Distributed Systems

CS425/ECE428

Feb 6 2023

Instructor: Radhika Mittal

Logistics Related

- HW2 release date has been pushed to Mon, Feb 20th. Accordingly, its due date has been pushed to Mon. Mar 6th.

- MP0 due on Wednesday.

- Note about exams on CampusWire:
  - Midterm: Mar 22-24, Finals: May 4
  - Reservation via PrairieTest.
    - You can reserve a slot for Midterms starting Mar 2nd
  - If you need DRES accommodations, please upload your Letter of Accommodations on the CBTF website.

Today's agenda

- **Global State**
  - Chapter 14.5
  - Goal: reason about how to capture the state across all processes of a distributed system without requiring time synchronization.
- **Multicast** (if time)

Recap

- State of each process (and each channel) in the system *at a given instant of time.*
  - Difficult to capture -- requires precisely synchronized time.

- Relax the problem: find a consistent global state.

- Chandy-Lamport algorithm to calculate global state.
  - Obeys causality (creates a consistent cut).
  - Does not interrupt the running distributed application.
  - Can be used to detect global properties.

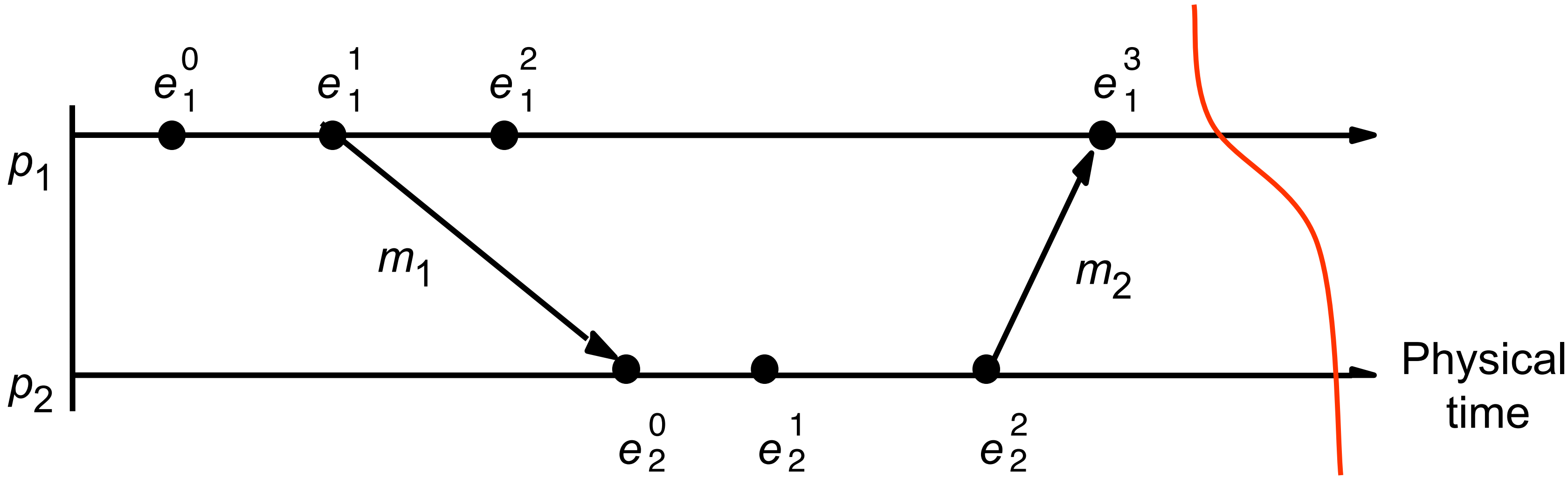
More notations and definitions

- history(p_i) = $h_i = \langle e_i^0, e_i^1, \ldots \rangle$
- global history: $H = \cup_i (h_i)$

- A run is a total ordering of events in H that is consistent with each h_i's ordering.

- A linearization is a run consistent with happens-before ($\rightarrow$) relation in H.

Example



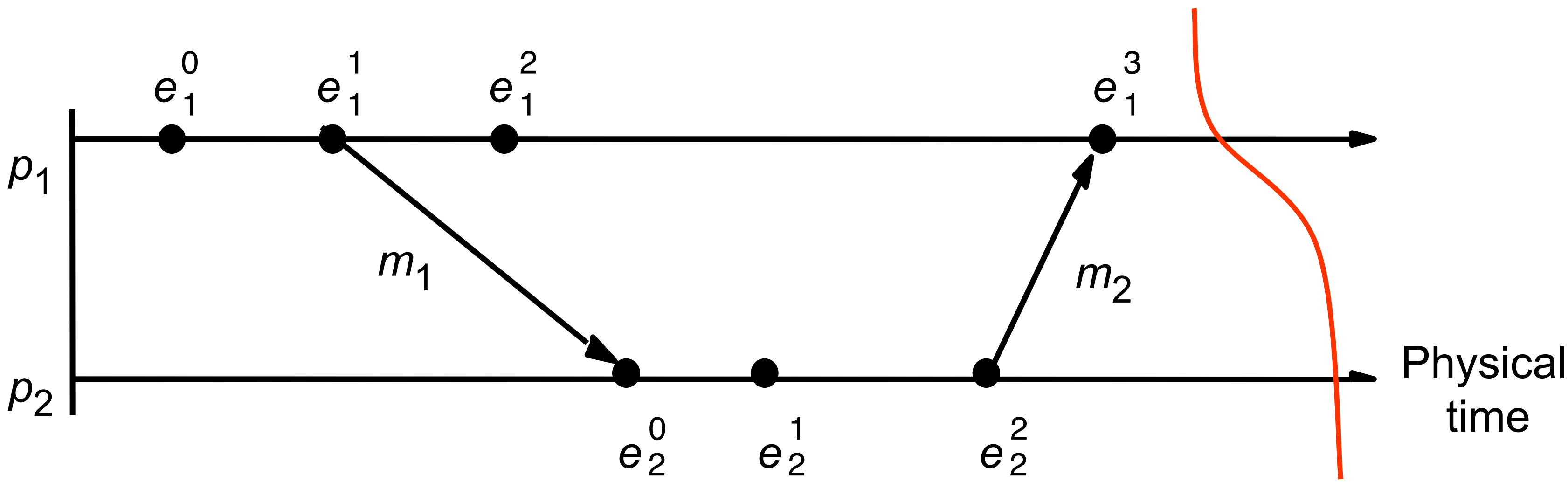
Order at p_1: $< e_1^0, e_1^1, e_1^2, e_1^3 >$ Order at p_2: $< e_2^0, e_2^1, e_2^2 >$

Causal order across p_1 and p_2: $< e_1^0, e_1^1, e_2^0, e_2^1, e_2^2, e_1^3 >$

Run: $< e_1^0, e_1^1, e_1^2, e_1^3, e_2^0, e_2^1\ e_2^2 >$

Linearization: $< e_1^0, e_1^1, e_1^2, e_2^0, e_2^1\ e_2^2, e_1^3 >$

Example



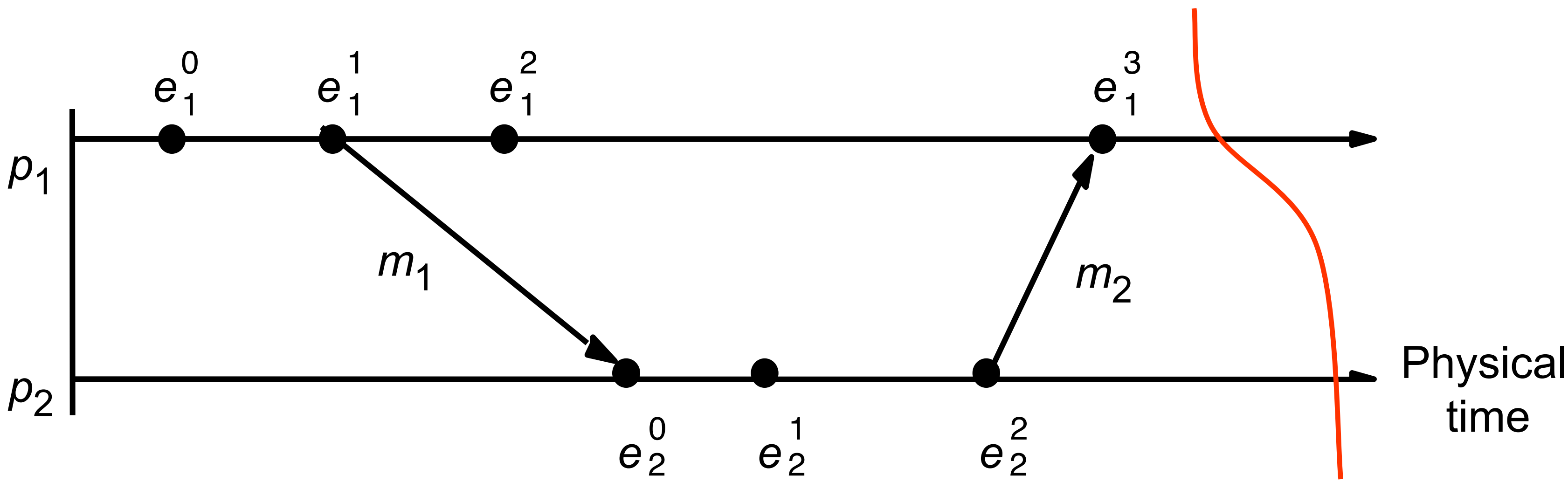
Order at p_1: $< e_1^0, e_1^1, e_1^2, e_1^3 >$ Order at p_2: $< e_2^0, e_2^1, e_2^2 >$

Causal order across p_1 and p_2: $< e_1^0, e_1^1, e_2^0, e_2^1, e_2^2, e_1^3 >$

Run: $< e_1^0, e_1^1, e_1^2, e_1^3, e_2^0, e_2^1\ e_2^2 >$

Linearization: $< e_1^0, e_1^1, e_1^2, e_2^0, e_2^1\ e_2^2, e_1^3 >$

Example

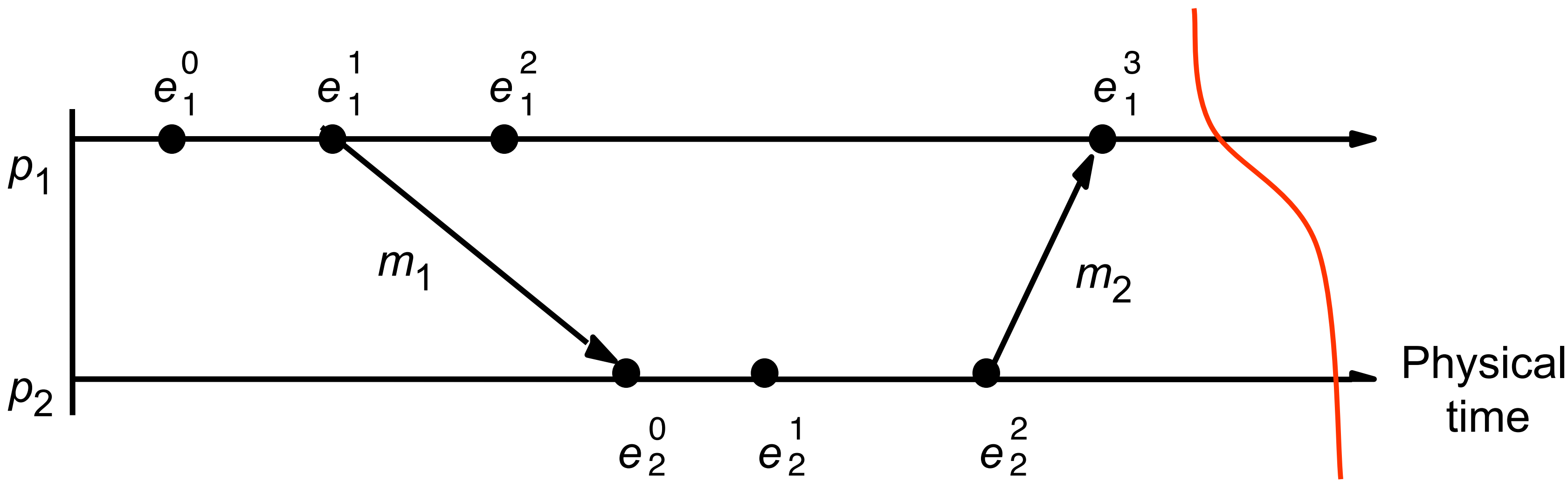


Order at p_1: < e_1^0, e_1^1, e_1^2, e_1^3 > Order at p_2: < e_2^0, e_2^1, e_2^2 >

