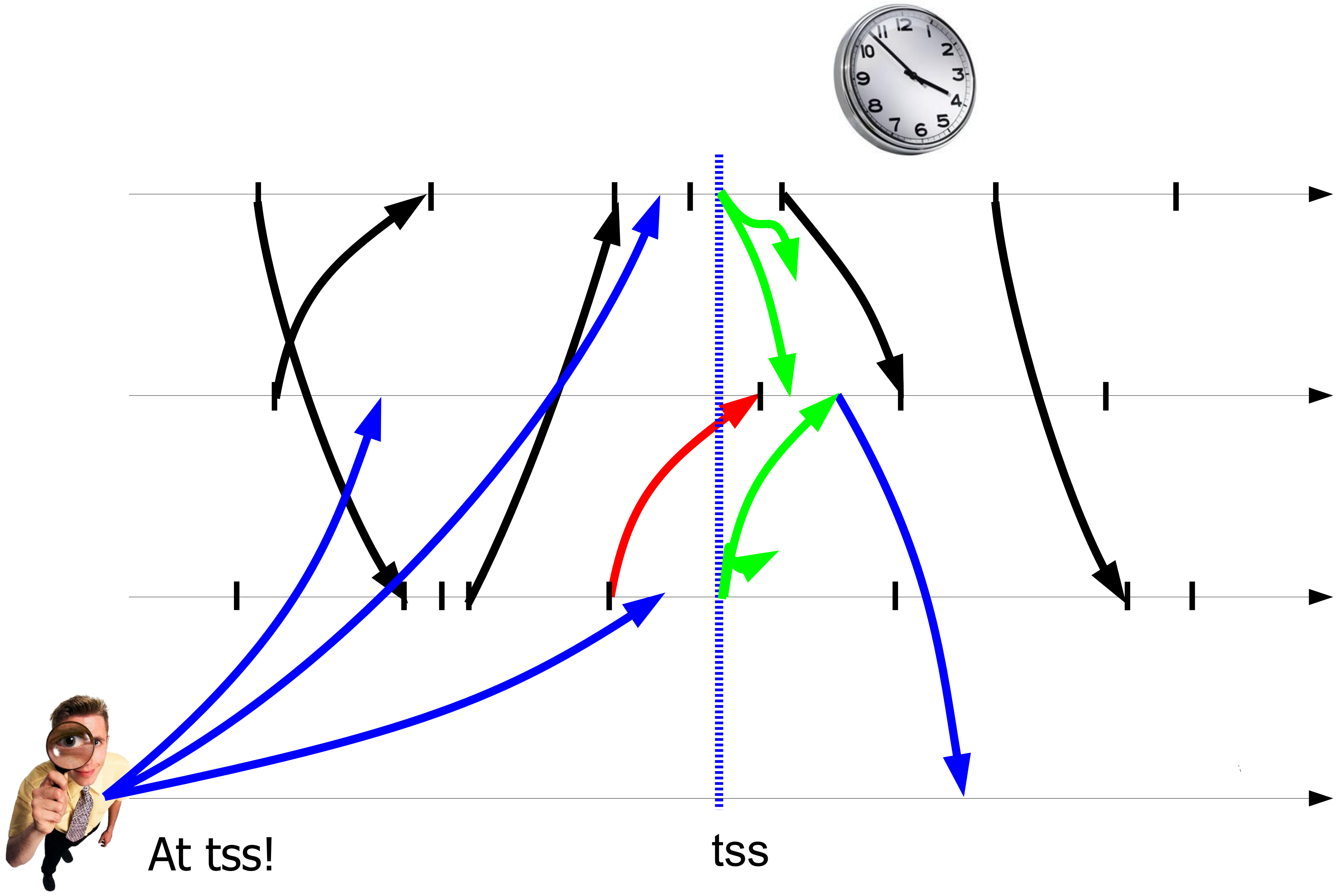
No reporting to monitor process

- Reporting all events to a monitor causes a large overhead
- Can a query be issued at some point in time?

