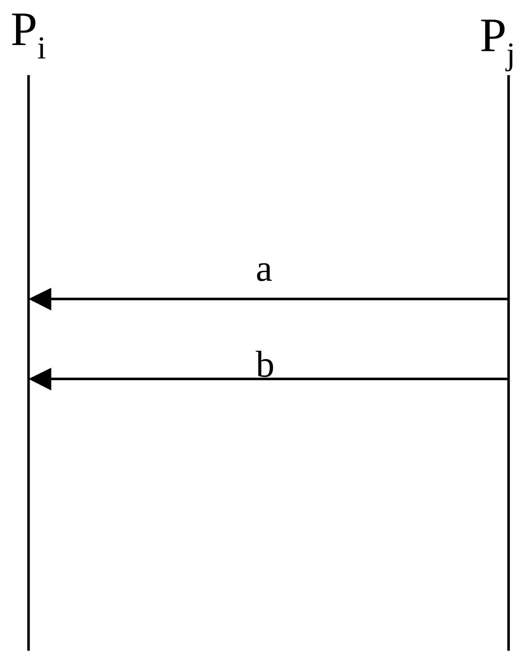
Snapshots

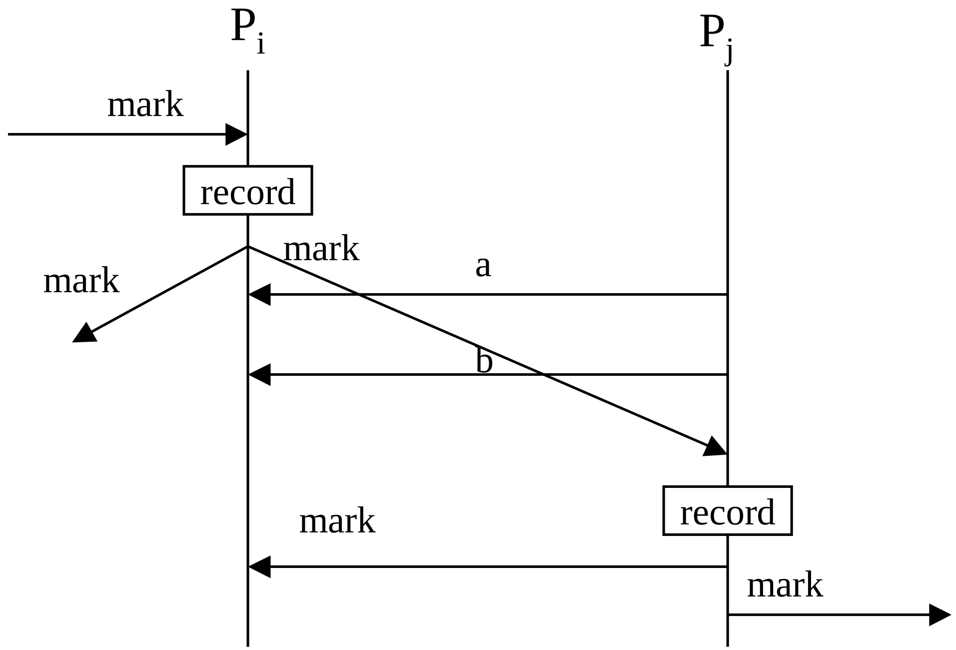
- **Chandy-Lamport Algorithm for the determination of consistent global states**
  - principle of operation
    - broadcast marker
    - upon receipt of marker record own state, and record any incoming message from another process until that process has recorded its state (these messages then belong to the channel between the processes)
    - processes may record their state at different points in time, but the differential is always accounted for by the state of the channel in between



  tele

Snapshots

- **Chandy-Lamport Algorithm for the determination of consistent global states**
  - principle of operation
    - broadcast marker
    - upon receipt of marker record own state, and record any incoming message from another process until that process has recorded its state (these messages then belong to the channel between the processes)
    - processes may record their state at different points in time, but the differential is always accounted for by the state of the channel in between



