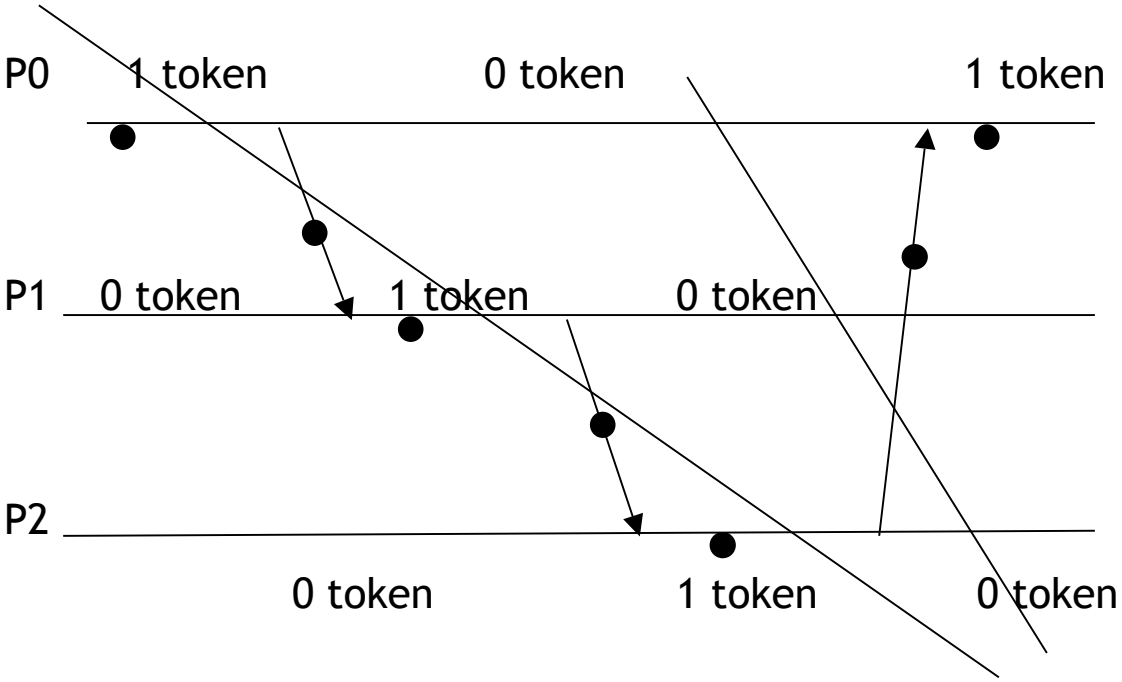
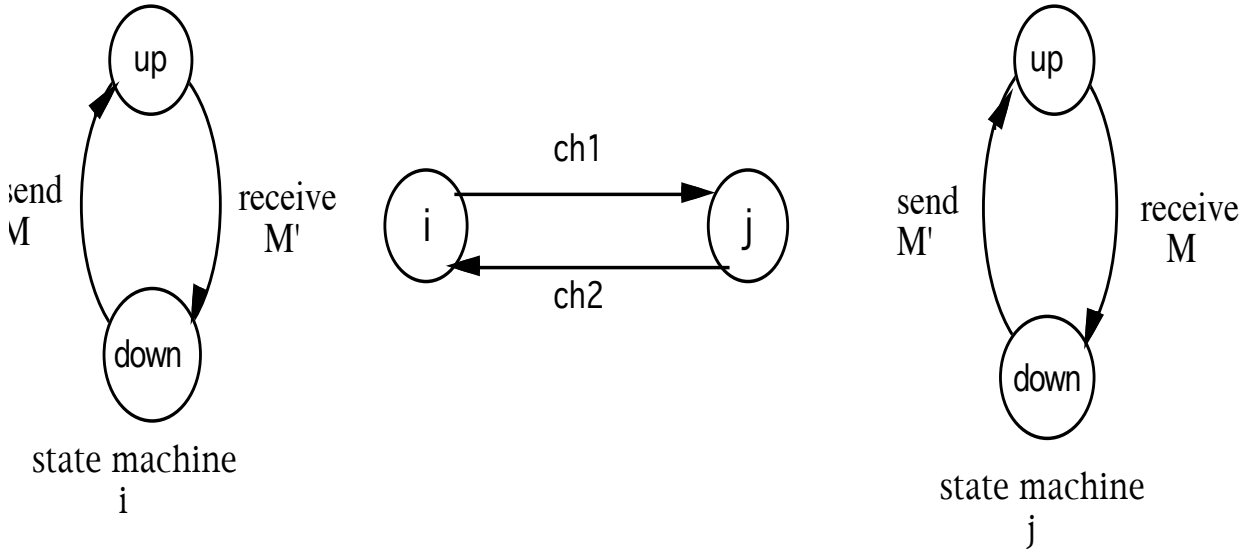
Distributed Snapshot