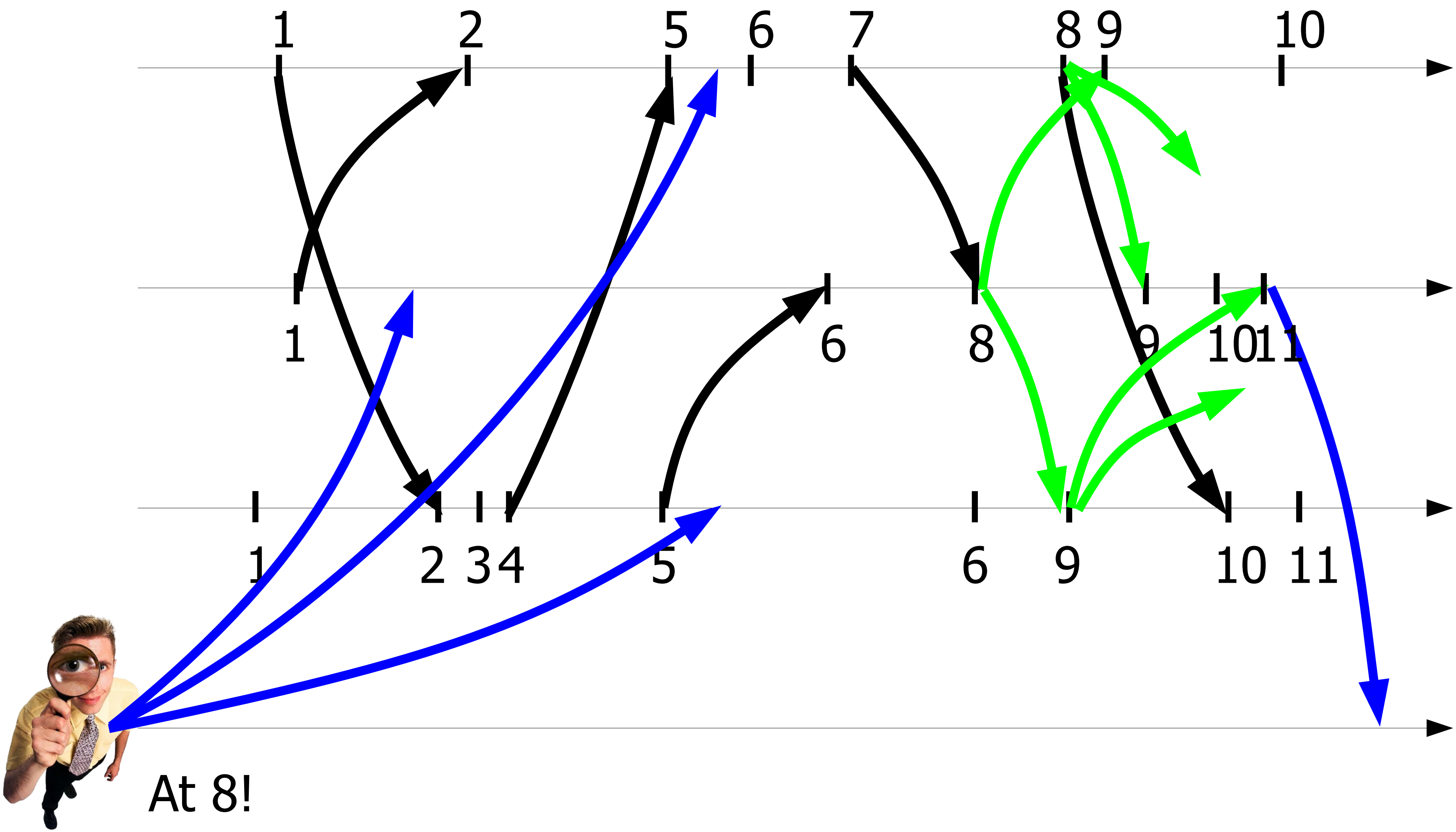
Fourth try: No reporting, synchronous

- Monitor broadcasts tss in the future
- At tss, each process:
  - Records state
  - Sends messages to all others
  - Starts recording messages until receiving a message with RC > tss
- After stopping, sends all data to monitor

