Distributed Systems ID2201

global state

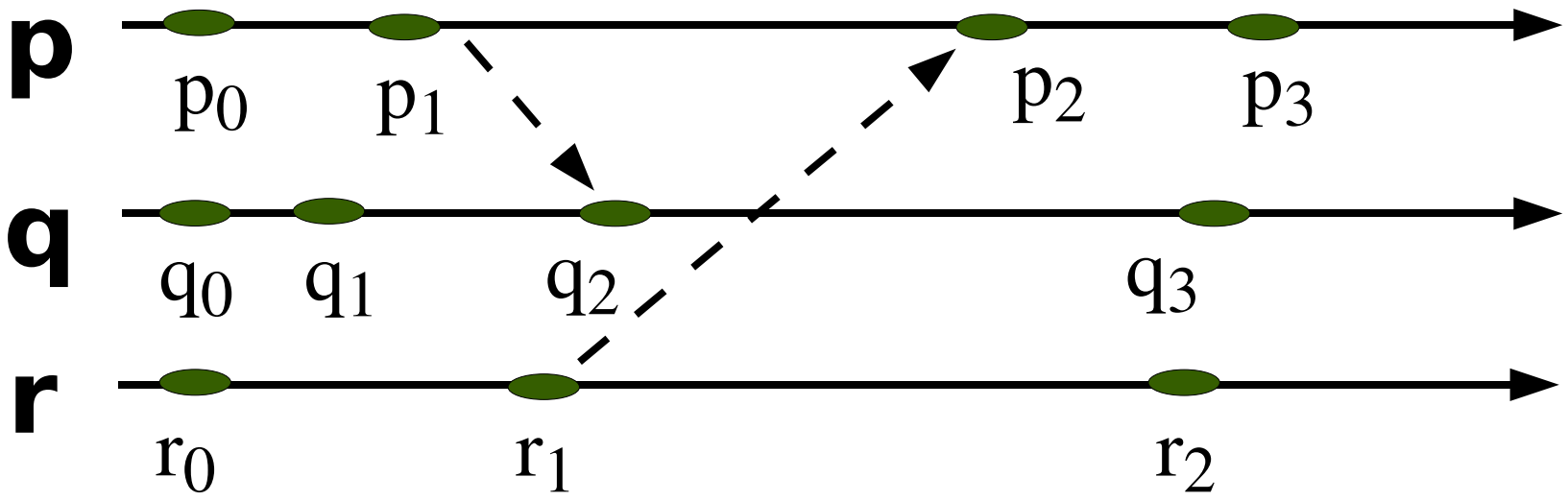
Johan Montelius

Global state

- Time is very much related to the notion of global state.
- If we cannot agree on a time how should we agree on a global state.
- Global state is important:
  - garbage collection
  - dead-lock detection
  - termination
  - debugging

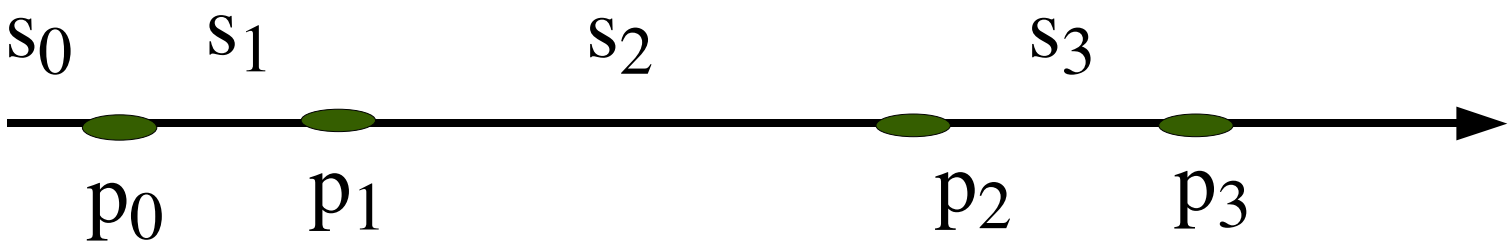
Global state

- Given a partial order of events, can we say anything about the state of the system?



History and state

- The history of a process is a sequence of events: <p_0, p_1, ..>
- The state of a process, s_i, is the state before the i'th event.

