Impossibility of Consensus with at most one faulty process

Shared memory and message passing

Assumptions

- n processes
- n 1-writer/n-reader R/W registers
- At most one process can fail by stopping
  - 1-resilient
- Last time: wait-free Consensus is impossible
  - n-1 resilient

Outline of the proof

- Let B be a 1-resilient Consensus algorithm
- We show how to use B to produce an algorithm A that solves wait-free Consensus for 2 processes
- ➔ B does not exist

The Algorithm B Structure

- n processes $q_0,\ldots,q_{n-1}$
- Each q_j writes to a single shared register r_j whose initial value is arbitrary
- The code of q_j:

Super step:
(1) Read some shared variable r_k
(2) Perform state transition
(3) Write the resulting state to r_j (full information model)

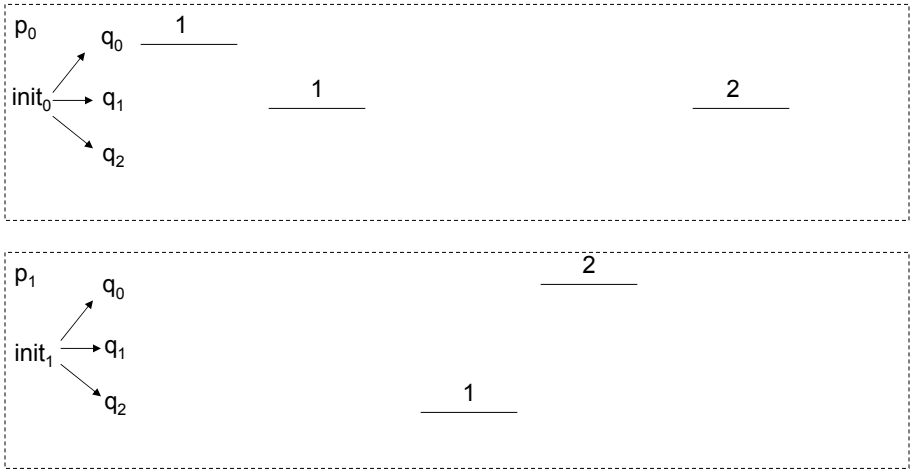
Exercise: Show that no generality is lost by assuming this structure

The Simulation Structure

- Two processes: p_0 and p_1
- Let each p_i, i=0,1, simulate half of the processes
  - Results in a weaker result. Which and Why?
- Let each p_i, i=0,1, simulate *each* process q_j, j=0,…,n-1, in a round-robin manner

Exercise: Prove by simulation that n/2-resilient Consensus cannot be implemented in an asynchronous shared memory model with R/W registers. Take care to be precise.

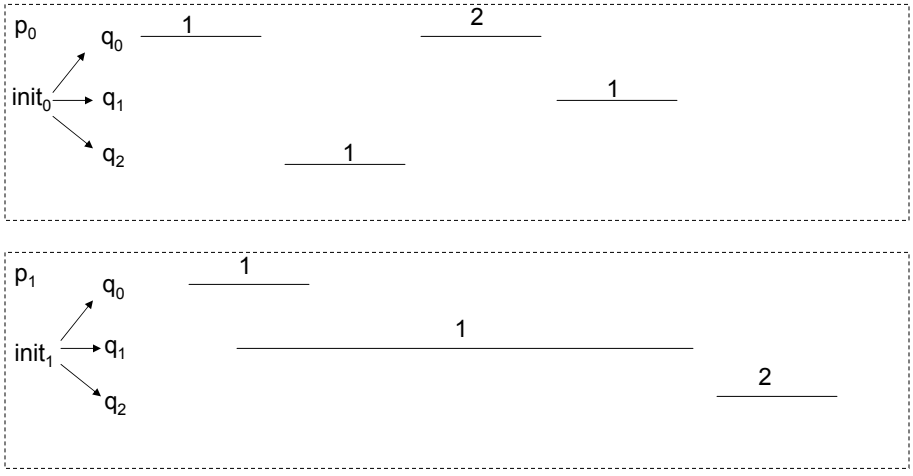
Overview of the simulation



Overview of the Simulation

- Process p_i, i=0,1:
  j := 0;
  while true do
    k:=next super-step of q_j to execute;
    s:=state of q_j at the end of super-step k-1;
    v:=value read by q_j; // simulate read
    s':=transition(j,s,v); // simulate transition
    write s' to q_j's register; // simulate write
    j:=j+1 mod n

Concurrent Execution of A



Data Structure

- For each simulating process p_i, i=0,1, for each simulated process q_j, j=0,…,n-1, and for each simulated super-step k≥0:

  Suggest[j,k,i]: The state of q_j at the end of super-step k as suggested by p_i; initially ⊥

  Flag[j,k,i]: Competition flag of p_i for super-step k of process p_j; initially ⊥