Fourth try: No reporting, synchronous



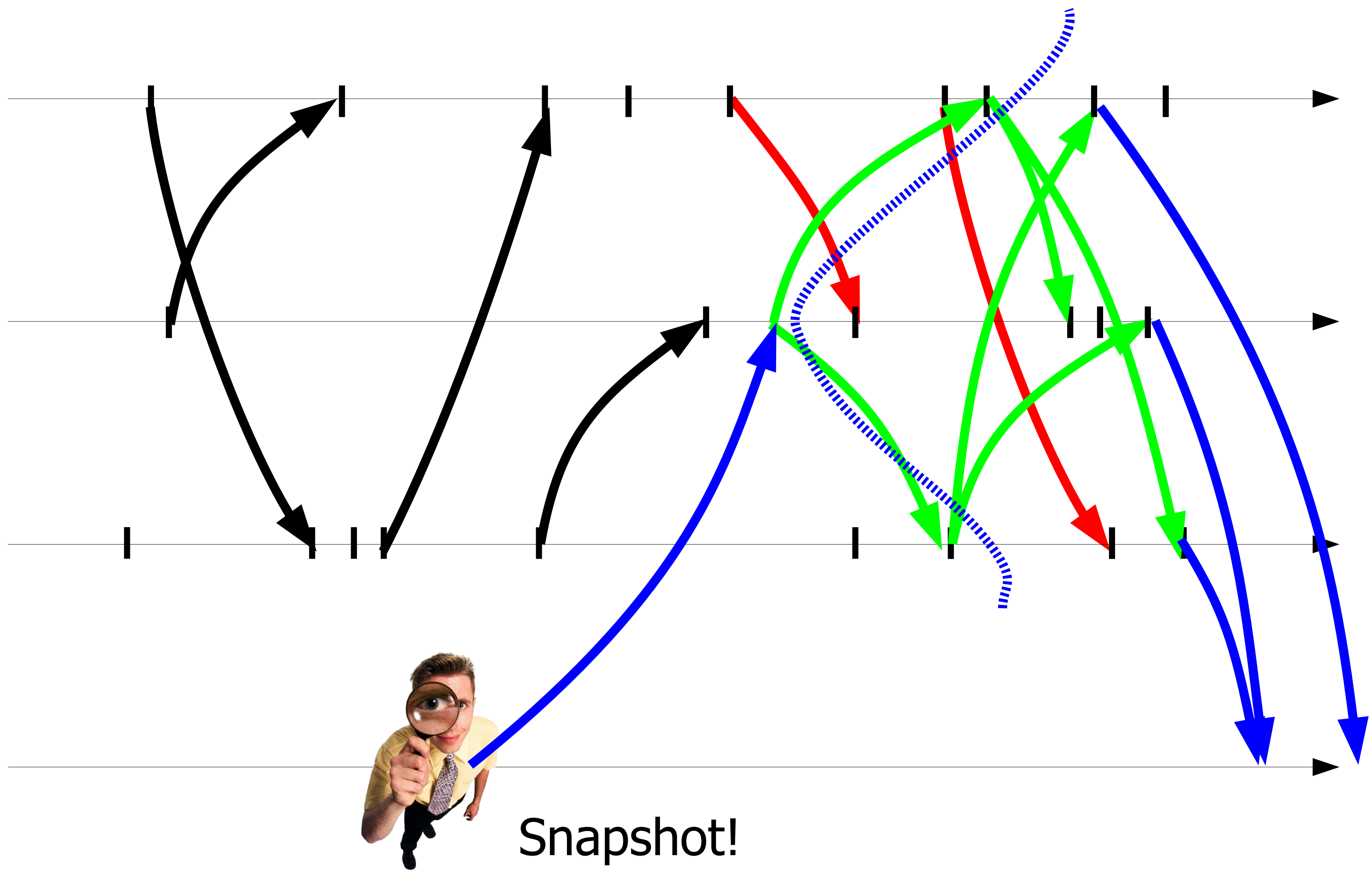
Fifth try: No reporting, logical clock



Chandy and Lamport

- Send a “Snapshot” message to some process
- Upon receiving for the first time:
  - Records state
  - Relays “Snapshot” to all others
  - Starts recording on each channel until receiving “Snapshot”
- Send all data to monitor

