

FLP Result

M. J. Fischer, N. A. Lynch and M. S. Paterson. *Impossibility of Distributed Consensus with One Faulty Process*. Journal of the ACM 32(2):374-382 (April 1985).

Very weak consensus problem

Each process has an *initial value* of 0 or 1, and eventually *decides* on a value of 0 or 1.

weak termination: some correct process decides

agreement: no two processes decide on different values

very weak validity: there is a run in which the decision is 0 and a run in which the decision is 1.

... not all processes need to decide.

System model (I)

- set P of processes.
- Process is a deterministic automaton executing atomic steps
⟨send 0 or more messages;
receive message or λ (null message);
change state deterministically based on
message received⟩

System Model (II)

- set M of messages (*message buffer*) containing undelivered messages $(p, q:m)$ with $p, q \in P$: p sent m to q .
Assume without loss of generality that contents m are unique.
- The choice of which message (or λ) to receive is nondeterministic.
 - not immediate;
 - not FIFO;
 - can not be infinitely postponed.

System Model (III)

p has:

- two distinct initial states σ_p^0 and σ_p^1 ;
- two distinct sets of decision states D_p^0 and D_p^1 .

... being in a decision state is stable.

System Model (IV)

Configuration $C=(s, M)$ represents a global state of the protocol.

- $s: P \rightarrow \text{process state}$
- M : message buffer

Initial configuration C_0 has

- $M = \emptyset$
- $\forall p \in P: s(p) \in \{\sigma_p^0, \sigma_p^1\}$

System Model (V)

Event e is applicable to $C = (s, M)$ if:

- $e \in M$
- $e = (\lambda, p)$

$e(C)$ is e applied to C :

- state of p changes;
- e removed from M if $e \neq (\lambda, p)$;
- 0 or more messages from p added to M .

System Model (VI)

S is a *schedule*: a (possibly unbounded) sequence of events.

$S = e_1 e_2 e_3 \dots$ applied to C

$C^1 = e_1(C)$

$C^2 = e_2(C^1)$

$C^3 = e_3(C^2)$

$\dots$

$C C^1 C^2 C^3 \dots$ is sequence of configurations that *corresponds to S* .

System Model (VII)

If S is finite then say $C^k = S(C)$ follows C .

If S is infinite then the corresponding sequence of configurations is a *run* if

- C is an initial configuration;
- $\forall p \in P$: no non-null e of p is applicable to infinitely many C^i unless p has only finitely many events in S .

System Model (VIII)

- If not infinitely many events of p in run then p is *faulty* in this run; else it is *correct*.

This is a common technique used in formal proofs about asynchronous systems, and doesn't imply anything fundamental.

Very weak consensus recap

weak termination: in any run R , $\exists p \in P$ correct in R containing configuration $C = (s, M)$: $s(p) \in D_p^0 \cup D_p^1$.

agreement: in any run R , there is no configuration $C = (s, M)$ and $p, q \in P$ correct in R : $s(p) \in D_p^0$ and $s(q) \in D_p^1$.

very weak validity: for both $v \in \{0, 1\}$: there is a run R_v containing configuration $C = (s, M)$: $s(p) \in D_p^v$.

