Correctness: Model Checking

Doug Woos

Logistics notes

PS 4 due Sunday night

No class on Monday

Distributed systems are hard!

Probably don’t have to convince you!

Simple(-ish) algorithms, complex systems

Many failure scenarios

Non-determinism

Correctness hugely important

How can we get them right?

Correctness options

Thinking really hard 🤔

Testing

Proofs on paper?

Model checking

Full formal verification

Correctness options

Thinking really hard 🤔

Testing

Proofs on paper?

<u>Model checking</u>

Full formal verification

Model checking

Model:

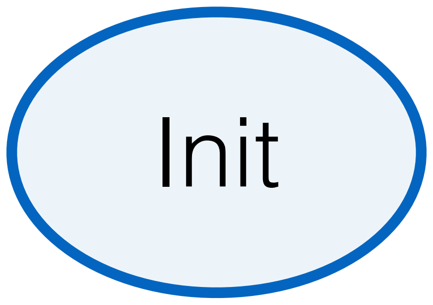
- A formal model of the system in a logic
- Abstracts implementation details
- A specification in the same logic

Checking:

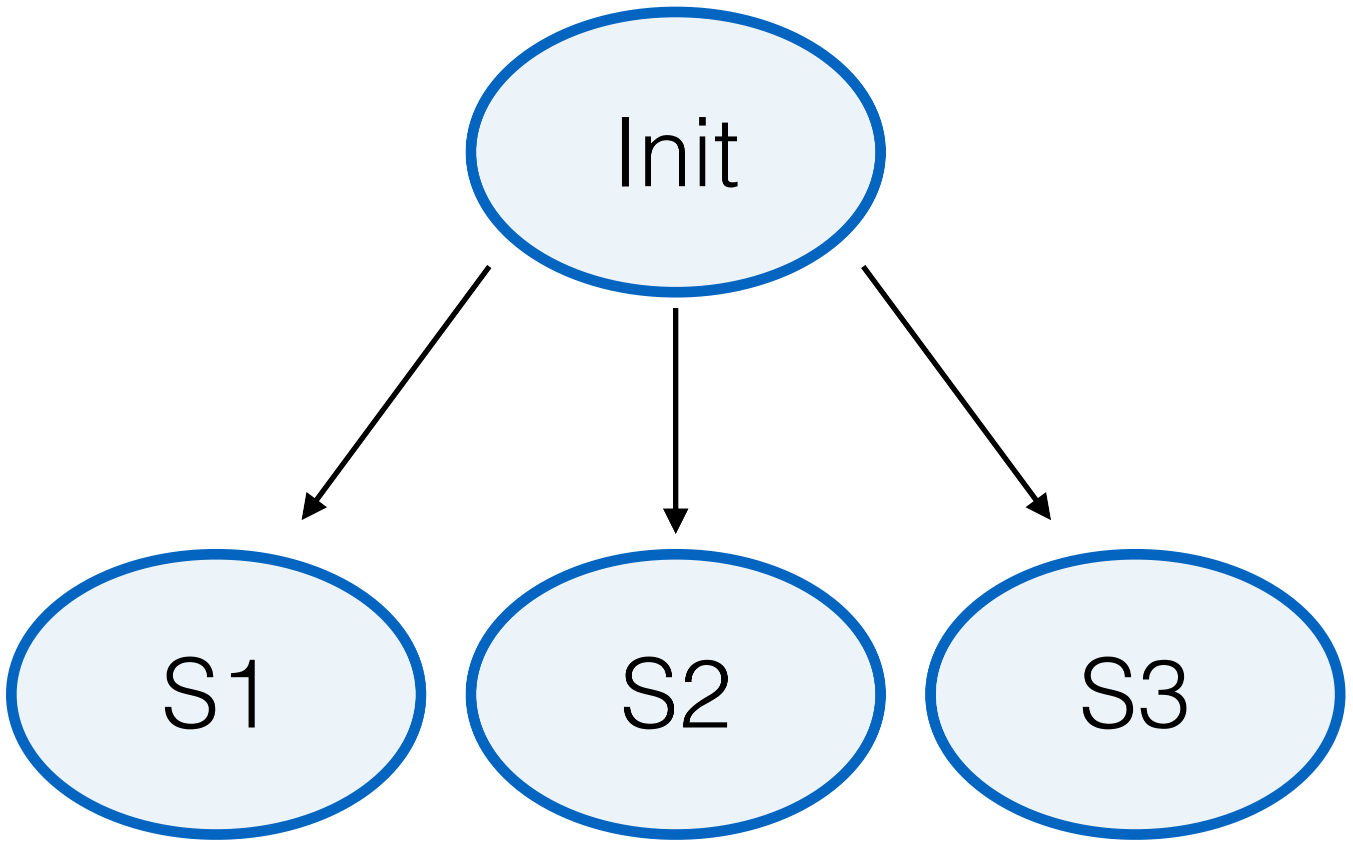
- Exhaustively test the model
- Ensures that it follows the specification