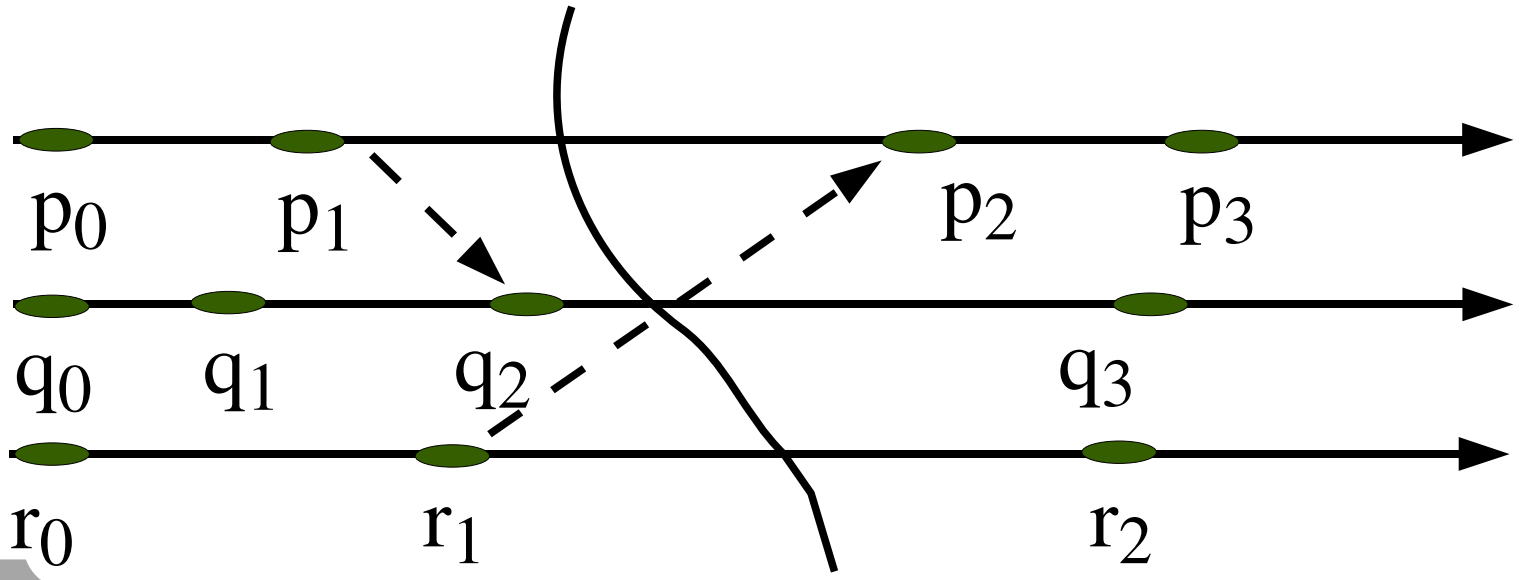


History and state

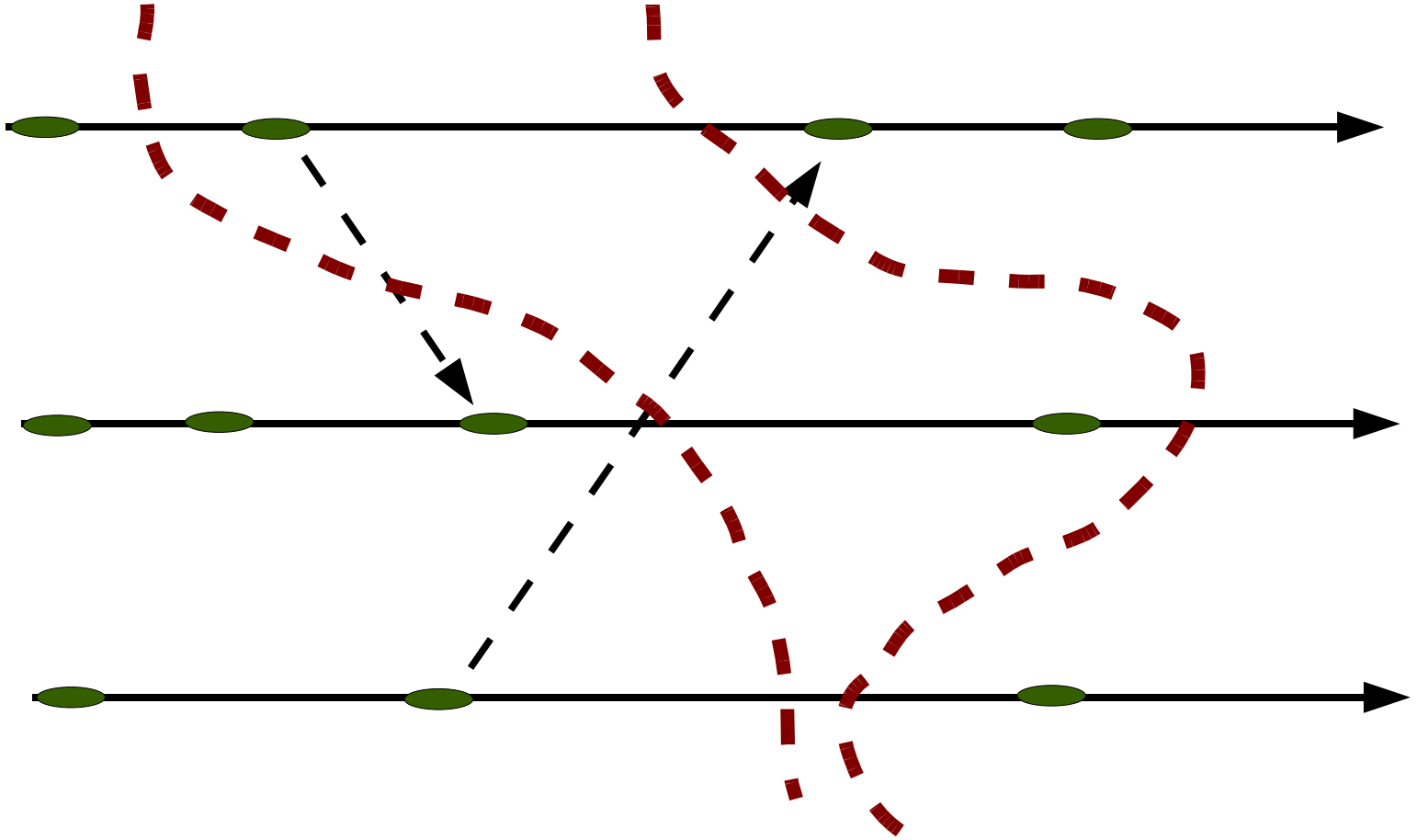
- Is the state of a process the history of events?
- What is the global state of a distributed system?
  - the union of histories of all processes
- Do all unions make sense?