At most one winner

- A super-step k of q_j is *computed* if either
  - Flag[j,k,0] ≠⊥ and Flag[j,k,1] ≠⊥, or
  - Flag[j,k,i]=⊥ and Flag[j,k,1-i]=1, i=0,1
- *Winner* of a computed super-step k of process q_j is the process p_i that sets Flag[j,k,i] if exists, or p_0 otherwise
- Claim 1: There is at most one winner for each computed super-step

Proof of Claim 1

W(Suggest[j,k,i])

R(Suggest[j,k,1-i])

W(Suggest[j,k,1-i])

R(Suggest[j,k,i])

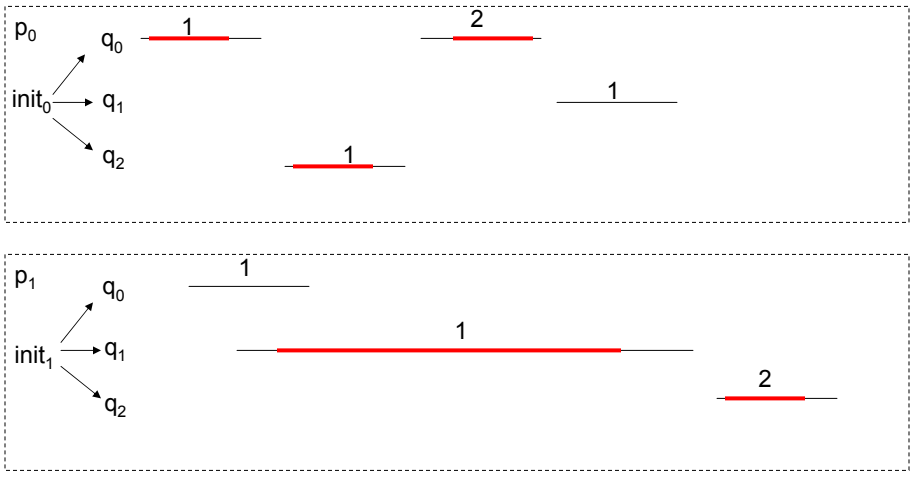
W(Suggest[j,k,i])

R(Suggest[j,k,1-i])

W(Suggest[j,k,1-i])

R(Suggest[j,k,i])

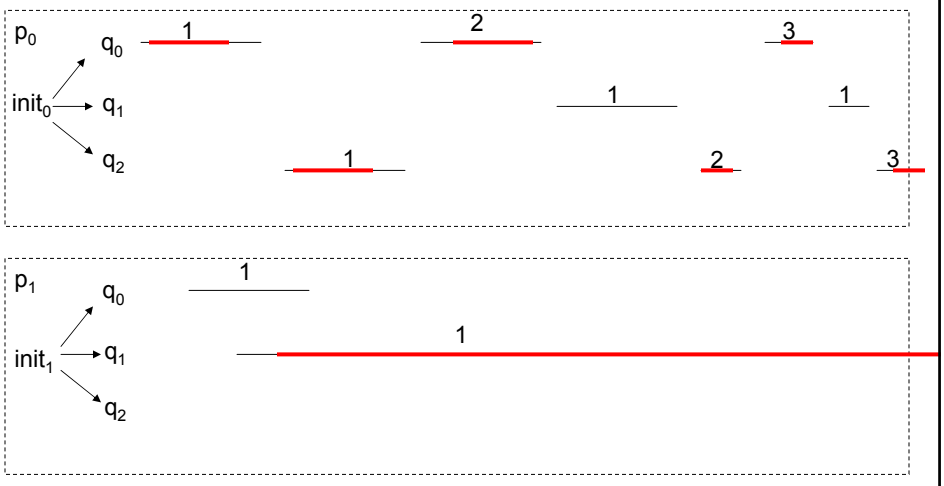
Concurrent Execution of A



Progress

- Simulation of at most one process q_j might get stuck
- If p_i is stuck on simulating a super-step k of q_j, then p_{1-i} fails after writing Suggest[j,k,1-i], but before setting Flag[j,k,1-i] ➔
- p_i will be able to attain progress on simulating all processes except for possibly q_j

Concurrent Execution of A



Simulation Correctness: Informally

- In each execution α of A, the values taken by Suggest registers are consistent with states reachable in an execution β of B
  - But not all Suggest registers! Which?
  - Only those corresponding to computed super-steps of winners

