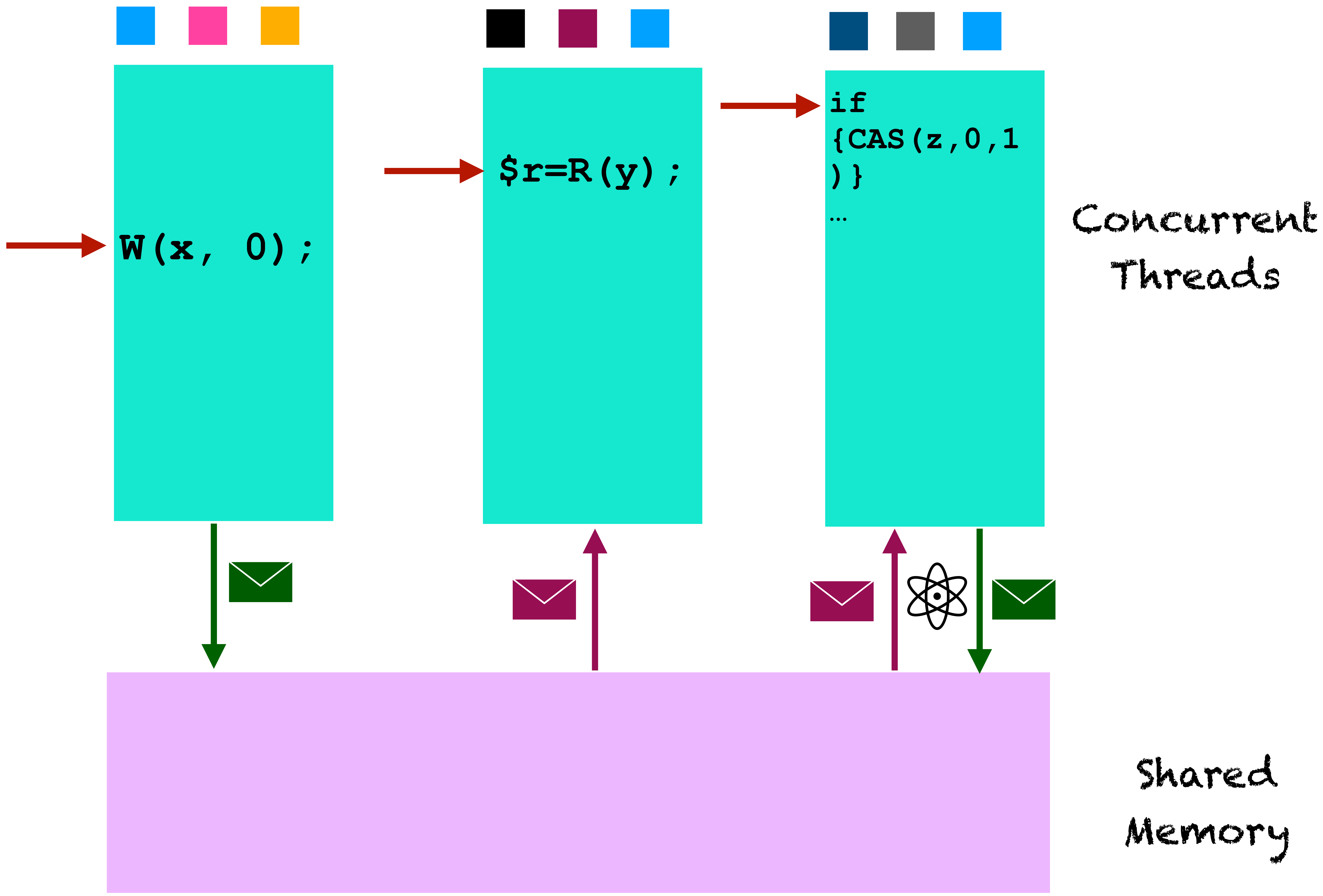
Overcoming Memory Weakness with Unified Fairness

- Parosh Aziz Abdulla (Uppsala University)
- Mohamed Faouzi Atig (Uppsala University)
- Adwait Godbole (University of California, Berkeley)
- Shankaranarayanan Krishna (IIT Bombay)
- **Mihir Vahanwala (MPI-SWS, Saarbrücken)**