Global history and cuts

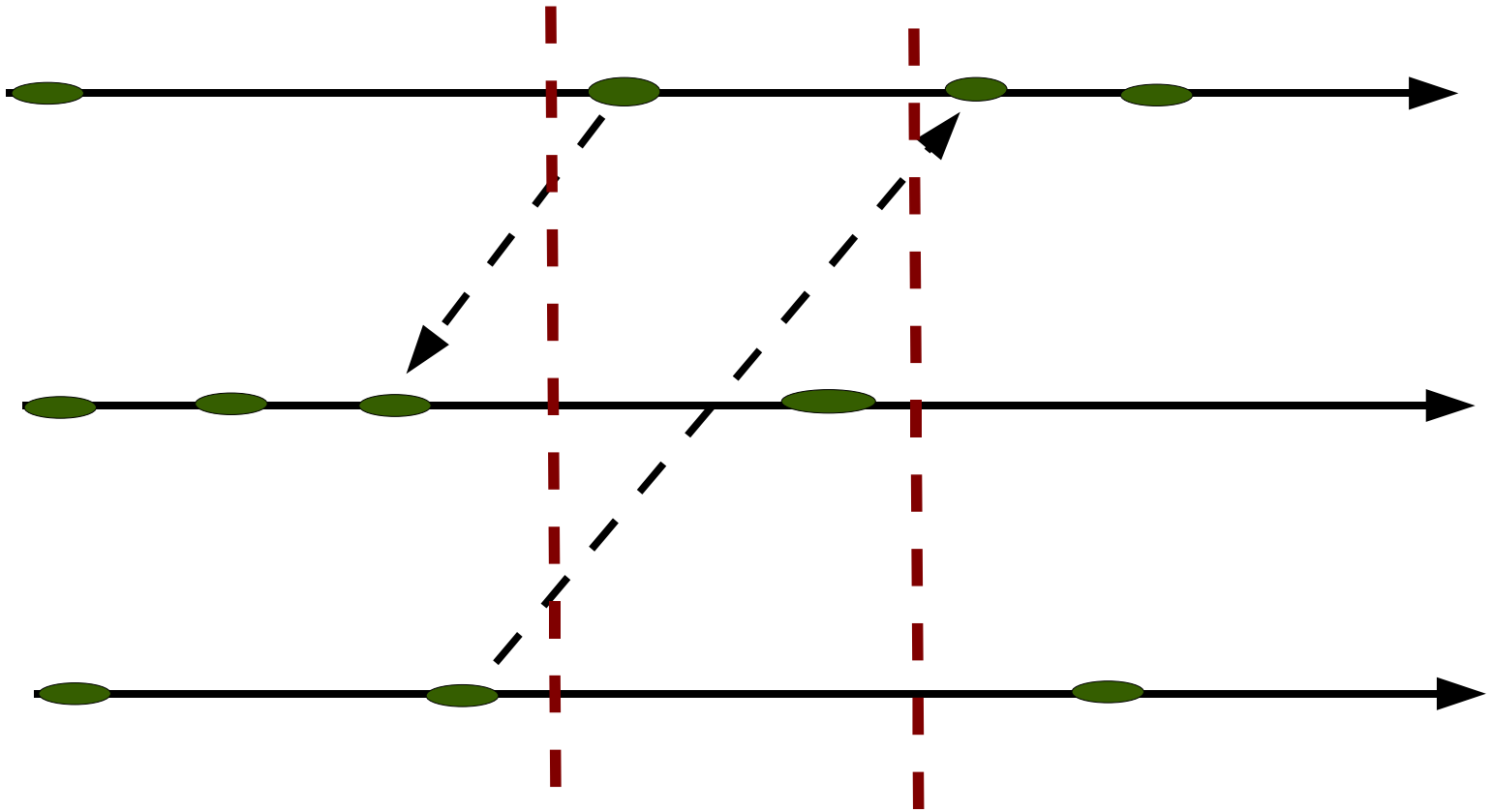
- The global history is the union of all histories.
- A *cut* is the global history up to a specific event in each history.



