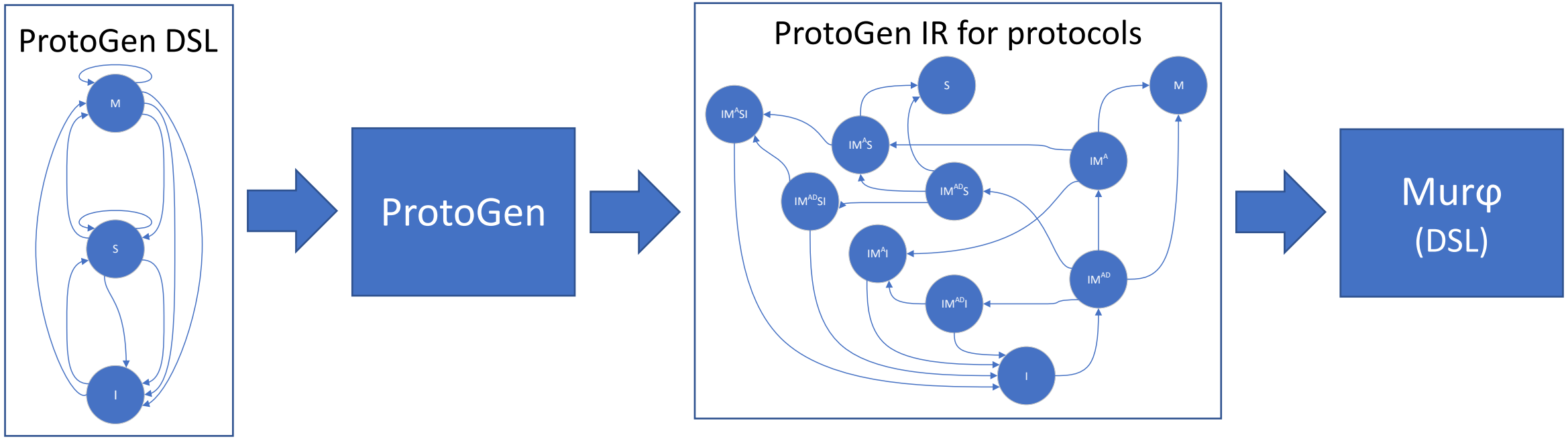
Rename Messages



ProtoGen Summary

- Infer serialization order from incoming messages
- Rename messages in order to achieve this
- React like in stable state

ProtoGen Tool



ProtoGen Verification

Verified Protocols	
MSI	✓
MESI	✓
MOSI	✓
MOESI	✓
TSO-CC	✓

How good are ProtoGen protocols?

- Protocol specifications from Primer
- Stalling protocols: Almost identical
- Non-stalling MSI protocol: 5 fewer stalls

ProtoGen as good (or better) than manually generated protocols

ProtoGen is work in progress…

- Only directory protocols (we believe snooping is possible)
- Needs a correct SSP (working on autocorrecting SSP protocols)
- Only flat protocols (working on hierarchical)
- Needs virtual channel assignment (working on automating it)