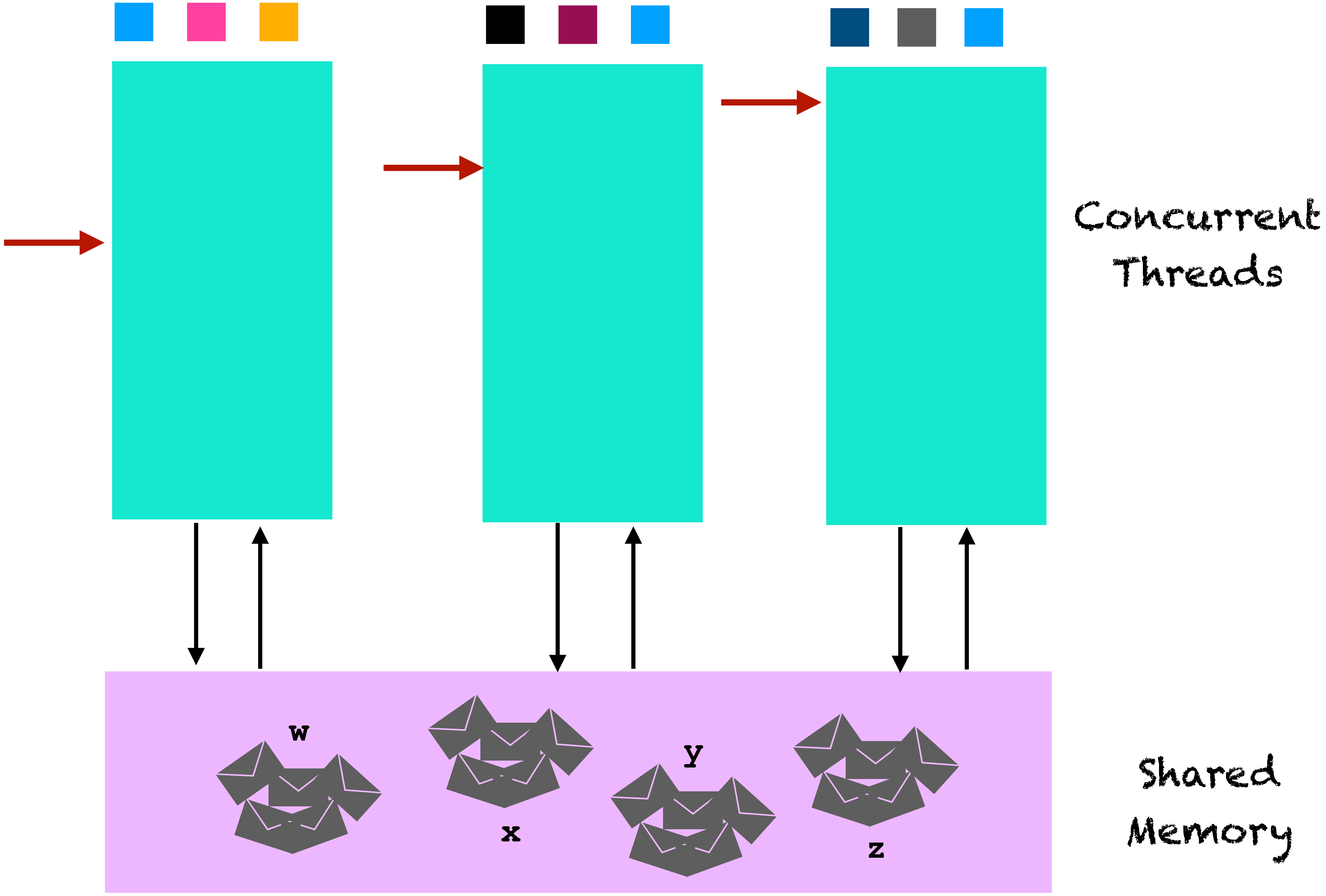
RHPL 2023, Hyderabad

December 20

Concurrency and Memory: The Setup



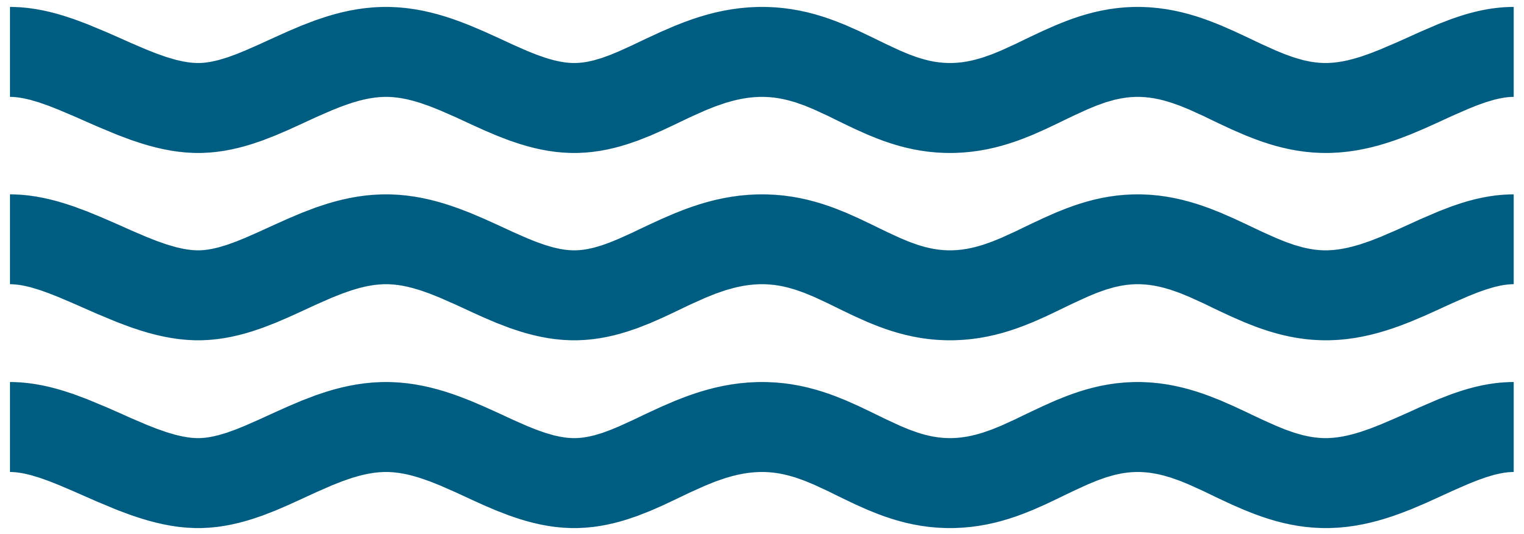
Weak memory: an abstract idea



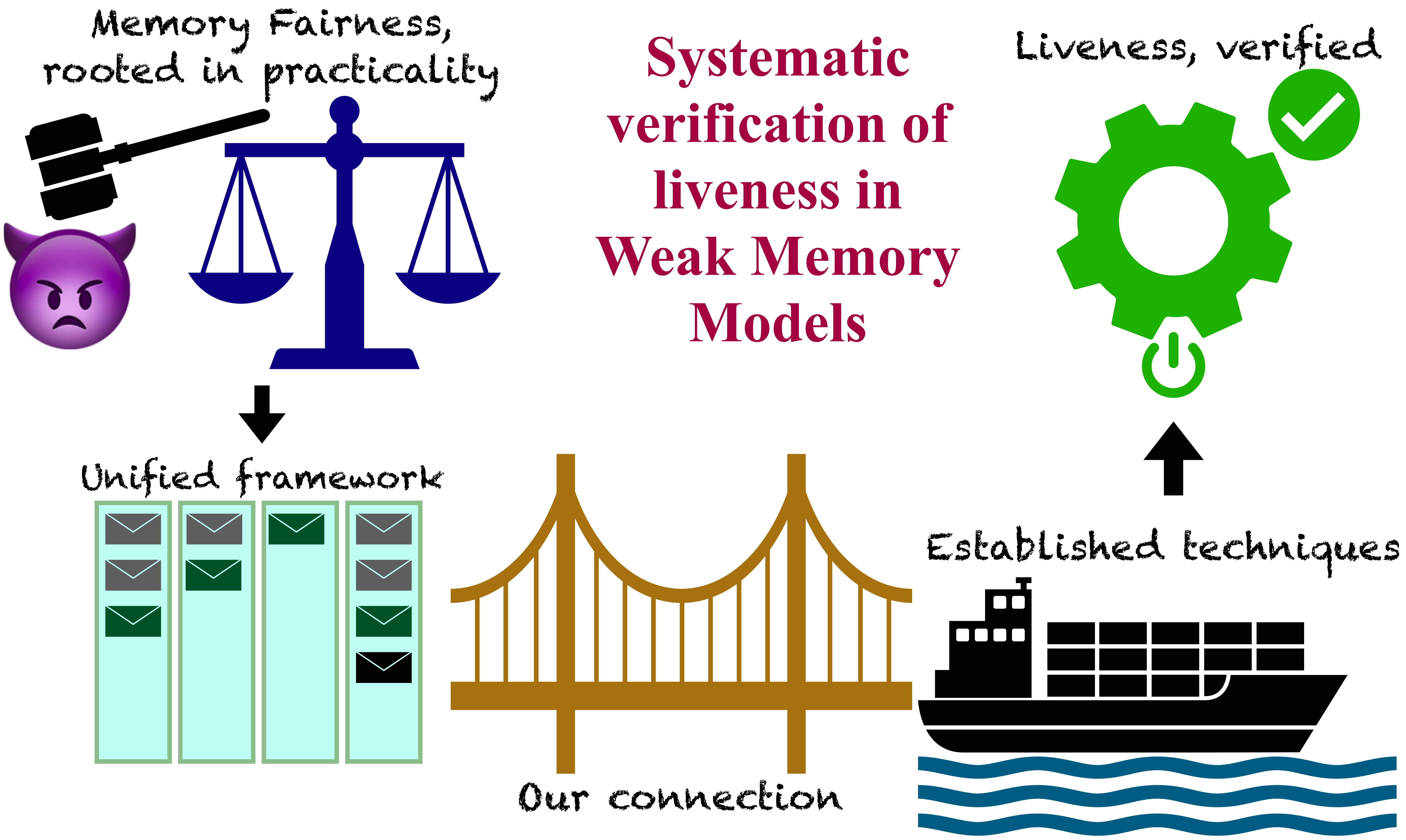
Weak Memory: The challenge

Demonic, impractical
Non-determinism

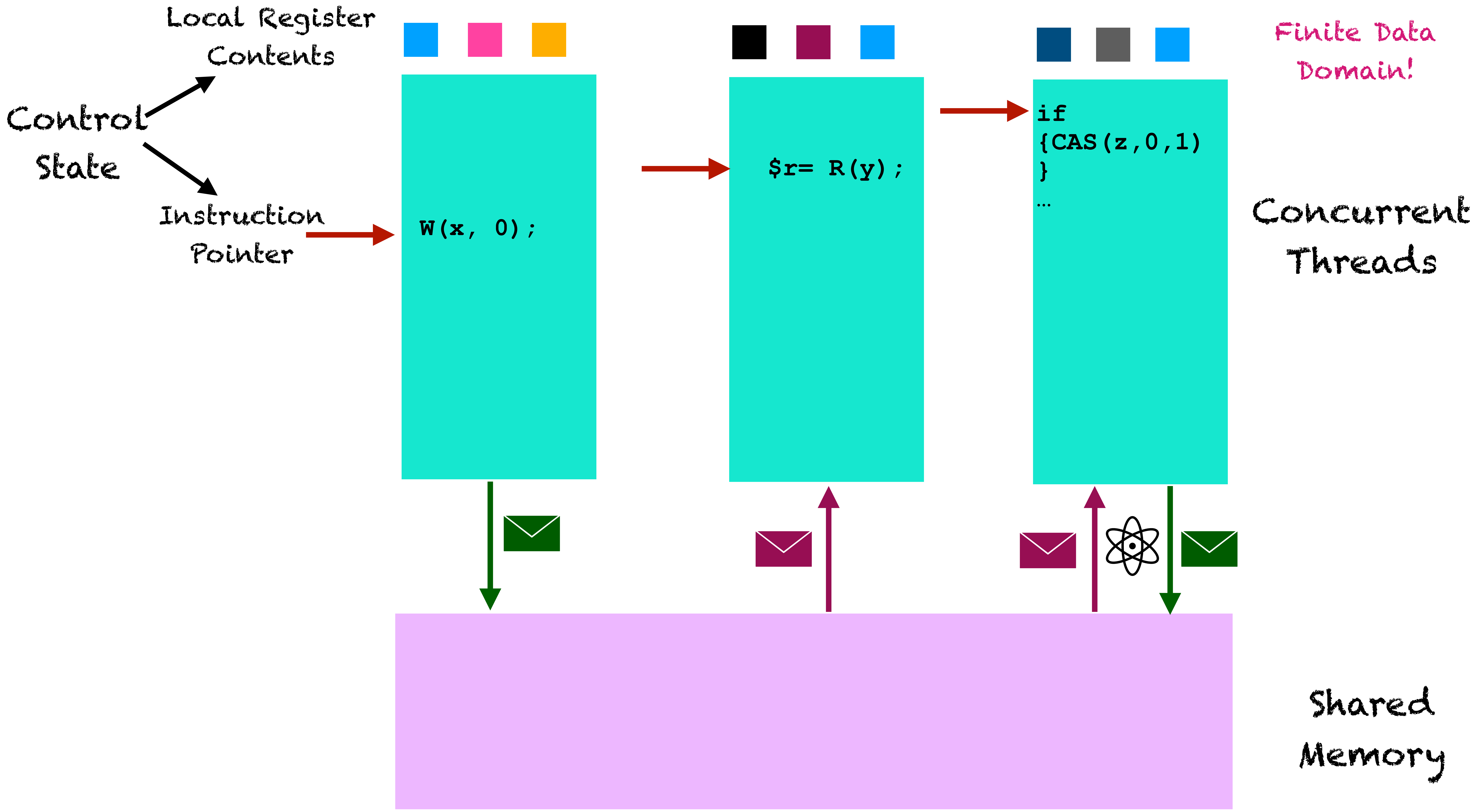
Algorithms: all at sea
Whole host of models



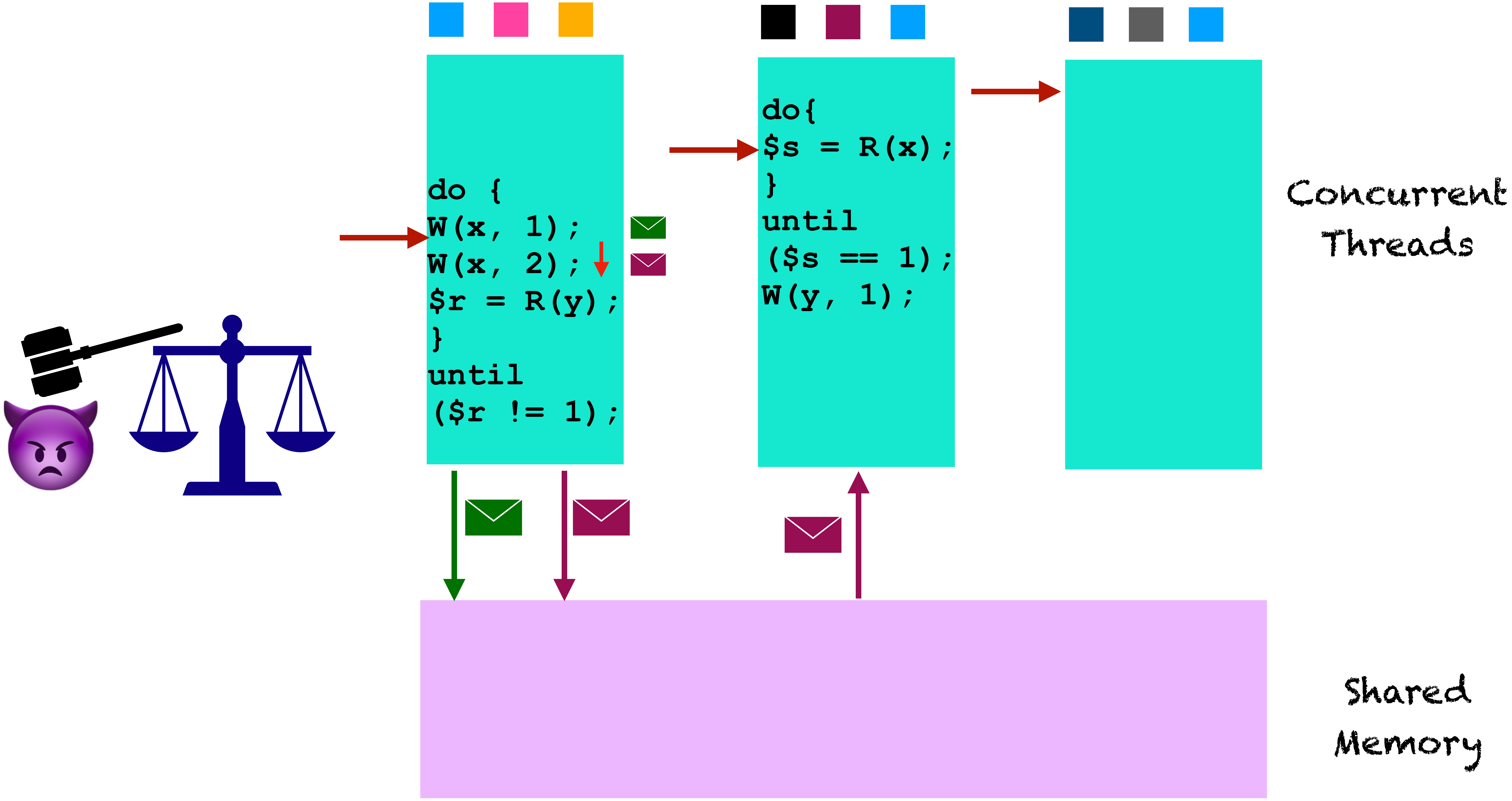
Liveness Verification:
- Program termination
- Repeated control state reachability


Memory Fairness,
rooted in practicality
Systematic verification of liveness in Weak Memory Models
Liveness, verified
Unified framework
Established techniques
Our connection