different cuts

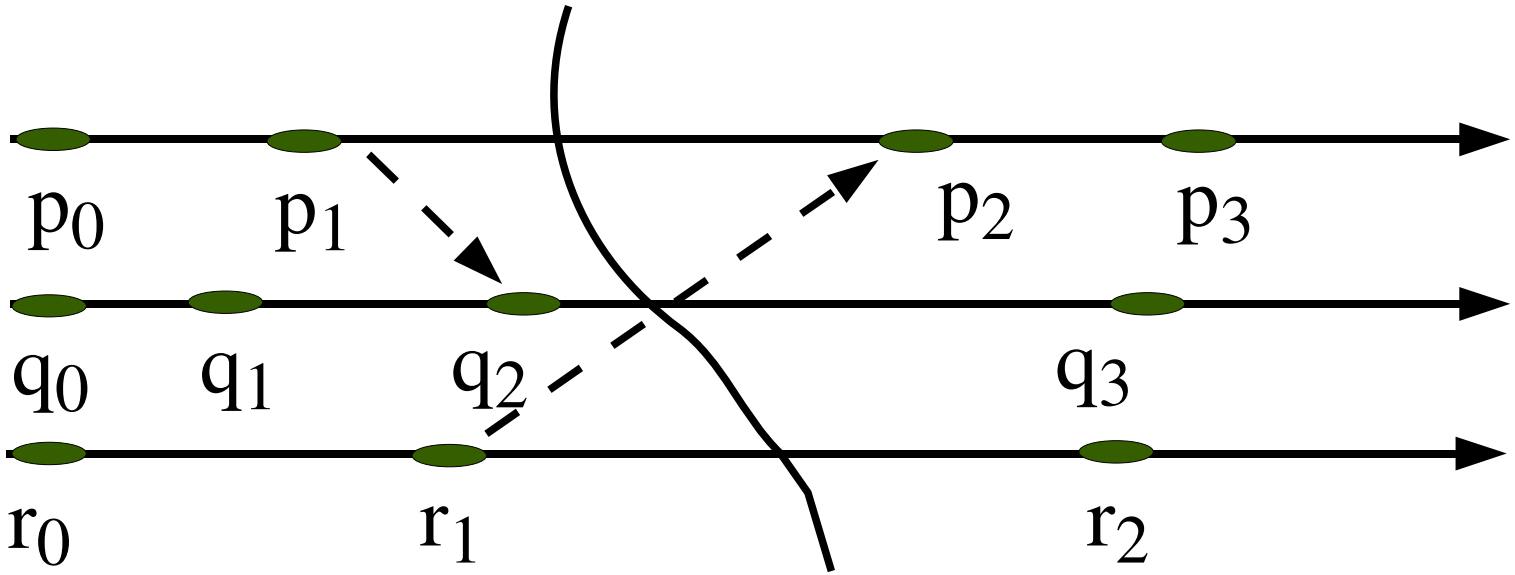


different cuts



Consistent cuts

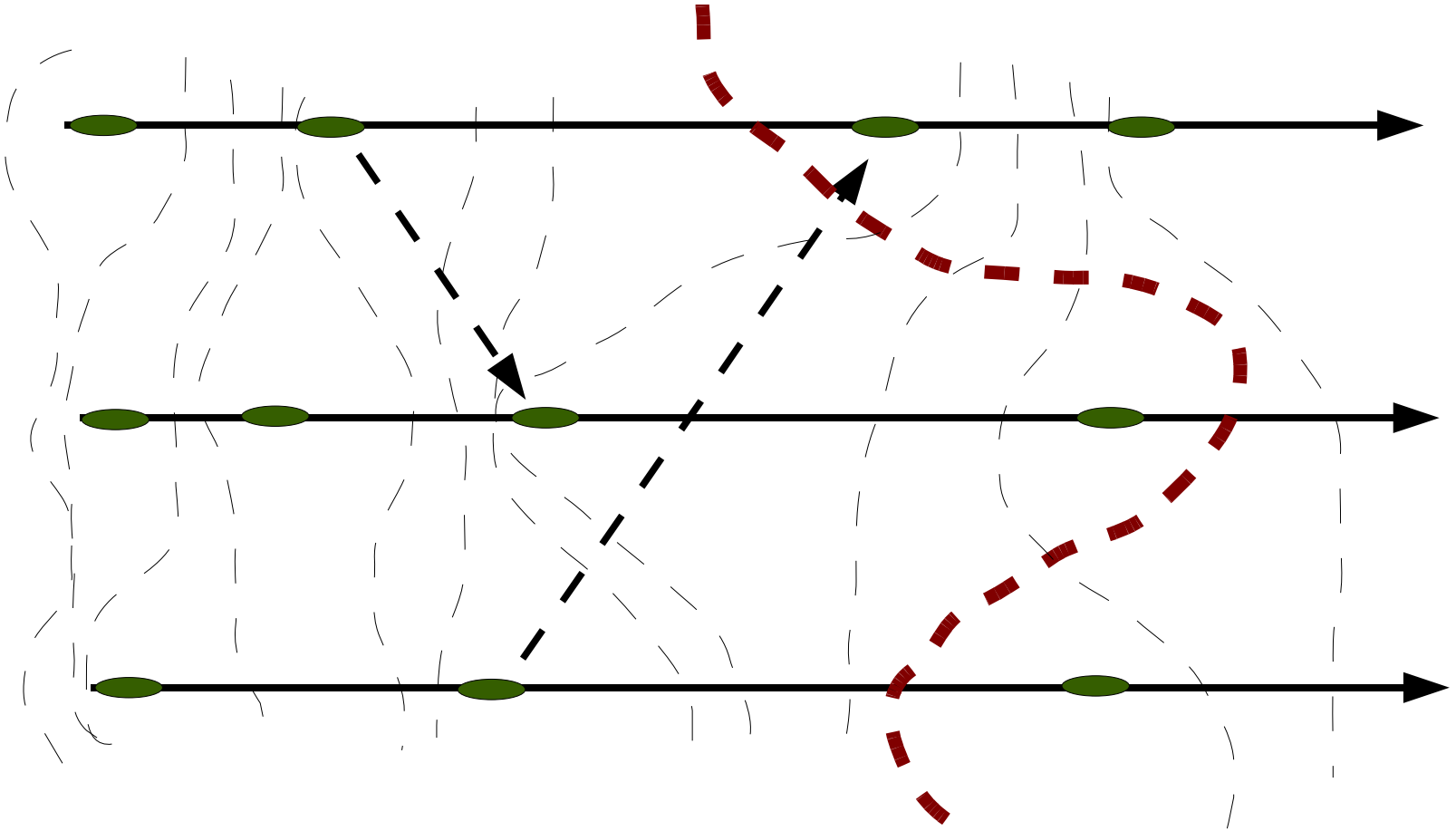
- How can we tell if a cut is *consistent?*
  - *For each event e in the cut:*
    - *if f happened before e then*
    - *f is also in the cut.*



Consistent global state

- A consistent cut corresponds to a consistent global state.
  - it is a possible state without contradictions
  - the actual execution might not have passed through the state