Chandy and Lamport



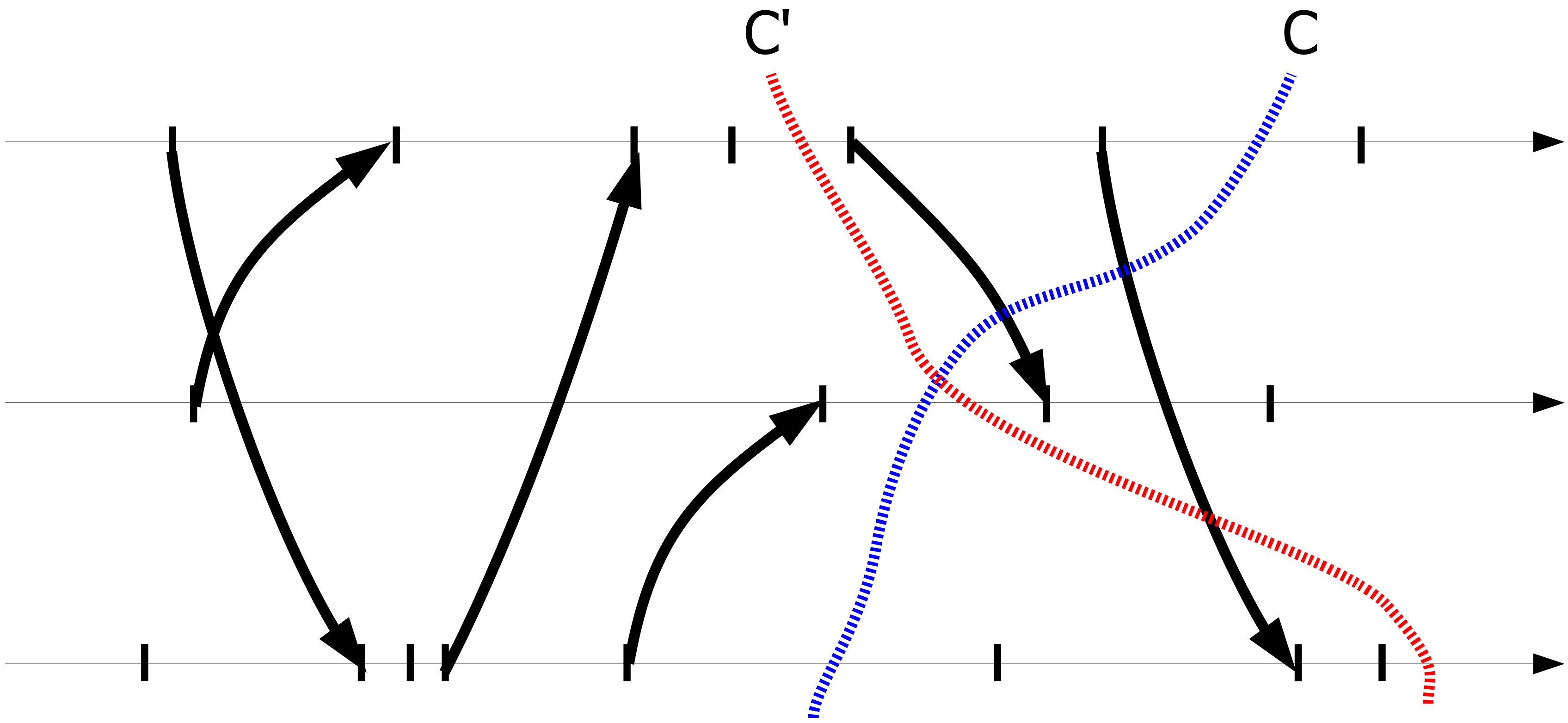
Summary

- Can observe the distributed system by:
  - Using scalar logical clocks
  - Using vectorial clocks and causal delivery
  - Using Chandy-Lamport algorithm to collect a discrete snapshot