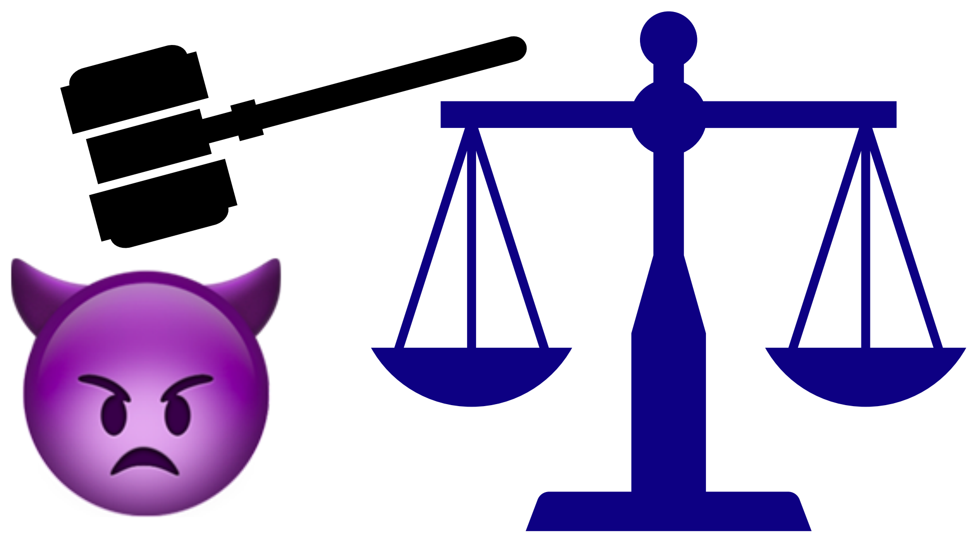
Concurrency and Memory: The Setup



Need for transition fairness

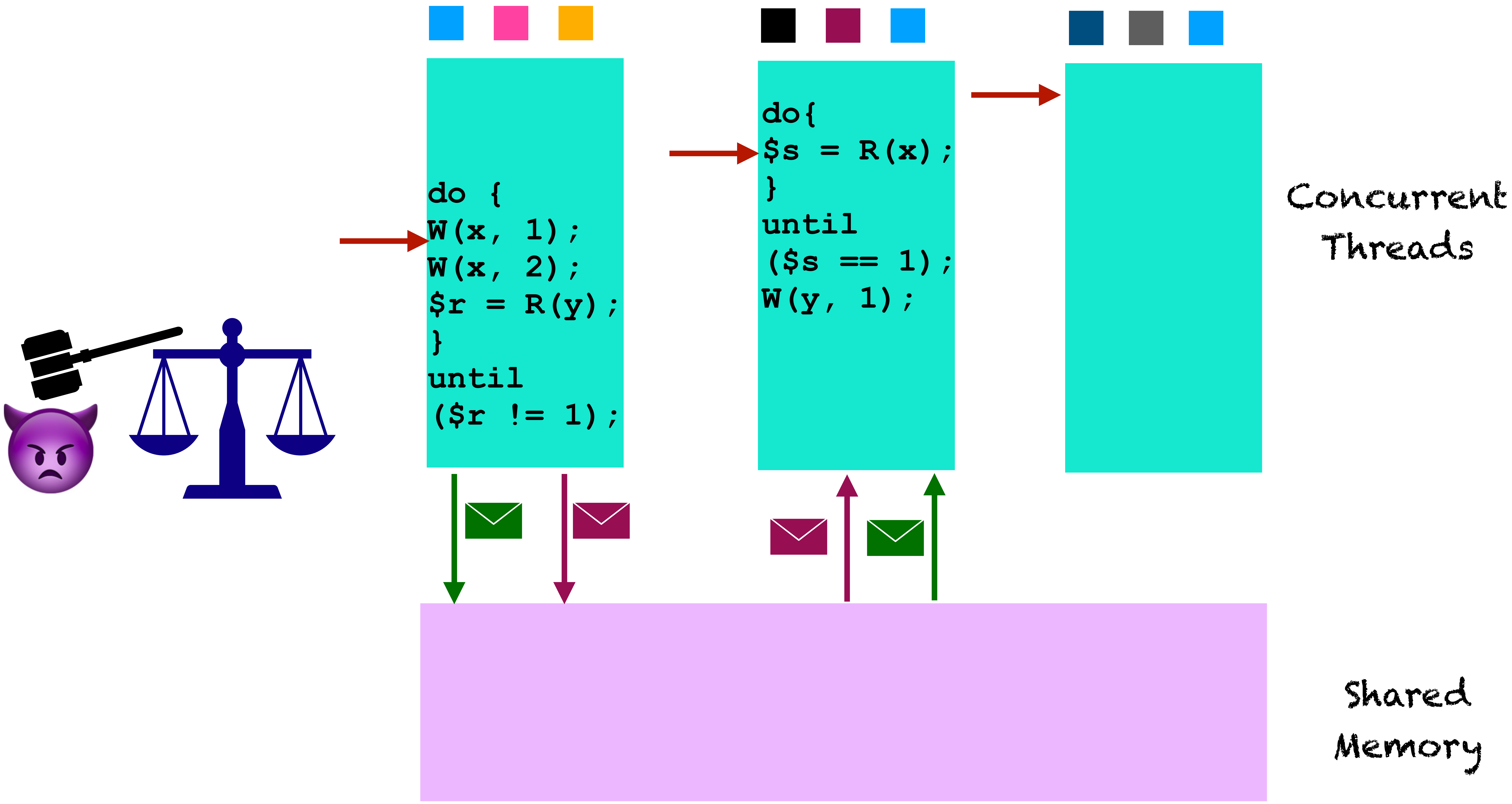


Transition fairness

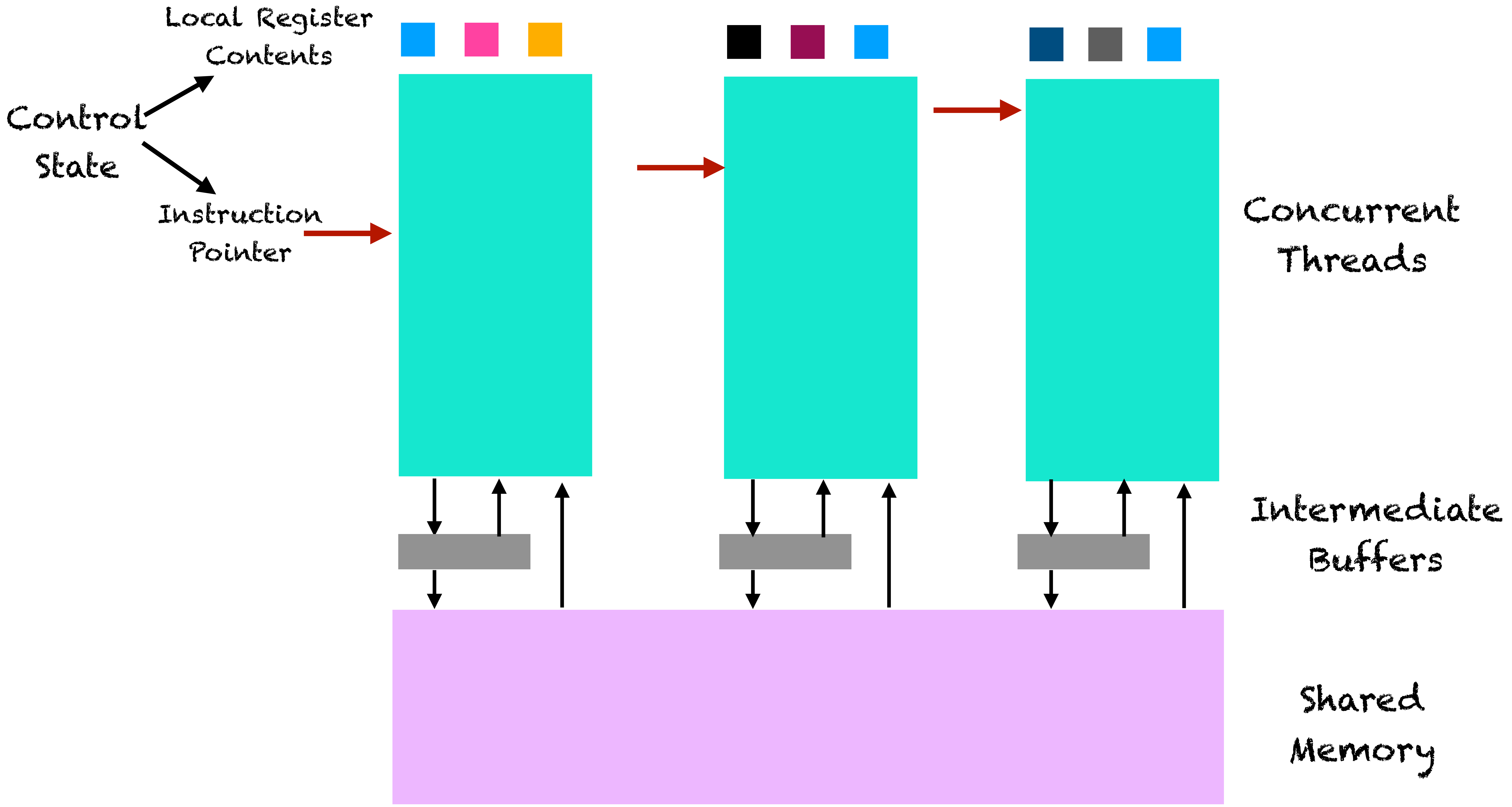


If a configuration c is visited infinitely often, then every transition (c, c’) that is enabled from c is taken infinitely often.

The resolution



Reality is more complex: example

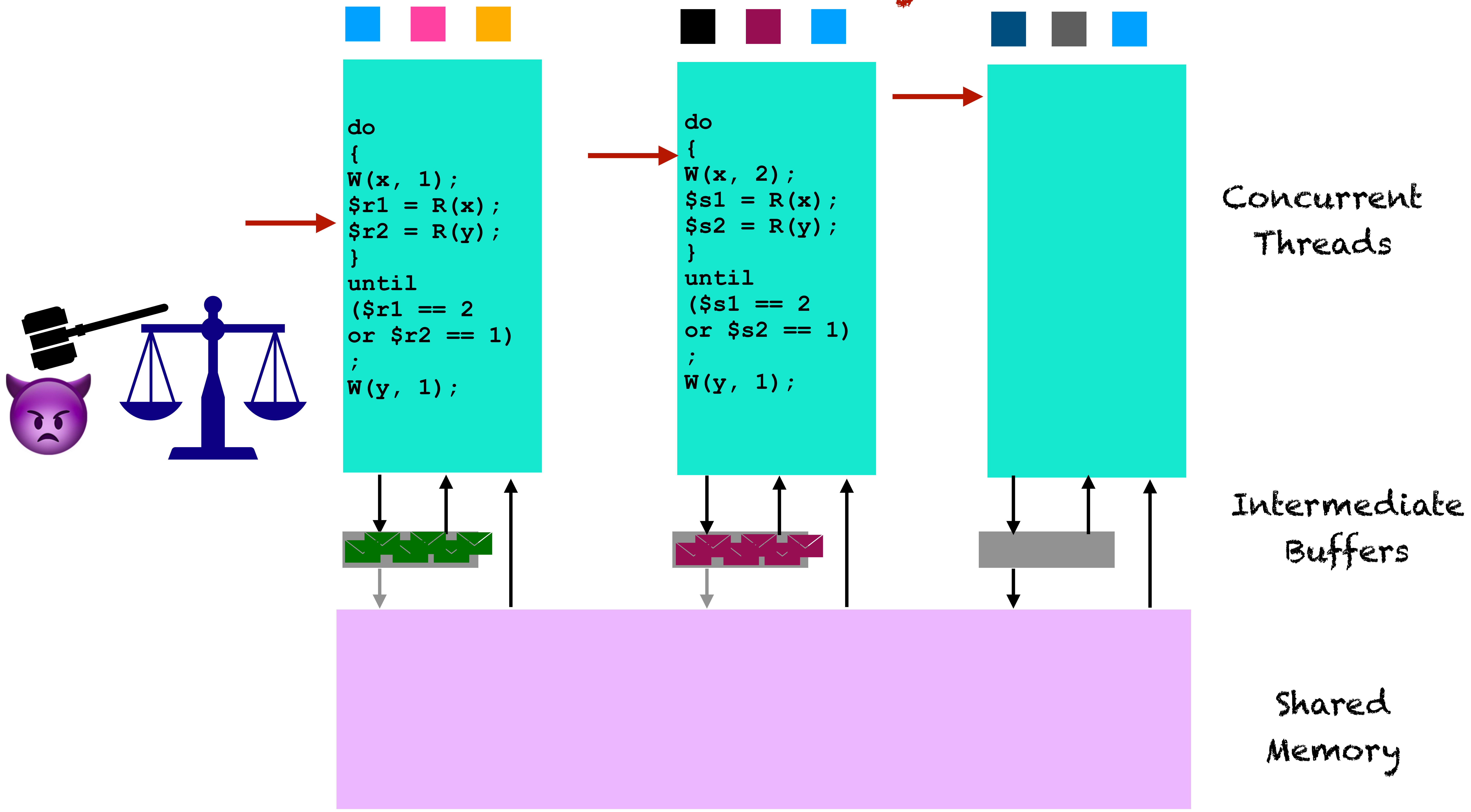


Transition fairness falls short

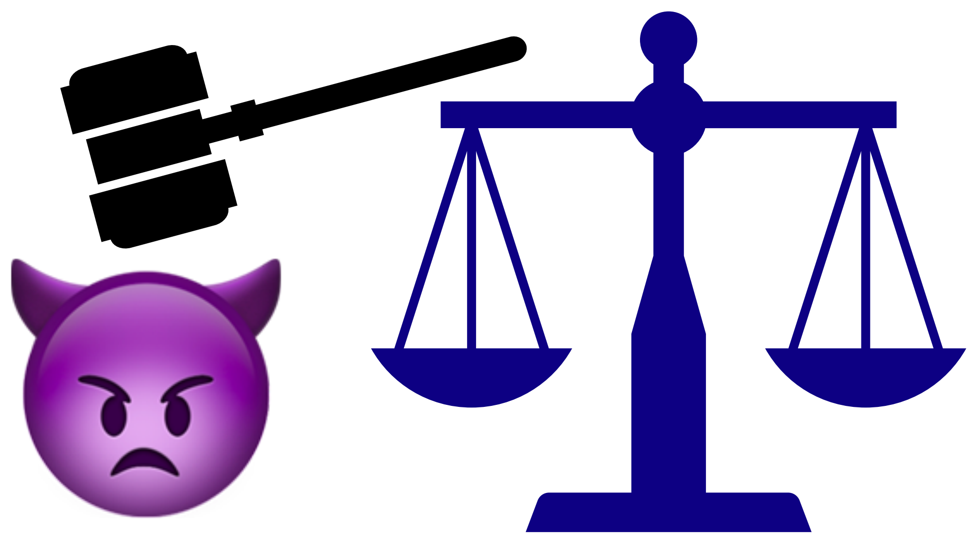
If a configuration c is visited infinitely often,
then every transition (c, c') that is enabled from c
is taken infinitely often.

But what if there are infinitely many configurations?
An infinite run need not visit any configuration repeatedly!

Need for memory fairness

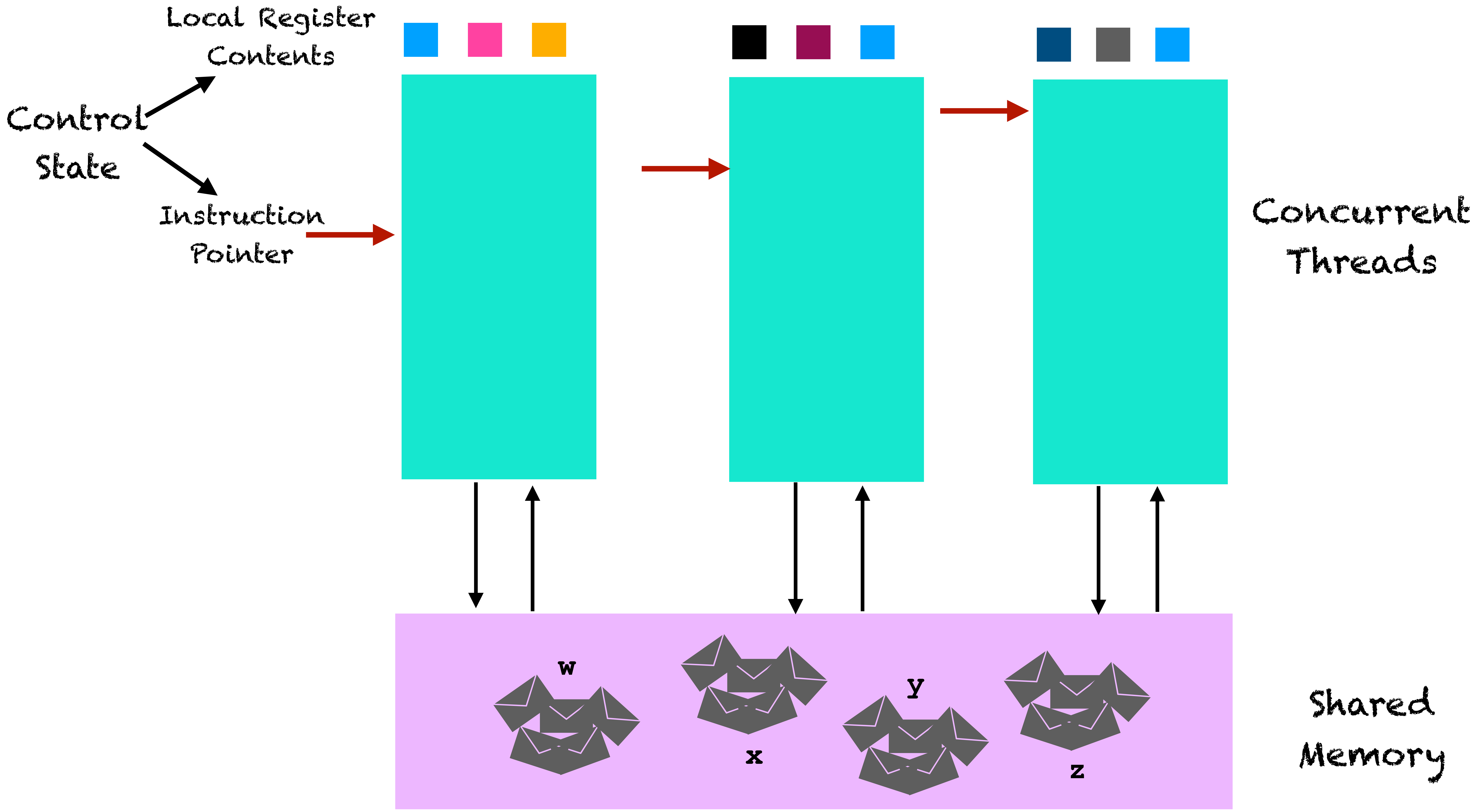


Memory fairness, informally



The “buffers” are flushed “regularly”.