consistent cut

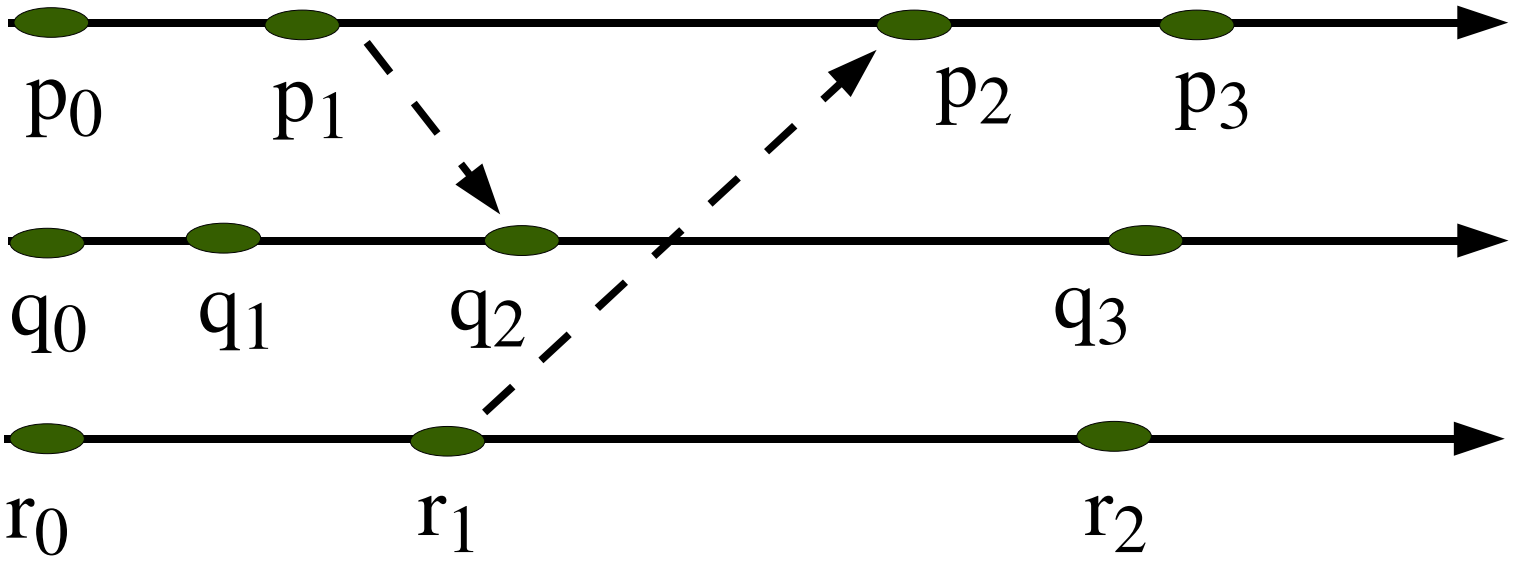


Linearization

- A *run* is a total ordering of all events in a global history that is consistent with each local history.
- A *linearization* or *consistent run* is a run that describes transitions between consistent global states.
- A state S' is reachable from state S if there is a linearization from S to S'.

Linearization

$\{p_0, p_1, q_0, r_0, q_1, r_1, p_2, p_3, q_2, r_2, q_3\}$



Why is this important?

- If we can collect all events and know the *happened before order*, then we can construct all possible linearizations.
- We know that the actual execution took one of these paths.
- Can we say something about the execution even though we do not know which path that was taken?

Global state predicates

- A *global state predicate* is a property that is true or false for a global state.
  - stable: if a predicate is true it remains true for all reachable states.
  - non-stable: if a predicate can become true and then later become false

Properties of executions

- Safety
  - a predicate is never true in any state reachable from S_0.
- Liveness
  - a predicate that eventually evaluates to true for any linearization from S_0.

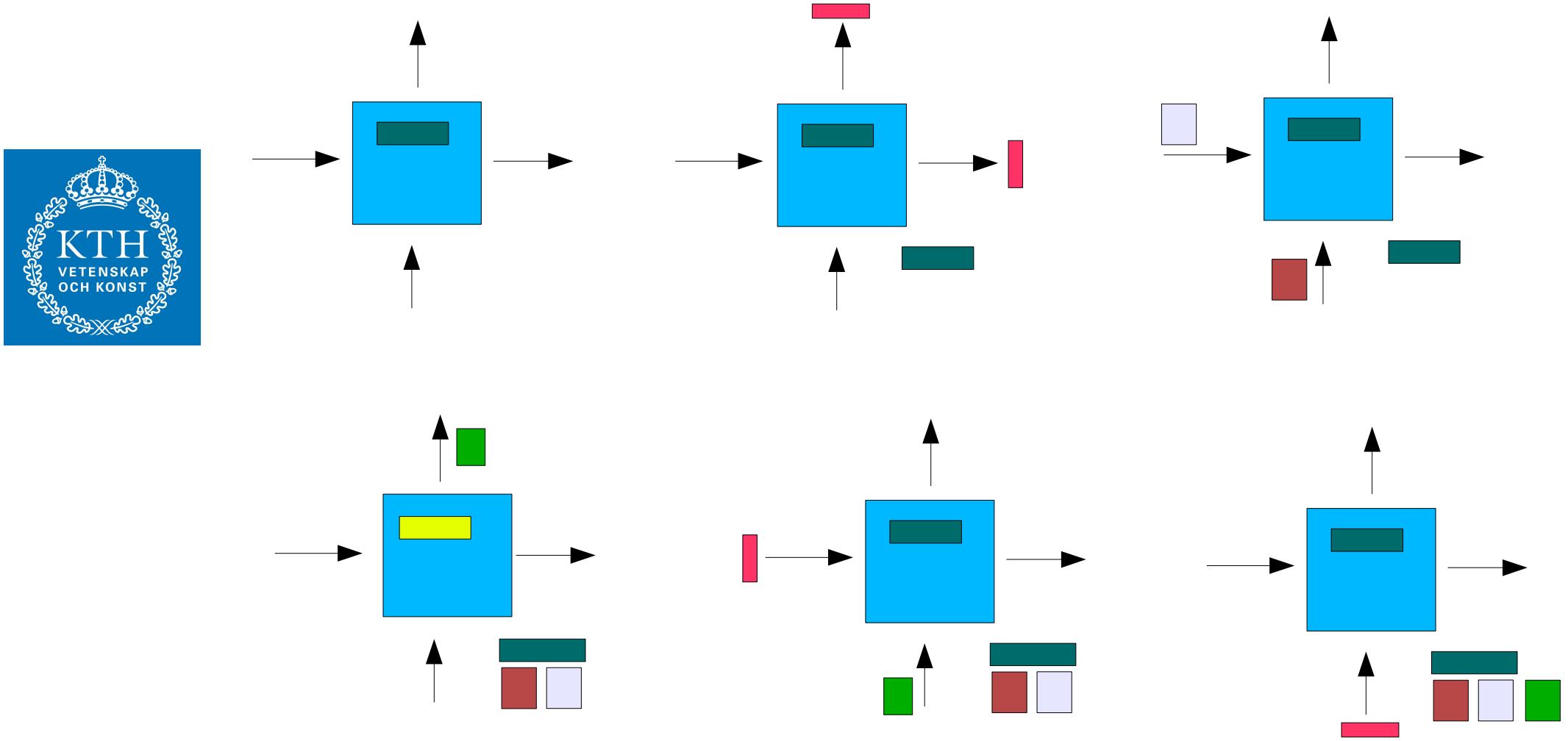
How do we record a global state?

