Distributed Snapshots: Determining Global States of Distributed Systems

by K. Mani Chandy and Leslie Lamport

Presenter: Ionut Trestian

Why is this useful?

- Many problems in distributed systems can be cast as a problem of detecting global states.
- Example: stable property detection.
- Stable properties are ones that persist, once it becomes true it stays true thereafter.
- Global state detection can be used for check pointing.
- Examples
  - “computation has terminated”
  - “the system is deadlocked”
  - “all tokens in a ring have disappeared

Determining Global state-Issues

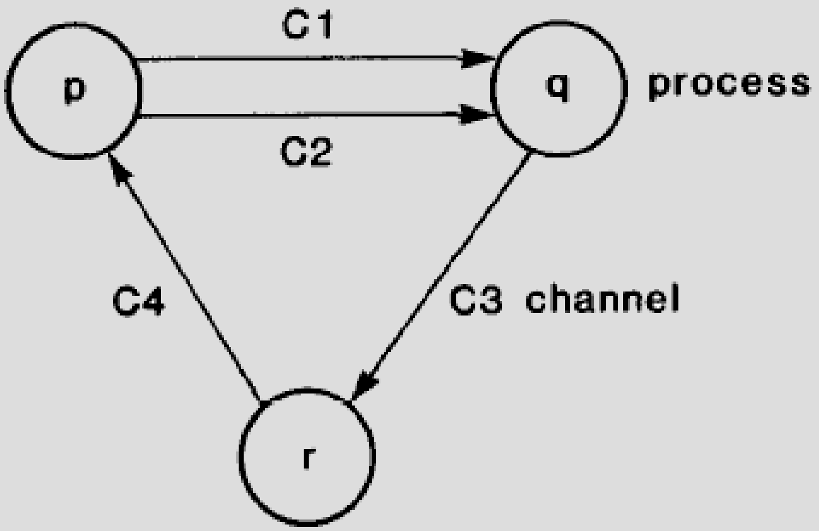
- Processes communicate by sending and receiving messages
- A process can record its own state and the messages it sends and receives, nothing else.