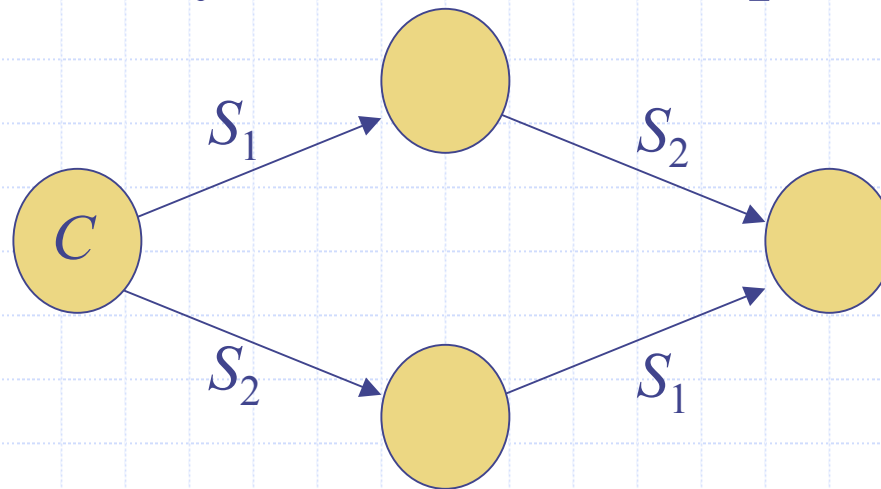
0- and 1-Determined

- Configuration C is v -*determined* if there is no schedule S and process p : $S(C) = (s, M)$ where $s(p) \in D_p^{\underline{v}}$
 $\underline{v}$ denotes the complement of v .
- C is *determined* if it is either 0- or 1-determined.
- C is *undetermined* if it is not determined.

A Simple Lemma

Let S_1, S_2 be schedules from C : no process p has events in both S_1 and S_2 . Then, $S_1(S_2(C)) = S_2(S_1(C))$.

Simple proof by induction. Graphically,



The Theorem

Very weak consensus is not solvable if one or more processes can be faulty.

Proof by contradiction: suppose protocol Π solves very weak consensus when at most one process is faulty.

Two Facts (to be proven later)

FACT 1: Π has an undetermined initial configuration.

FACT 2: Let C be an undetermined configuration and e an event applicable to C .

- $CS = \{C' : C' = S(C) \text{ for some finite } S \text{ not containing } e\}$
- $DS = \{D : \exists C' \in CS : D = e(C')\}$

DS contains an undetermined configuration.

The contradiction

- Let C_0 be an undetermined initial configuration [FACT 1].
 - Construct run R in which each corresponding configuration is undetermined.
- ... such a run violates *weak termination*.

Constructing the run

Q : queue of processes

$C = C_0$;

while (1) {

p = first process in Q

if (M contains e of p) then let e be the oldest such event

else let e be (λ, p)

D = an undetermined configuration $D = e(S(C))$

 for some S

[FACT 2]

$C = D$

 move p to end of Q

}

Notes on the construction

- The run is valid: each event in M is eventually received.
- No process is faulty in R : each process takes infinitely many steps.
- Each configuration is undetermined.
- Once we show **FACT 1** and **FACT 2** are valid, we've proven that there can be no such protocol Π .

Showing **FACT 1** (I)

Proof by contradiction: assume that all initial configurations are determined.

- From *very weak validity*, we know that there are both 0-determined and 1-determined configurations.
- Define two initial configurations C and C' as *neighbors* if all but one process in the two configurations have the same initial values.

Showing **FACT 1** (II)

- Since there are both 0- and 1-determined configurations, there must be two neighboring configurations C_0 which is 0-determined and C_1 which is 1-determined.

Consider some 0-determined configuration and some 1-determined configuration. There is some path connecting them by the neighbor relation. By following this path, one must find a pair of configurations that differ in how they are determined.

Showing **FACT 1** (III)

Let p be the process whose value differs between C_0 and C_1 .

- By *weak termination* and the fact that Π can tolerate a single failure, there is a schedule S that contains no events of p such that

$$S(C_0) = C = (s, M): \exists q: s(q) \in D_q^0$$

- But since S contains no events of p and because C_0 and C_1 differ only in p 's initial value, $S(C_1)$ results in the same configuration C in which $s(q) \in D_q^0$
... and so C_1 is not 1-determined.

FACT 1 holds.

Showing **FACT 2** (I)

Proof by contradiction: assume that all configurations in DS are determined.

CLAIM 1: There are both 0- and 1-determined configurations in DS .

Proof: Since C is undetermined, by weak termination there is a schedule S_0 that leads to a 0-determined configuration C_0 and a schedule S_1 that leads to a 1-determined configuration C_1 .

Showing **FACT 2** (II)

CLAIM 1 (continued)

Consider the schedule S_v such that $C_v = S_v(C)$ where (by construction) C_v is v -determined.

Either $e \notin S_v$ or $e \in S_v$.

If $e \notin S_v$ then $C_v \in CS$ and so $e(C_v) \in DS$. Since C_v is v -determined, $e(C_v)$ is v -determined.