Symbolic state vs. <u>concrete state</u>

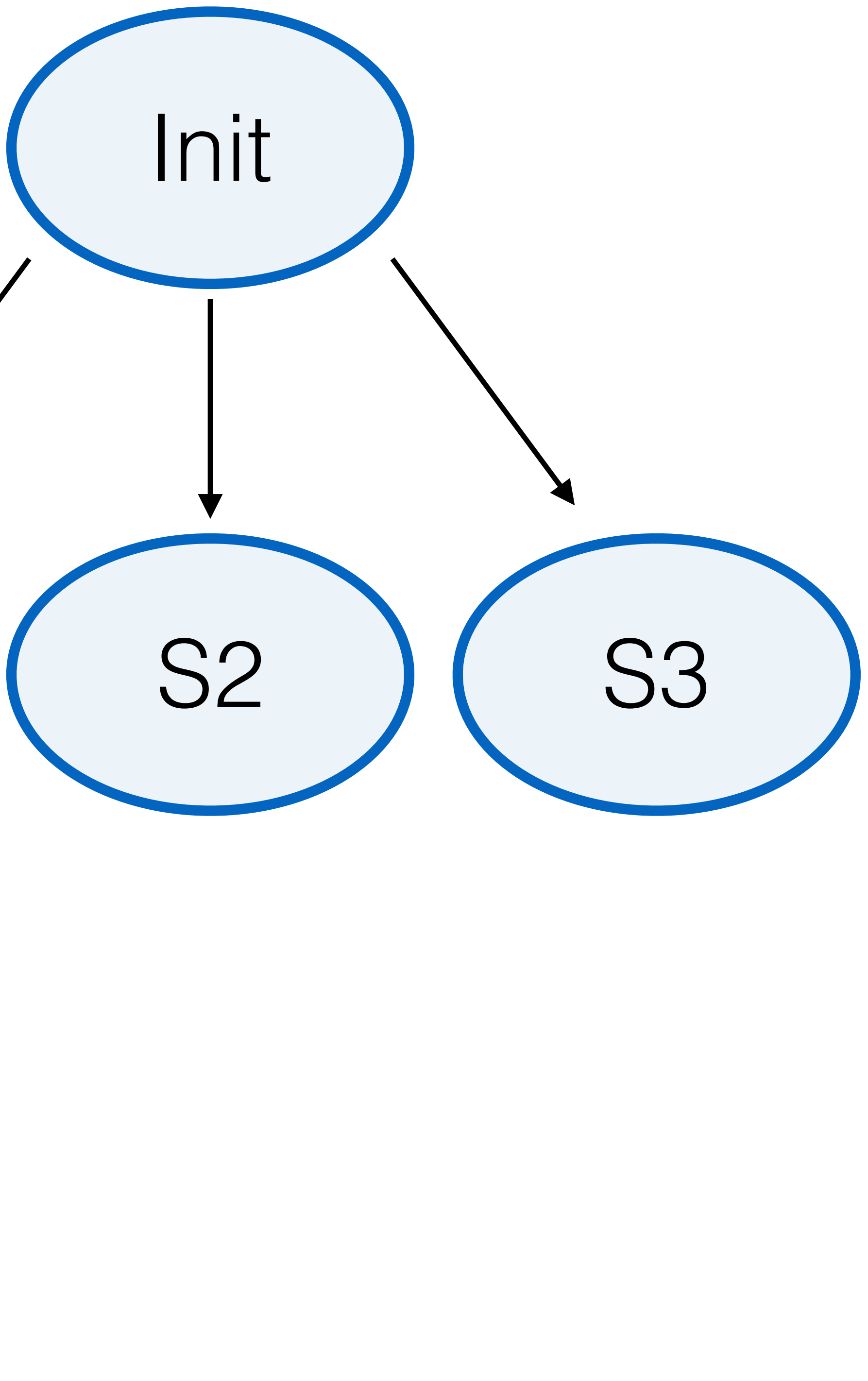
Concrete state model checking



Concrete state model checking



Concrete state model checking


$$\begin{tikzcd}
 & \text{Init} \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{S1} \arrow[d] & \text{S2} & \text{S3} \\
\text{S2} & &
\end{tikzcd}$$

Model checking challenges

State space is probably infinite

- Need to add bounds
- Up to 3 nodes, all clocks <= 5, etc.

Even with bounds, state space very large!

- Lots of clever techniques to reduce the space
- Symmetry-breaking, state hashing, etc.

Does model match implementation?

- Errors in translating to real code
- Code can have typical errors (NPEs, overflow, etc.)

Model checking Demo

Mutual exclusion

Use clocks to implement a lock

Goals:

- Only one process has the lock at a time
- Requesting processes eventually acquire the lock, in same order they request it

Assumptions:

- Reliable in-order channels (TCP)
- No failures

Mutual exclusion implementation

Timestamp all messages

Three message types:

- *request*
- *release*
- *acknowledge*

Each node's state:

- A queue of *request* messages, ordered by T_m
- The latest message it has received from each node

Mutual exclusion implementation

On receiving a *request*:

- Record message timestamp
- Add request to queue

On receiving a *release*:

- Record message timestamp
- Remove corresponding request from queue

On receiving an *acknowledge*:

- Record message timestamp

Mutual exclusion implementation

To acquire the lock:

- Send *request* to everyone, including self
- The lock is acquired when:
    - My request is add the head of my queue, and
    - I’ve received higher-timestamped messages from everyone

Next time

How to do distributed systems proofs

- Safety and liveness
- Invariants and induction

Machine-checked proofs

- Proof for all possible executions
- About actual implementation!
- Downside: lots and lots of work