Examine a few cuts for the token circulation system



Are these consistent cuts?

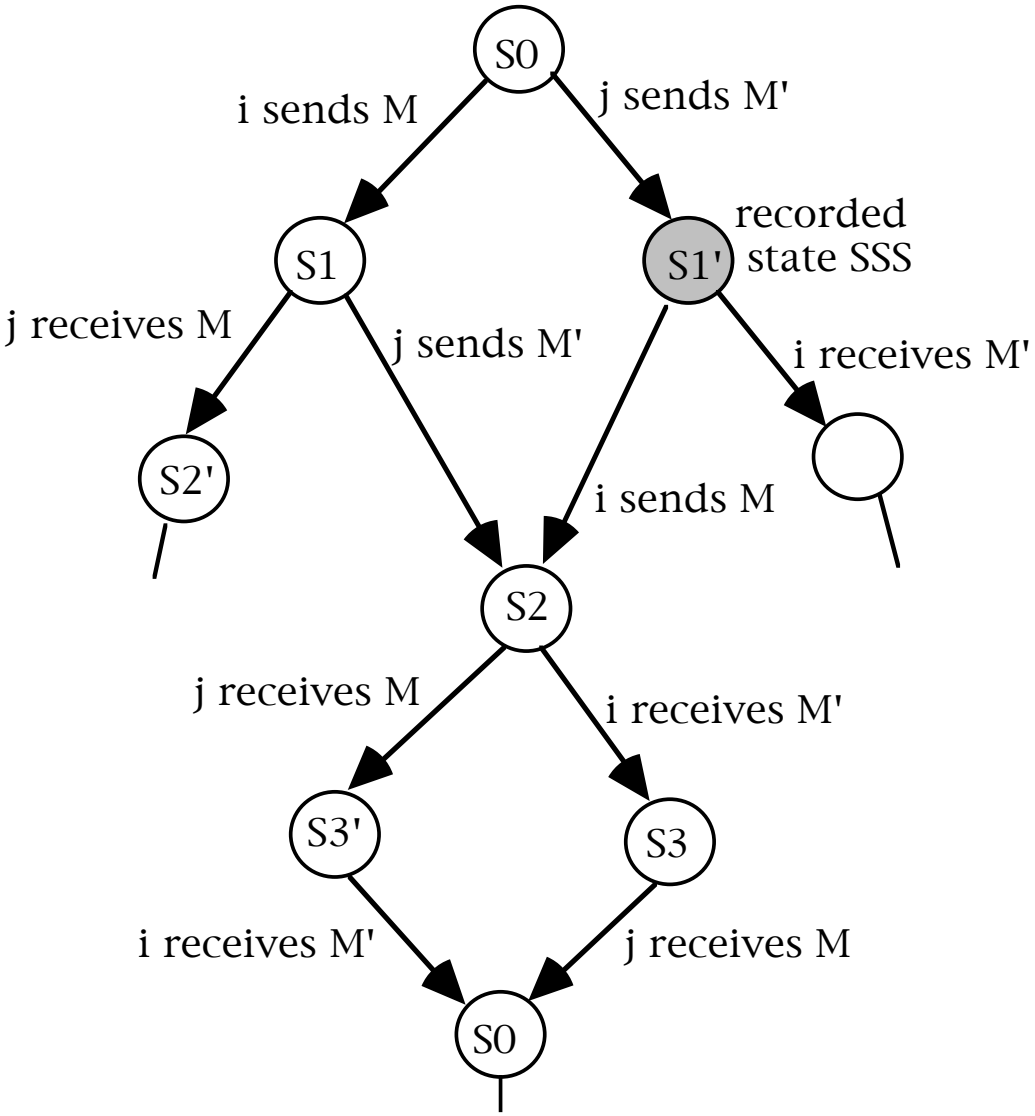
Limitations of the snapshot algorithm



global state	i	ch1	j	ch2
S0	down	□	down	□
S1	up	M	down	□
S2	up	M	up	M'
S3	down	M	up	□