Snapshots

- **Chandy-Lamport Algorithm for the determination of consistent global states**
  - any number of processes may at any time concurrently initiate snapshot-taking
  - a process initiating snapshot-taking follows the marker receiving rule (see below)
  - marker sending rule
    - a) record own state
    - b) broadcast marker
    - a) and b) must preceed any other local actions or message send / receive events
  - marker receiving rule
    - if P_i has not yet recorded own state (first marker is being received)
      - record own state
      - start recording all messages received on all incoming channels
    - if P_i has already recorded own state
      - record state of channel on which marker was received
      - stop recording that channel

Snapshots

- **Chandy-Lamport Algorithm for the determination of consistent global states**

Chandy and Lamport's 'snapshot' algorithm

Marker receiving rule for process p_i

On p_i's receipt of a *marker* message over channel c:

if (p_i has not yet recorded its state) it

records its process state now;

records the state of c as the empty set;

turns on recording of messages arriving over other incoming channels;

else

p_i records the state of c as the set of messages it has received over c since it saved its state.

end if

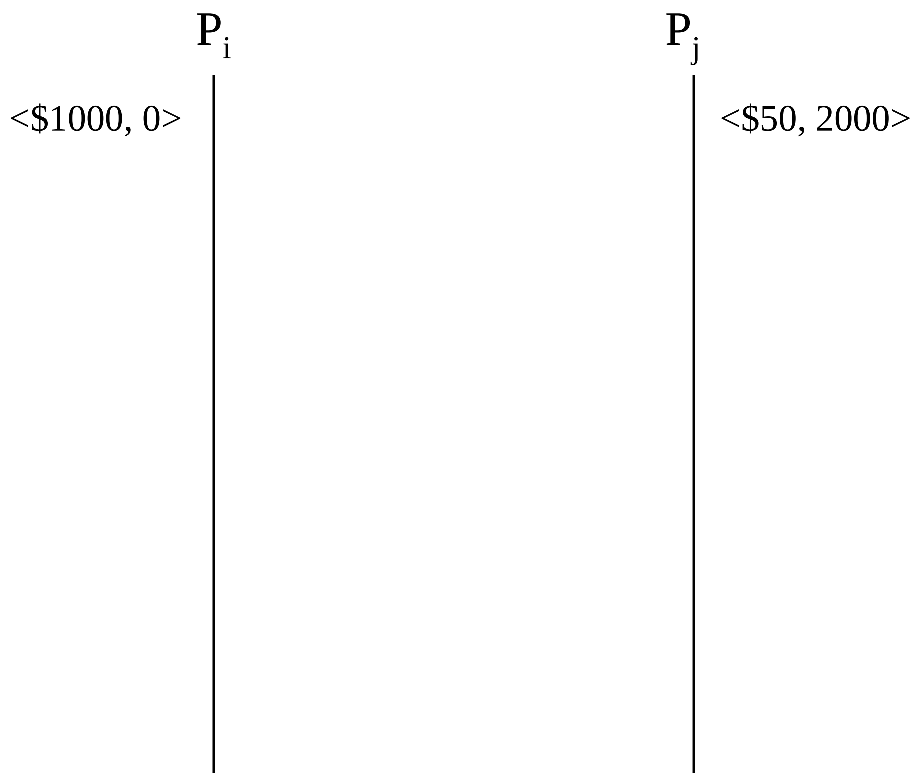
Marker sending rule for process p_i

After p_i has recorded its state, for each outgoing channel c:

p_i sends one marker message over c (before it sends any other message over c).

Snapshots

- **Chandy-Lamport Algorithm for the determination of consistent global states**
  - example:

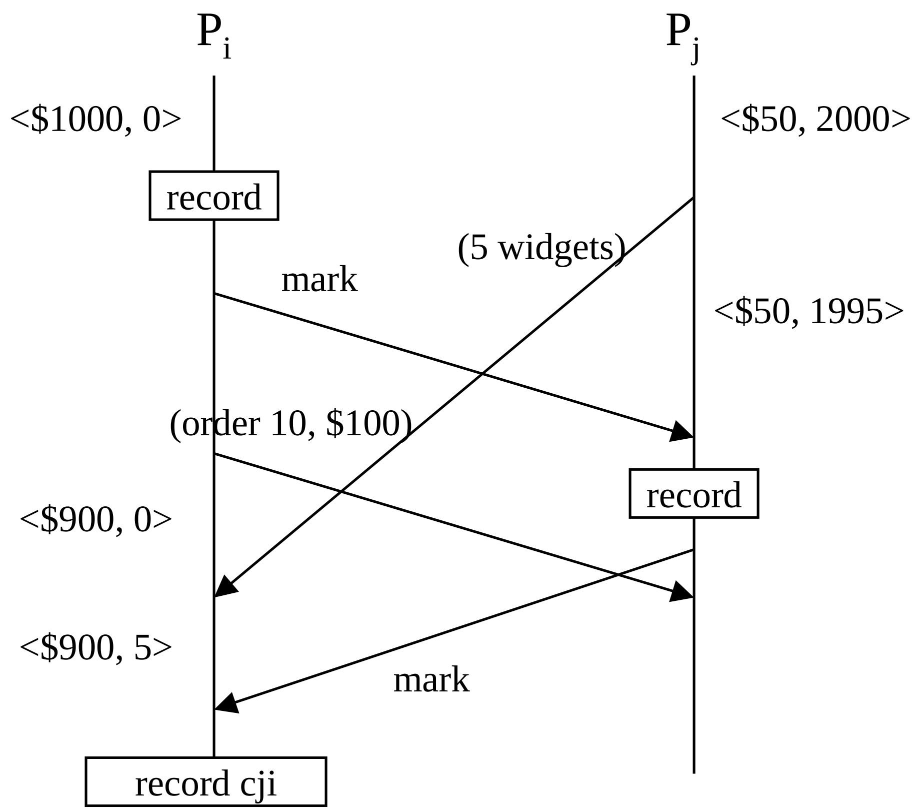


  - recorded state

    $P_i: \langle \$1000, 0\rangle, P_j: \langle \$50, 1995\rangle, c_{ij}: \langle\rangle, c_{ji}: \langle(5 \text{ widgets})\rangle$

Snapshots

- **Chandy-Lamport Algorithm for the determination of consistent global states**
  - example:



  - recorded state

    P_i: <$1000, 0>, P_j: <$50, 1995>, c_{ij}: <>, c_{ji}: <(5 widgets)>

Snapshots

- **Chandy-Lamport Algorithm for the determination of consistent global states**
  - Theorem: The Chandy-Lamport Algorithm terminates
    - Proof sketch:
      - Assumption: a process receiving a marker message will record its state and send marker messages via each outgoing channel in finite period of time.
      - If there is a communication path from p_i to p_k, then p_k will record its state a finite period of time after p_i
      - Since the communication graph is strongly connected, all process in the graph will have terminated recording their state and the state of incoming channels a finite time after some process initiated snapshot taking.

Snapshots

- **Chandy-Lamport Algorithm for the determination of consistent global states**
  - Theorem: Snapshots taken by the Chandy-Lamport Algorithm correspond to consistent global states
  - Proof:

    Let e_i and e_k be events at P_i and P_k, and let $e_i \rightarrow e_k$.

    Then, if e_k is in the cut, so is e_i.

    That means, if e_k occurred before P_k recorded its state, then e_i must have occurred before P_i recorded its state

    k=i: obvious.