Causal order across p_1 and p_2: < e_1^0, e_1^1, e_2^0, e_2^1, e_2^2, e_1^3 >

< e_1^0, e_1^1, e_2^0, e_2^1, e_1^2, e_2^2, e_1^3 >

Example



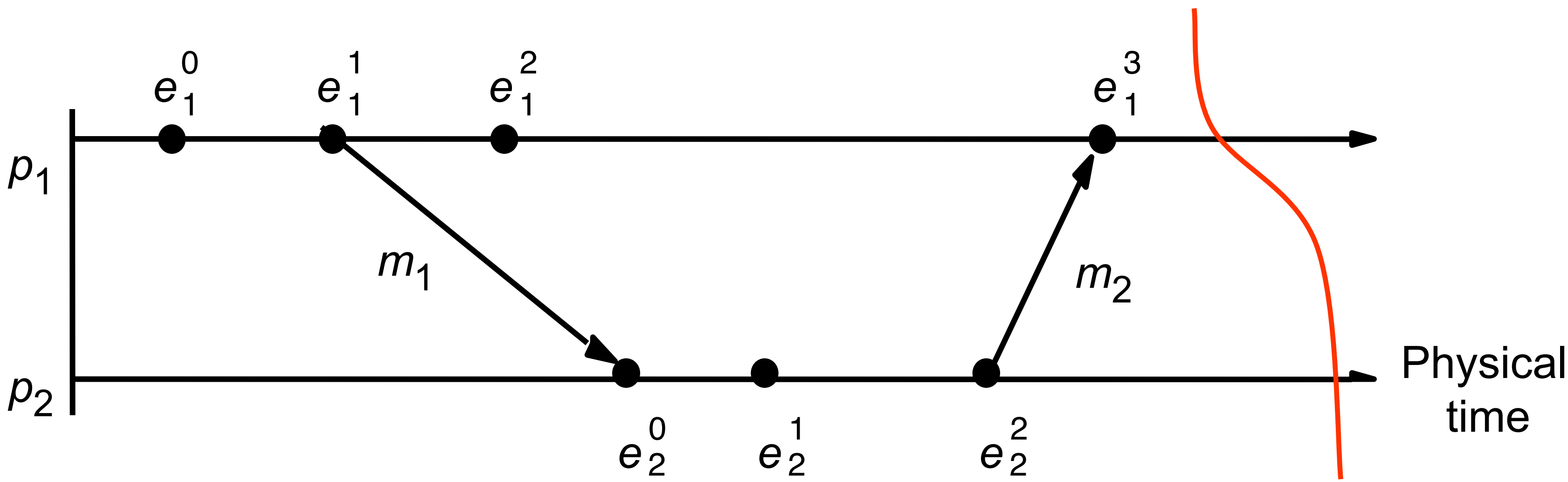
Order at p_1: $< e_1^0, e_1^1, e_1^2, e_1^3 >$ Order at p_2: $< e_2^0, e_2^1, e_2^2 >$

Causal order across p_1 and p_2: $< e_1^0, e_1^1, e_2^0, e_2^1, e_2^2, e_1^3 >$

$< e_1^0, e_1^1, e_2^0, e_2^1, e_1^2, e_2^2, e_1^3 >$: Linearization

$< e_1^0, e_2^1, e_2^0, e_1^1, e_1^2, e_2^2, e_1^3 >$:

Example



Order at p_1: $< e_1^0, e_1^1, e_1^2, e_1^3 >$ Order at p_2: $< e_2^0, e_2^1, e_2^2 >$

Causal order across p_1 and p_2: $< e_1^0, e_1^1, e_2^0, e_2^1, e_2^2, e_1^3 >$

$< e_1^0, e_1^1, e_2^0, e_2^1, e_1^2, e_2^2, e_1^3 >$: Linearization

$< e_1^0, e_2^1, e_2^0, e_1^1, e_1^2, e_2^2, e_1^3 >$: Not even a run

More notations and definitions

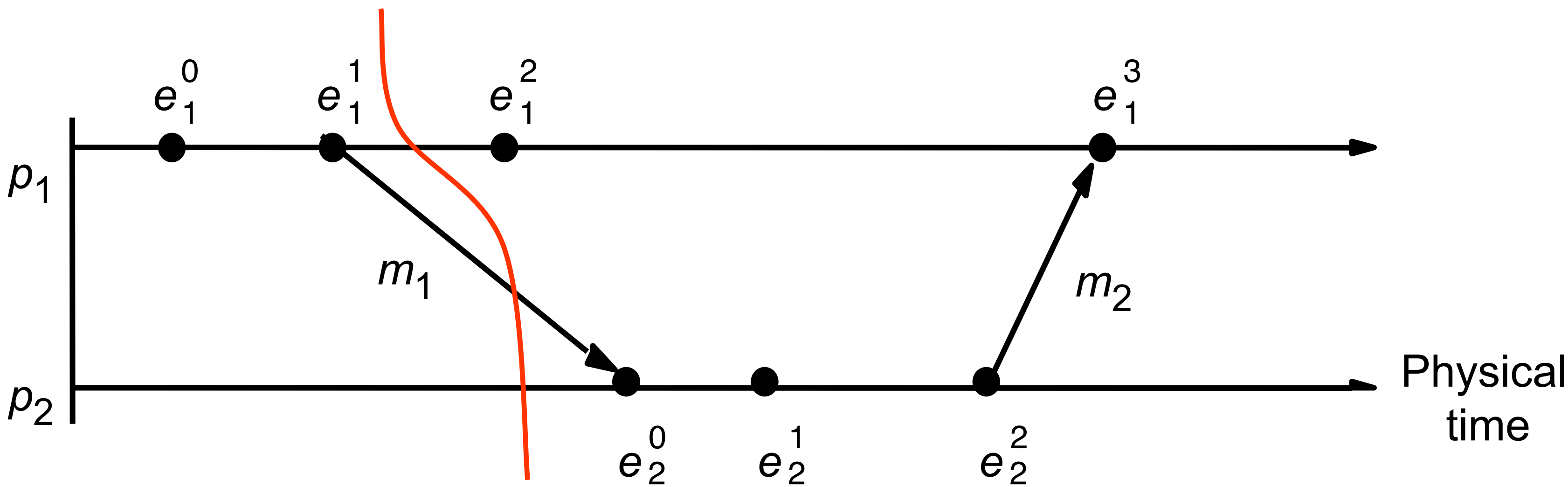
- **history(p_i)** = $h_i = \langle e_i^0, e_i^1, \ldots \rangle$
- **global history**: $H = \cup_i (h_i)$

- A **run** is a total ordering of events in H that is consistent with each $\mathbf{h}_i$'s ordering.

- A **linearization** is a run consistent with happens-before ($\rightarrow$) relation in H.

- Linearizations pass through consistent global states.

Example

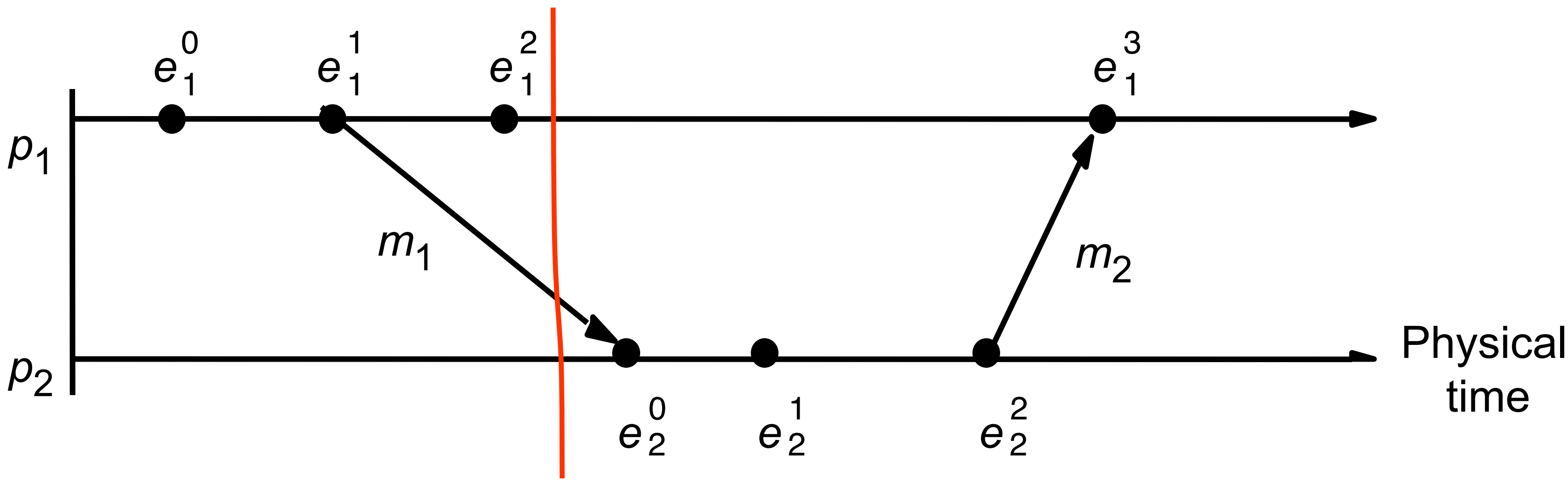


Order at p_1: $< e_1^0, e_1^1, e_1^2, e_1^3 >$ Order at p_2: $< e_2^0, e_2^1, e_2^2 >$

Causal order across p_1 and p_2: $< e_1^0, e_1^1, e_2^0, e_2^1, e_2^2, e_1^3 >$

Linearization: $< e_1^0, e_1^1, e_1^2, e_2^0, e_2^1\ e_2^2, e_1^3 >$

Example

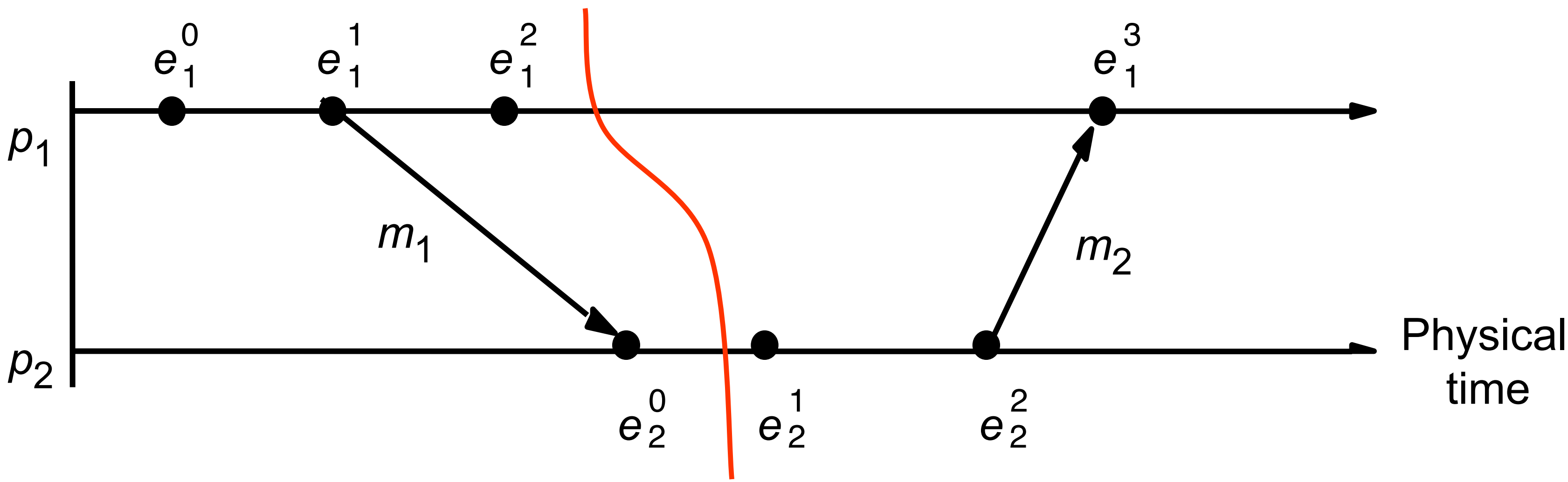


Order at p_1: < $e_1^0, e_1^1, e_1^2, e_1^3$ > Order at p_2: < e_2^0, e_2^1, e_2^2 >

Causal order across p_1 and p_2: < $e_1^0, e_1^1, e_2^0, e_2^1\ e_2^2, e_1^3$ >

Linearization: < $e_1^0, e_1^1, e_1^2, e_2^0, e_2^1\ e_2^2, e_1^3$ >

Example

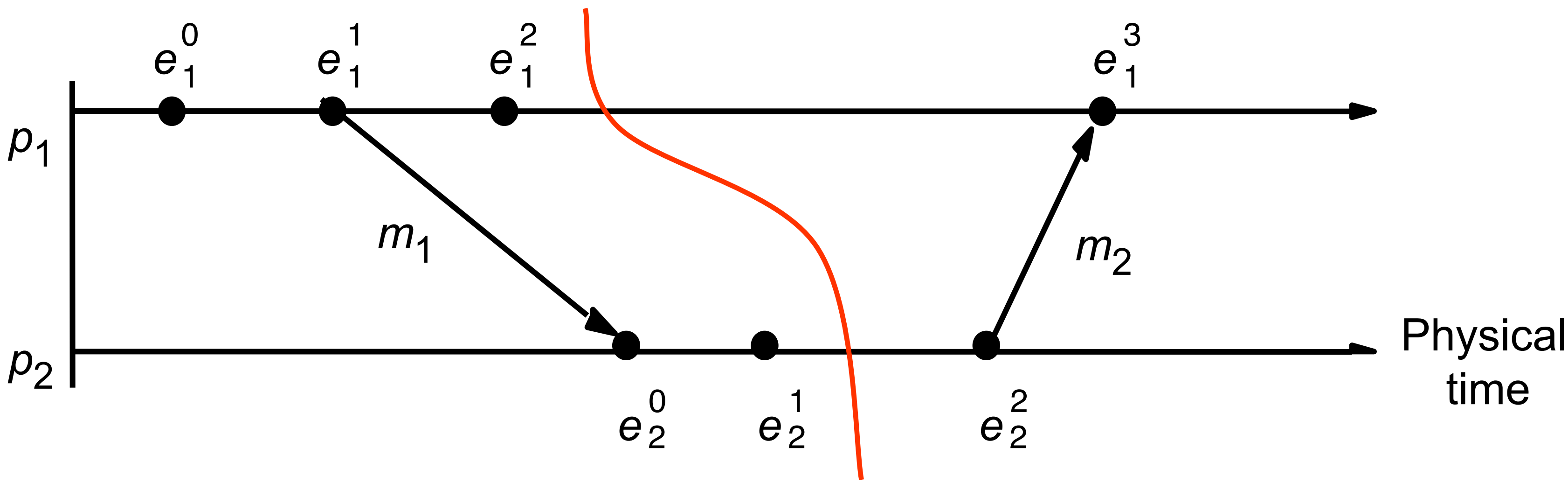


Order at p_1: < $e_1^0, e_1^1, e_1^2, e_1^3$ >   Order at p_2: < e_2^0, e_2^1, e_2^2 >