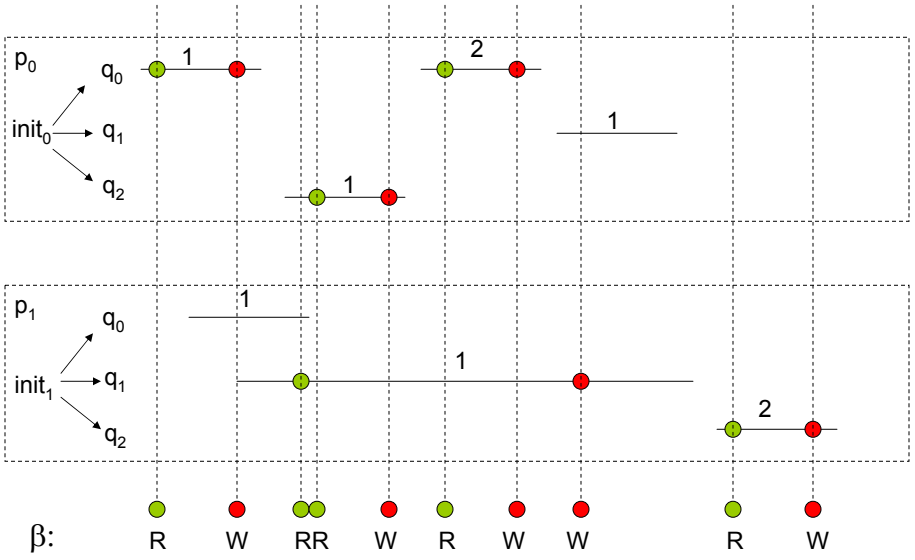
Simulation Correctness: Precisely

- For every q_j and $k \geq 1$:

  Read point of super-step k of q_j in α: The point where the winner of k of q_j reads the state from some other process register
  - After line 2 of get-read

  Write point of k of q_j in α: The point where the winner of k of q_j sets Flag[j,k,i] to 1, or the second process sets Flag[j,k,i] to 0

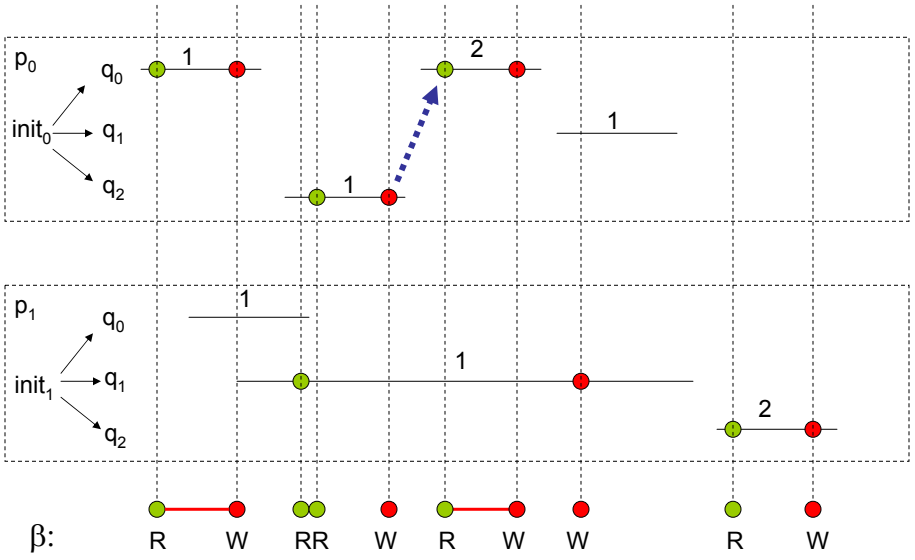
Concurrent Execution of A



Correspondence Lemma

- If q_j executes at least k super-steps in β, then in α:
  1. Eventually computed(j,k)=true, and after that point
  2. Suggest[j,k,w]=q_j's state and register after k super-steps in β, where w is the winner of k of j

Concurrent Execution of A



Proof of Correspondence Lemma

(1) is trivial

(2) is proved by induction on the length of α

Finally…

- get-state(j,k-1) returns Suggest[j,k-1,w] where w is the winner of super-step k-1 of q_j ➔ By Correspondence Lemma, Suggest[j,k-1,w]=state of q_j after k-1 super-steps in β ➔ If the state is deciding, then the decision satisfies Agreement and Validity
- Termination: Simulation can get stuck for at most one q_j ➔ in β at most one process fails ➔ Since B is 1-resilient, all correct processes decide in β and therefore, in α

Exercise: Show that the simulation works with regular registers.

Message Passing

- Simulate a send/receive system on top of a shared memory system with R/W registers
- The result: a shared memory system whose external behavior is indistinguishable from message passing

Simulation

- 1-writer/1-reader register $R_{i,j}$ for each ordered pair of processes (i,j)
- Send(m)$_{i,j}$: Append m to $R_{i,j}$
- Receive:
  - Process i polls $R_{j,i}$ for each j, $1 \leq j \leq n$
  - If m is at the head of $R_{k,i}$, Receive(m)$_{k,i}$, advance the (local) head pointer

Exercise: Fill in missing details of the simulation.

Impossibility in message-passing

- Theorem [aka FLP]: There is no algorithm for solving Consensus problem in an asynchronous message passing system with n processes, one of which can fail by crashing
- Proof?