If $e \in S_v$ then rewrite S_v as $S_v^a e S_v^b$. Note that

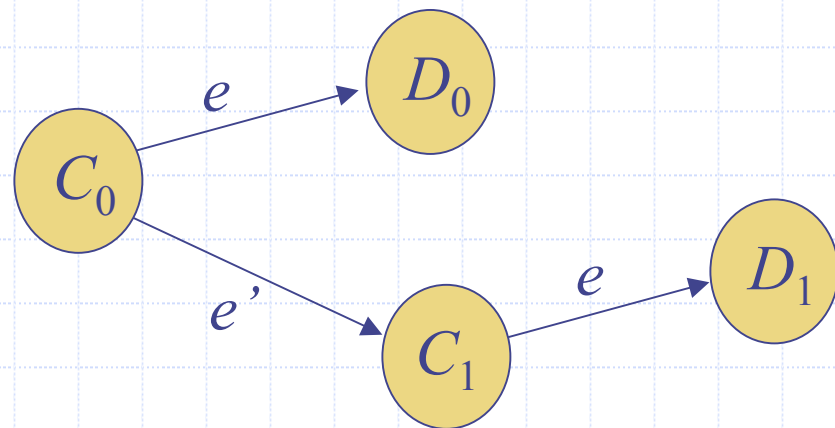
$$D = S_v^a(e(C)) \in DS.$$

Since $S_v^b e(D)$ is v -determined and by assumption D is determined, D must be v -determined.

Showing **FACT 2 (III)**

Define two configurations E_1 and E_2 to be *adjacent* if there is some event e' such that either $E_1 = e'(E_2)$ or $E_2 = e'(E_1)$.

CLAIM 2: There are adjacent configurations C_0 and C_1 in CS such that $D_v = e(C_v)$ is v -determined.



Showing **FACT 2** (IV)

Proof:

Construct a graph with nodes representing configurations in CS and edges indicating adjacency.

Each node C' can be labeled 0 or 1 depending on whether $e(C')$ is 0- or 1-determined (recall that by assumption, all configurations in DS are determined.)

Showing **FACT 2 (V)**

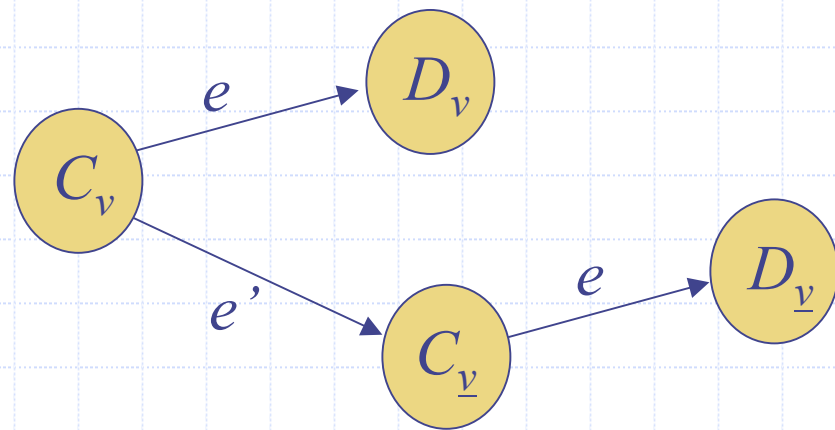
Since C is in CS and all configurations in CS are generated by a finite schedule from C , there is a path from each node to C , and the graph is connected.

From CLAIM 1 we know that there are nodes in this graph labeled 0 and nodes labeled 1.

Choose a path from a 0-labeled node to a 1-labeled node. There must be some point in which two adjacent nodes are found with different labels.