But the recorded state using the distributed snapshot algorithm is: (down □ up M'). Note that this state was never reached!

Reachability graph



Summary

1. Every snapshot state recorded by Chandy-Lamport Algorithm *is reachable from the initial state* through a feasible sequence of actions. However there is no guarantee that this state will actually be attained during a particular computation.

2. Every final state that is reachable from the initial state is also reachable from the recorded snapshot state through a feasible sequence of actions.

Lai-Yang Algorithm

Works with non-FIFO channels. Once again, the cornerstone is the old property:

No red message is received in a white action.

So what if a red message reaches a white process? The process *first turns red* (i.e. records its own state) before accepting the red message. The steps are as follows:

LY1. The initiator records its own state. When it needs to send a message **m** to another process, it sends a message **(m, red).**

LY2. When a process receives a message **(m, red**), it records its state if it has not already done so, and then accepts the message **m.**

No guarantee that the snapshot will eventually be taken!

Global State Collection

Various applications

Computing of the network topology

Computing the size of a network

Detecting termination

Detecting deadlock

Detecting failures

Various mechanisms

Wave algorithms

One or more initiators

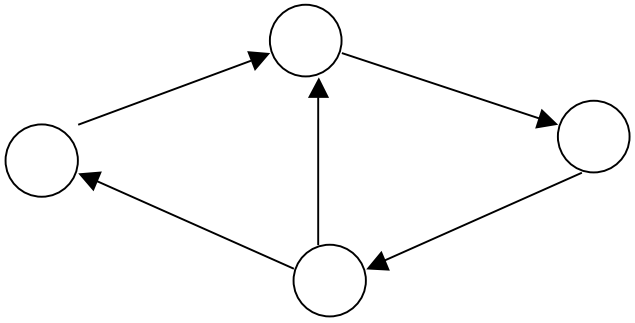
Non-initiators triggered by initiators

Local and global computations terminate after a bounded number of steps

Heartbeat algorithms

An elementary broadcasting algorithm

Goal. Every process will compute the global state



Strongly connected network of processes

program broadcast (for process i}

define V.i, W.i : **set of values**;

initially V.i = s(i), W.i = □ {and every channel is empty}

do V.i ≠ W.i □ send (V.i \ W.i) to each outgoing channel; (1)

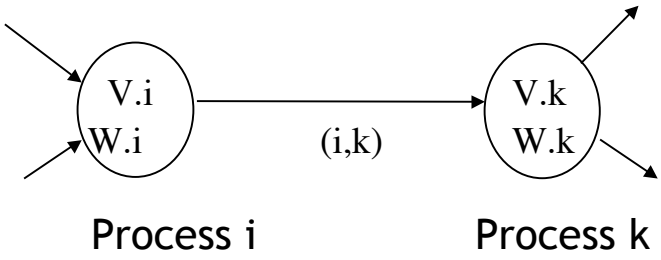
W.i := V.i

¬empty (k, i) □ receive X from channel (k, i); (2)

V.i := V.i □ X

od

Correctness



We need to show that

(1) The algorithm eventually terminates, and

(2) □i: V.i = {s(0), s(1), …, s(n-1)}

Lemma. **empty (i. k) □ W.i □ V.k**.

- Initially, **W.i □ V.k** holds
- *Total data* pumped into (i,k) = W.i (after step 1)
- When channel (i,k) becomes empty, V.k includes the content of (i,k) (after step 2)

Prove that Deadlock is not possible.

Prove that the algorithm terminates with V.i = {s(0), s(1), …, s(n-1)}