- Stop all processes, record all process states and all messages that are in transit.
  - tough
  - how do we decide when to record the state
  - we don't really want to do this

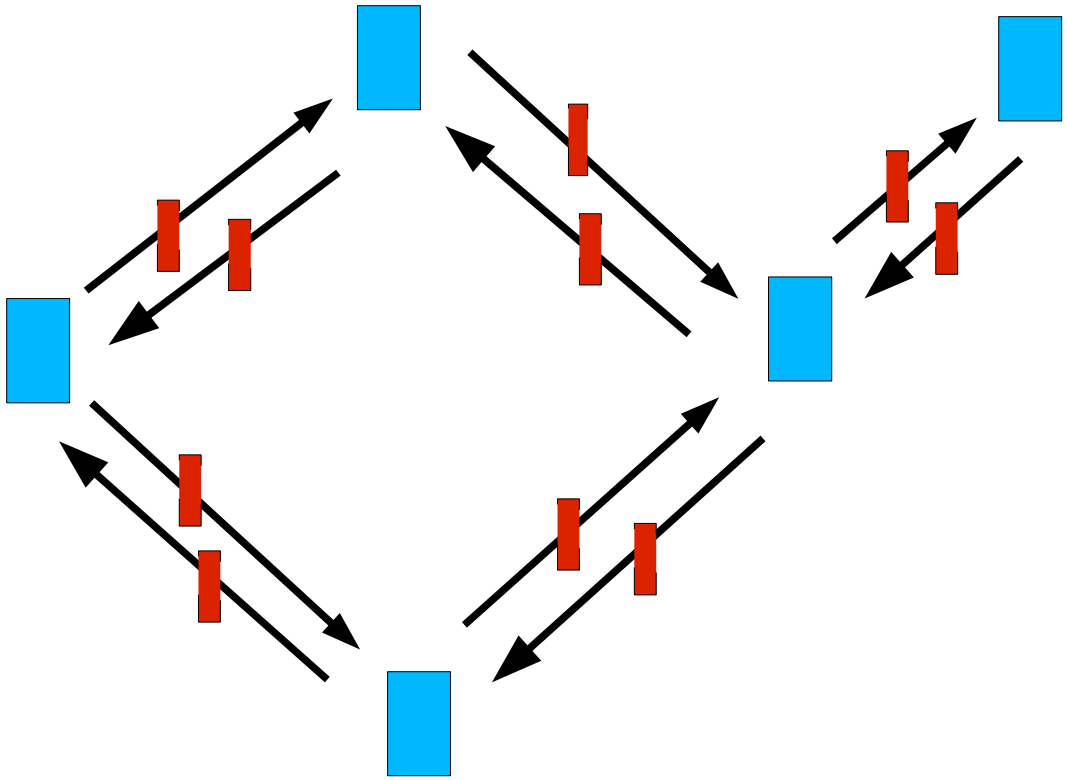
Snapshot - Chandy and Lamport

- Take a snapshot by:
  - record the internal state
  - sends a marker on each outgoing channel
  - start recording incoming messages on each channel until a marker is received on the channel
- If a process receives a marker it will start the snapshot procedure.