- To determine global systems state all processes must record their state and send it the recorded local state to a process p.
- **Problem:** this must be synchronized (common clock)

Determining global state-Issues

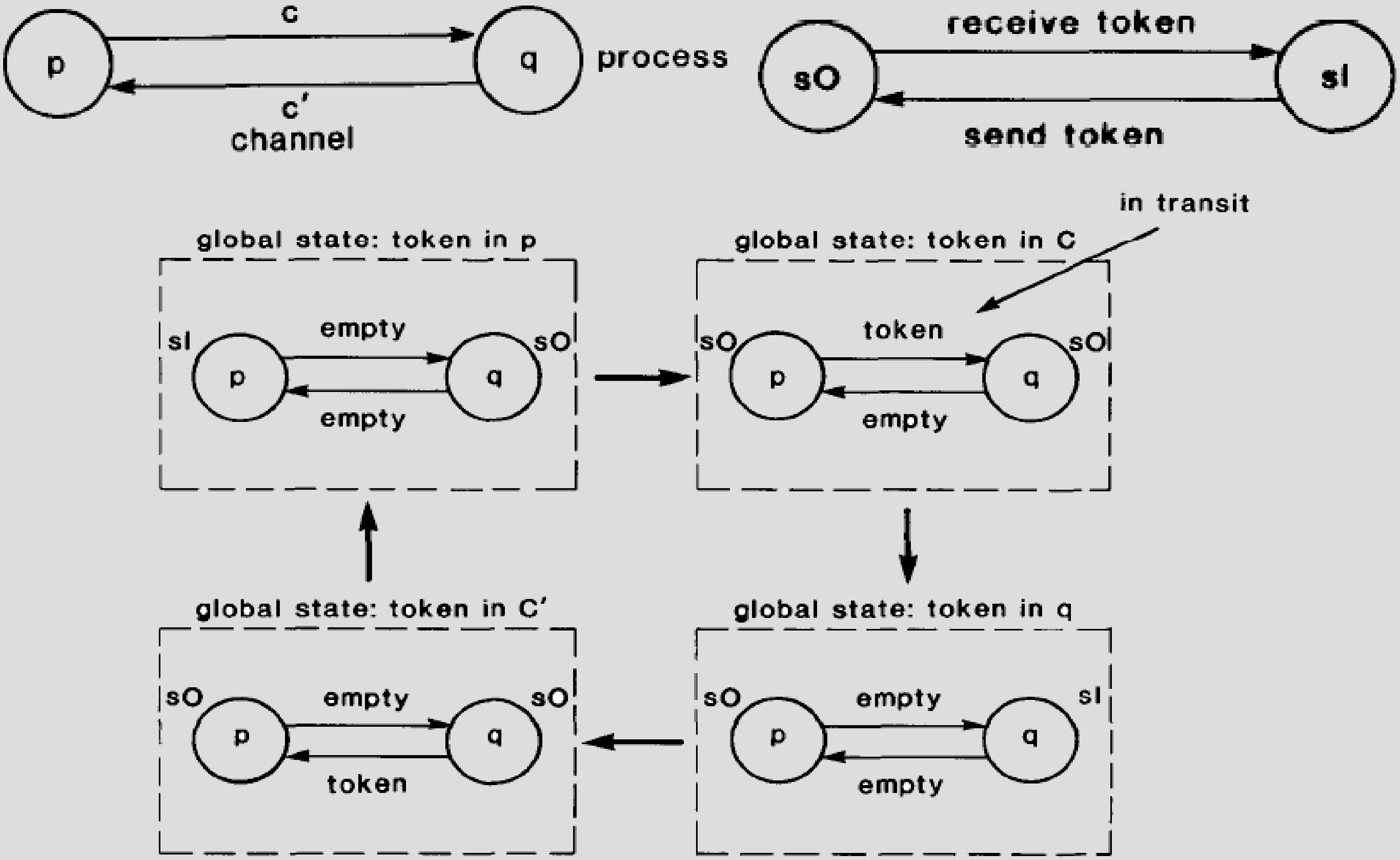
- Current algorithms which determine the global state of a system to solve deadlock and termination problems seem incorrect and impractical.
- Mainly because the relationships among local process states, global system states and points in a distributed computation are not well understood
- Distributed algorithms consist of a sequence of phases
  - A transient part in which useful work is done
  - A stable part in which the system cycles endlessly