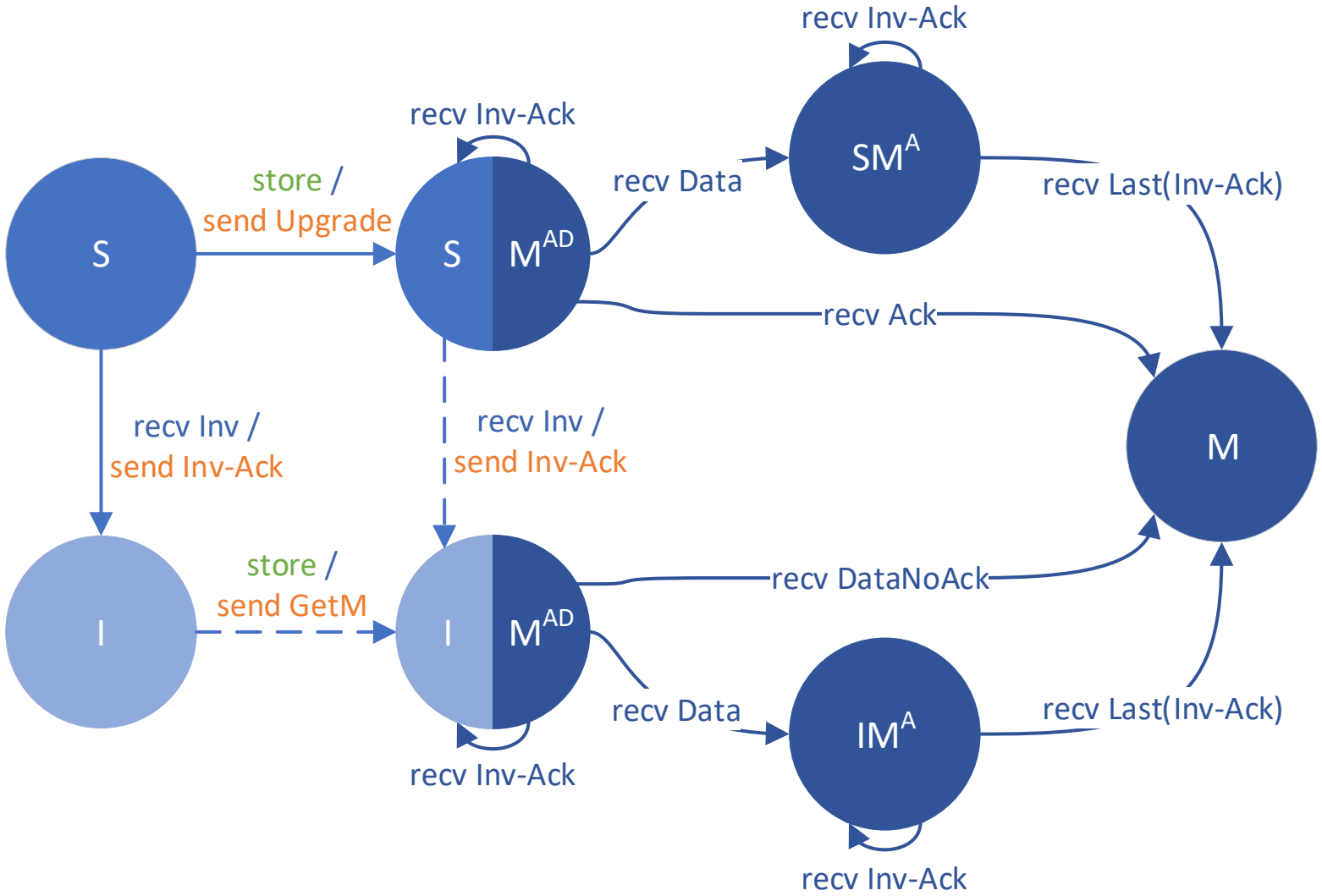
ProtoGen makes coherence protocols easy!