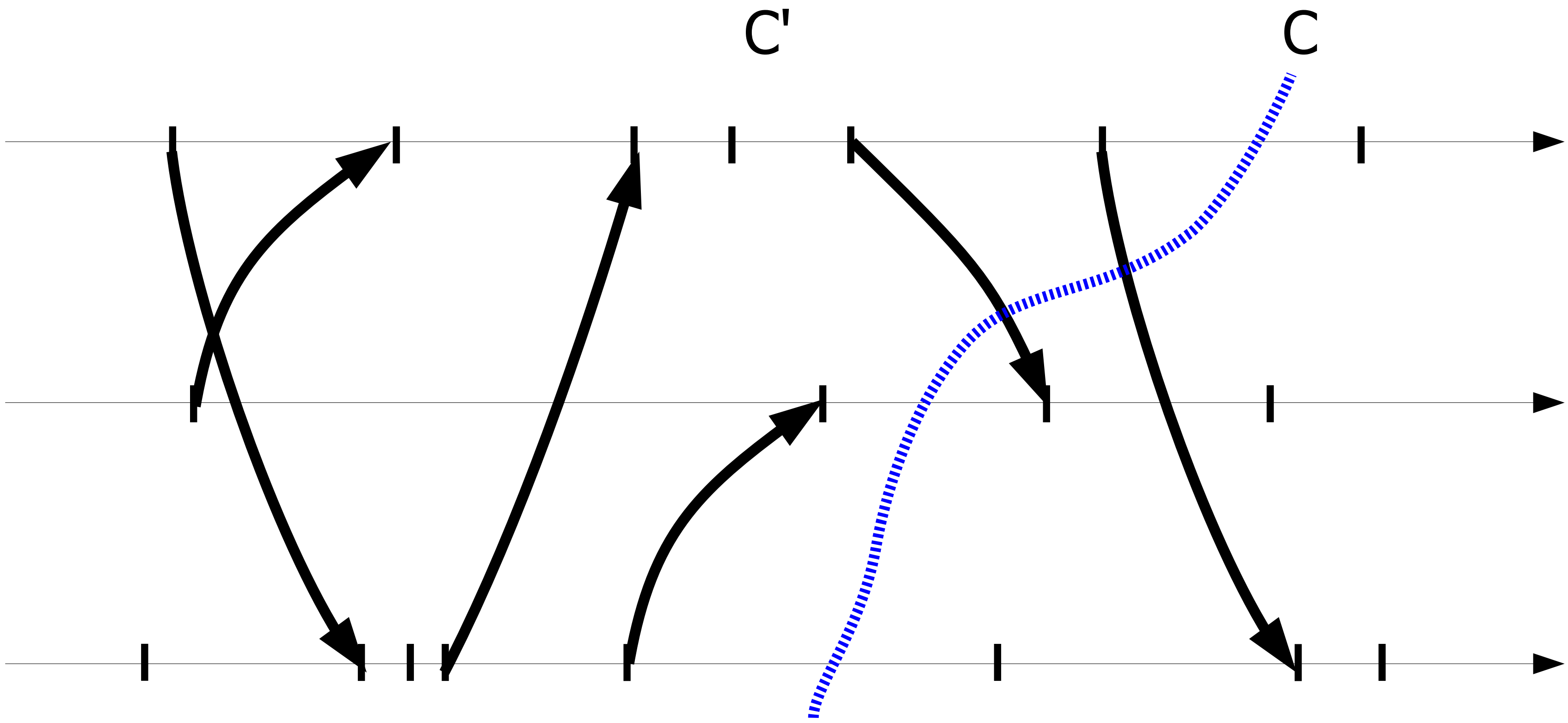
Cuts and consistency

- A cut is the union of prefixes of process history
- A consistent cut includes all causal predecessors of all events in the cut
- Intuitive methods:
  - If a cut is an instant, there are no messages from the future
  - In the diagram, no arrows enter the cut
  - All events in the frontier are concurrent