Causal order across p_1 and p_2: < $e_1^0, e_1^1, e_2^0, e_2^1$ e_2^2, e_1^3 >

Linearization: < $e_1^0, e_1^1, e_1^2, e_2^0,$ | e_2^1 e_2^2, e_1^3 >

Example

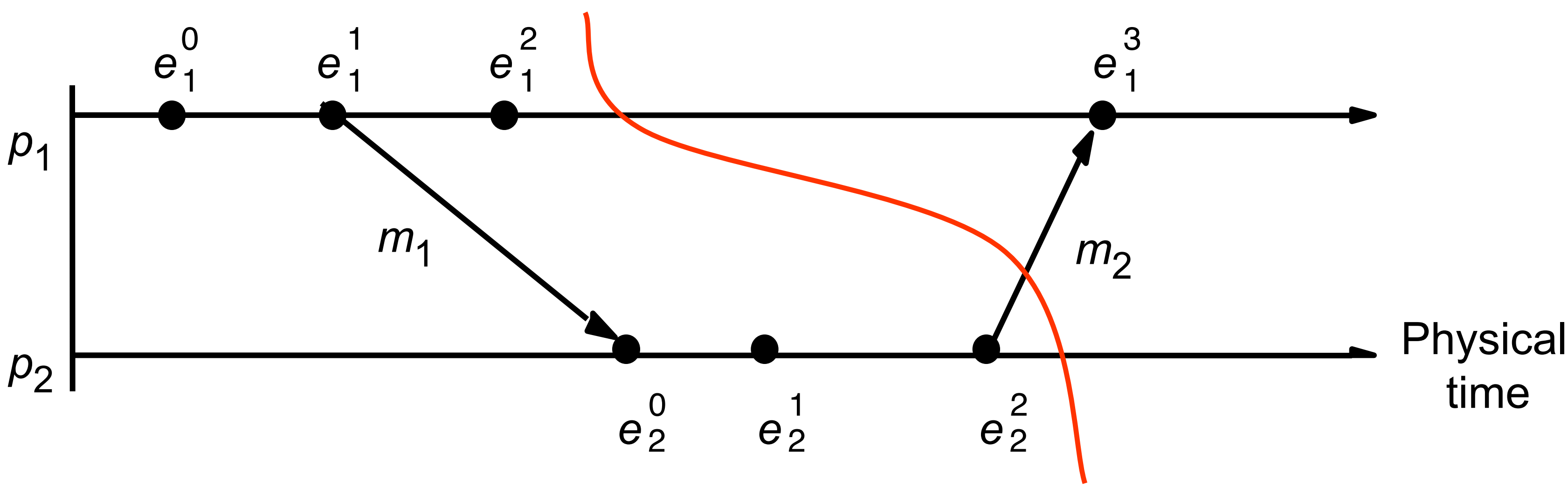


Order at p_1: $< e_1^0, e_1^1, e_1^2, e_1^3 >$ Order at p_2: $< e_2^0, e_2^1, e_2^2 >$

Causal order across p_1 and p_2: $< e_1^0, e_1^1, e_2^0, e_2^1\ e_2^2, e_1^3 >$

Linearization: $< e_1^0, e_1^1, e_1^2, e_2^0, e_2^1 \,|\, e_2^2, e_1^3 >$

Example

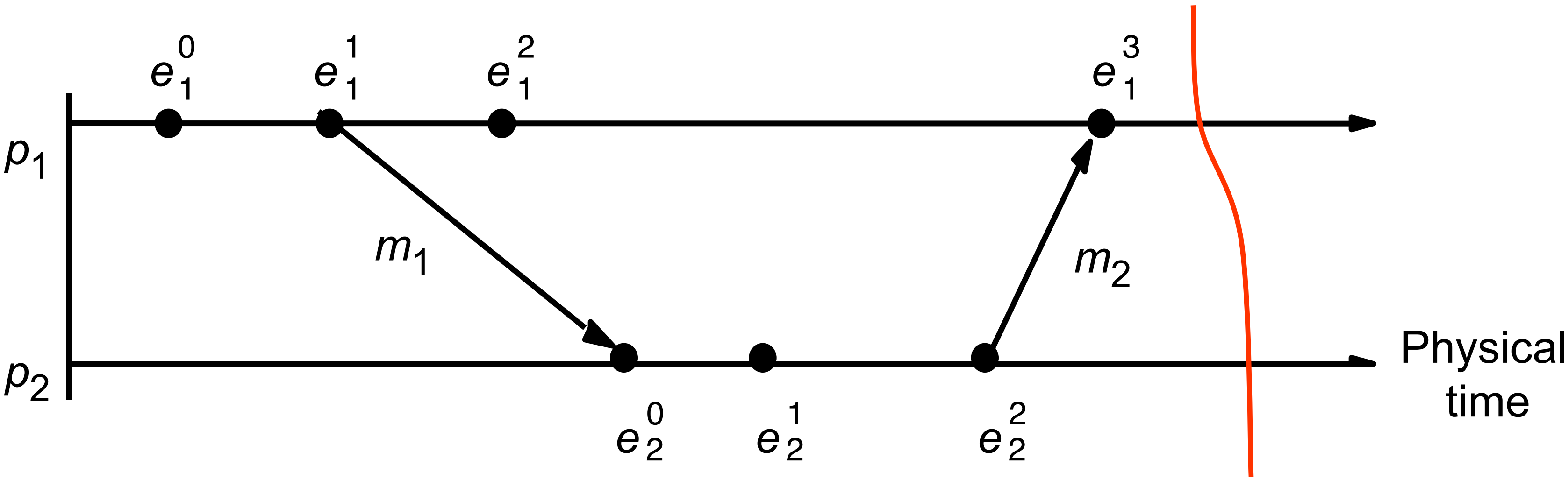


Order at p_1: $< e_1^0, e_1^1, e_1^2, e_1^3 >$ Order at p_2: $< e_2^0, e_2^1, e_2^2 >$

Causal order across p_1 and p_2: $< e_1^0, e_1^1, e_2^0, e_2^1\ e_2^2 , e_1^3 >$

Linearization: $< e_1^0, e_1^1, e_1^2, e_2^0, e_2^1\ e_2^2 , e_1^3 >$

Example

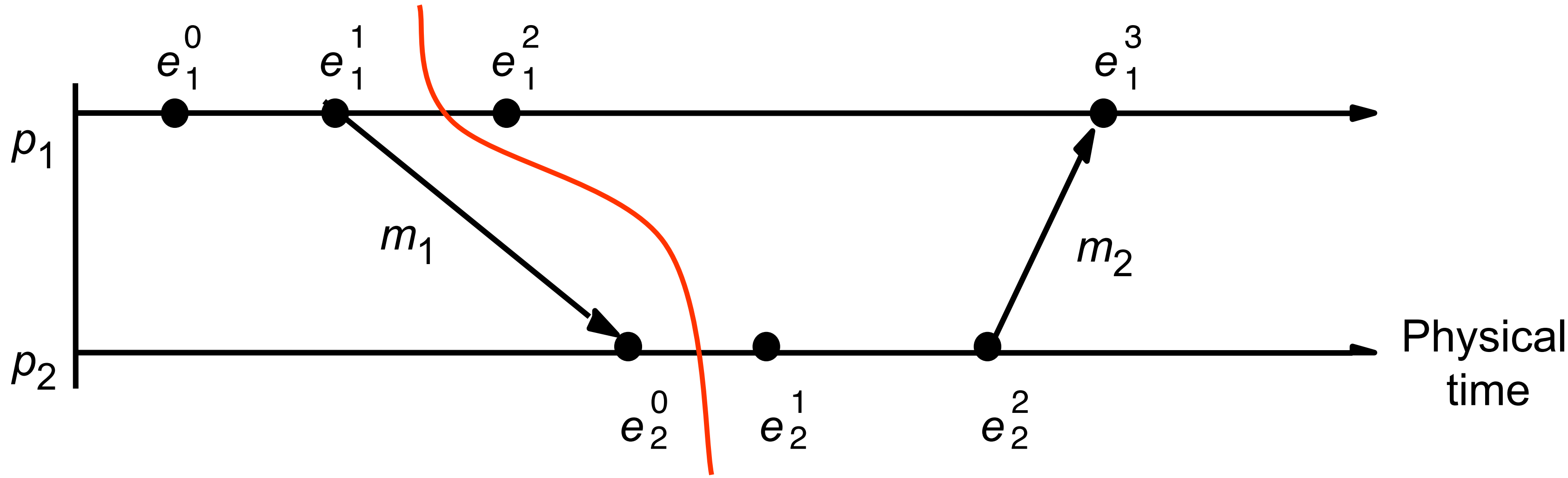


Order at p_1: < e_1^0, e_1^1, e_1^2, e_1^3 >   Order at p_2: < e_2^0, e_2^1,e_2^2>

Causal order across p_1 and p_2: < e_1^0, e_1^1, e_2^0, e_2^1 e_2^2 , e_1^3 >

Linearization: < e_1^0, e_1^1, e_1^2, e_2^0, e_2^1 e_2^2 , e_1^3 >

Example



Order at p_1: $< e_1^0, e_1^1, e_1^2, e_1^3 >$ Order at p_2: $< e_2^0, e_2^1, e_2^2 >$

Causal order across p_1 and p_2: $< e_1^0, e_1^1, e_2^0, e_2^1 \; e_2^2, e_1^3 >$

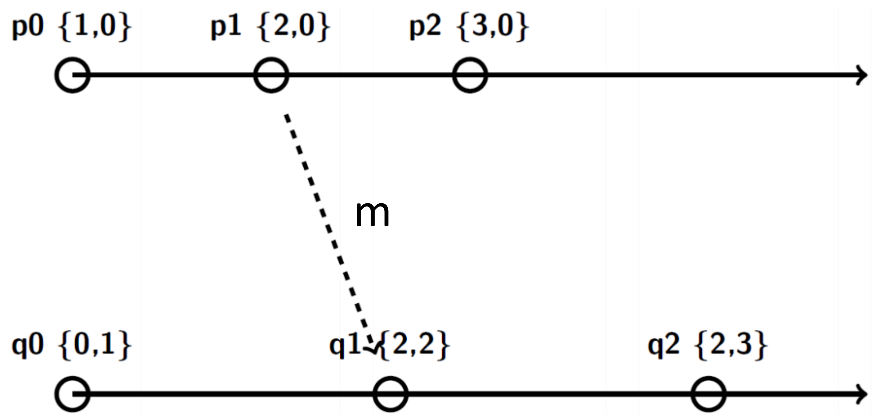
Linearization: $< e_1^0, e_1^1, e_1^2, e_2^0, e_2^1 \; e_2^2, e_1^3 >$

Linearization $< e_1^0, e_1^1, e_2^0, e_2^1, e_1^2, e_2^2, e_1^3 >$

More notations and definitions

- Linearizations pass through consistent global states.
- A global state $\mathbf{S_k}$ is reachable from global state $\mathbf{S_i}$, if there is a linearization that passes through $\mathbf{S_i}$ and then through $\mathbf{S_k}$.
- The distributed system evolves as a series of transitions between global states S_0 , S_1 , ….

State Transitions: Example



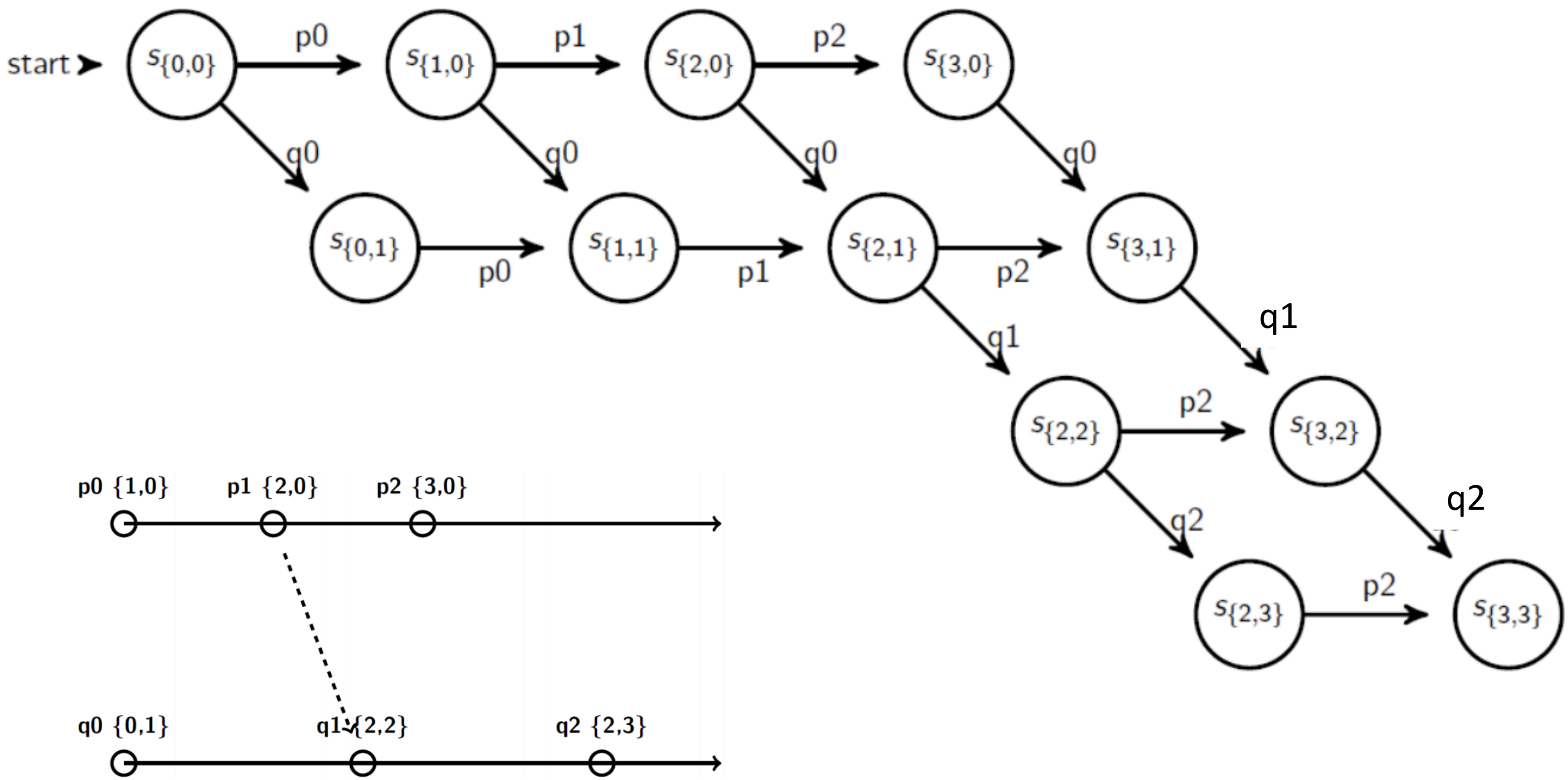
- Causal order:
  - p0 → p1 → p2
  - q0 → q1 → q2
  - p0 → p1 → q1 → q2
- Concurrent:
  - p0 || q0
  - p1 || q0
  - p2 || q0, p2 || q1, p2 || q2