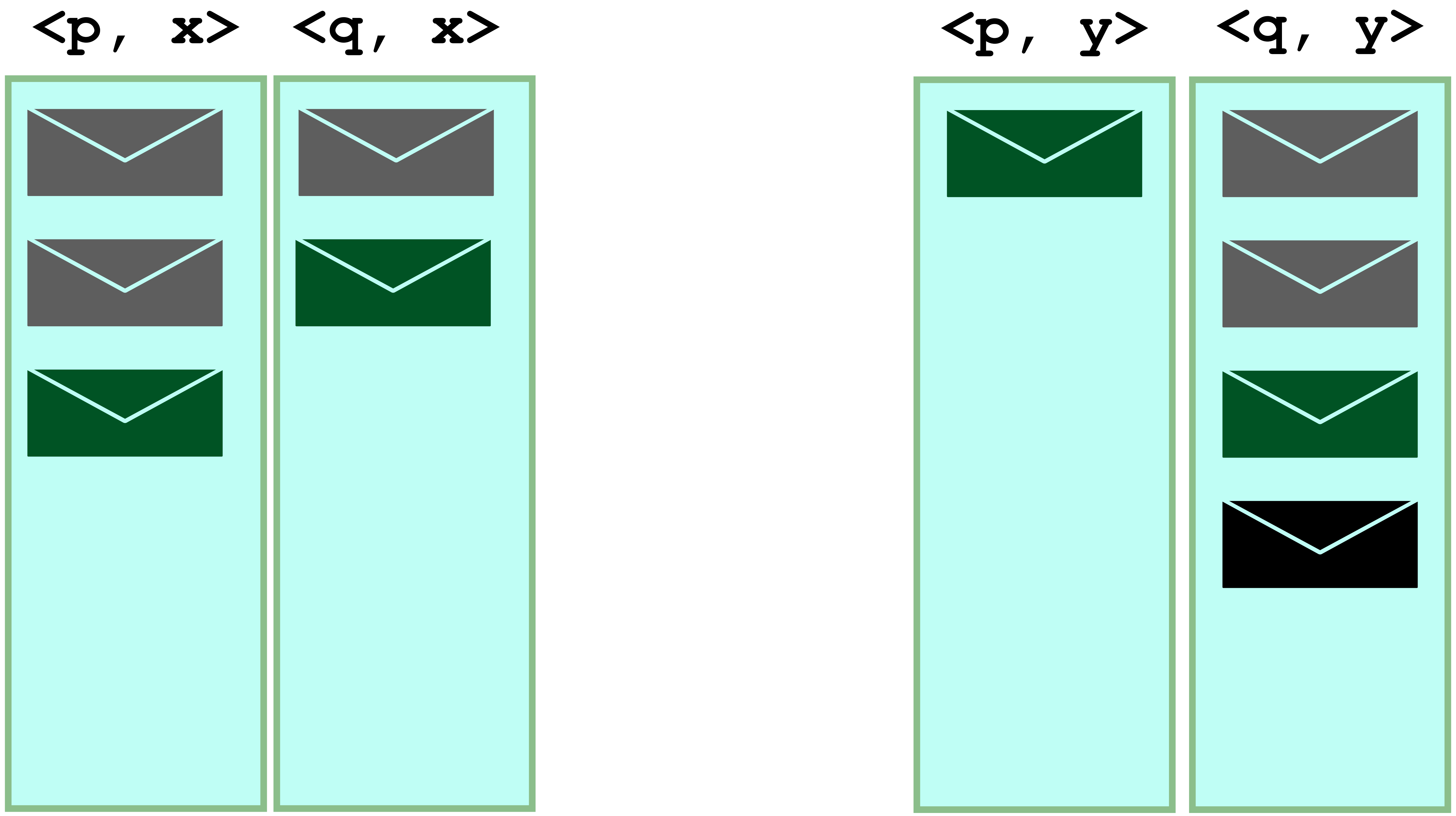
Weak memory: an abstract idea



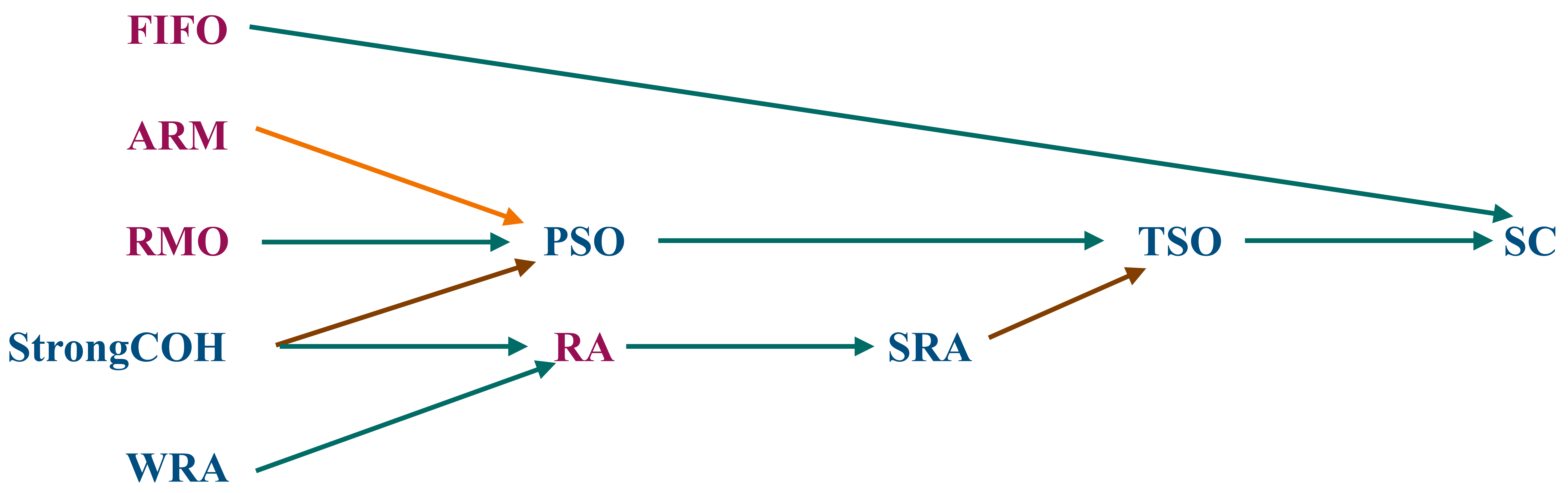
How does weak memory propagate messages?

Consider writes by thread p to variable x.

They will always be observed in the same order in which they were made!



Relative Strength of Memory Models: An arrow from A to B denotes that all behaviours of B are allowed by A.



Blue denotes that the underlying reachability is decidable, purple denotes it is undecidable.

Turquoise arrows indicate that relative strength follows from **design**.
The **orange** arrow indicates the enforcement of **acquire semantics** on reads.
Brown arrows indicate the enforcement of **multi copy atomicity** on the memory model.

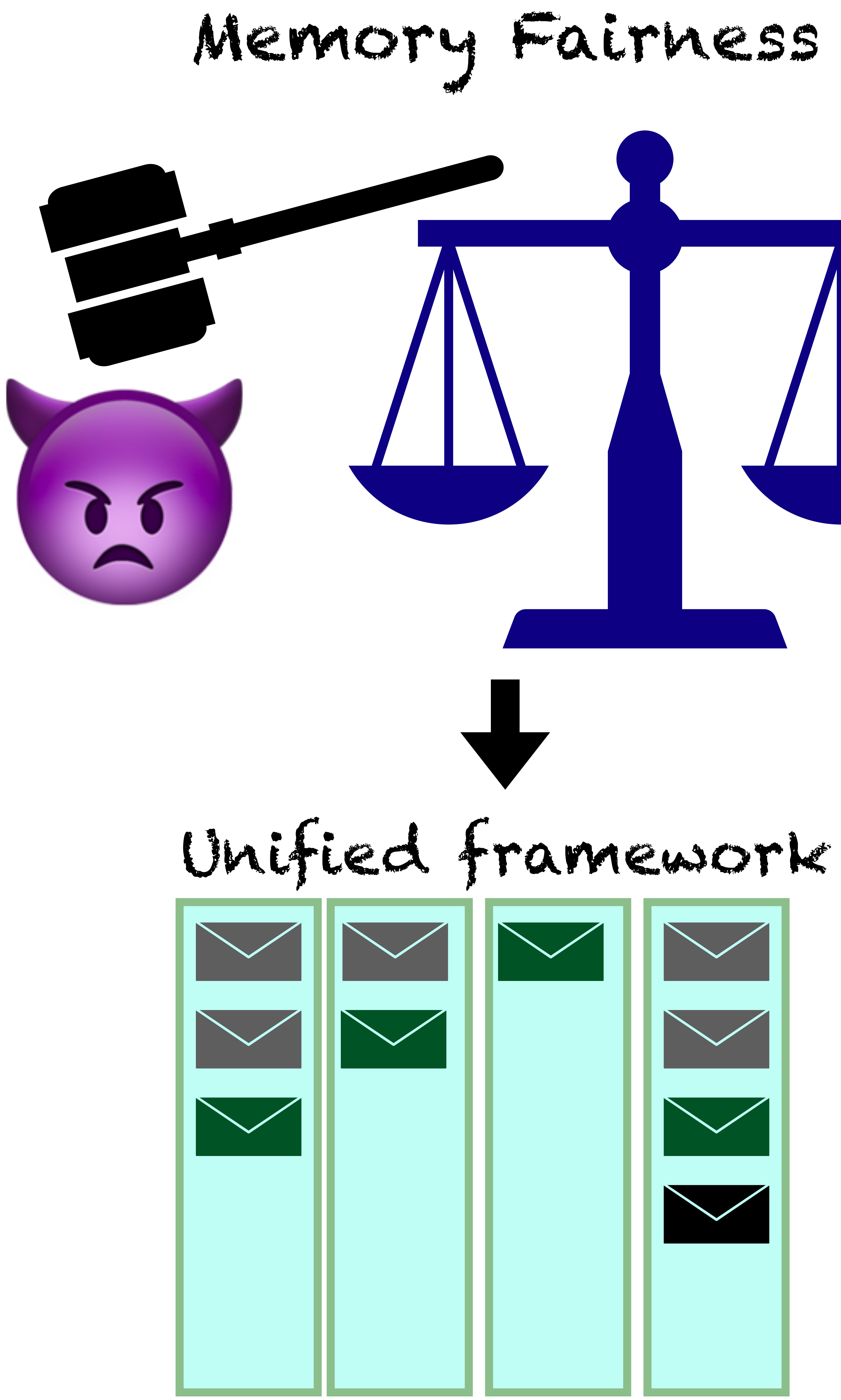
Configuration Size

Constraints imposed by the memory model make messages redundant as the run progresses

We only keep track of messages that are not redundant!

The number of messages stored in a memory configuration is called its size.

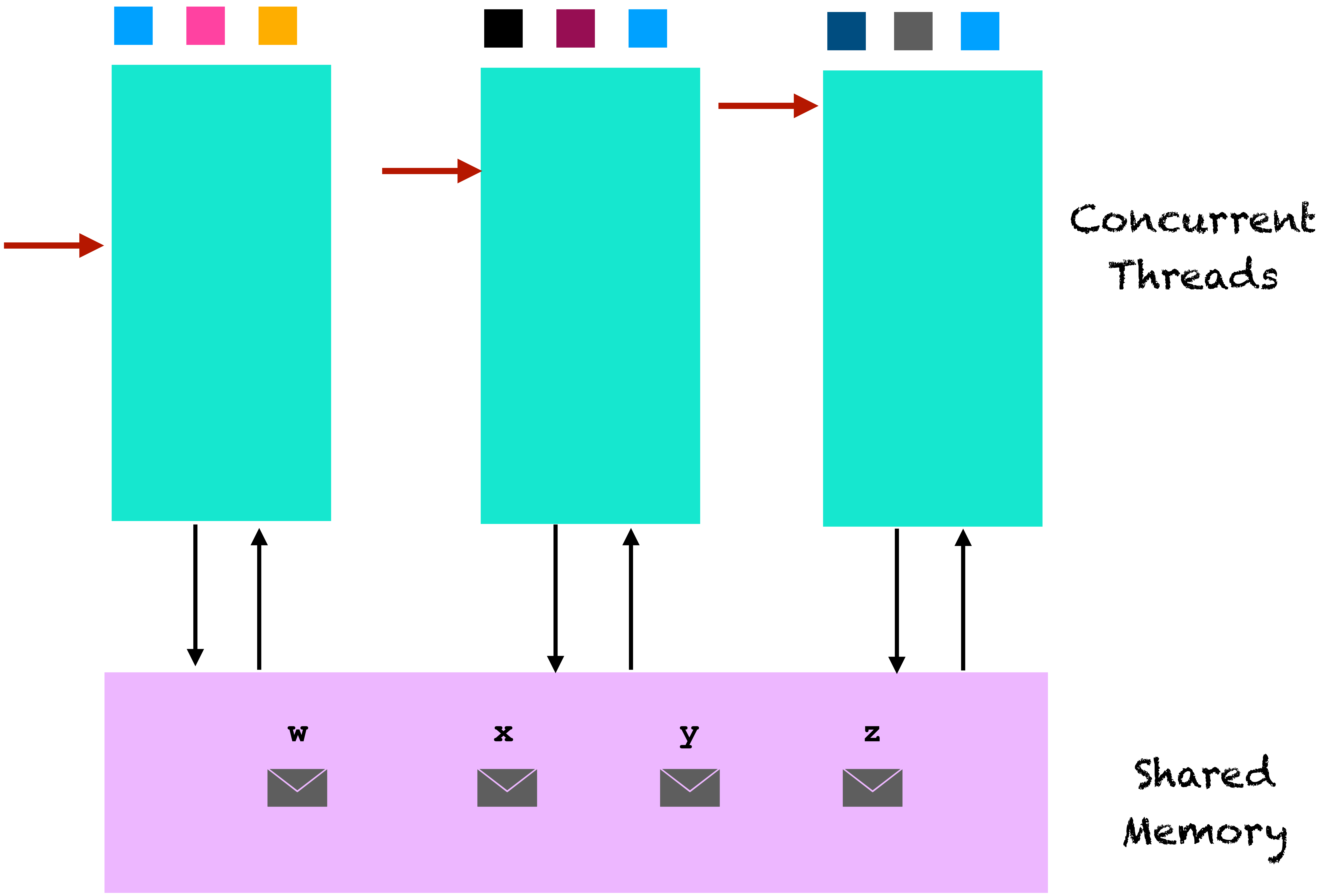
Size Bounded Executions



An execution is called size bounded if there exists an N such that each configuration is of size at most N.

If N is specified, we refer to the execution as N-bounded.