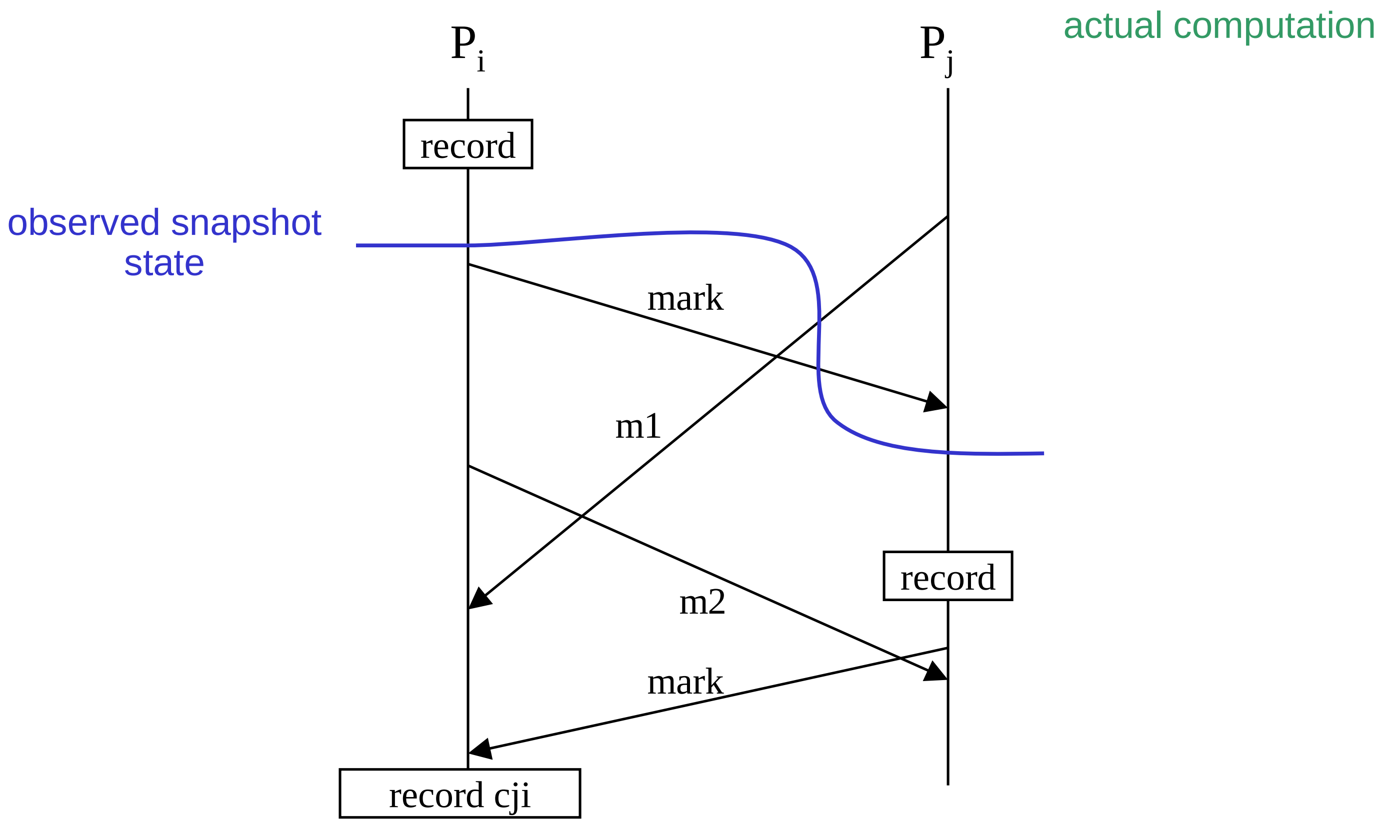
    $k \neq i$: assume P_i recorded its state before e_i occurred

    - as $k \neq i$ there must be a finite sequence of messages $m_1, \ldots, m_n$ that induced $e_i \rightarrow e_k$

    - then, before any of the $m_1, \ldots, m_n$ had arrived, a marker must have arrived at P_k , and P_k must have recorded it's state before e_k occurred, hence a contradiction to the above assumption