Consistent cuts

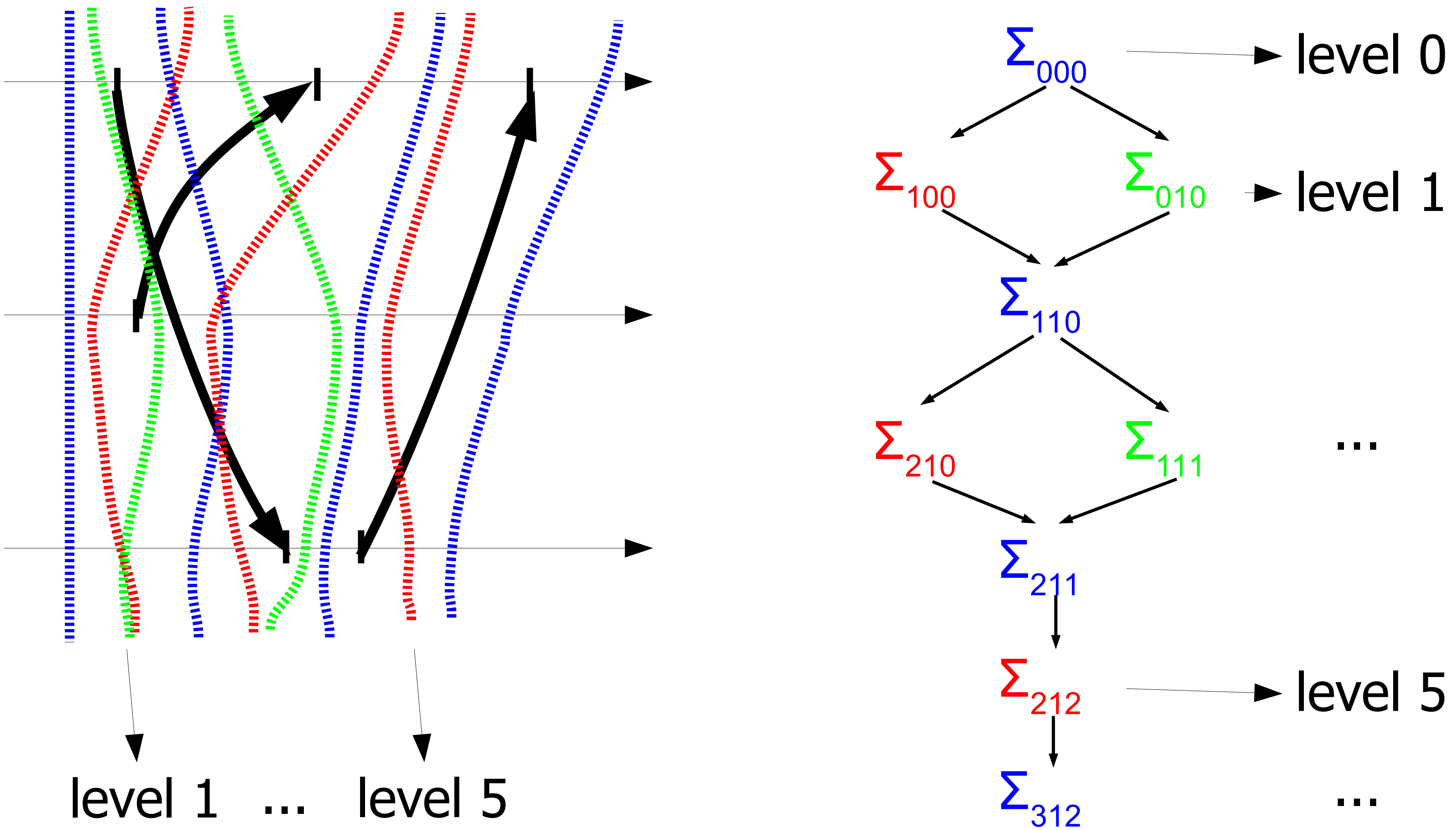


Consistent cuts and global states



- Notation Σ_{625} means state after 6 events in first process, 2 events in second, ...
- This is a *level 13* state (after 6+2+5 events)

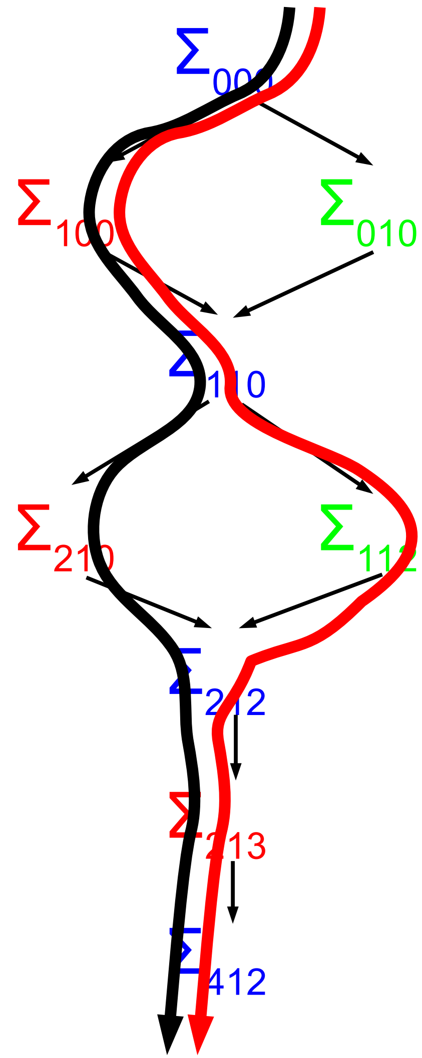
State lattice



(*) Not all possible configurations a represented!