- https://github.com/icsa-caps/ ProtoGen

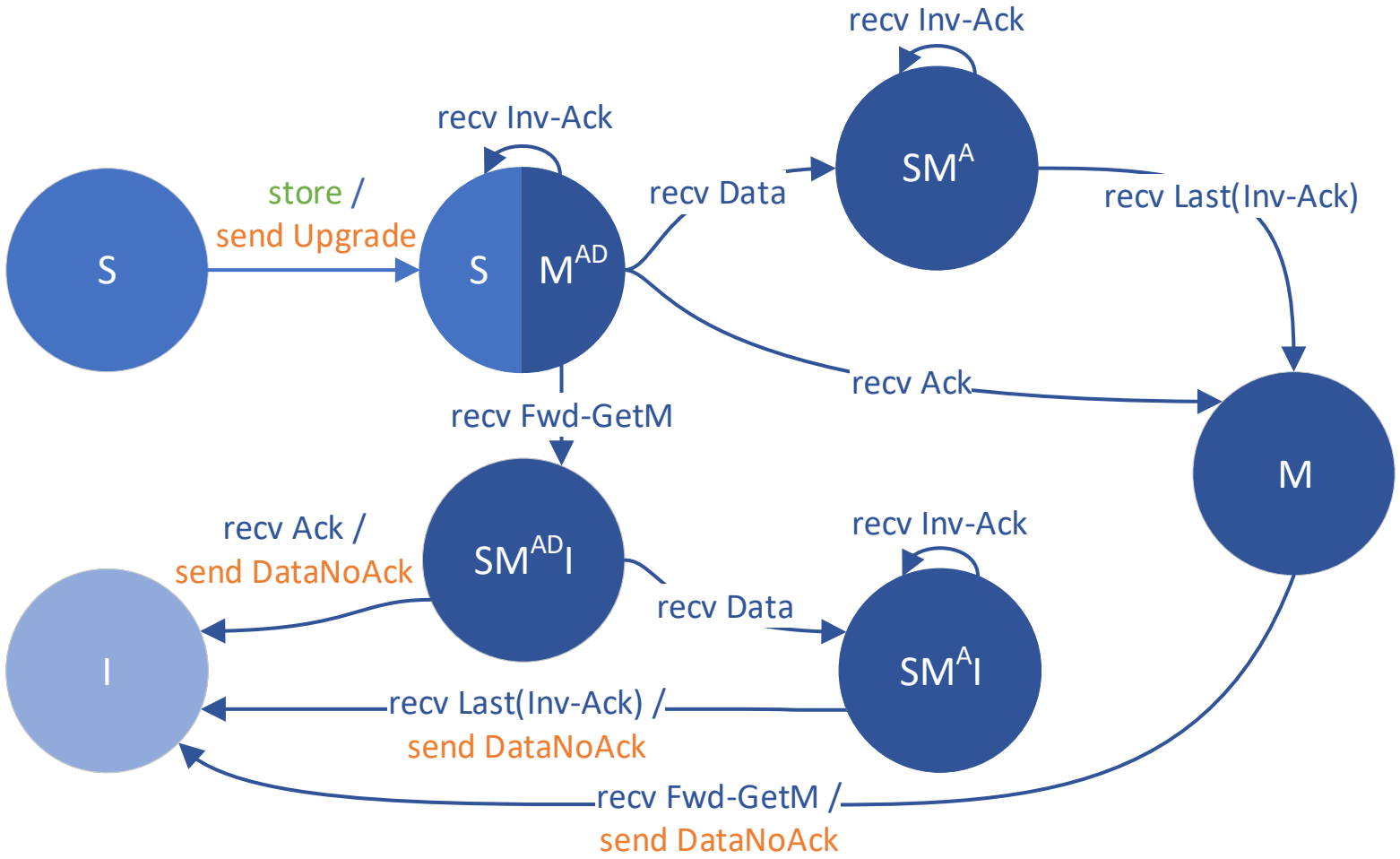
N. Oswald, V. Nagarajan and D.J. Sorin
ProtoGen: Automatically Generating Directory Cache Coherence Protocols from Atomic Specifications
ISCA 2018.