Snapshots

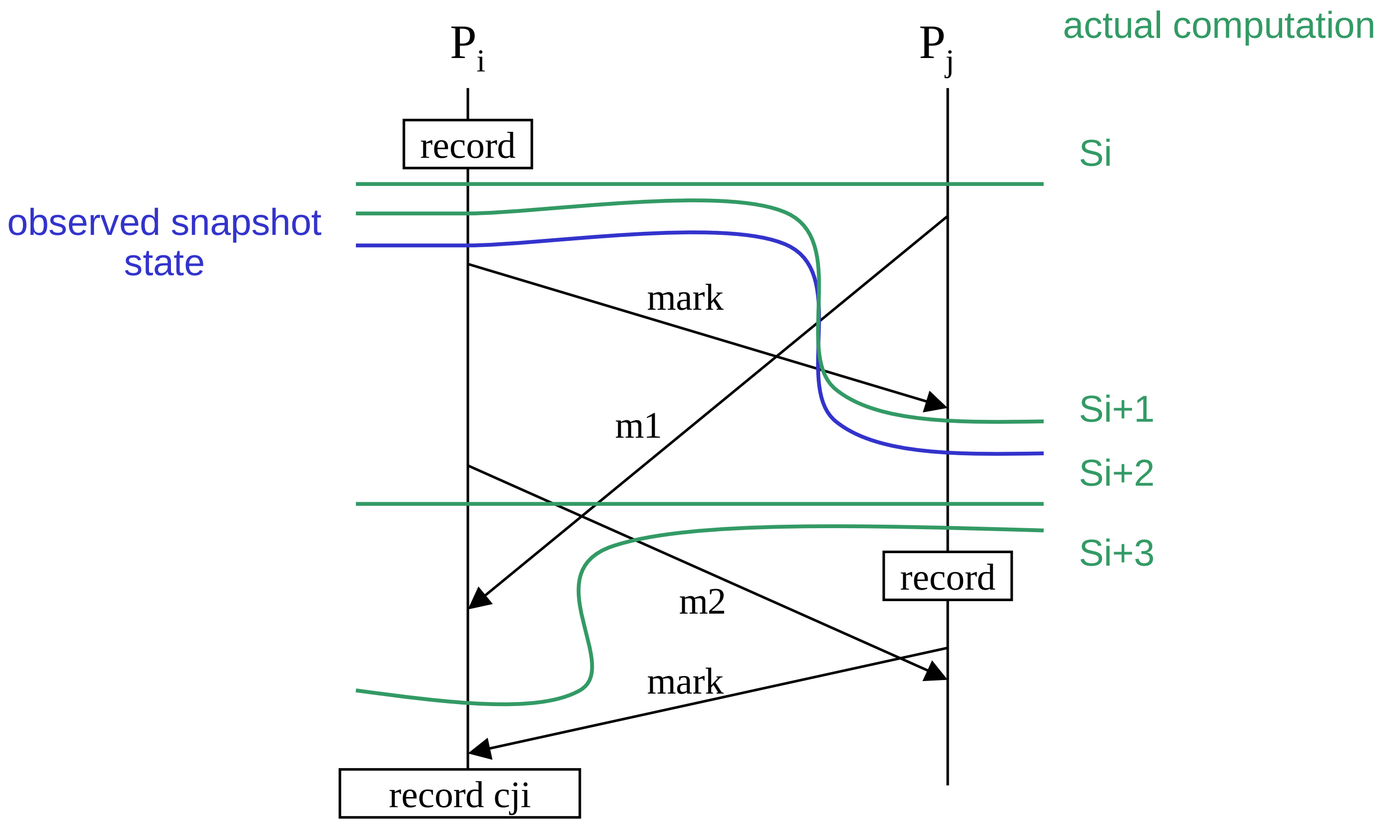
- **Chandy-Lamport Algorithm for the determination of consistent global states**
  - Observation: Chandy-Lamport algorithm records a possible global system state, but the actual execution of the system that initiated the snapshot taking may never have reached this global system state.
  - Example:



tele

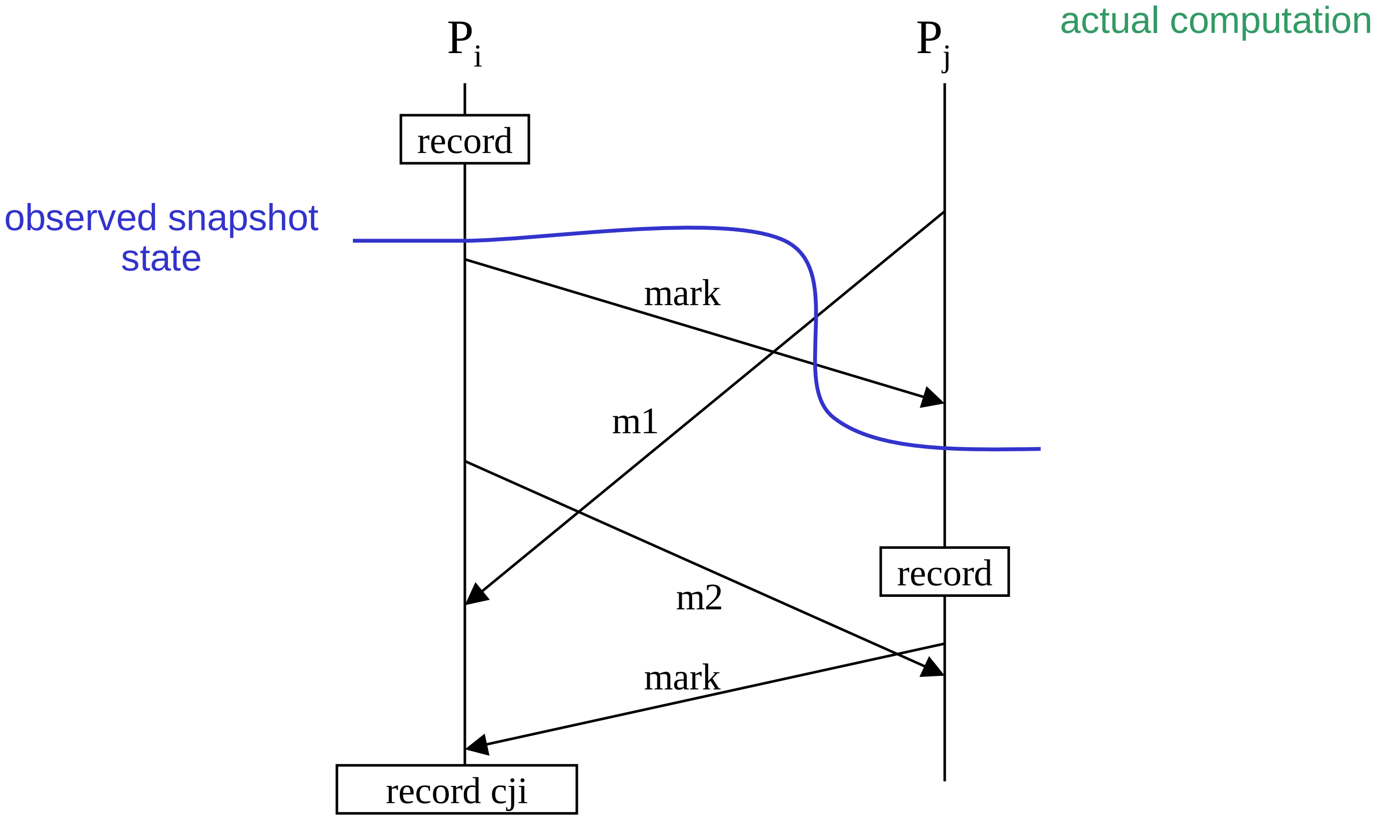
Snapshots

- **Chandy-Lamport Algorithm for the determination of consistent global states**
  - Observation: Chandy-Lamport algorithm records a possible global system state, but the actual execution of the system that initiated the snapshot taking may never have reached this global system state.
  - Example:



Snapshots

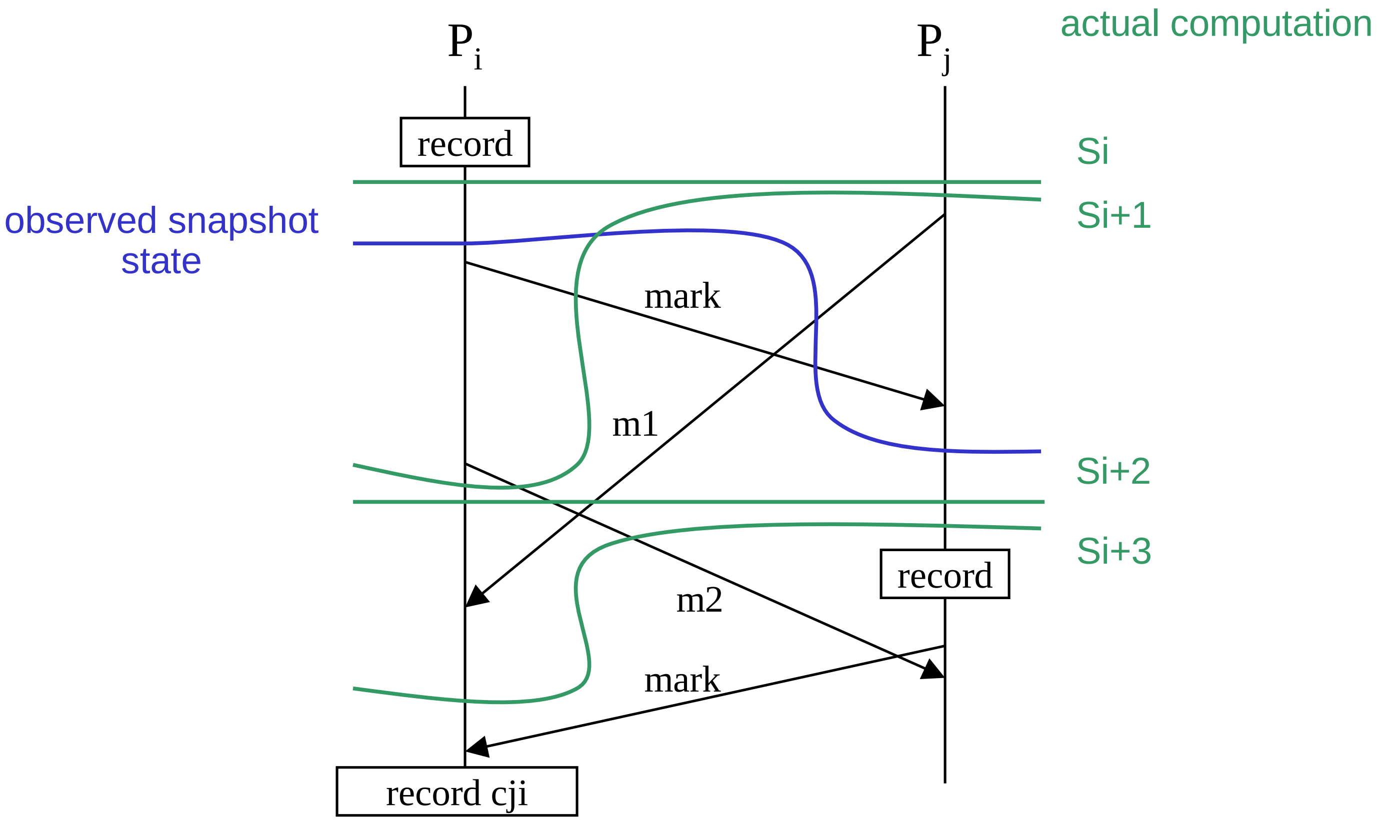
- **Chandy-Lamport Algorithm for the determination of consistent global states**
  - Observation: Chandy-Lamport algorithm records a possible global system state, but the actual execution of the system that initiated the snapshot taking may never have reached this global system state.
  - Example:



  tele

Snapshots

- **Chandy-Lamport Algorithm for the determination of consistent global states**
  - Observation: Chandy-Lamport algorithm records a possible global system state, but the actual execution of the system that initiated the snapshot taking may never have reached this global system state.
  - Example:



Snapshots

- **Chandy-Lamport Algorithm for the determination of consistent global states**
  - Reachability Theorem: Let Sys = e_0, e_1, .. the linearization of a system execution. Let
    - S_{init} the initial global state of the system immediately before Chandy-Lamport snapshot-taking was initiated by the first process,
    - S_{snap} the recorded snapshot state, and
    - S_{final} the global system state after the algorithm terminated.

    Then there is a permutation Sys'= e'_0, e'_1, .. of Sys such that
    - S_{init}, S_{snap} and S_{final} occur in Sys' and
    - S_{snap} is reachable from S_{init}, and
    - S_{final} is reachable from S_{snap} .