Consistent global states

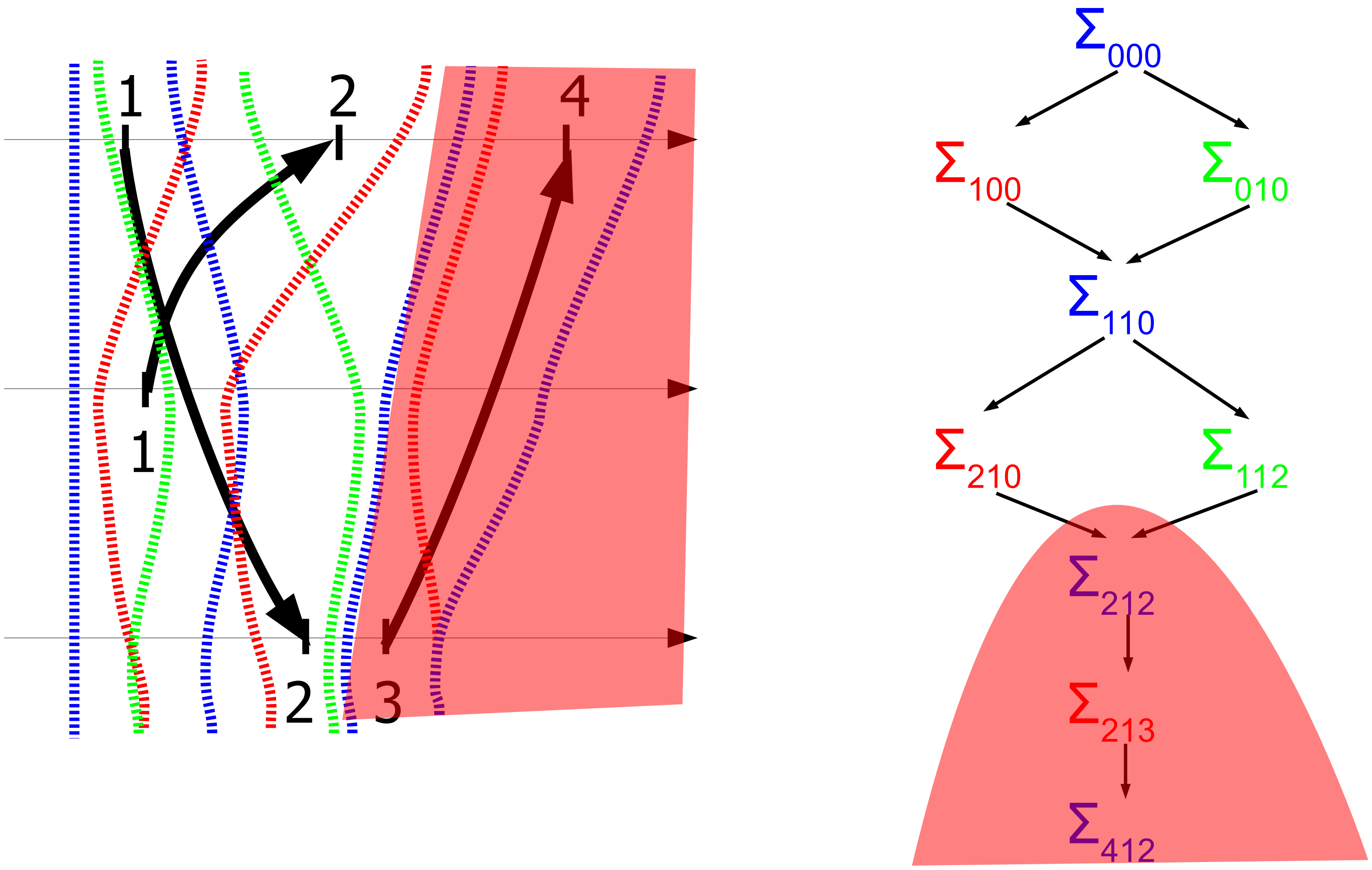
- Includes the true sequence of states in the system
- An observer within the system cannot deny any of the possible paths



Stable predicates

- Once true, always true
- Examples:
  - Deadlock detection
  - Termination
  - Loss of token
  - Garbage collection
- Can be evaluated periodically on snapshots