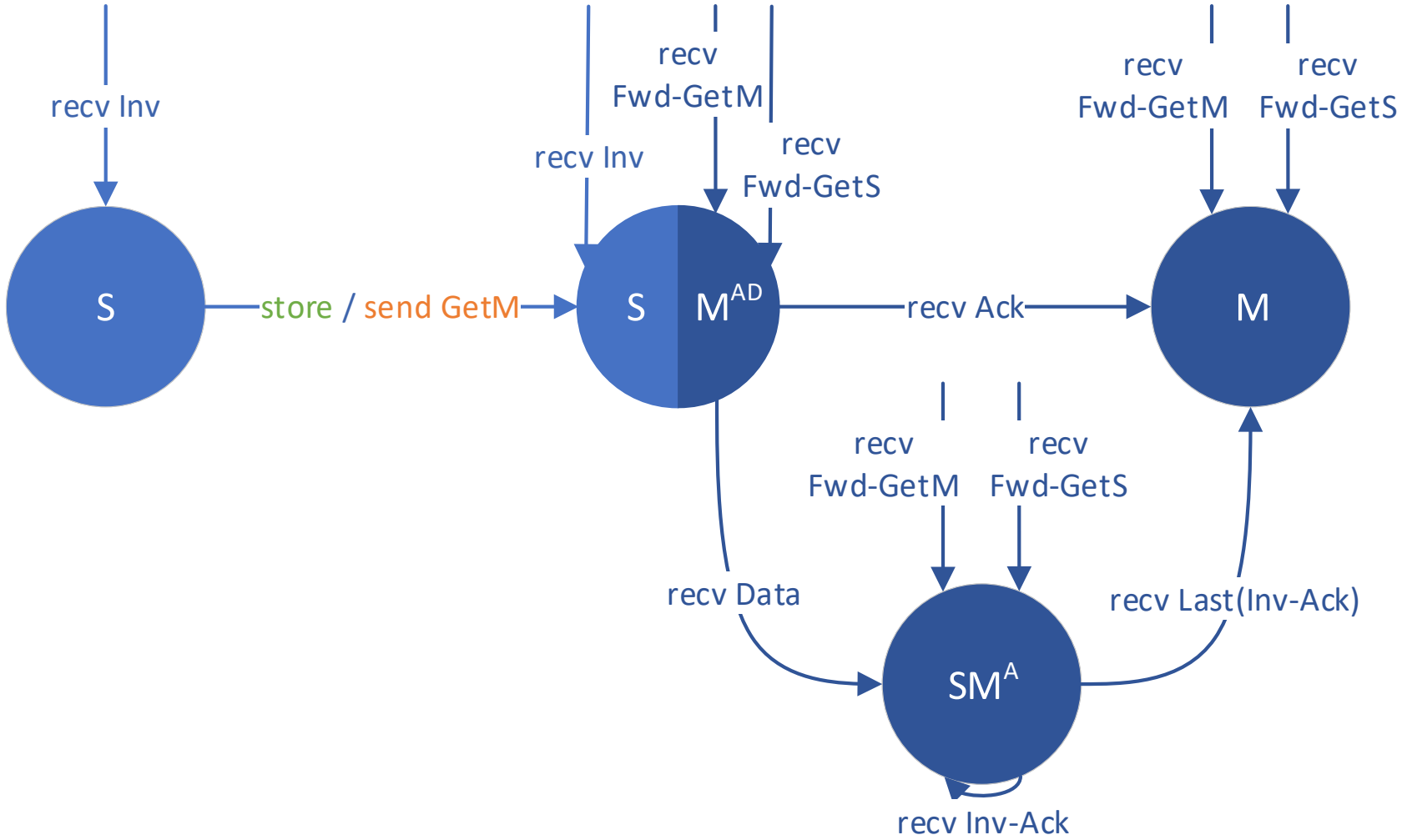
THE UNIVERSITY *of* EDINBURGH
informatics

Own Transaction after Remote Transaction



Own Transaction before Remote Transaction




recv Inv
S
store / send GetM
recv Inv
recv Fwd-GetM
recv Fwd-GetS
S
M^AD
recv Ack
recv Fwd-GetM
recv Fwd-GetS
M
recv Fwd-GetM
recv Fwd-GetS
recv Data
SM^A
recv Last(Inv-Ack)
recv Inv-Ack

Bluespec

- Idea: Guarded atomic actions
  - Atomic updates of multiple participants

- Dave et al. implemented non-blocking coherence protocol
  - Input was not SSP protocol, but complete non-blocking MSI protocol description

Teapot

- Language Support for Writing Memory Coherence Protocols
- Similar to ProtoGen DSL
- Does not automatically generate transient states nor transitions
  - Input is not SSP protocol, but complete non-blocking MSI protocol description

Atomic Coherence

- Atomic Coherence: Leveraging Nanophotonics Similar to Build Race-Free Cache Coherence Protocols
- Atomic transactions
  - Mutex based approach
- Performance achieved by leveraging optical interconnects