Distributed System Model

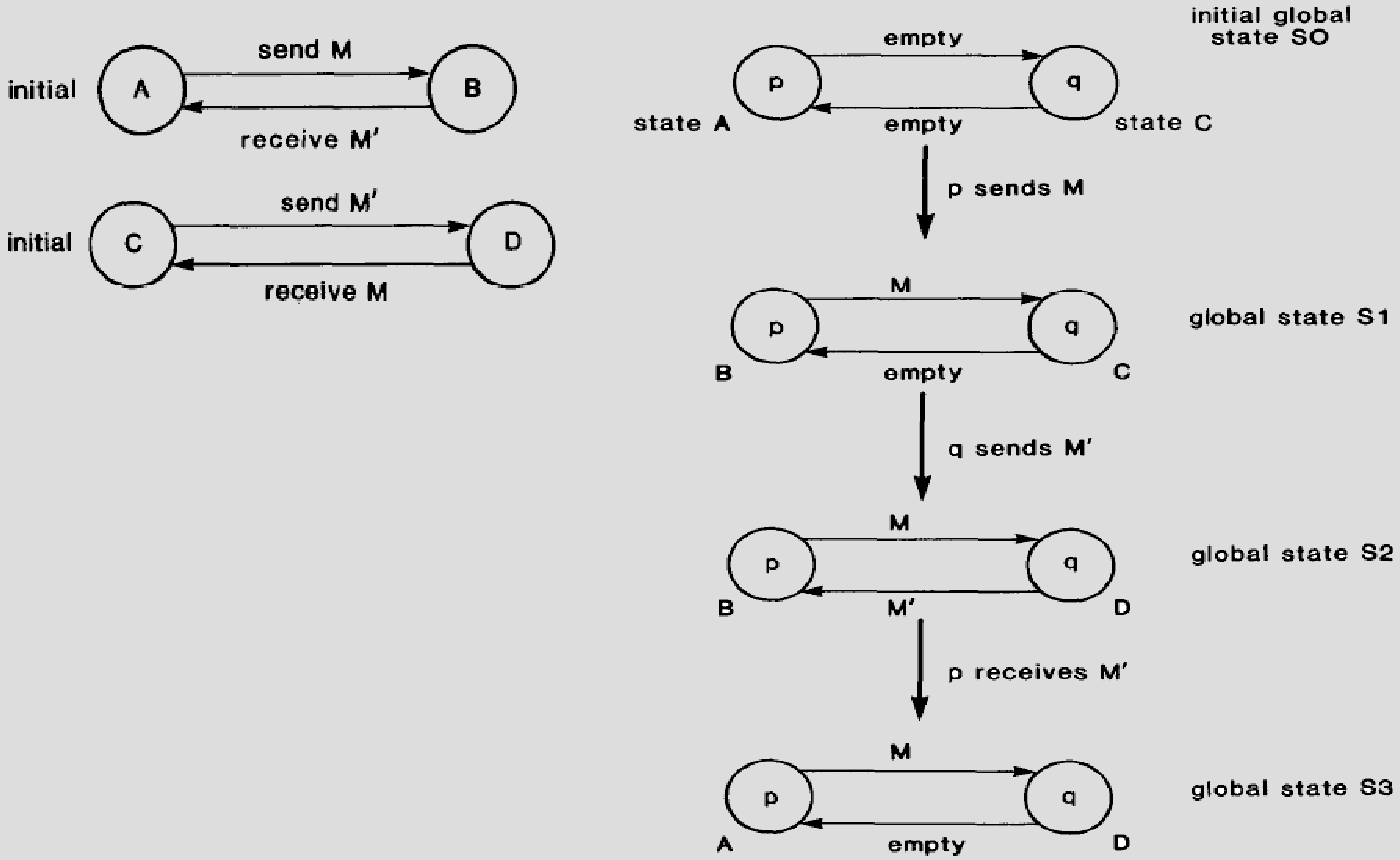
- A distributed system consists of a finite set of processes and a finite set of channels.
- No failures and all messages arrive intact.
- Communication channels are unidirectional and FIFO ordered. They also have infinite buffers. Delays on the channel are arbitrary but finite.
- Processes are defined by an initial state, a set of states and a set of events.