Many linearizations:

- < p0, p1, p2, q0, q1, q2>
- < p0, q0, p1, q1, p2, q2>
- <q0, p0, p1, q1, p2, q2 >
- <q0, p0, p1, p2, q1,q2 >
- ……

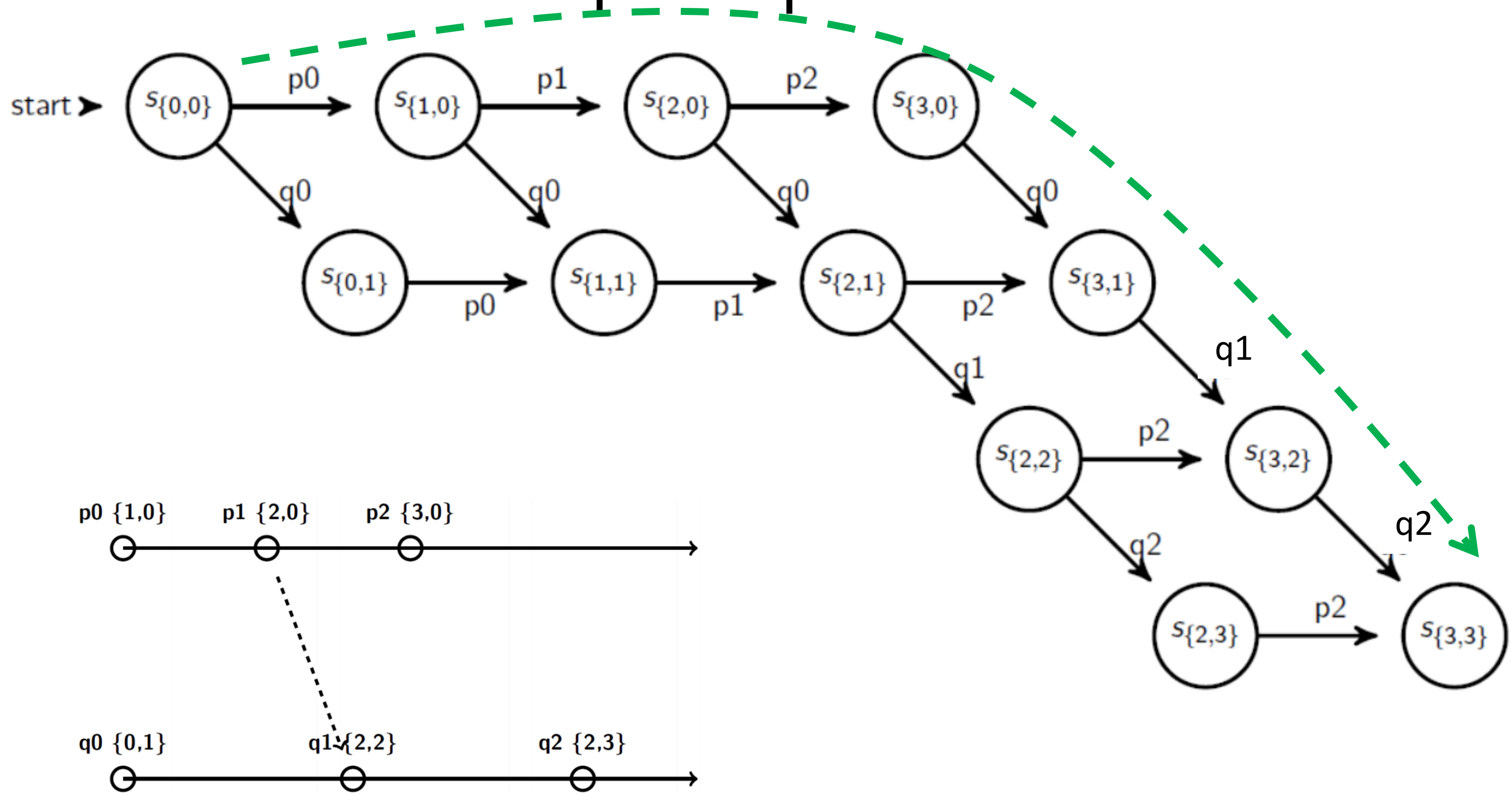
State Transitions: Example

Execution Lattice. Each path represents a linearization.



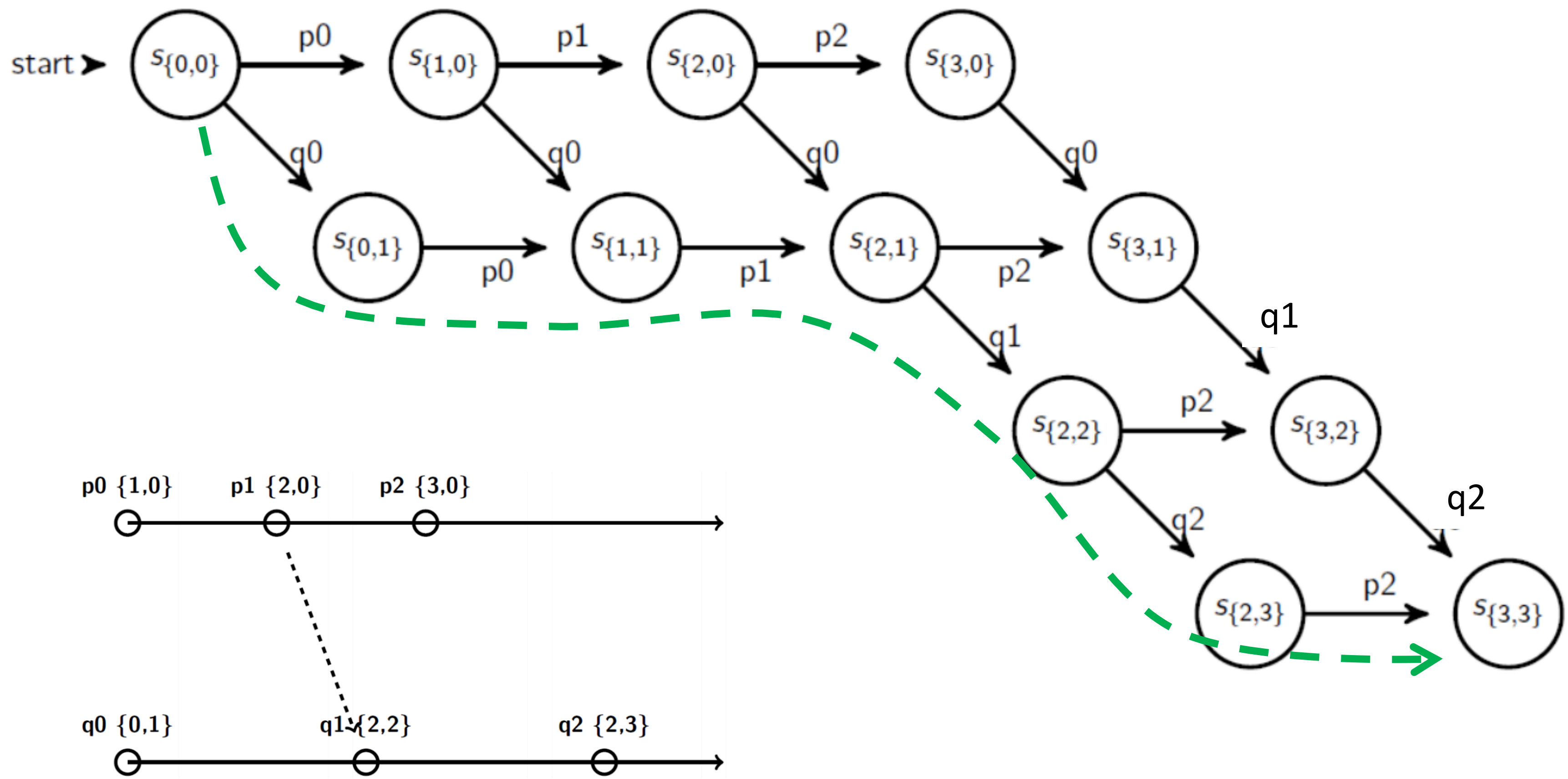
State Transitions: Example

Execution Lattice. Each path represents a linearization.



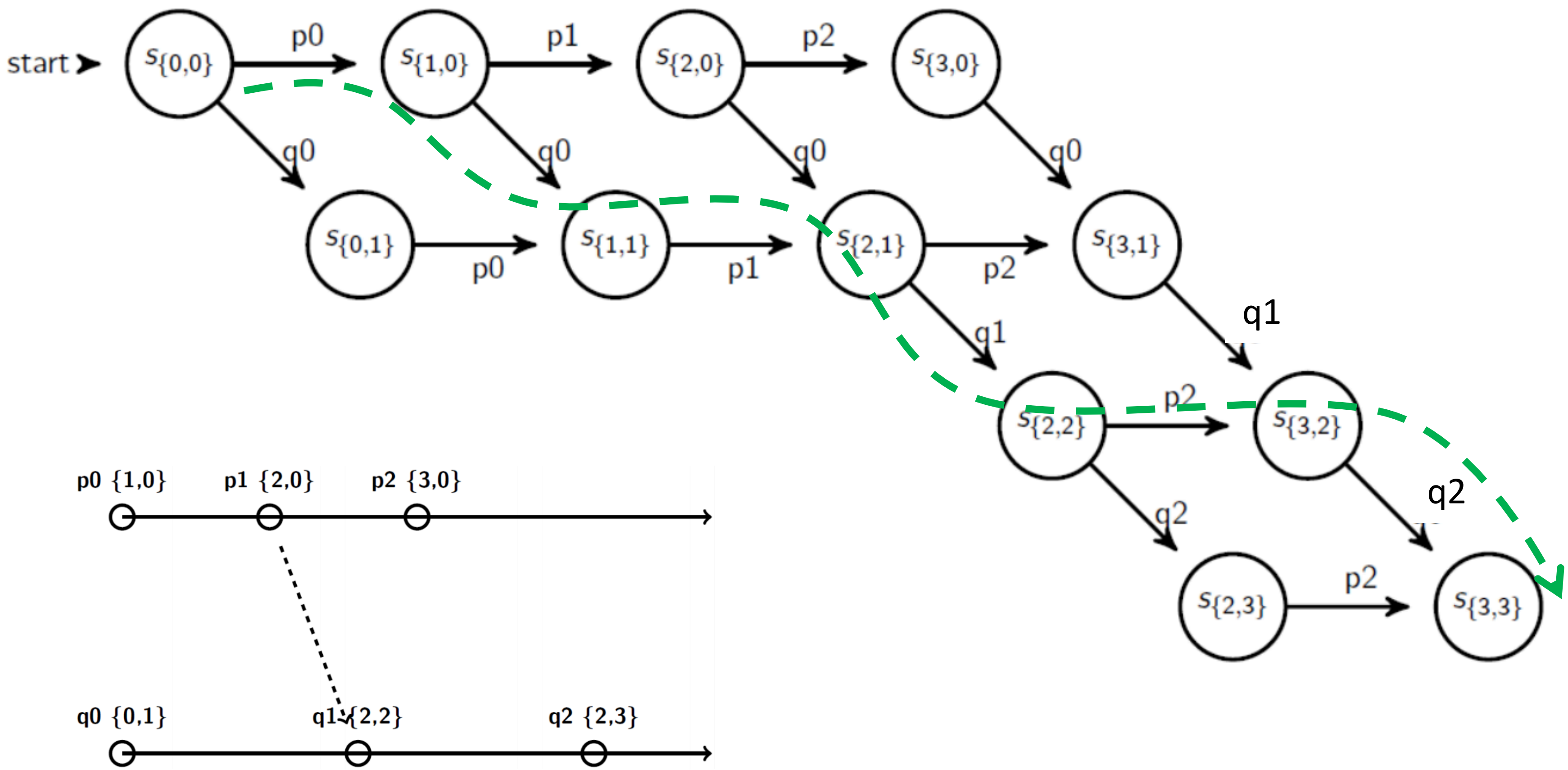
State Transitions: Example

Execution Lattice. Each path represents a linearization.