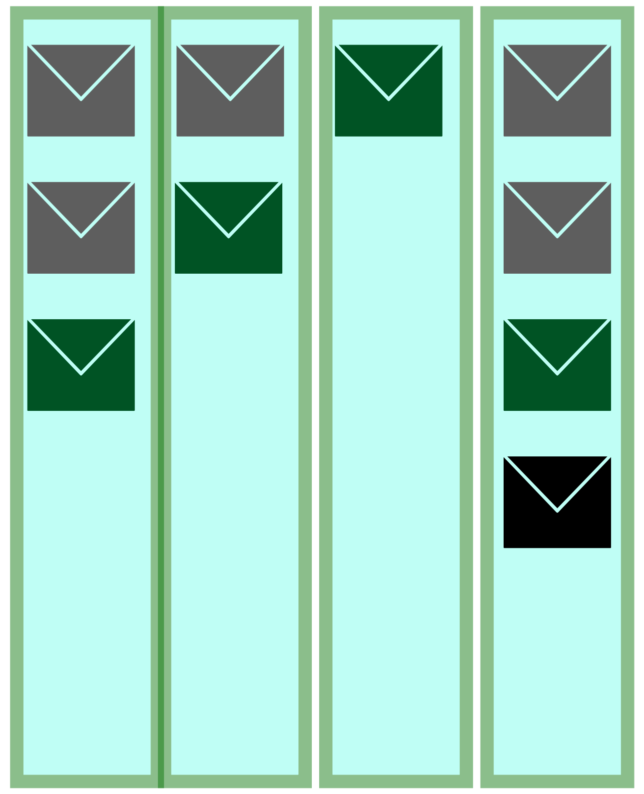
Configuration size and weakness



An SC configuration.

No weak behaviour. Note its minimal size, i.e. one message per variable

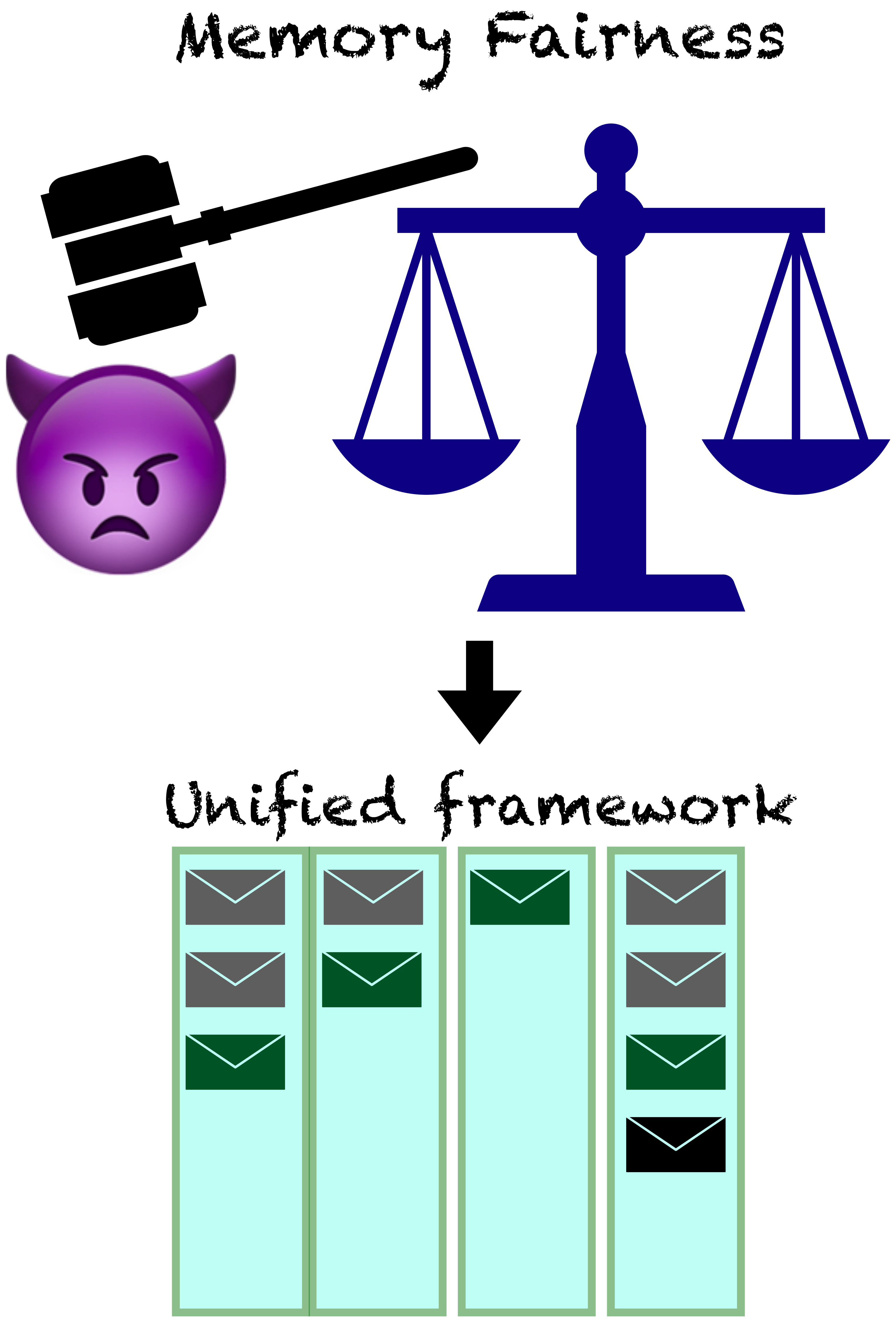
Plain Configurations



Configurations with exactly one message per variable are called plain.

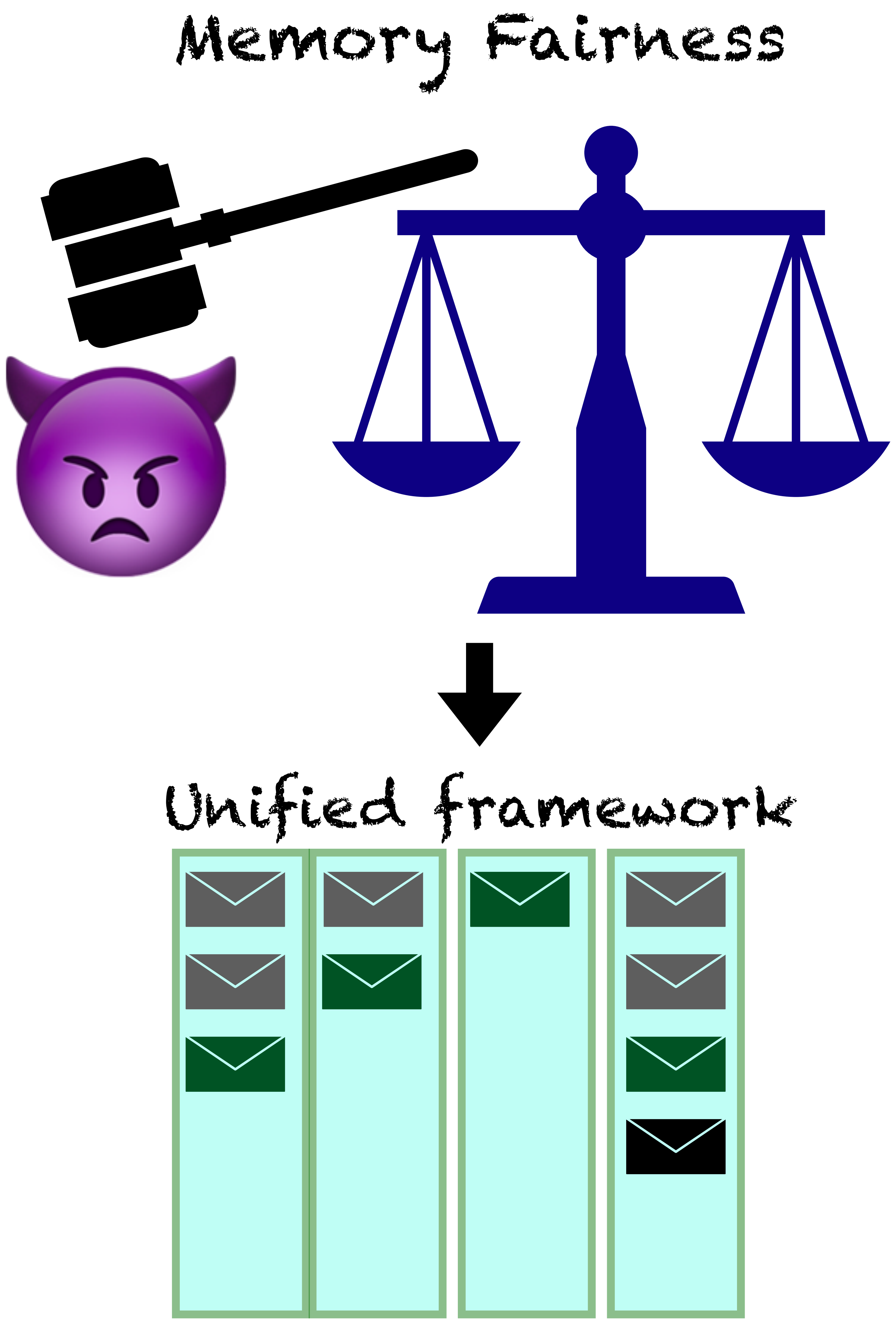
There are finitely many plain configurations

Repeatedly Plain Executions



An infinite execution is repeatedly plain if plain configurations occur infinitely often.

Transition + Memory Fairness, Formally



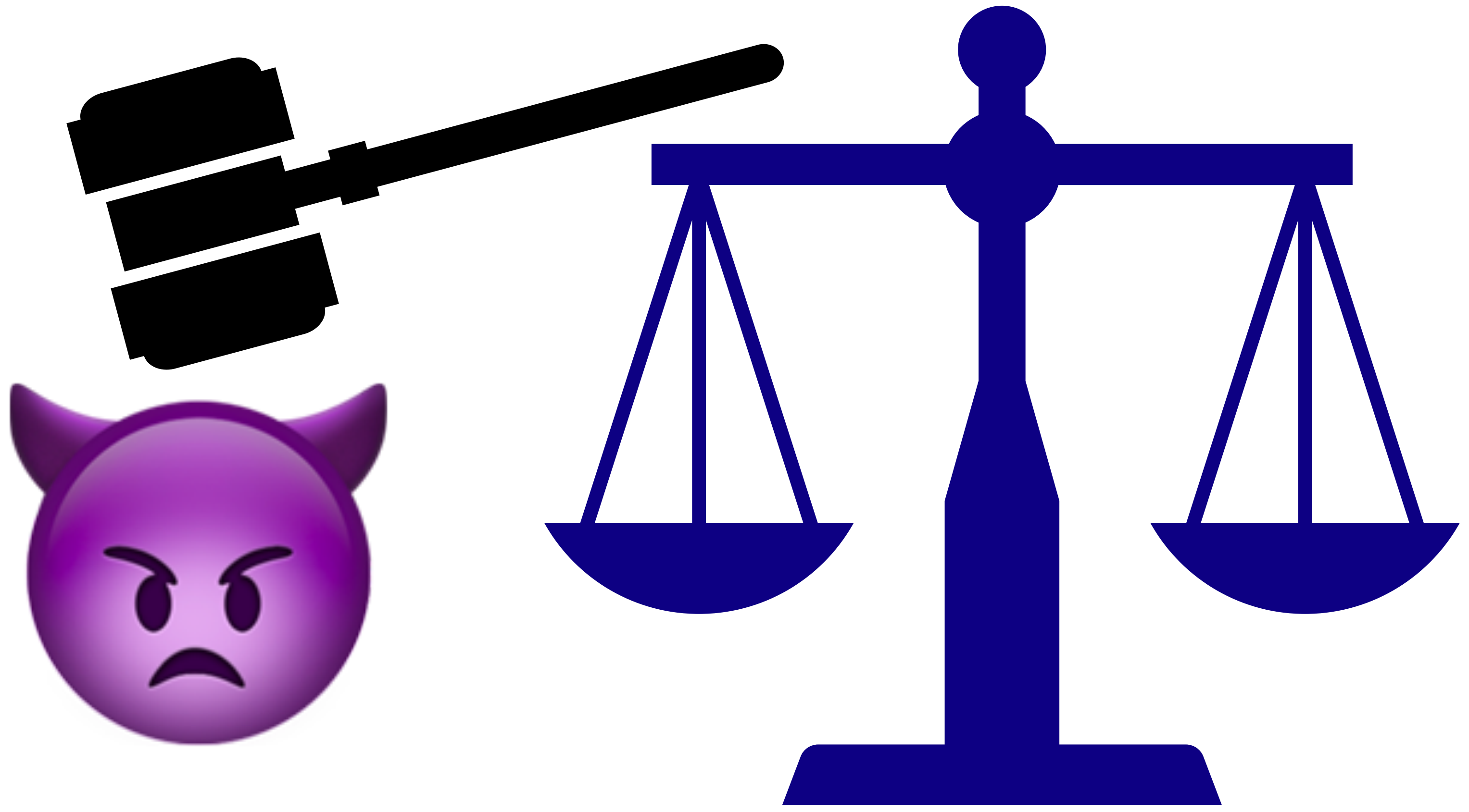
We have the following fairness conditions on infinite executions

- N-bounded transition fairness

- Repeatedly plain transition fairness

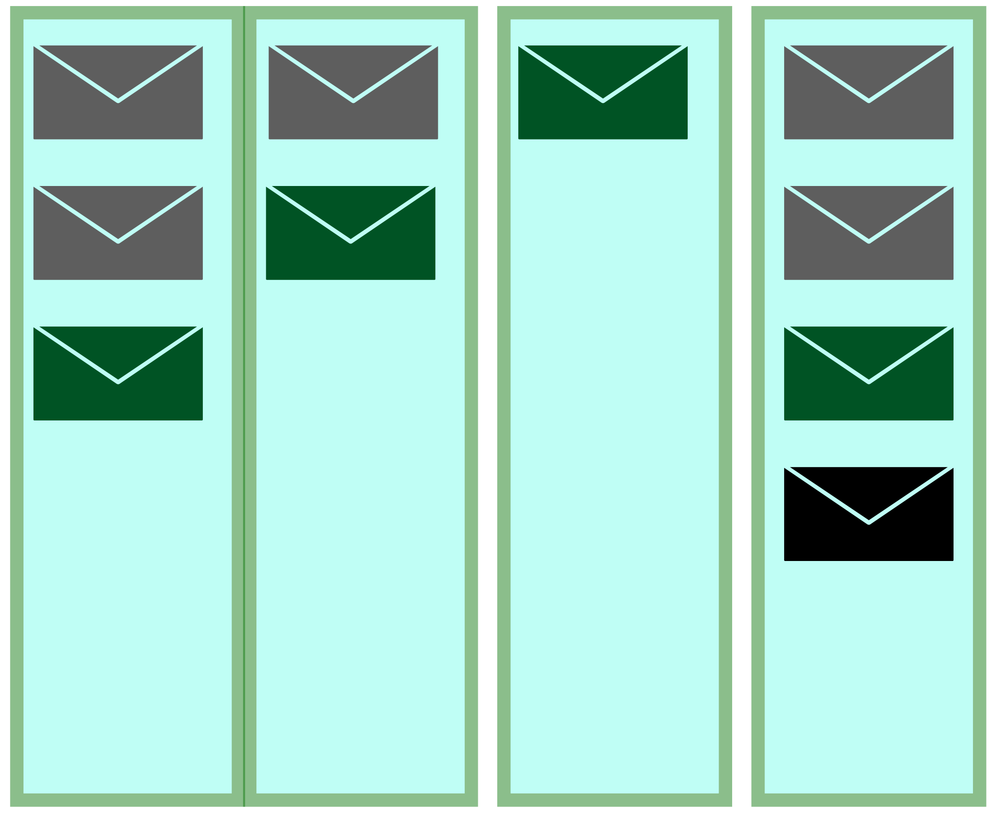
A Probabilistic Analog

Memory Fairness



A Markov chain induced by the system satisfies Probabilistic Memory Fairness if the set of plain configurations is visited infinitely often with probability 1.