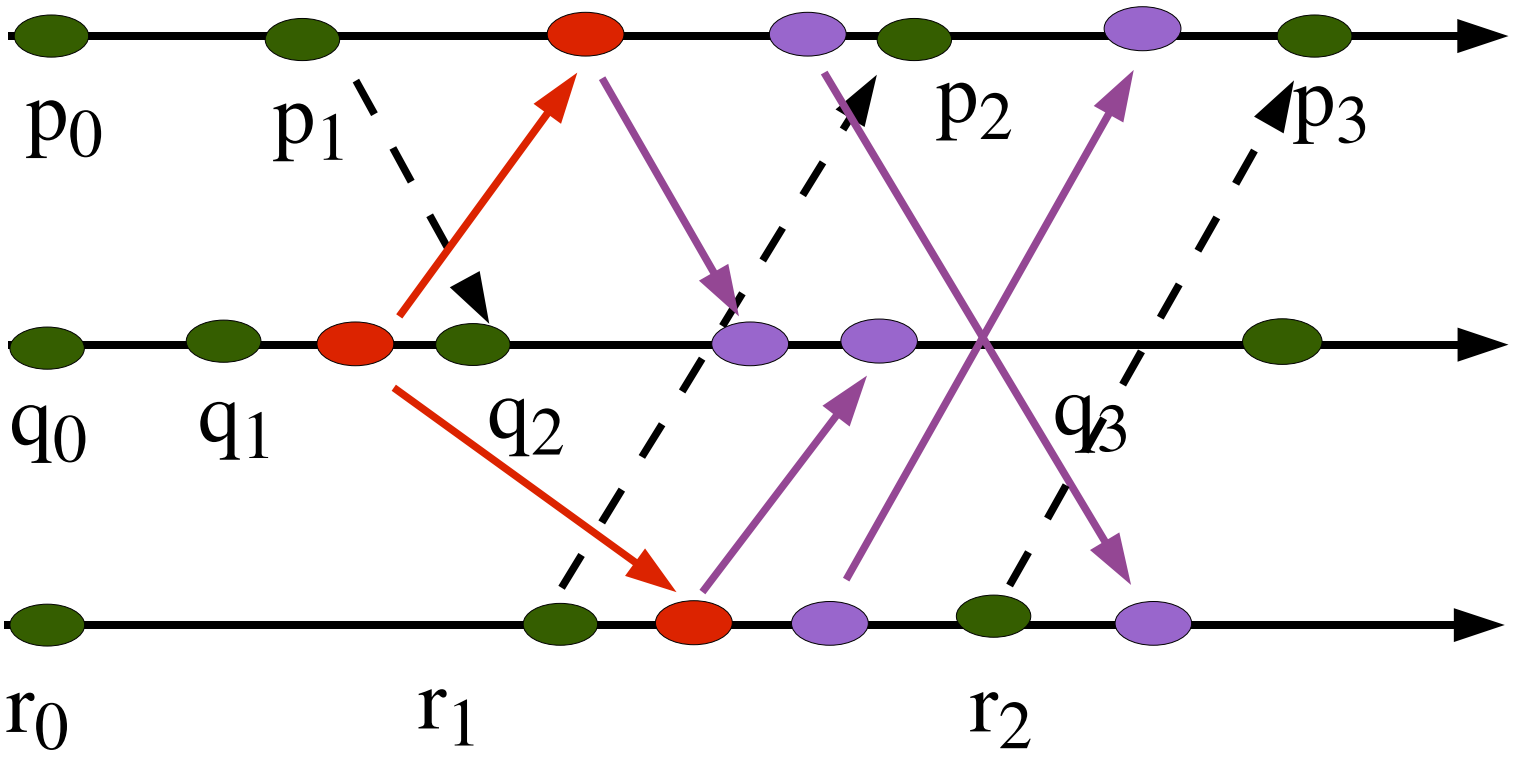
Snapshot



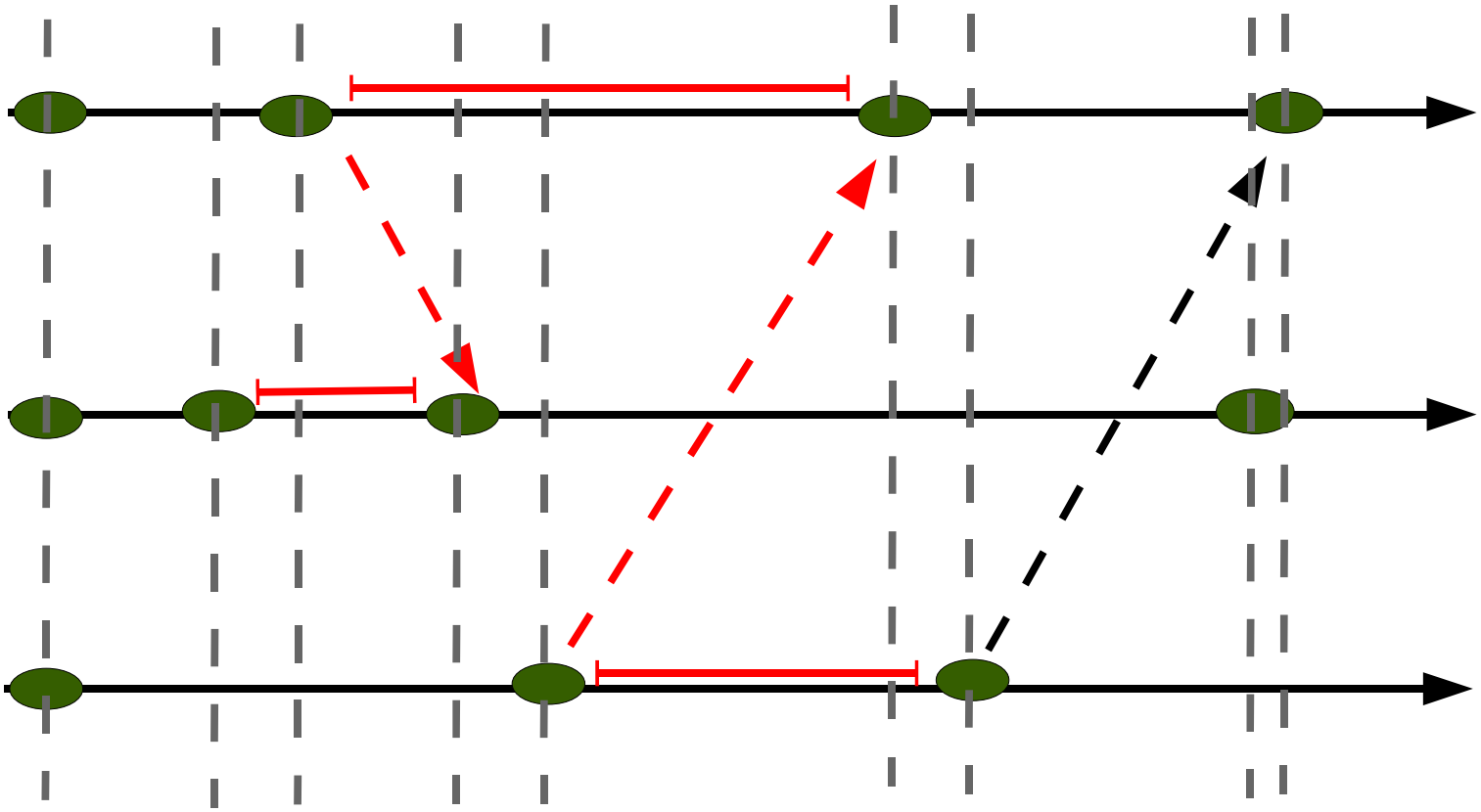
Snapshot



Snapshot



Snapshot



Snapshot

- The snapshot is a recording of a consistent cut.
- So what does it give us:
  - we have a recording of a state that the execution might have passed through
  - all for nothing?
- There is a *linearization* from the snapshot state to a state that occurred during the execution.

Predicates

- If we can show that a *stable predicate* is true given a snapshot then it is also true in the execution.
  - garbage collection
  - dead-lock detection
- A *non-stable* predicate might be true in the snapshot but not necessarily during the actual execution.