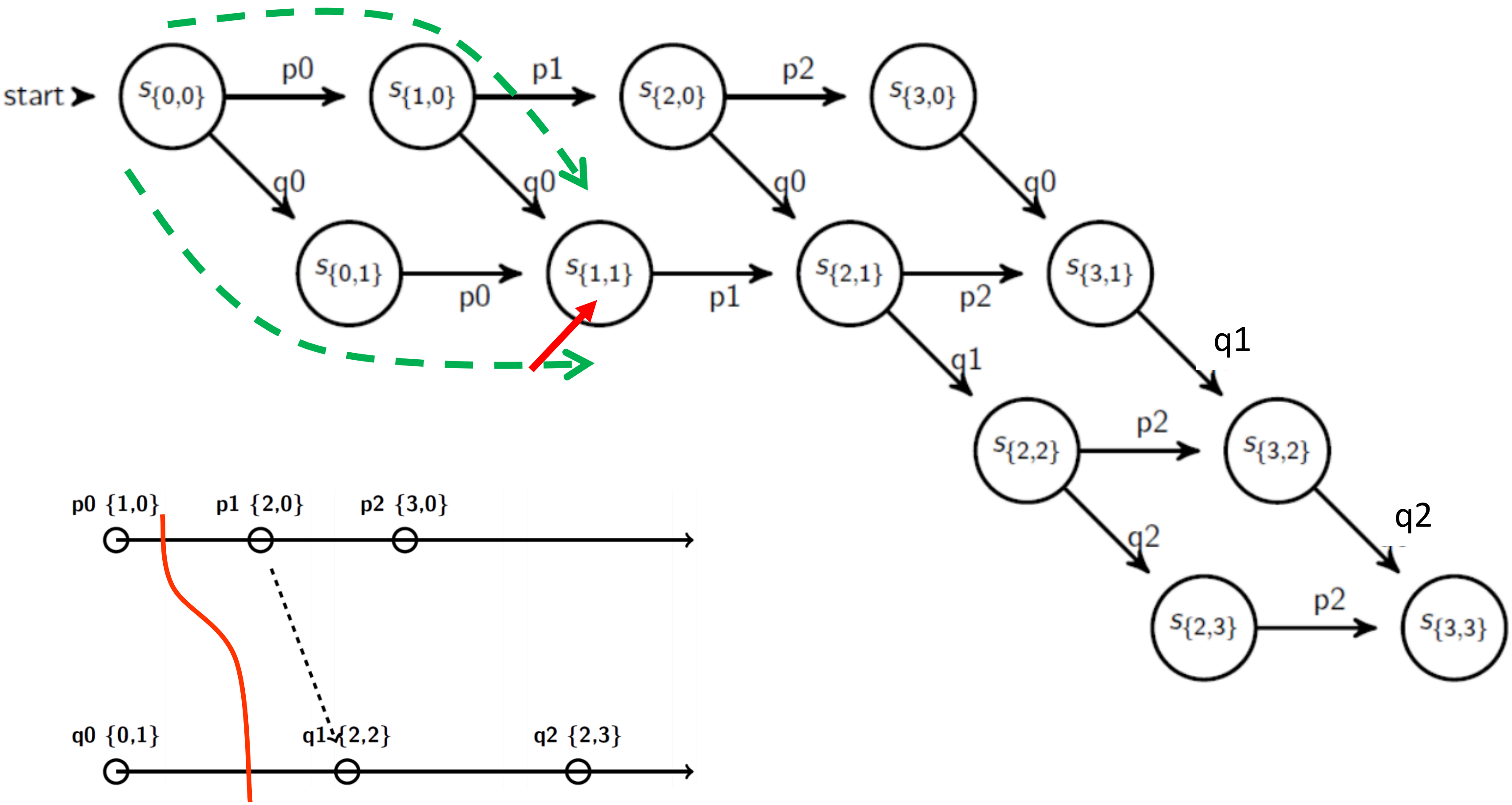


State Transitions: Example

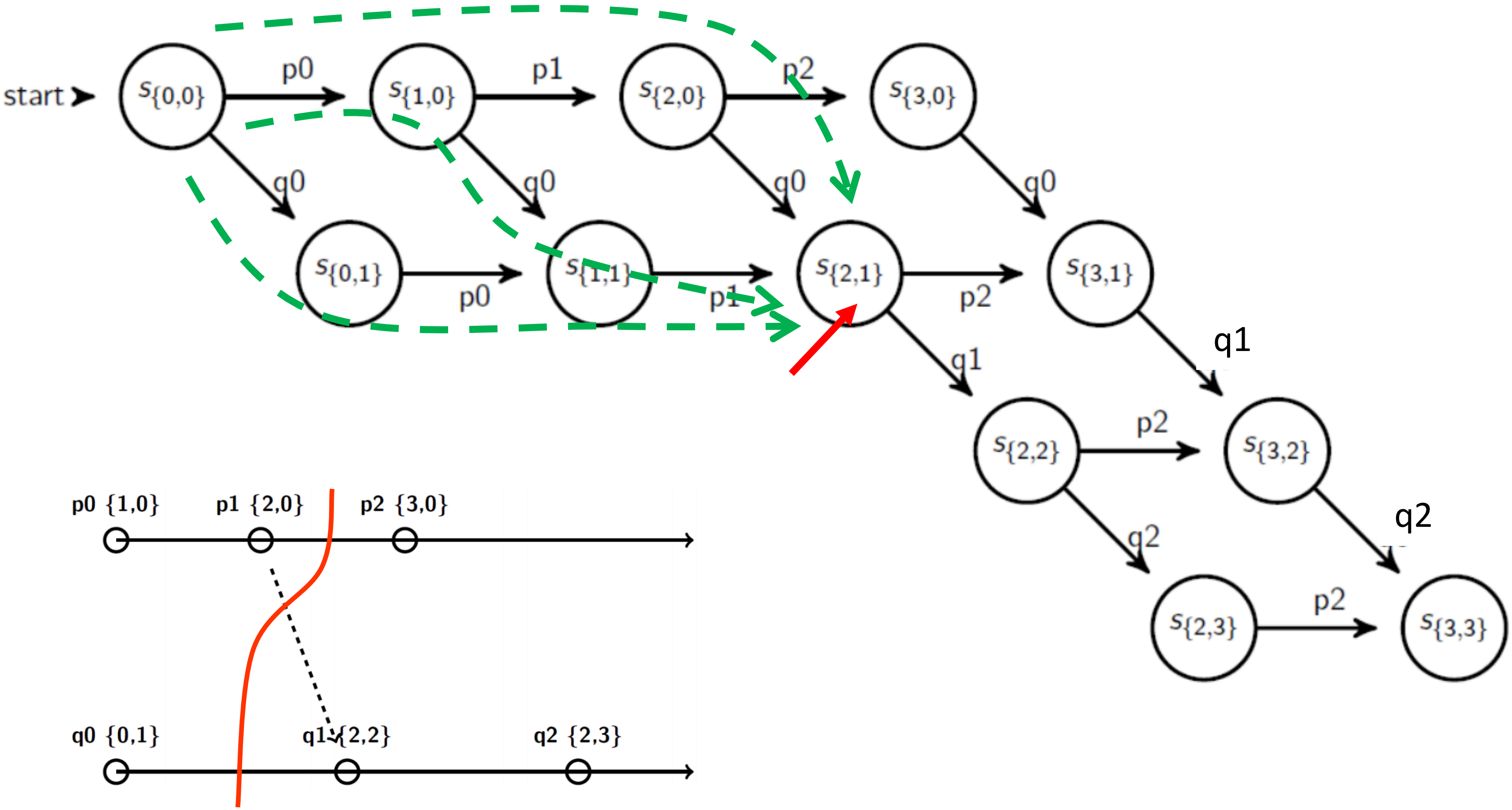
Execution Lattice. Each path represents a linearization.