Unified framework



Such Markov Chains are "decisive" by dint of having the set of plain configurations as a "finite attractor"

Decisive Markov Chains are well studied

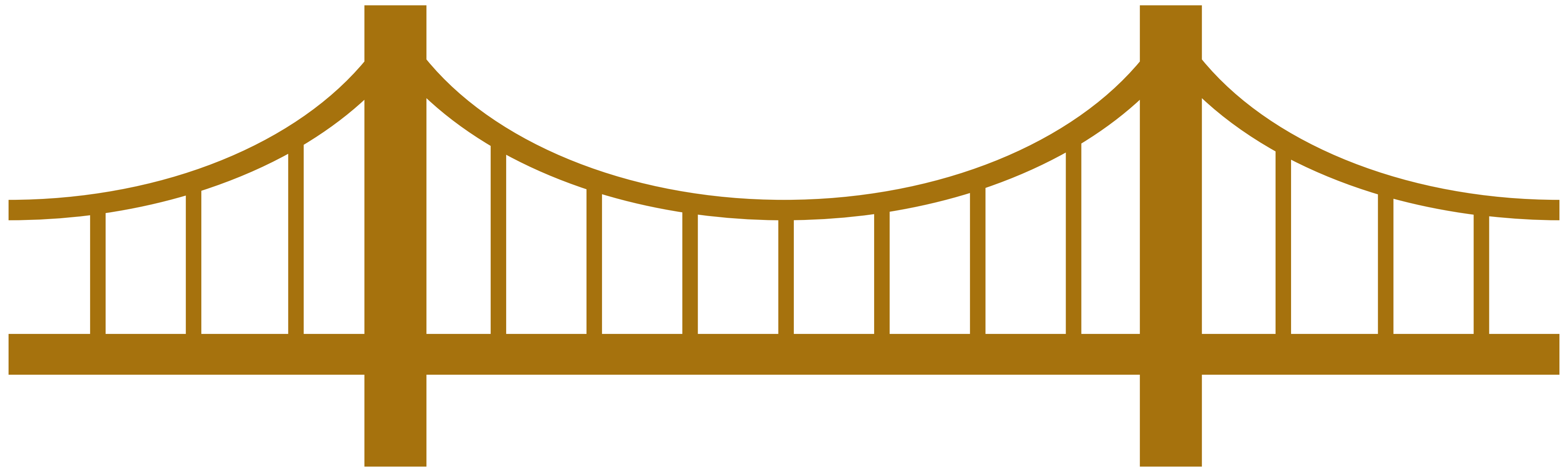
The connection

The following fairness conditions are equivalent for termination and repeated control state reachability