Possibly and definitely

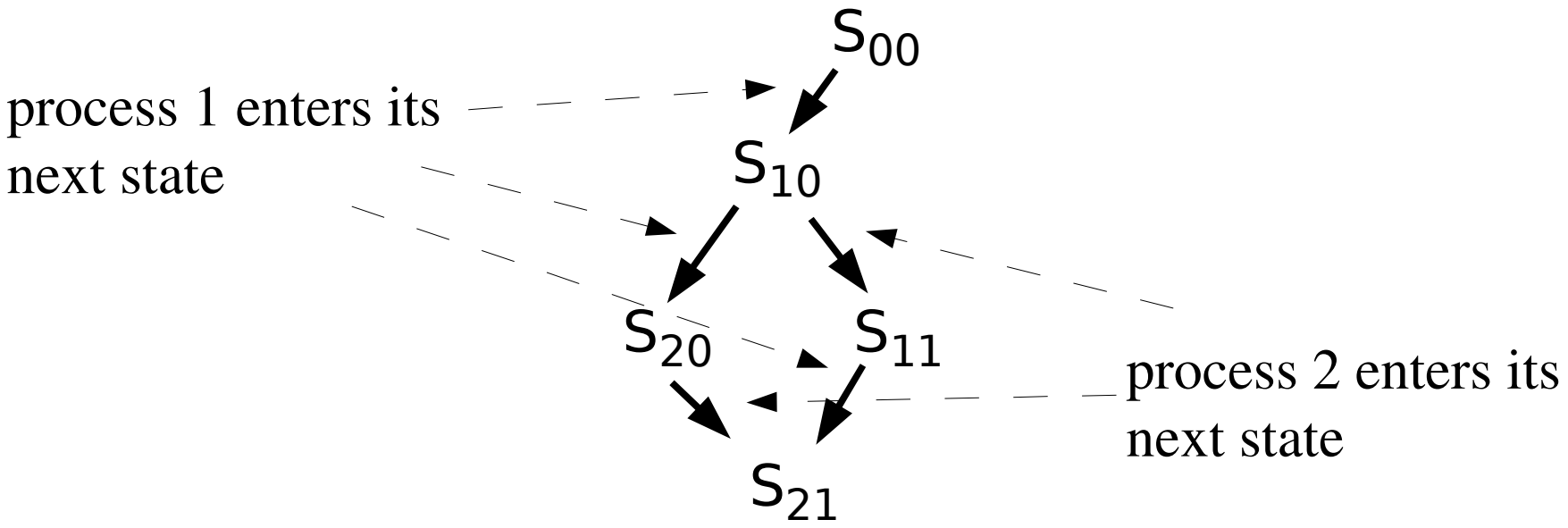
- We want to know if a non-stable predicate *possibly* occurred or *definitely* occurred.
- We have to look at all possible execution states.
  - How do we reconstruct all possible execution states?
  - Is the simple snapshot enough?

Vector time stamped state

- When a process, *i* ,records its state, *C*, it also time stamp it, $V(s_i)$, with its vector clock.
- States are collected by a monitor and grouped into *global states* $S = \{s_0,... s_n\}$.
- A global state is consistent iff
  - any event *e* in a process *k*, that *happened before* s_i is included in s_k
  - $V(s_k)[k] \geq V(s_i)[k]$

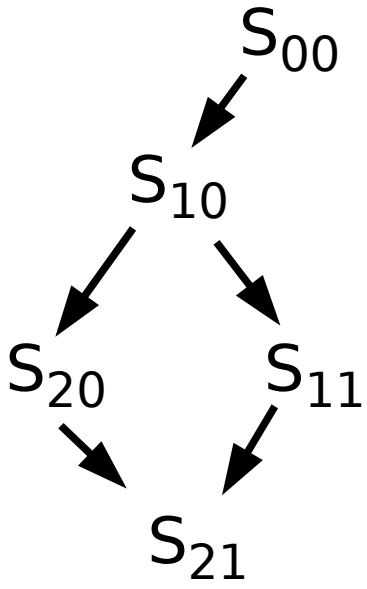
Global state lattice

- Consistent global states form a lattice where each edge is a *possible transition* of a process.



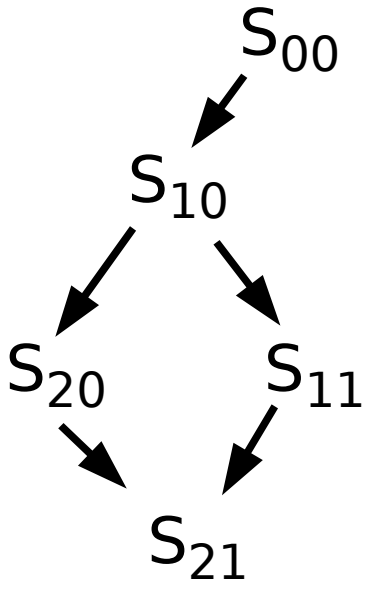
Possibly true

$p(S_{20})$ = true



Definitely true

$p(S_{20})$ = true
$p(S_{11})$ = true



Possibly and definitely

- If a predicate is *true in any* consistent global state of the lattice then it is *possibly true* in the execution.
- If we *cannot find a path* from the initial state to the final state without reaching a state for which a predicate is true then the predicate is *definitely true* during the execution.

Implementation issues

- What is the state of a process?
  - depends on the predicates we want to evaluate
- How often do we have to send a state report?
  - only when the state changes
- How do we best construct the lattice and can we monitor it during execution?
  - yes we can incrementally construct global states and build the lattice

Summary

- Knowing the *happened before order* gives us the notion of
  - consistent cuts or
  - consistent global states
- A snapshot can record a consistent state that the execution possibly entered.
  - evaluate stable predicates
- Using vector clocks we can time stamp states and construct all possible linearizations.
  - evaluate non-stable predicates