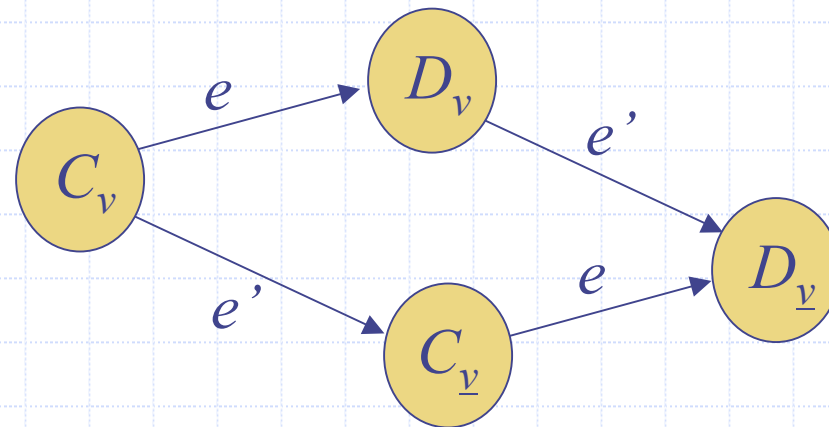
Showing **FACT 2 (VI)**

As defined in CLAIM 2, let C_v for $v \in \{0, 1\}$ be the adjacent configurations with respect to event e' . Recall e is of some process p . Let e' be of process q . Either $p \neq q$ or $p = q$.



Showing **FACT 2 (VII)**

If $p \neq q$ then by the lemma $e'(D_v) = D_{\underline{v}}$ which is a contradiction.



Showing **FACT 2 (VIII)**

If $p = q$ then find some schedule S containing no events of p such that $S(C_v) = E$ is determined.

Recall that C_v need not be determined.

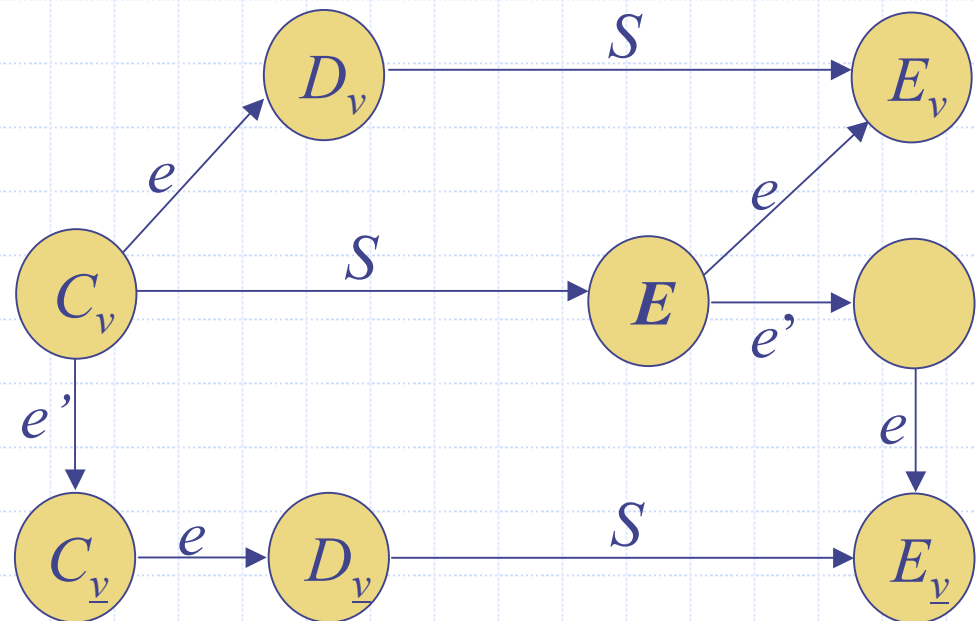
There must be such a schedule S because Π can tolerate one failure and *weak termination* holds.

Showing **FACT 2 (VIII)**

By the lemma, we have $e(E)$ is v -determined and $e'(E)$ is $\underline{v}$ -determined.

Hence, E is not determined; a contradiction.

FACT 2 holds.



Observation (I)

It is often stated that *FLP holds because a process can not distinguish between an arbitrarily slow process and a crashed process.*

This is a bit simplistic, as we will shortly see.

Observation (II)

A more accurate statement is that *there is a point in a failure-free execution in which the execution of an event at p makes the run determined one way or another*. Indeed, as we've seen, p makes a choice between becoming 0- or 1-determined.

From weak termination and being able to tolerate a faulty process, though, we can hold back p long enough to reach a determined state.

What happens if allow p to proceed after that? It's state hasn't changed...