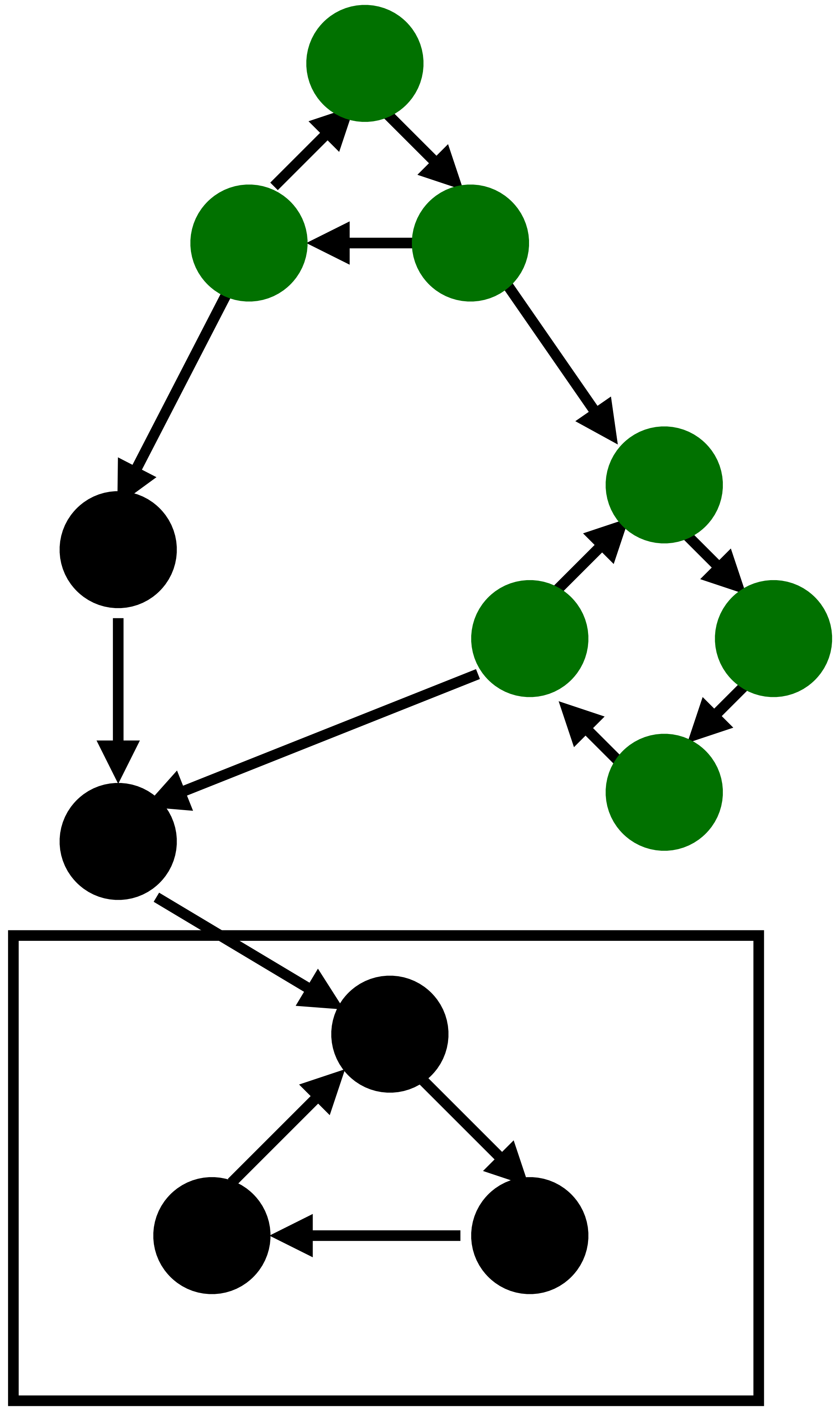
- Probabilisitic Memory Fairness

- N-bounded transition fairness for sufficiently large N

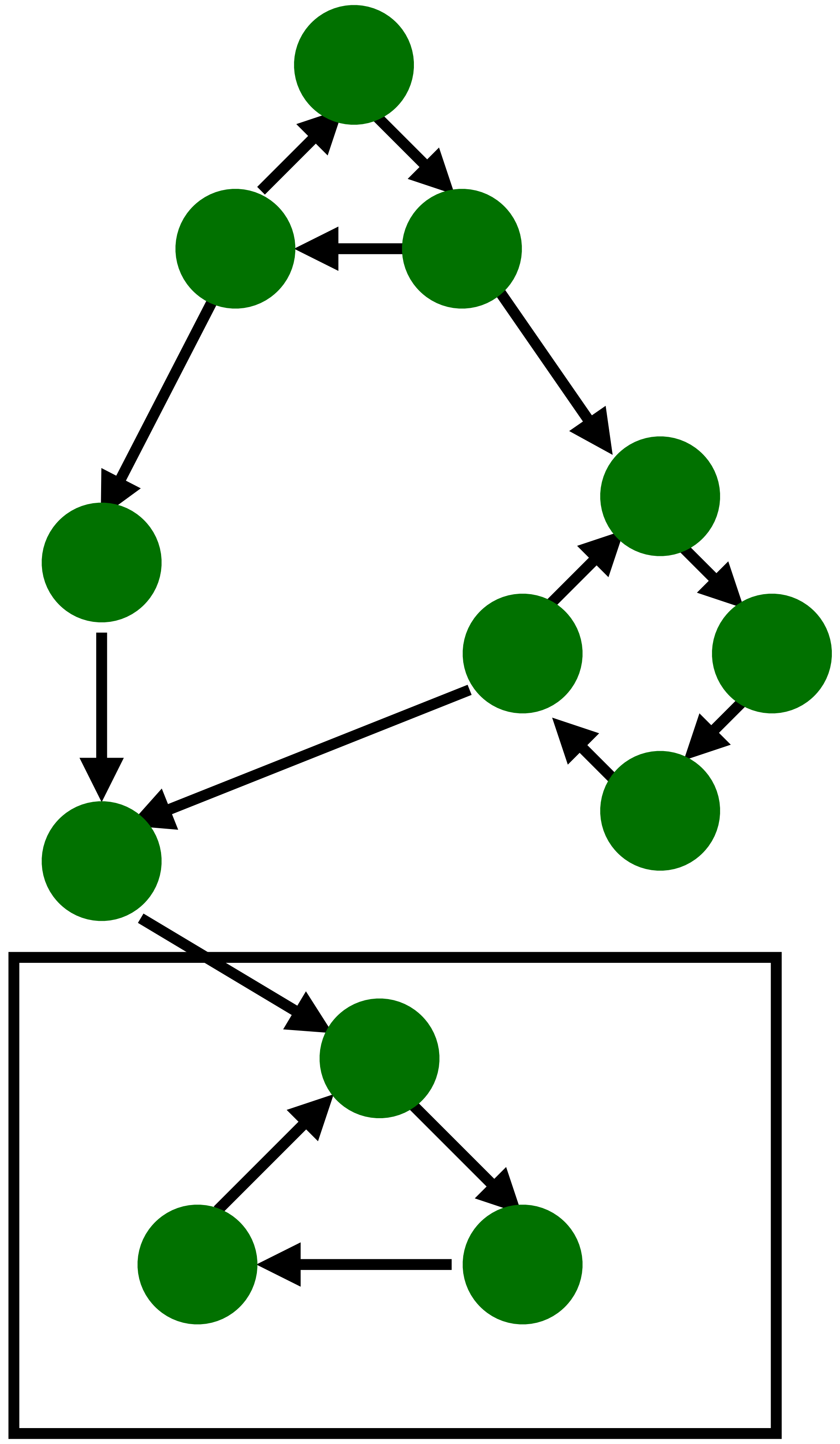
- Repeatedly plain transition fairness



Proof Sketch



Bad

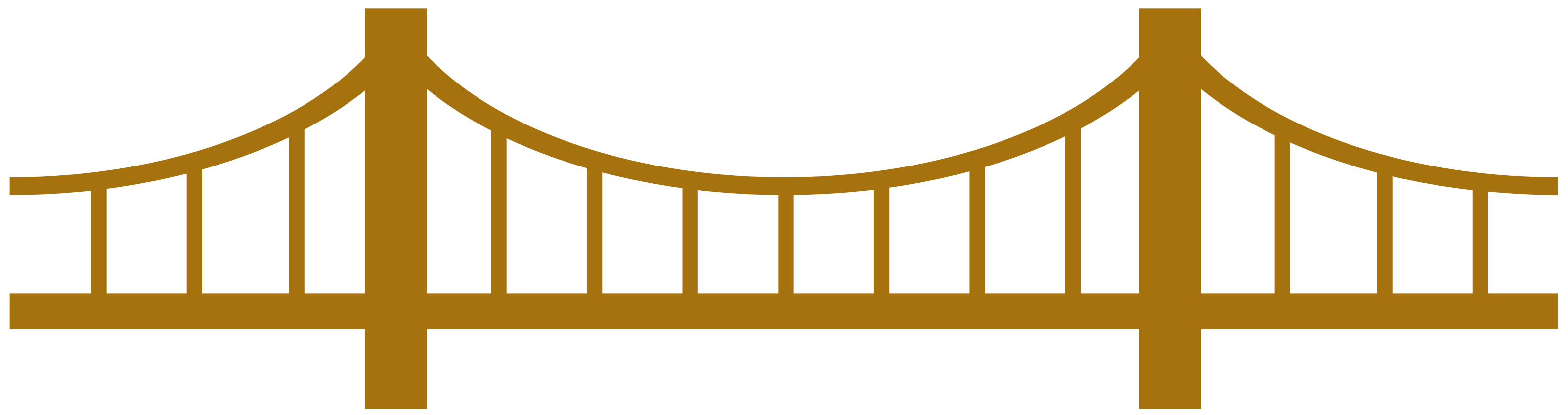


Good

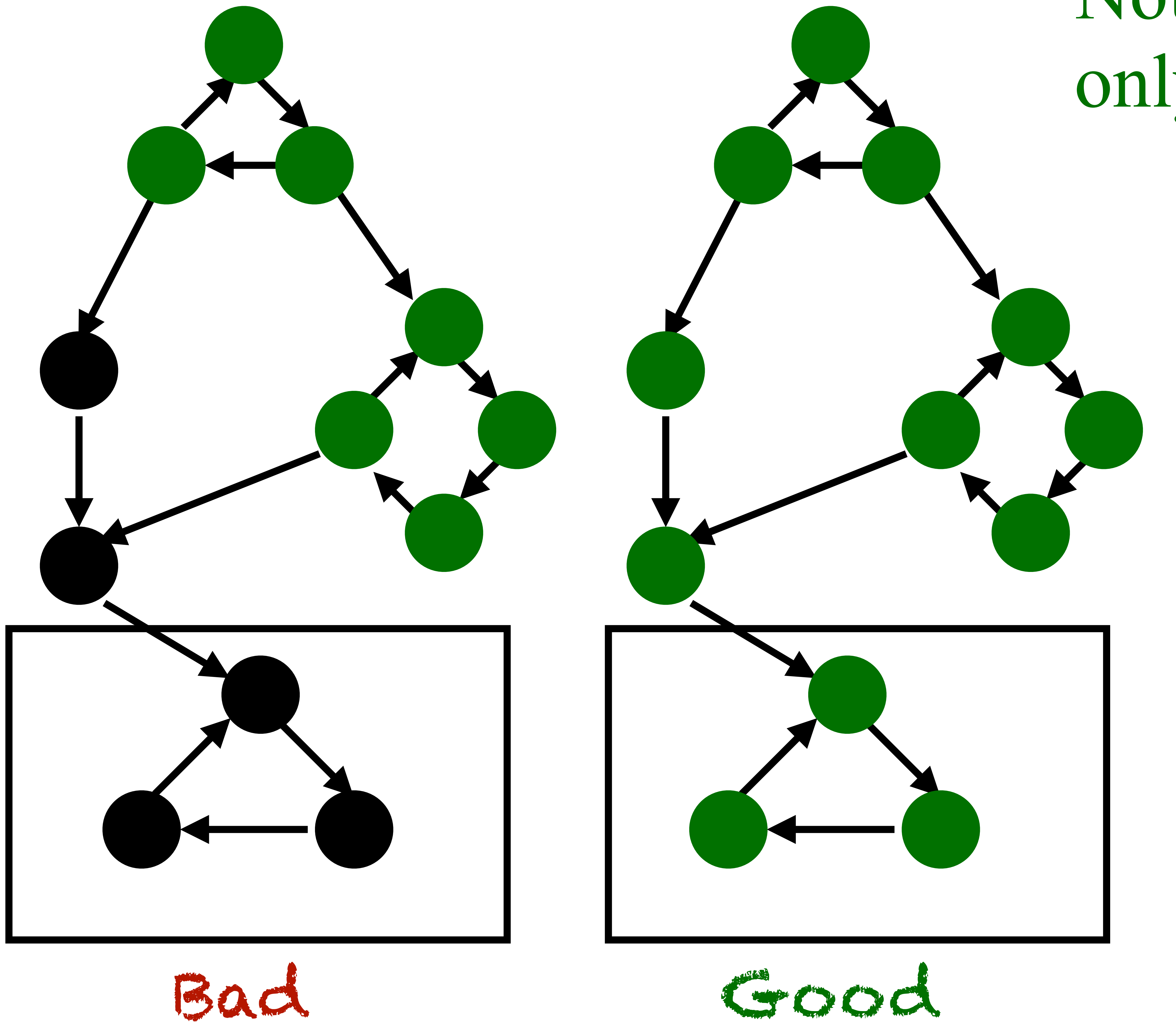
1) For each N, construct a graph G(N) with plain configurations as vertices

2) Draw an edge (γ, γ') if γ' is reachable from γ via configurations of size at most N

3) Paint a node green if the control state of interest is reachable via configurations of size at most N



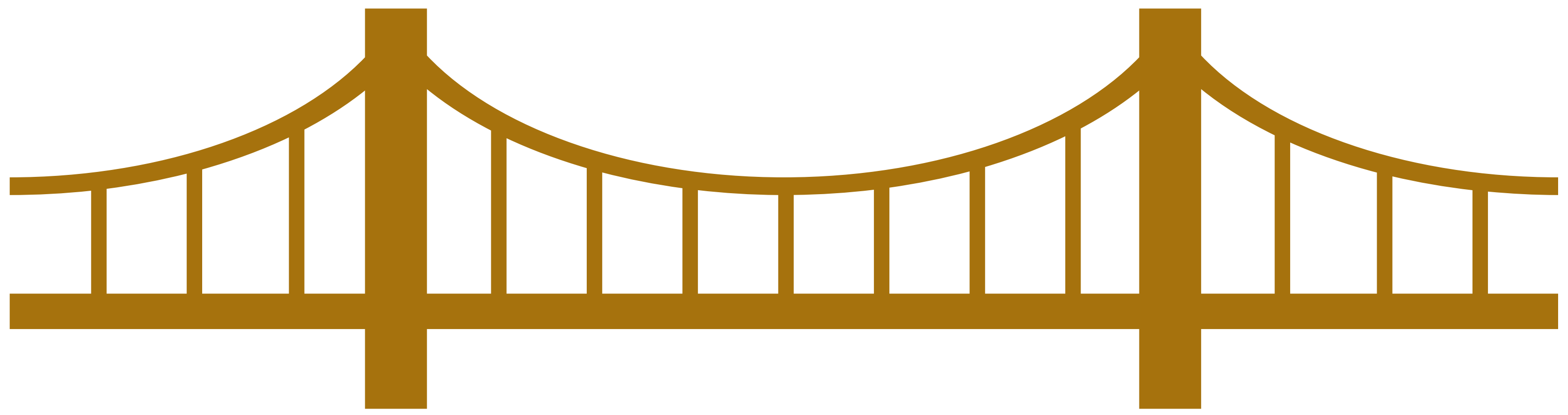
Proof Sketch



Notice, edges can only be added, and nodes can only go from black to green!

The finite graph saturates; let it be G for all sufficiently large N

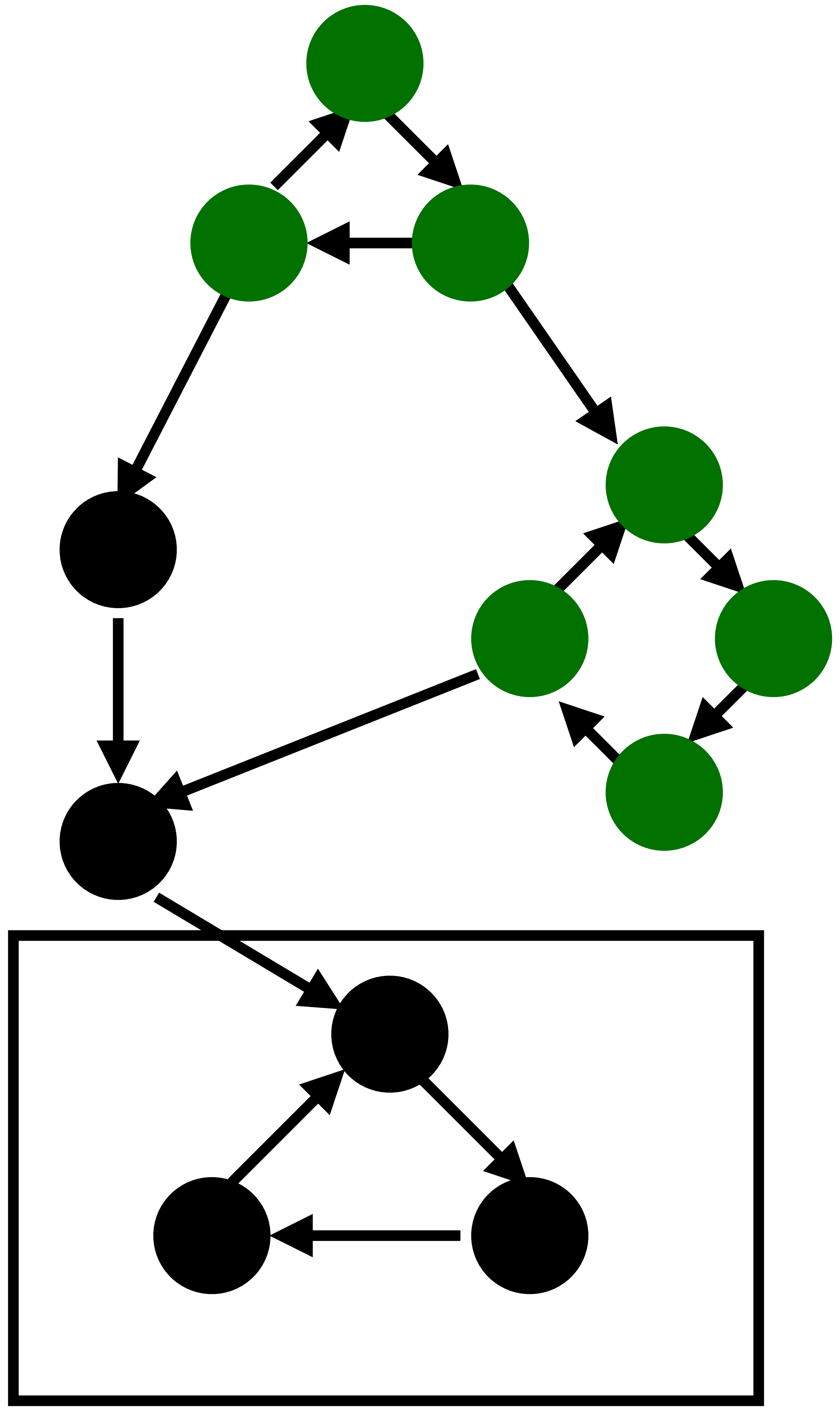
For all our fairness notions, liveness holds if and only if all bottom scc's of G are green



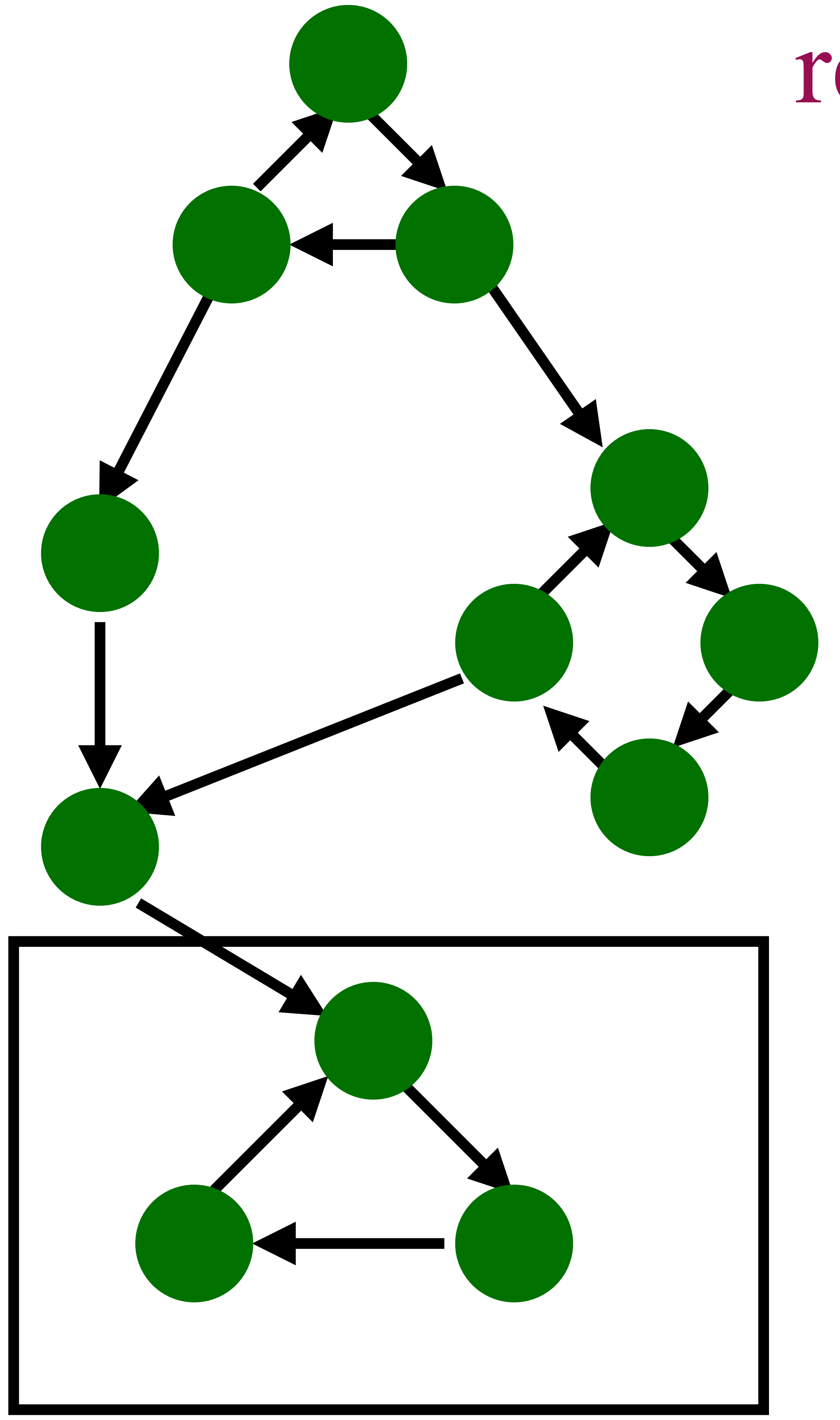
Liveness, Verified

All that remains is to construct G using reachability queries

This can be done by translating our framework into those used for verifying safety