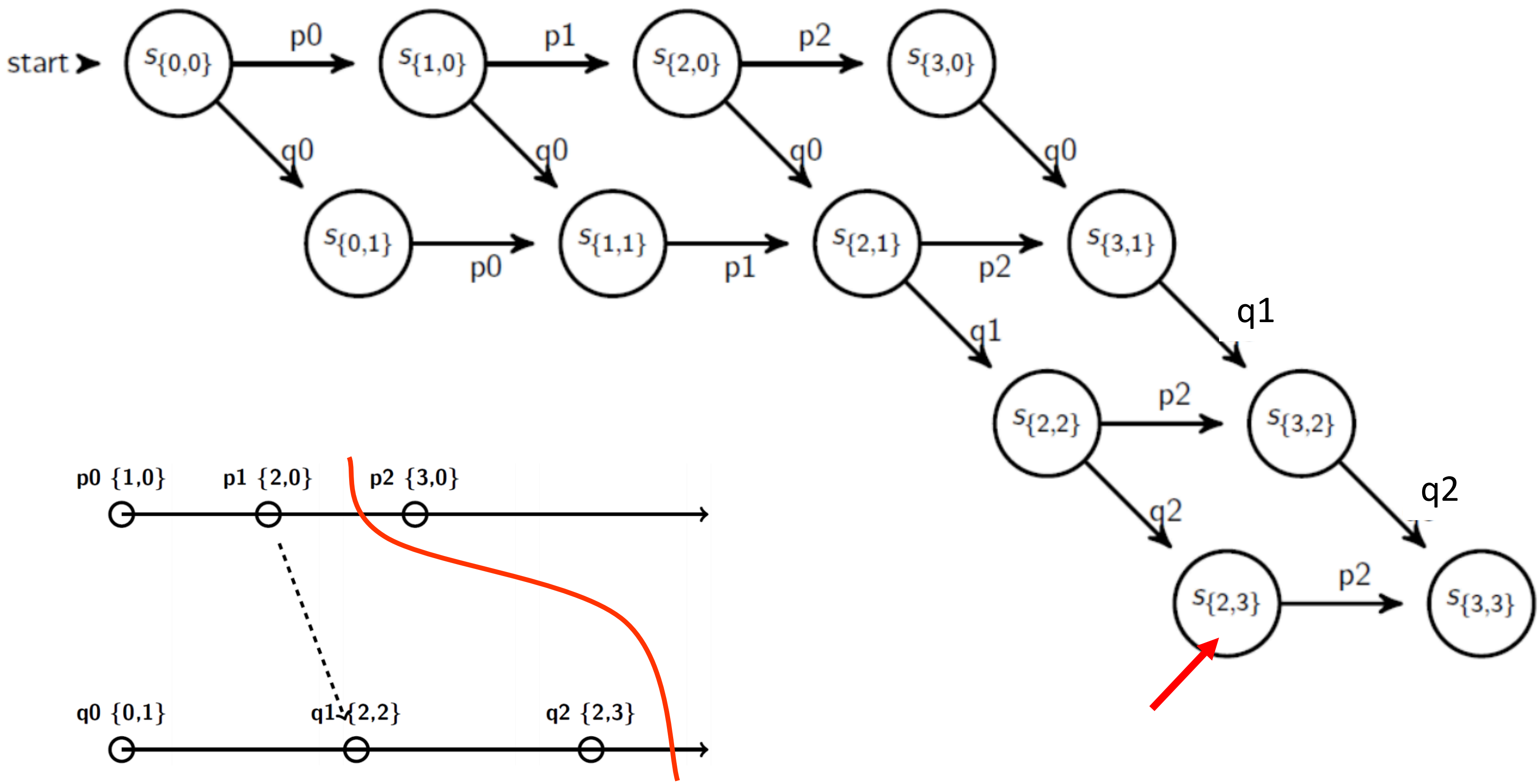
State Transitions: Example



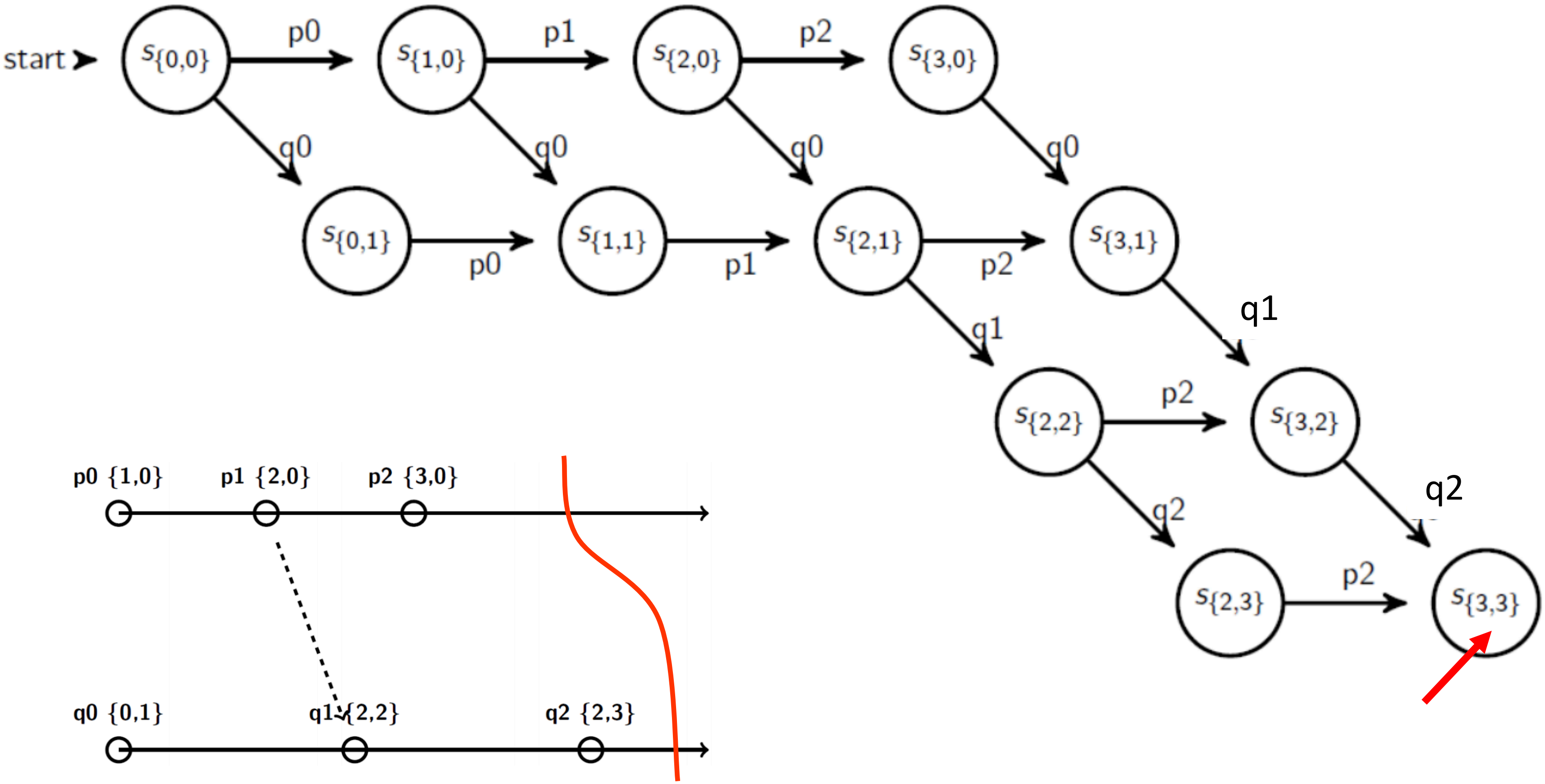
State Transitions: Example



State Transitions: Example



State Transitions: Example



More notations and definitions

- A run is a total ordering of events in H that is consistent with each $\mathbf{h_i}$'s ordering.
- A linearization is a run consistent with happens-before ($\rightarrow$) relation in H.
- Linearizations pass through consistent global states.
- A global state $\mathbf{S_k}$ is reachable from global state $\mathbf{S_i}$, if there is a linearization that passes through $\mathbf{S_i}$ and then through $\mathbf{S_k}$.
- The distributed system evolves as a series of transitions between global states S_0 , S_1 , ….

Global State Predicates

- A global-state-predicate is a property that is *true* or *false* for a global state.
  - Is there a deadlock?
  - Has the distributed algorithm terminated?
- Two ways of reasoning about predicates (or system properties) as global state gets transformed by events.
  - Liveness
  - Safety

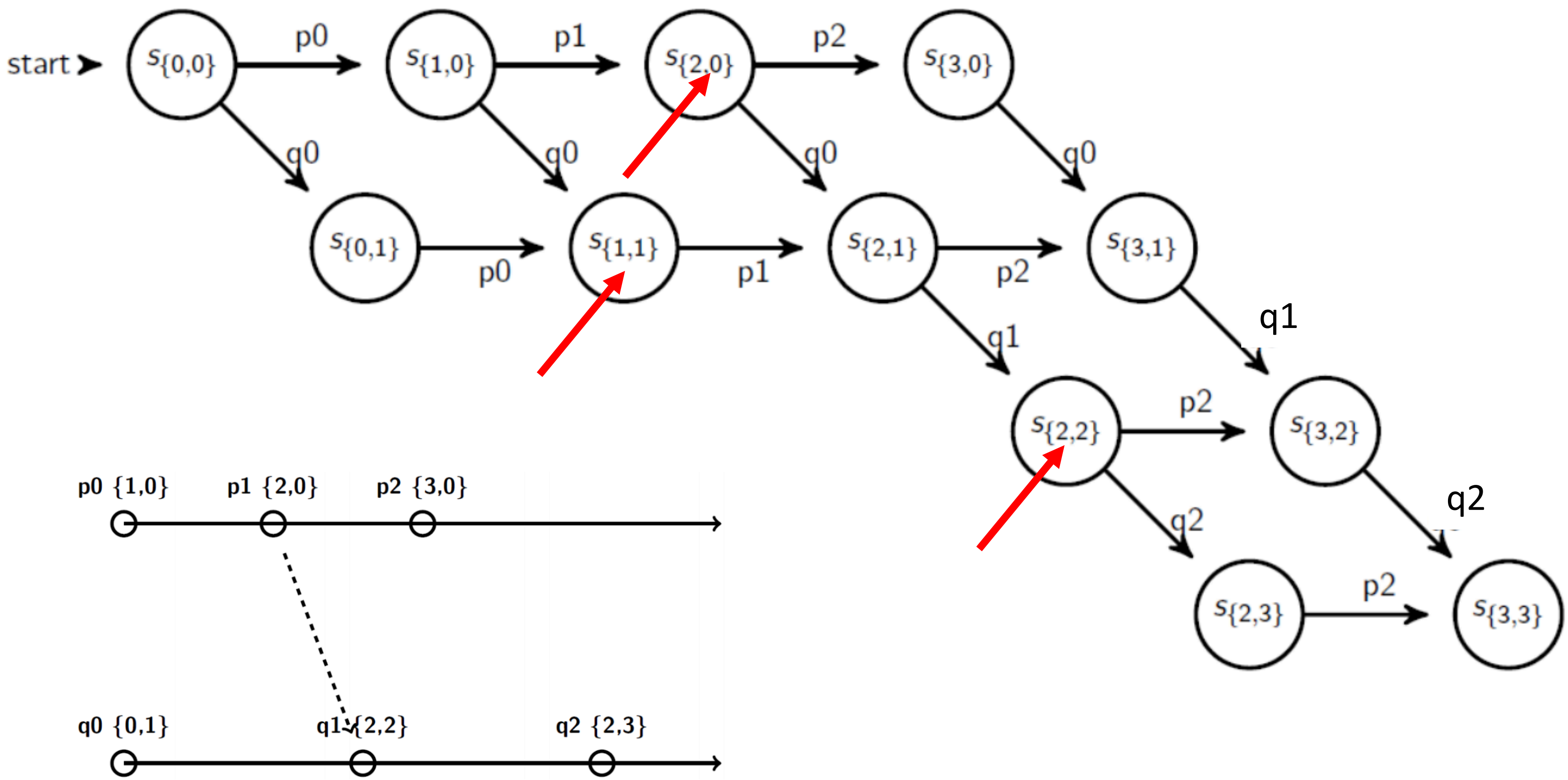
Liveness

- **Liveness = guarantee that something good will happen, eventually**
- **Examples:**
  - A distributed computation will terminate.
  - "Completeness" in failure detectors: the failure will be detected.
  - All processes will eventually decide on a value.
- A global state S_0 satisfies a **liveness** property P iff:
  - For all linearizations starting from S_0, P is true for some state S_L reachable from S_0.
  - $liveness(P(S_0)) \equiv \forall L \in$ linearizations from S_0, L passes through a S_L & $P(S_L)$ = true

Liveness Example

If predicate is true only in the marked states, does it satisfy liveness?

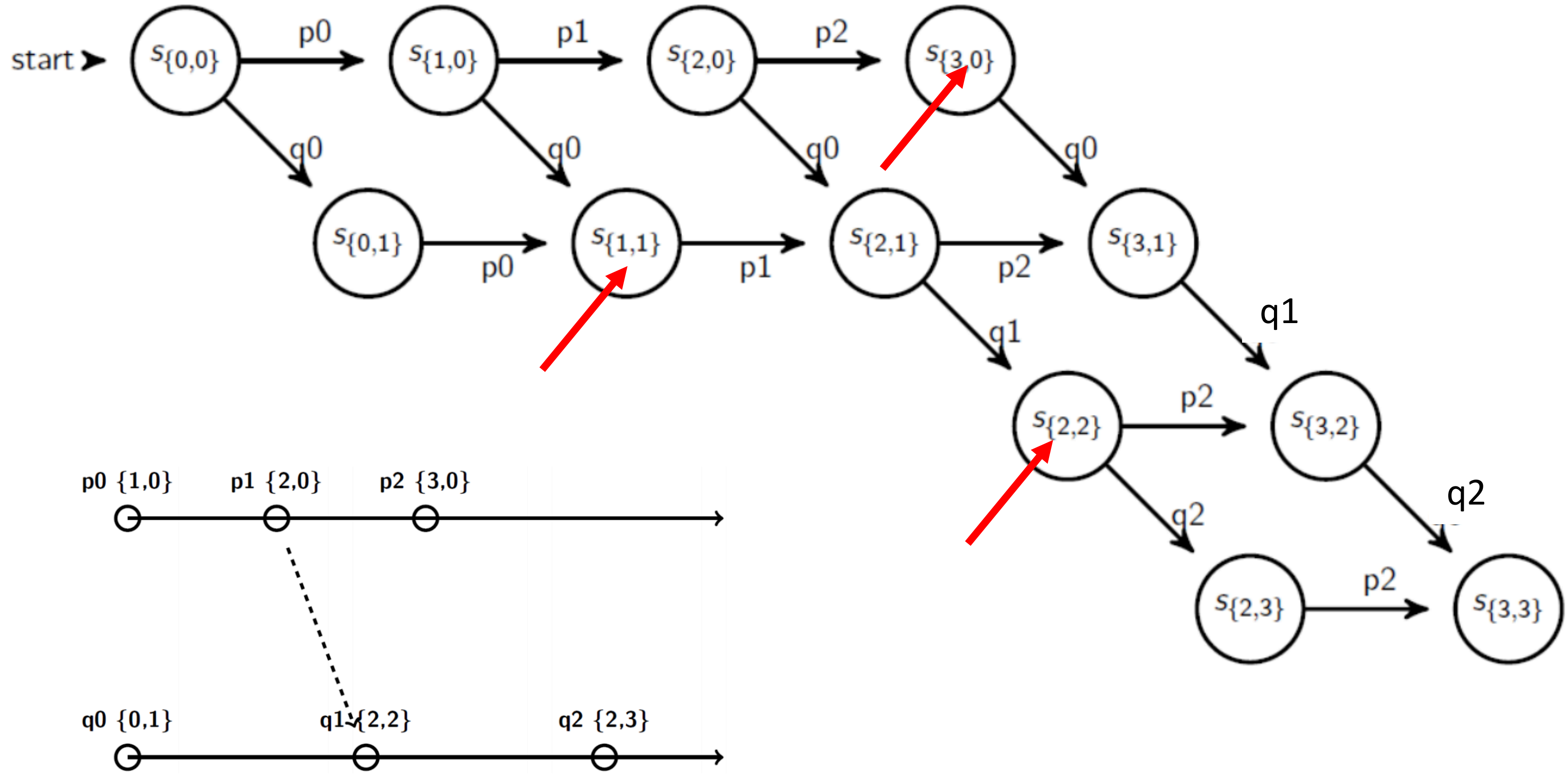
Yes



Liveness Example

If predicate is true only in the marked states, does it satisfy liveness?

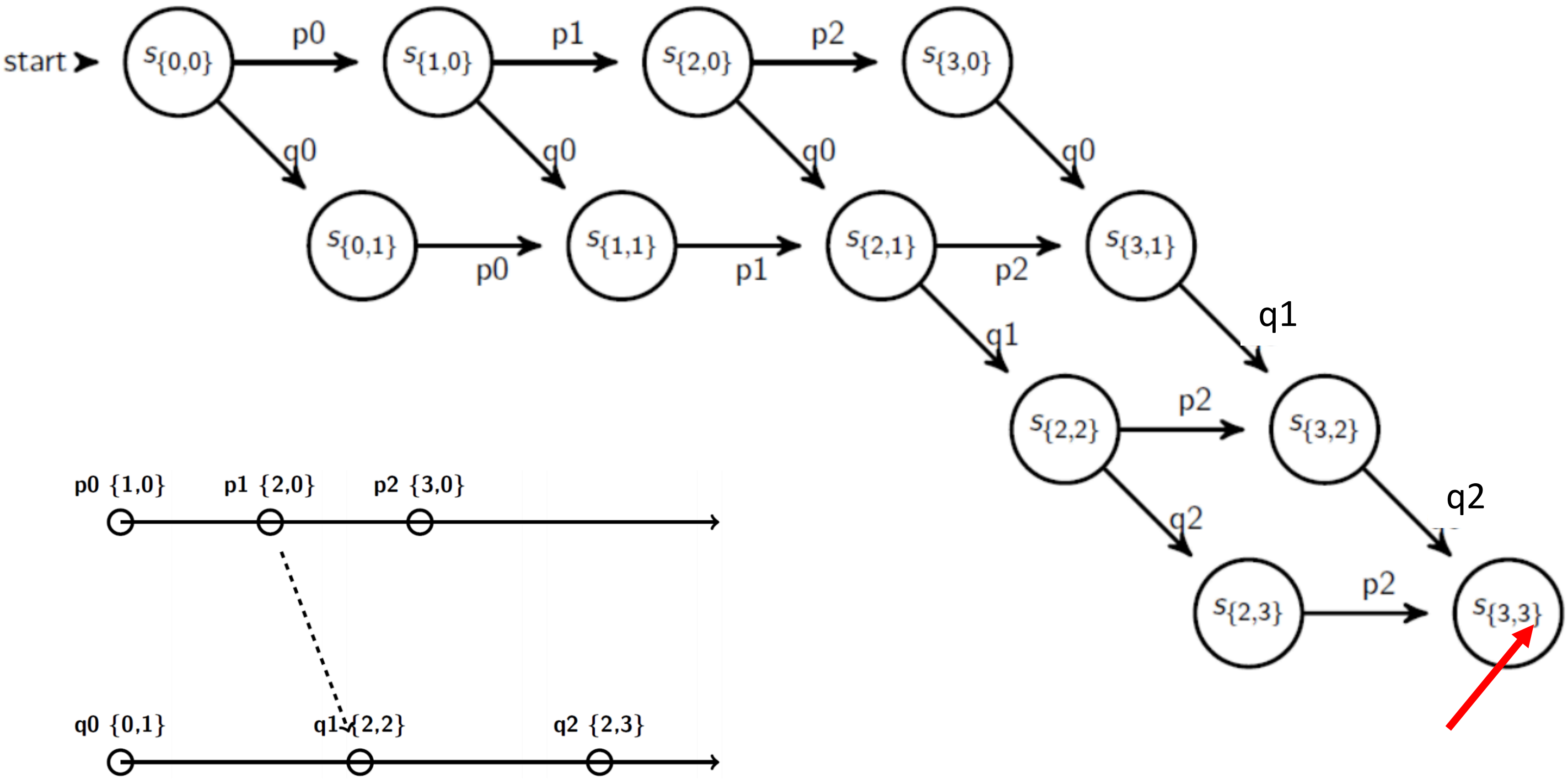
No



Liveness Example

If predicate is true only in the marked states, does it satisfy liveness?