Stable predicates

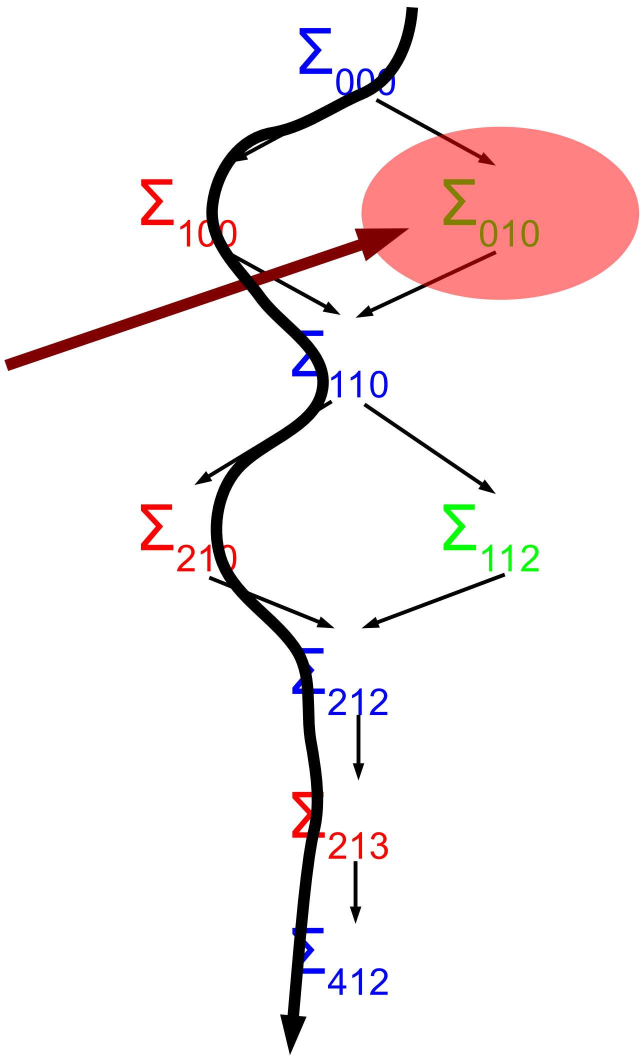


Non-stable predicates

- Examples:
  - Total size of queues in the system
  - Number of messages in transit
  - Amount of memory used
- Can be detected by full monitoring of all (relevant) events

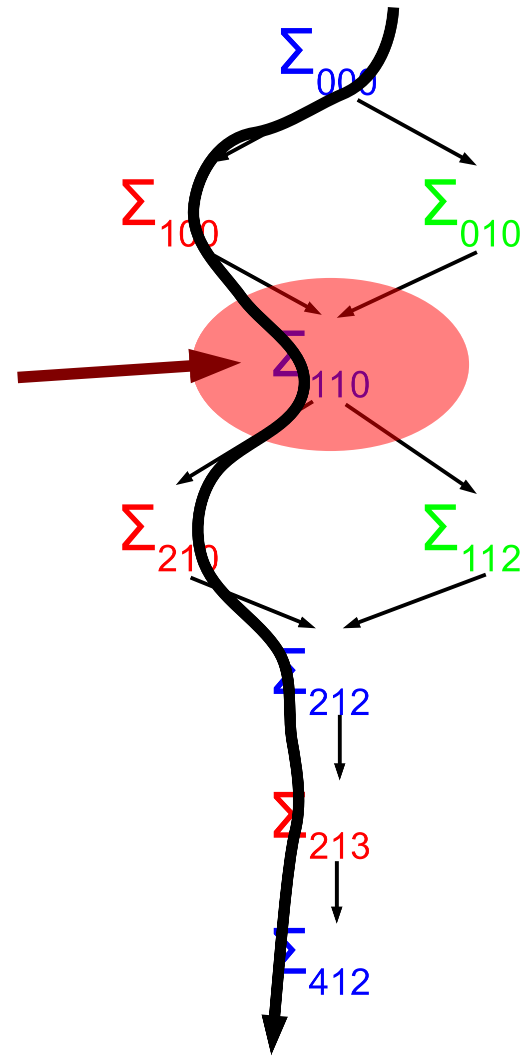
Non-stable predicates

- True in a subset of observable states
- Some are <u>possibly true</u>: an observer in the system cannot deny having been true
- The predicate does not hold on some paths



Non-stable predicates

- True in a subset of observable states
- Some are definitely true: an observer in the system is sure of having been true
- The predicate holds on all possible paths



Non-stable predicates

- Start with level n=1
- Loop while more states can be found:
  - Generate all level n states (by selecting all messages that can be accepted in state n-*1*)
  - If true in all of these states:
    - return **DEFINITELY TRUE**
  - If true in any of these states:
    - return **POSSIBLY TRUE**
  - Increment n
- return **FALSE**

Summary

- An instant in the asynchronous system model is defined by a consistent cut
  - Safe opportunities to observe the system
- Considering each consistent cut:
  - We cannot deny that the system might have been in that state
  - We can only assert that the system has been in that state if there are no concurrent cuts