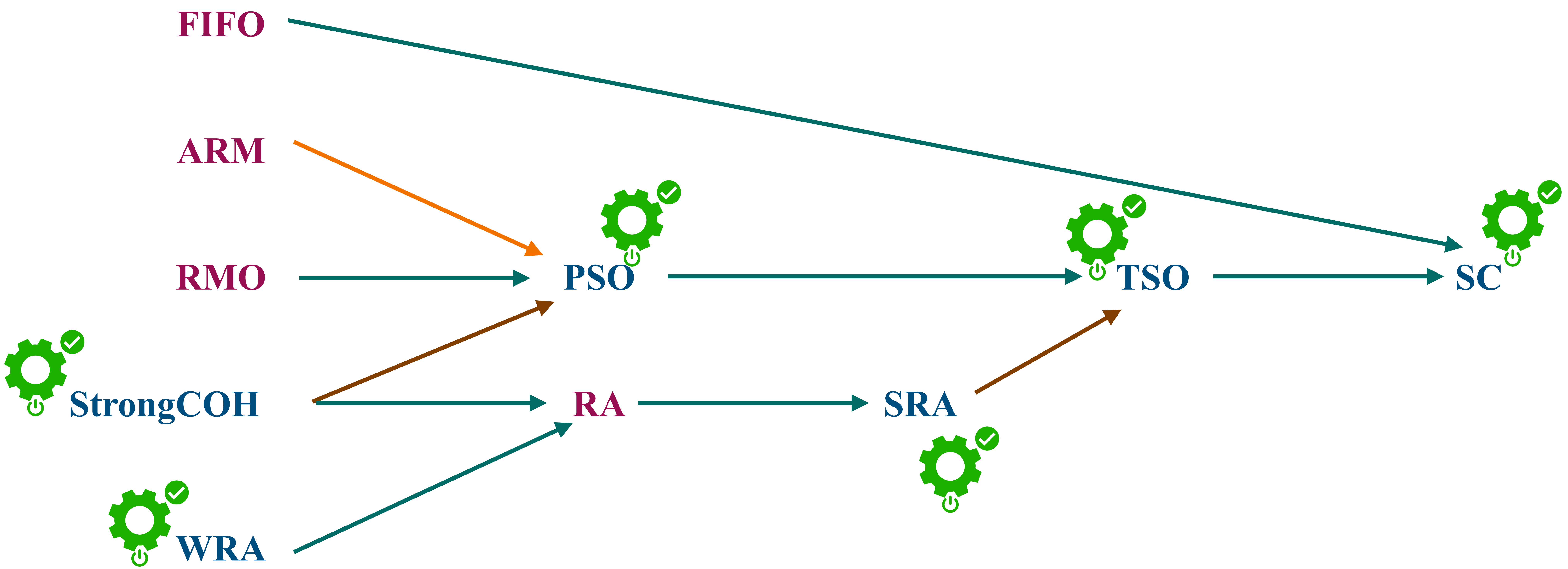


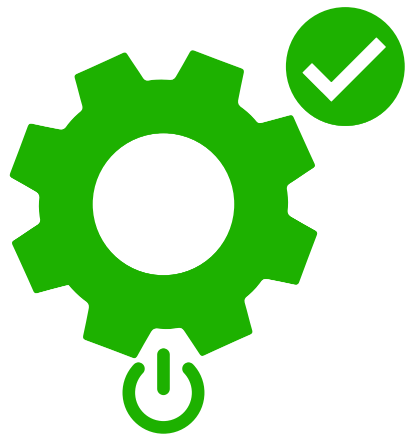
Bad



Good

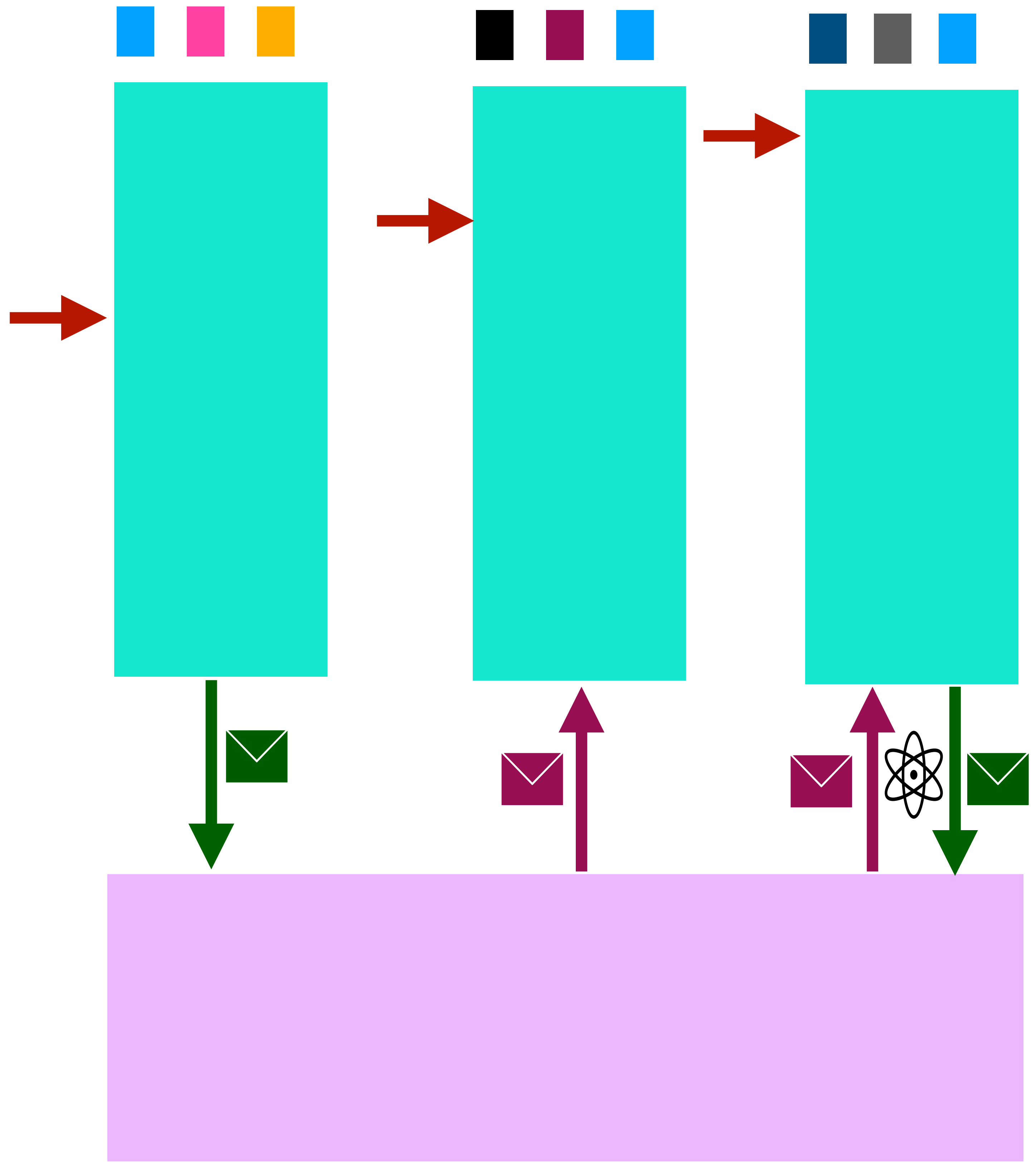
Verifying concrete models



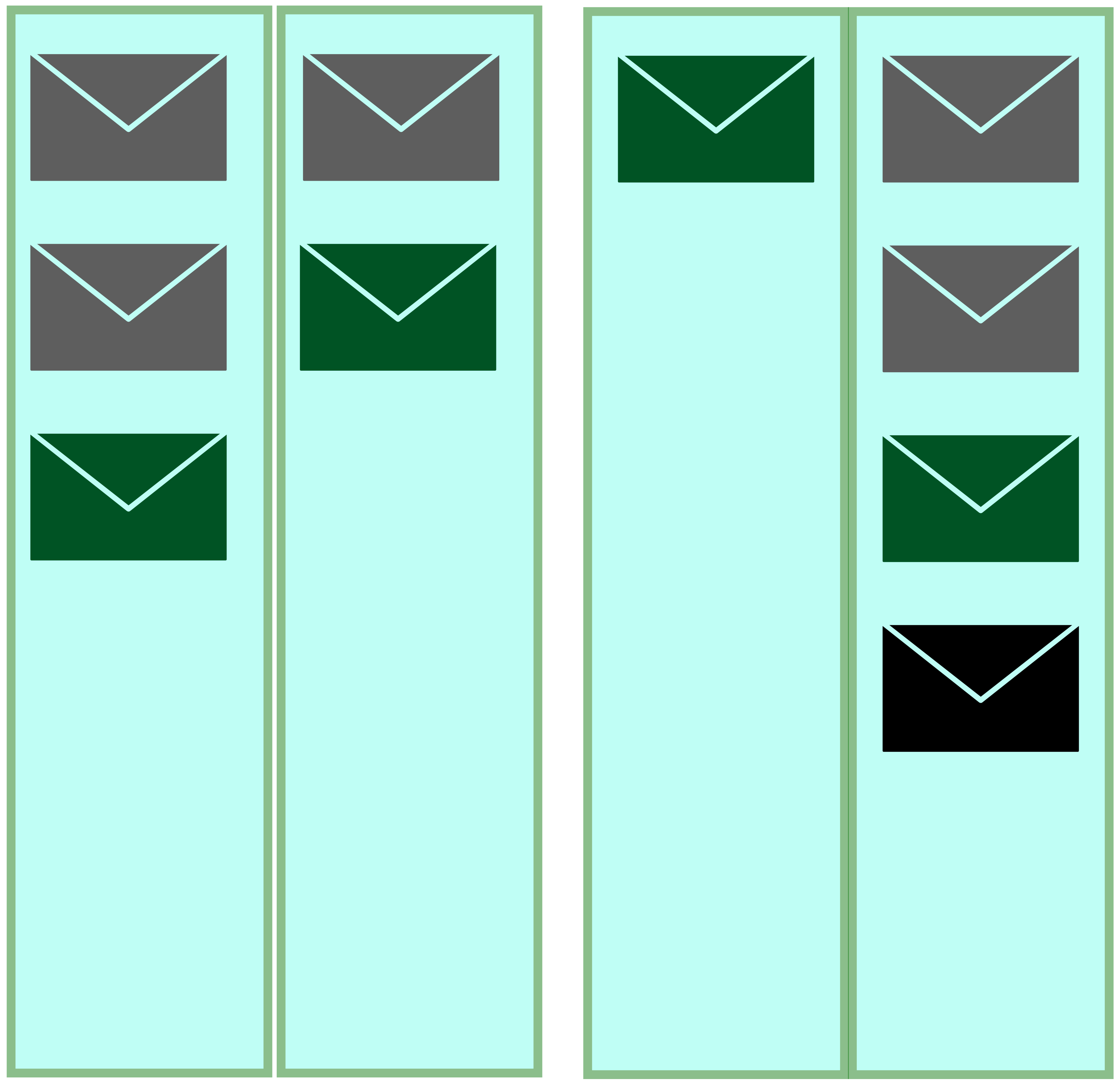


Reachability queries result in liveness decision procedures

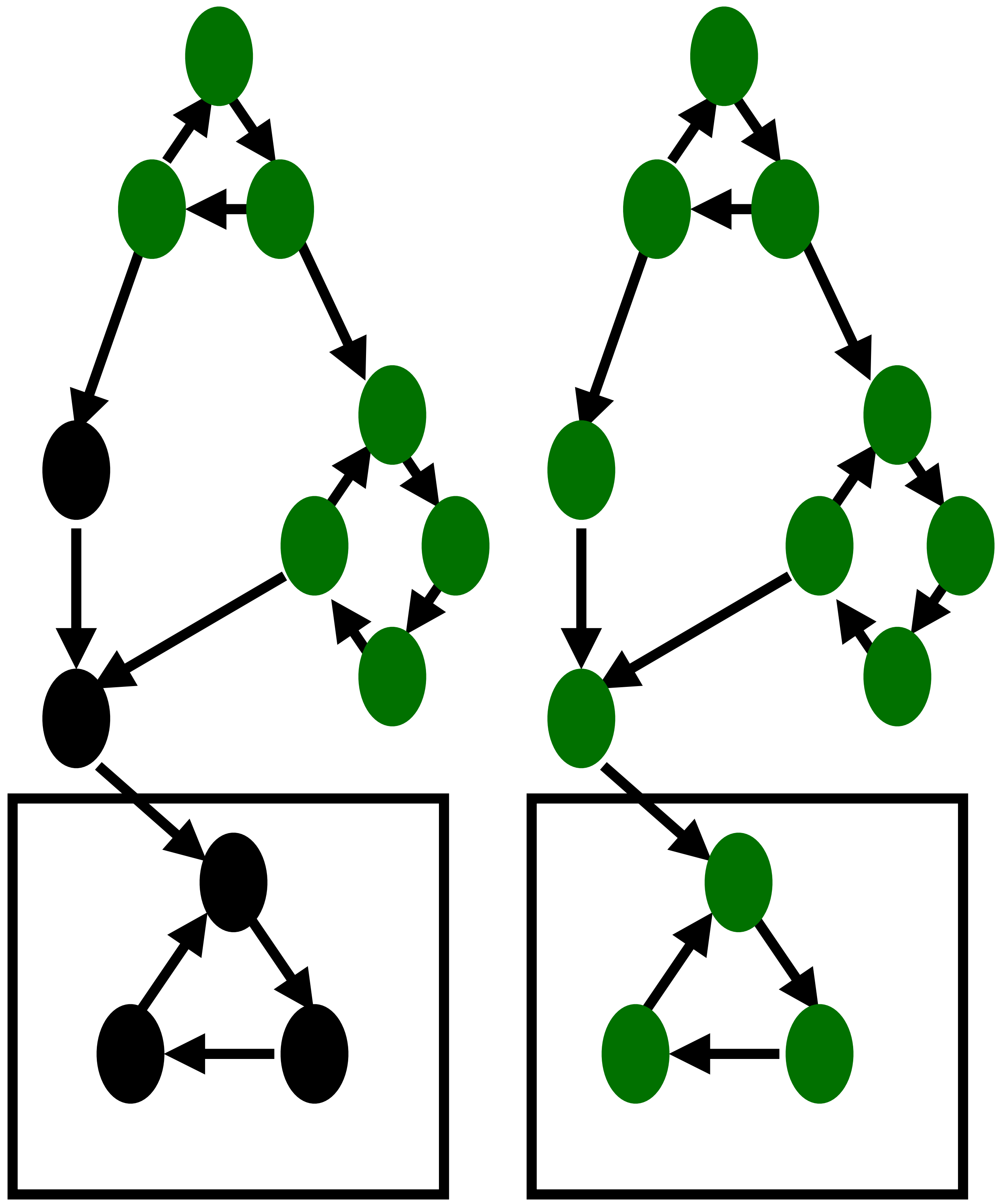
Thank You!



The Setup



The Model



The Procedure