Distributed System - Example



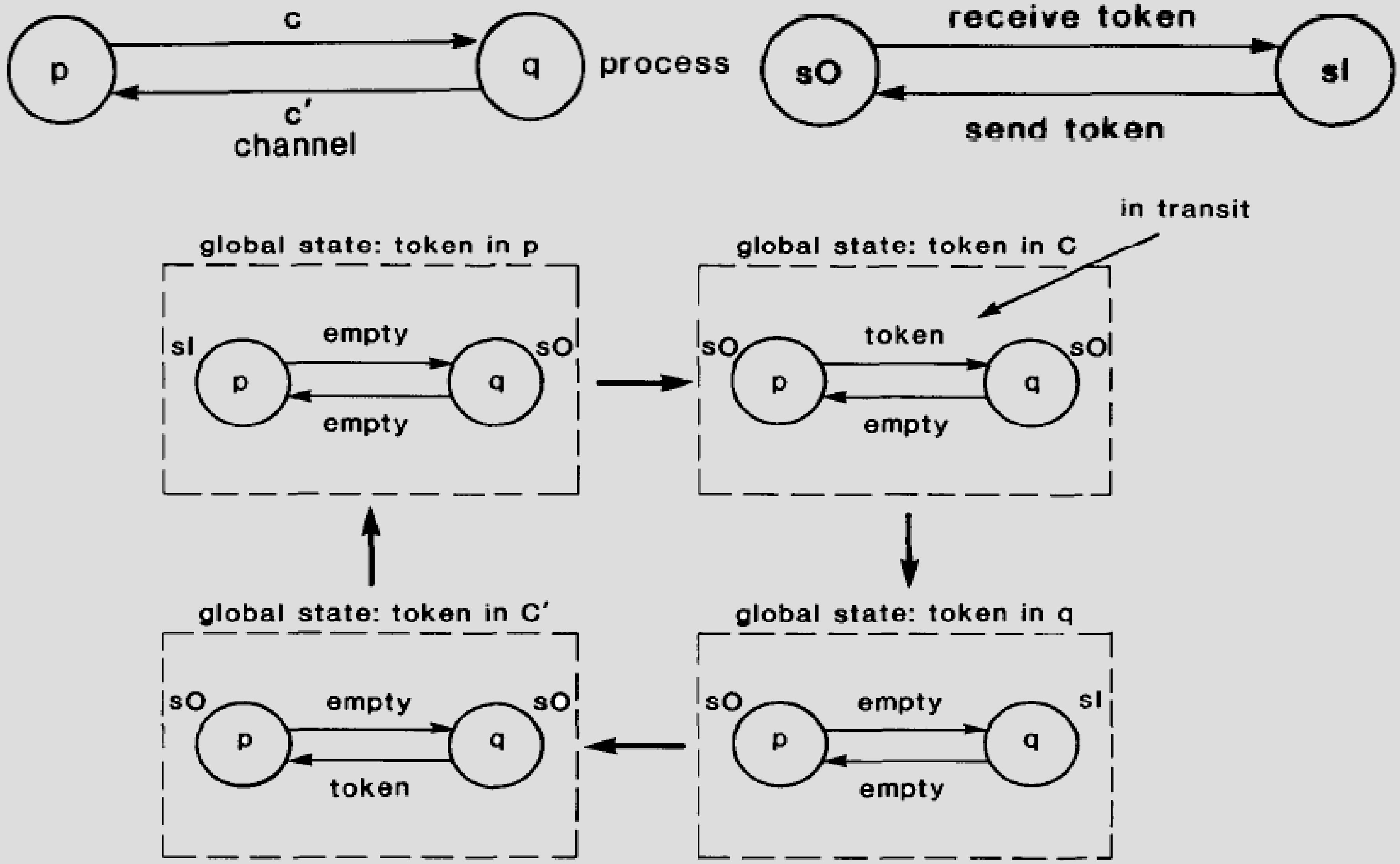
Distributed System - Example



Algorithm – Considerations

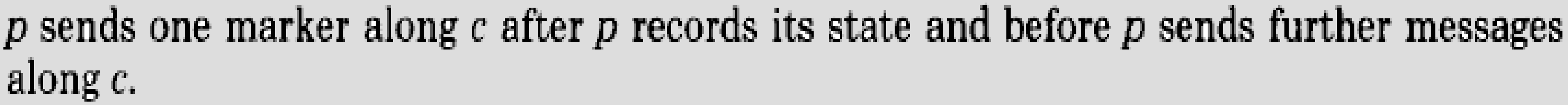
- Each process records its own state
- Two processes that a channel is incident on cooperate to record the channel state.
- Process and channel state must form a meaningful state.
- Computations required to record state must not interfere with underlying computations.