Yes



Liveness

- Liveness = **guarantee that something good will happen,** eventually

- **Examples:**
  - A distributed computation will terminate.
  - "Completeness" in failure detectors: the failure will be detected.
  - All processes will eventually decide on a value.

- A global state S_0 satisfies a **liveness** property P iff:
  - liveness(P(S_0)) ≡ ∀L∈ linearizations from S_0, L passes through a S_L & P(S_L) = true
  - For any linearization starting from S_0, P is true for some state S_L reachable from S_0.

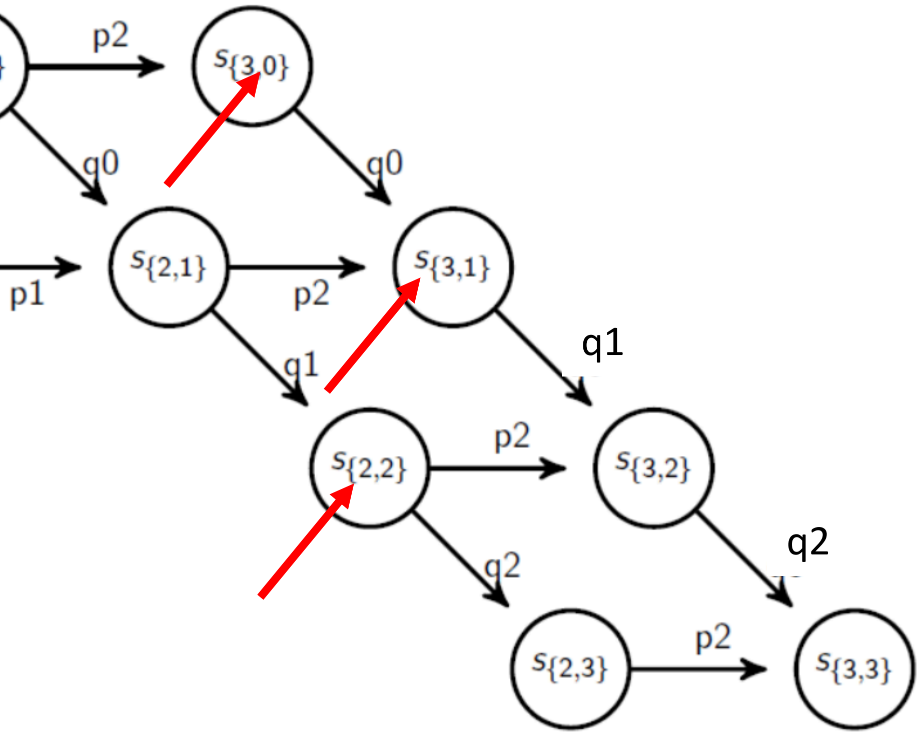
Safety

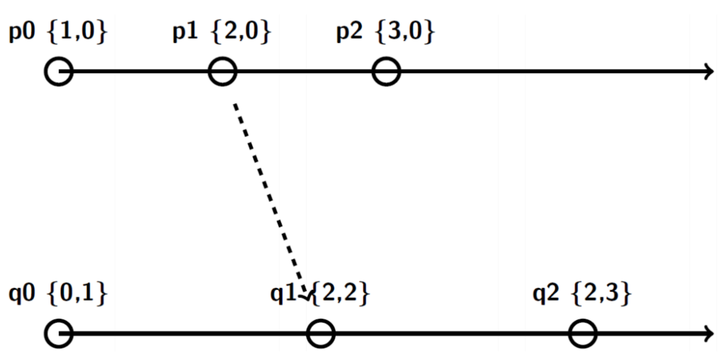
- Safety = guarantee that something bad will never happen.
- Examples:
  - There is no deadlock in a distributed transaction system.
  - "Accuracy" in failure detectors: an alive process is not detected as failed.
  - No two processes decide on different values.
- A global state S_0 satisfies a **safety** property P iff:
  - For all states S reachable from S_0, P(S) is true.
  - $safety(P(S_0)) \equiv \forall S$ reachable from S_0, $P(S) = true$.

Safety Example

If predicate is true only in the marked states, does it satisfy safety?

No

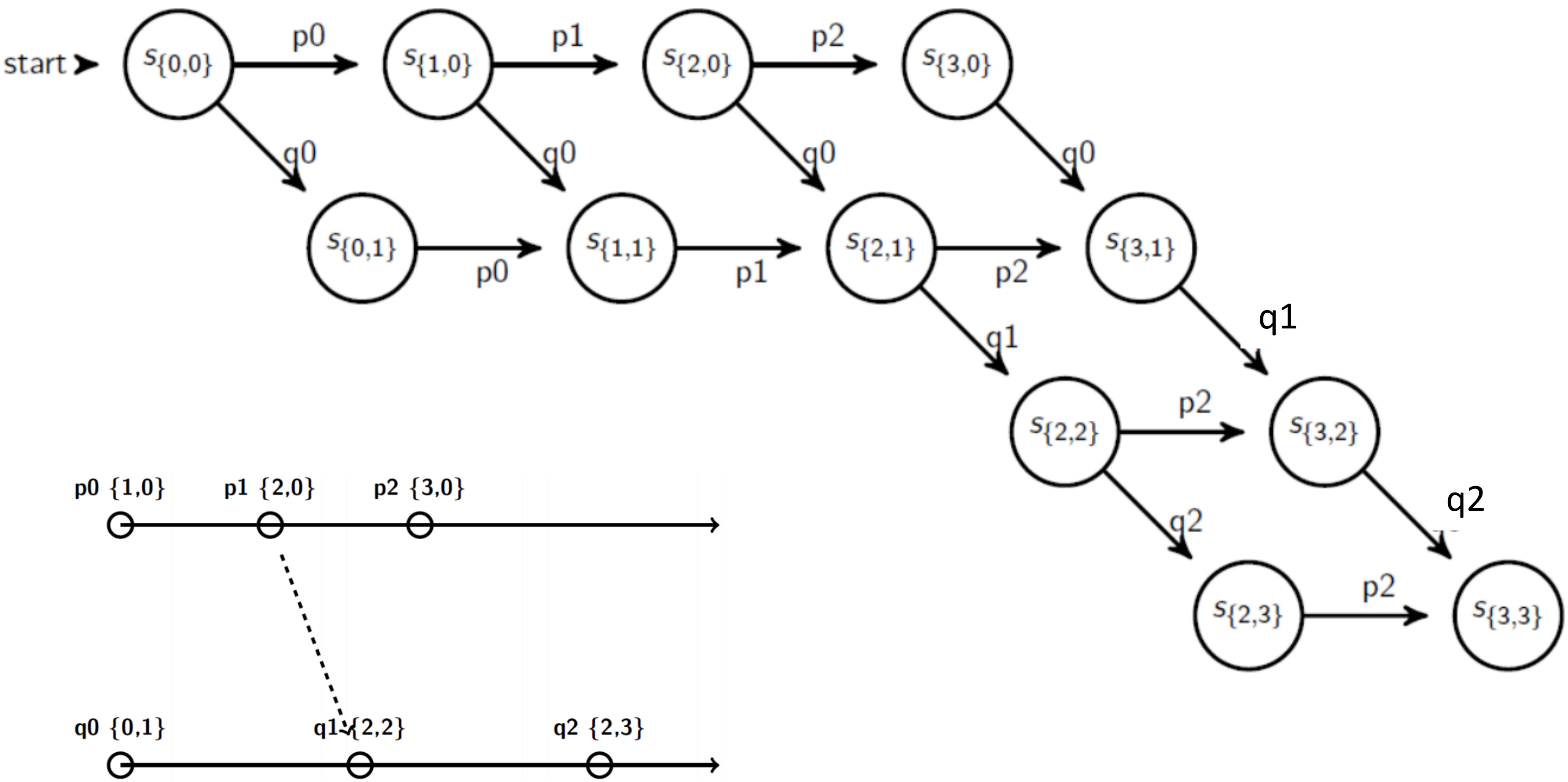




Safety Example

If predicate is true only in the **unmarked** states, does it satisfy safety?

Yes

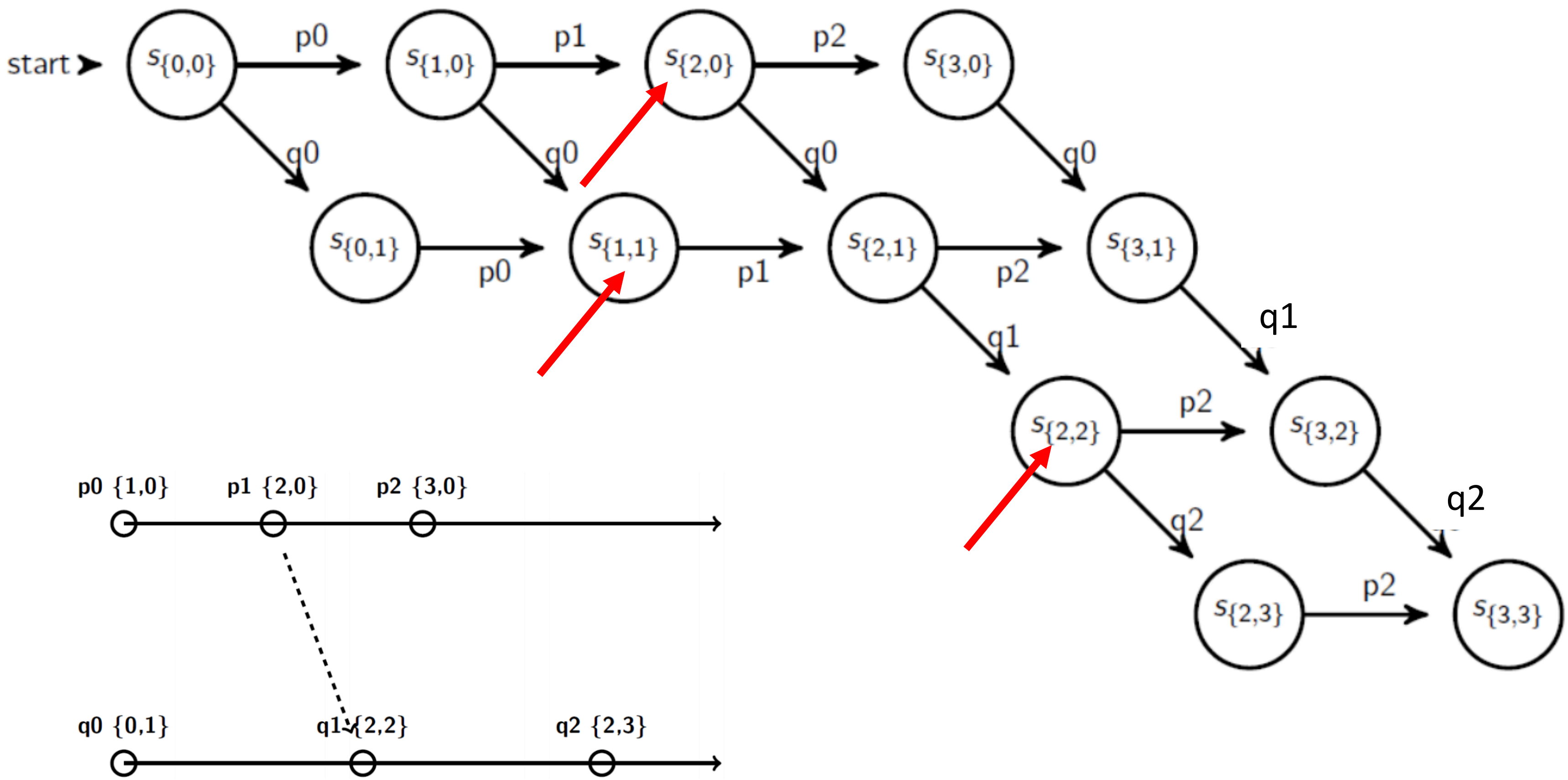


Safety

- **Safety = guarantee that something bad will never happen.**
- **Examples:**
  - There is no deadlock in a distributed transaction system.
  - "Accuracy" in failure detectors: an alive process is not detected as failed.
  - No two processes decide on different values.
- A global state S_0 satisfies a **safety** property P iff:
  - $\text{safety}(P(S_0)) \equiv \forall S$ reachable from S_0, $P(S) = \text{true}$.
  - For all states S reachable from S_0, P(S) is true.

Liveness Example

Technically satisfies liveness, but difficult to capture or reason about.