Algorithm - Discussion

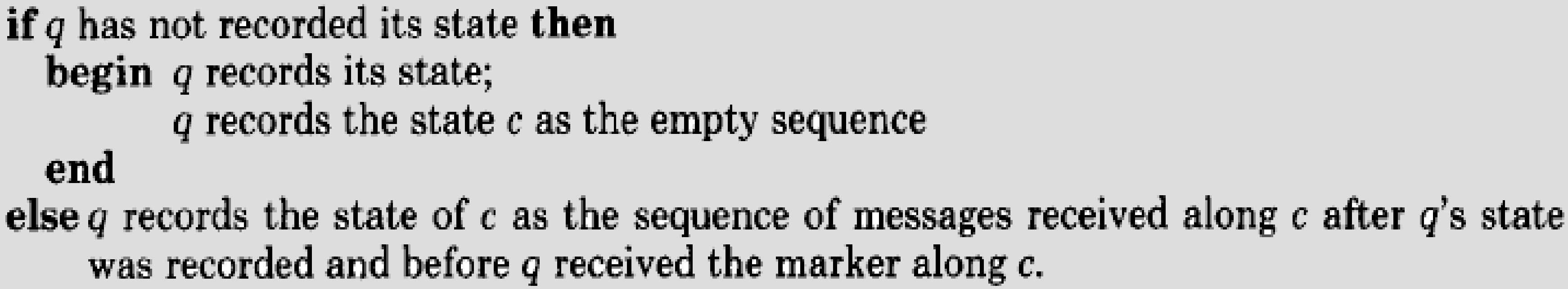


Algorithm outline

- Marker sending rule
- Marker receiving rule



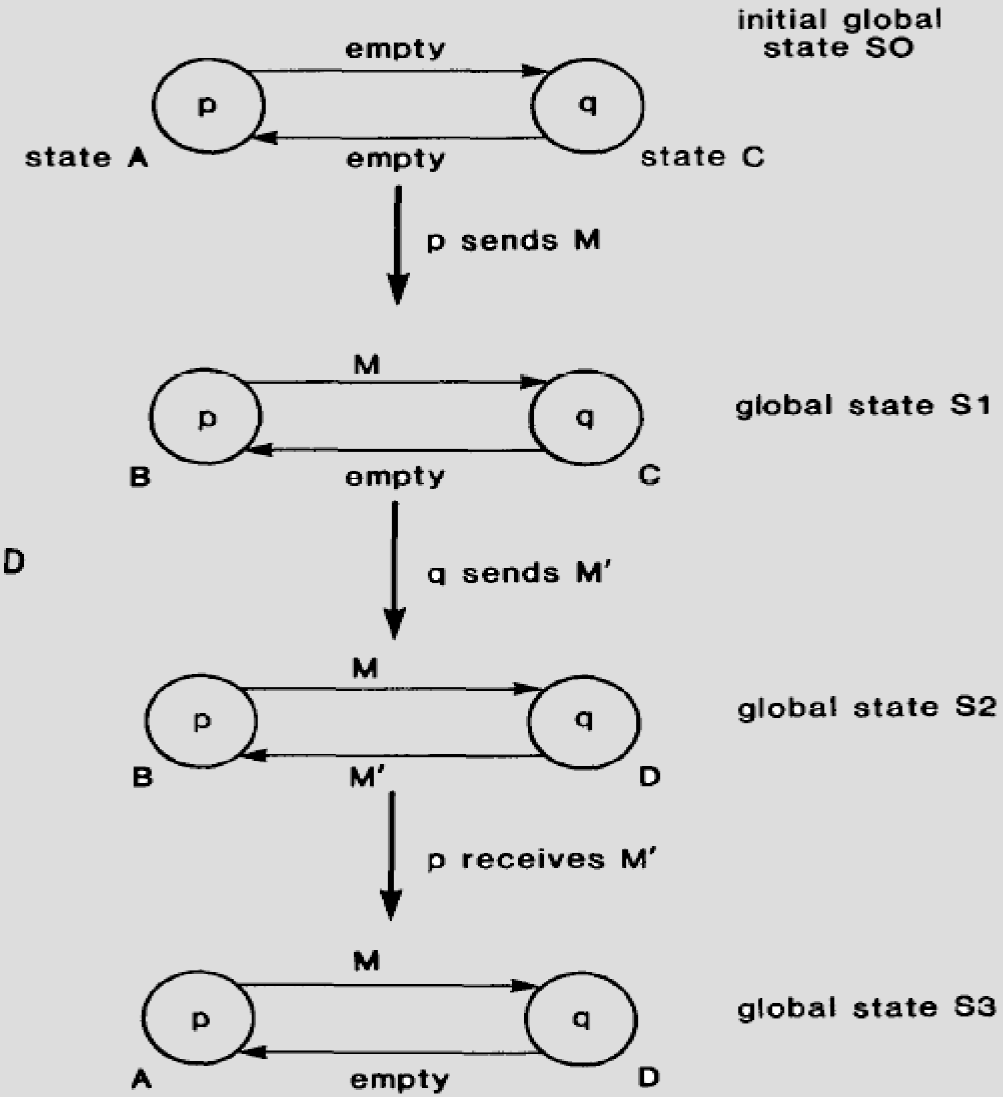
p sends one marker along c after p records its state and before p sends further messages along c.

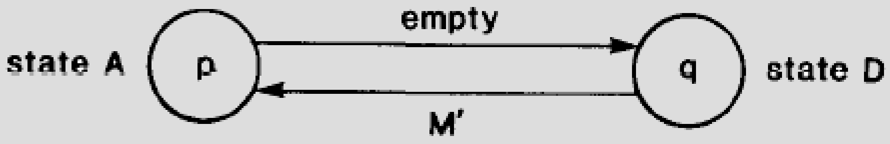


```
if $q$ has not recorded its state then
   begin  $q$ records its state;
          $q$ records the state $c$ as the empty sequence
   end
else $q$ records the state of $c$ as the sequence of messages received along $c$ after $q$'s state
      was recorded and before $q$ received the marker along $c$.
```

Algorithm - Results

- Prerecording event
- Postrecording event





Stability detection

- Input = a stable property y
- Output = a Boolean variable definite with the property:
  - y(Si) => definite and definite => y(So)
- Algorithm:
  - Begin
  - record a global state S*
  - Definite => y(S*)
  - end

Thank you !
Questions?