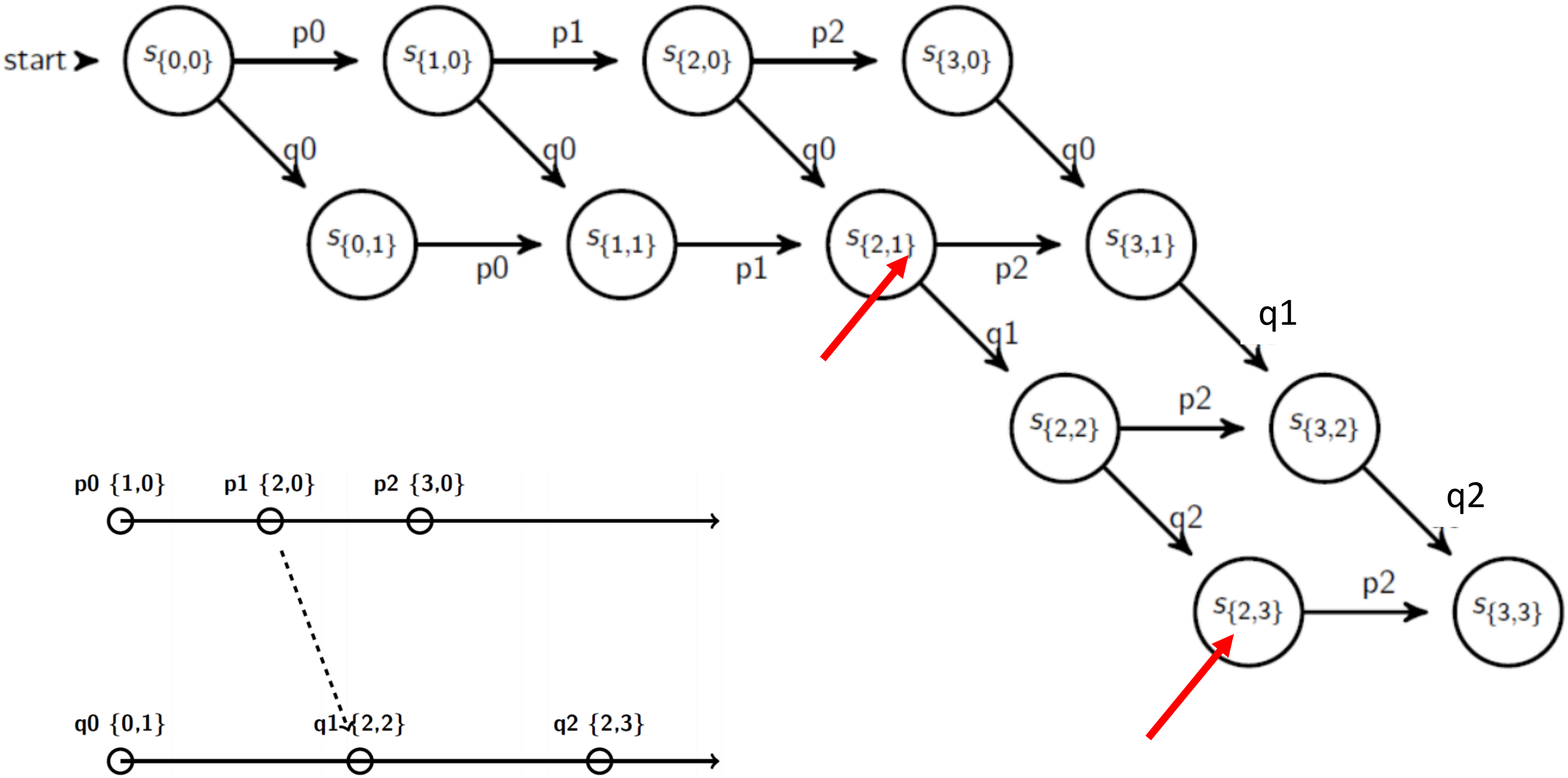
Stable Global Predicates

- once true, stays true forever afterwards (for stable liveness)

Stable Global Predicates

If predicate is true only in the marked states, is it stable?

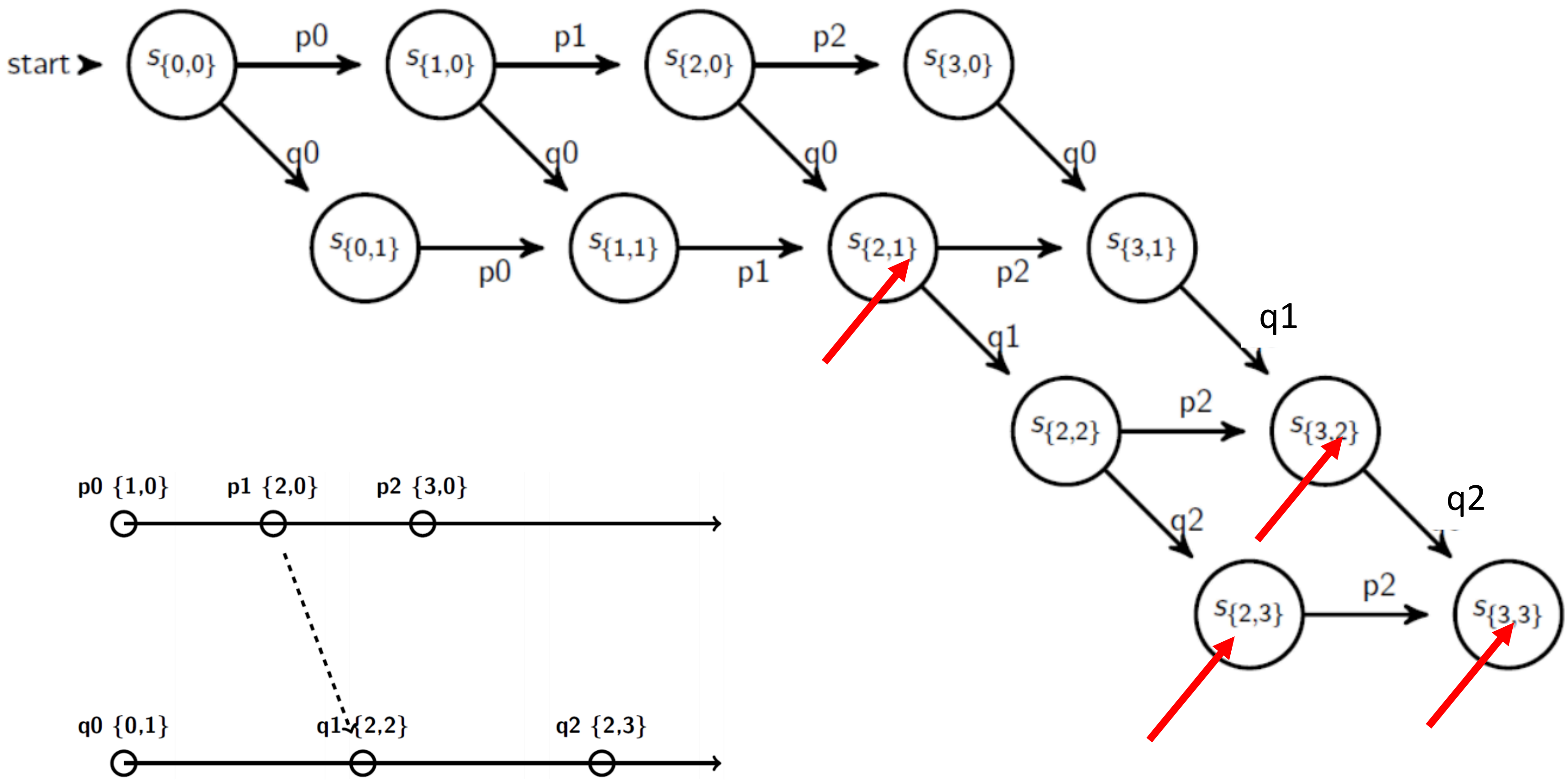
No



Stable Global Predicates

If predicate is true only in the marked states, is it stable?

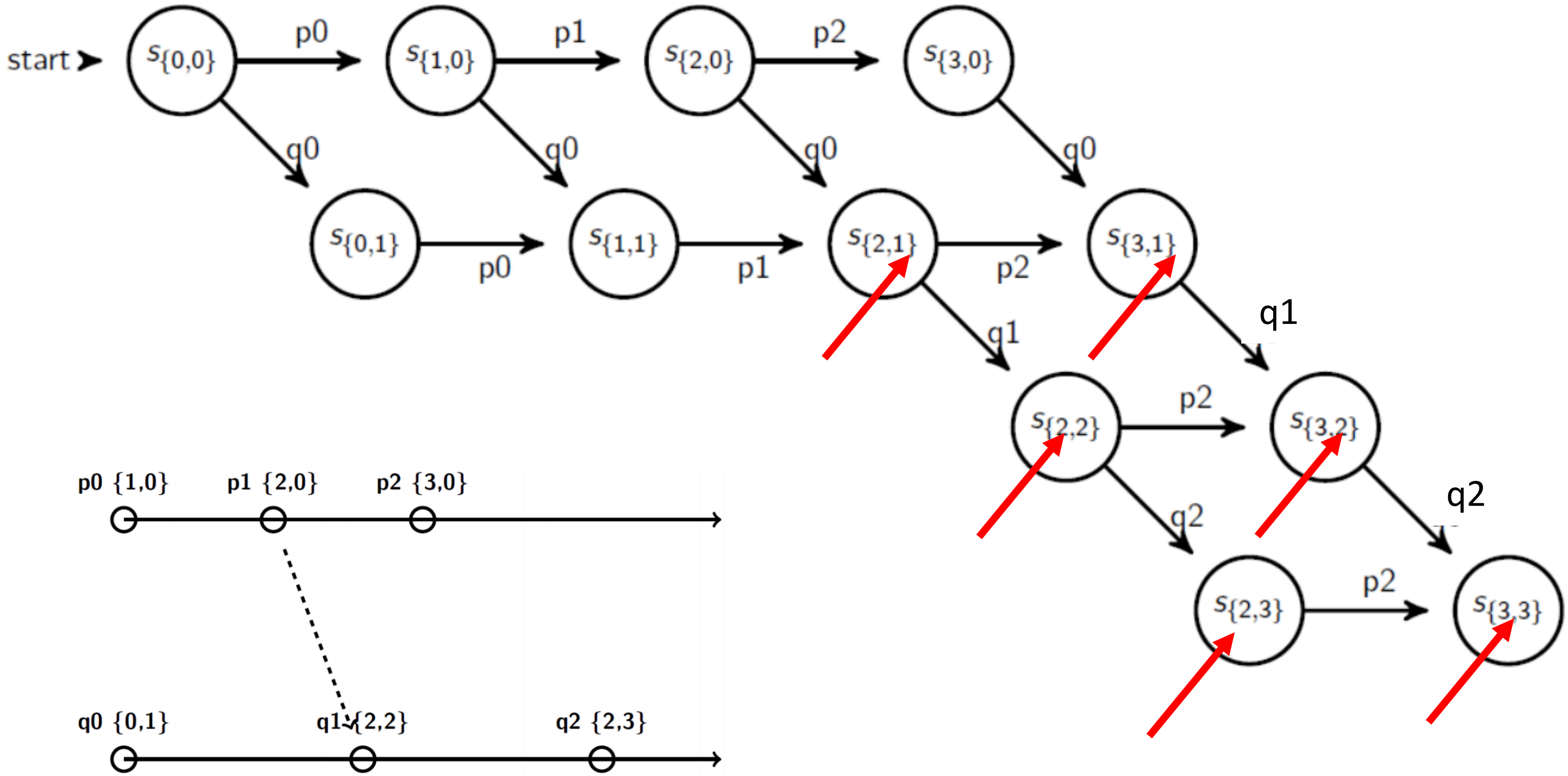
No



Stable Global Predicates

If predicate is true only in the marked states, is it stable?

Yes



Stable Global Predicates

- once true for a state S, stays true for all states reachable from S (for stable liveness)
- once false for a state S, stays false for all states reachable from S (for stable non-safety)
- Stable liveness examples (once true, always true)
  - Computation has terminated.
- Stable non-safety examples (once false, always false)
  - There is no deadlock.
  - An object is not orphaned.
- *All stable global properties can be detected using the Chandy-Lamport algorithm.*

Global Snapshot Summary

- The ability to calculate global snapshots in a distributed system is very important.
- But don't want to interrupt running distributed application.
- Chandy-Lamport algorithm calculates global snapshot.
- Obeys causality (creates a consistent cut).
- Can be used to detect global properties.
- Safety vs. Liveness.

Rest of today's agenda

- **Multicast**
  - Chapter 15.4

- **Goal:** reason about desirable properties for message delivery among a group of processes.

Communication modes

- **Unicast**
  - Messages are sent from exactly one process to one process.
- **Broadcast**
  - Messages are sent from exactly one process to all processes on the network.
- **Multicast**
  - Messages broadcast within a group of processes.
  - A multicast message is sent from any one process to a group of processes on the network.

Where is multicast used?

- Distributed storage
  - Write to an object are multicast across replica servers.
  - Membership information (e.g., heartbeats) is multicast across all servers in cluster.
- Online scoreboards (ESPN, French Open, FIFA World Cup)
  - Multicast to group of clients interested in the scores.
- Stock Exchanges
  - Group is the set of broker computers.
- ……

Communication modes

- **Unicast**
  - Messages are sent from exactly one process to one process.
    - *Best effort:* if a message is delivered it would be intact; no reliability guarantees.
    - *Reliable:* guarantees delivery of messages.
    - *In order:* messages will be delivered in the same order that they are sent.
- **Broadcast**
  - Messages are sent from exactly one process to all processes on the network.
- **Multicast**
  - Messages broadcast within a group of processes.
  - A multicast message is sent from any one process to the group of processes on the network.
  - *How do we define (and achieve) reliable or ordered multicast? (next class)*