The FLP Theorem

Jacopo Notarstefano
jacopo.notarstefano [at] gmail.com

The Distributed Consensus Problem

Definition

Consensus protocol

Definition (Consensus protocol)

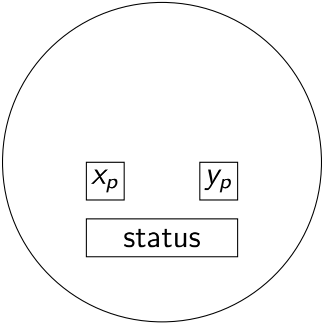
A **consensus protocol** is an asynchronous system of N processes ($N \geq 2$). Each process p has a one-bit **input register** x_p, an **output register** y_p with values in $\{\perp, 0, 1\}$ and an unbounded amount of internal storage.

Initial states prescribe fixed starting values for all but the input register; in particular, the output register starts with value $\perp$.

p acts deterministically according to a **transition** function.

Decision states

The states in which the output register has value 0 or 1 are distinguished as being **decision states**. The transition function cannot change the value of the output register once the process has reached a decision state; that is, the output register is “write-once”.



Message system

A **message** is a pair (p, m), where p is the name of the destination process and m is a "message value" from a fixed universe M.

Definition (Message system)

The **message system** mantains a multiset, called the **message buffer**, of messages that have been sent but not yet delivered. It supports two abstract operations:

`send(p,m)`: Places (p, m) in the message buffer.

`receive(p)`: Deletes some message (p, m) from the buffer and returns m, in which case we say (p, m) is **delivered**, or returns the special null marker $\varnothing$ and leaves the buffer unchanged.

Partial correctness

A configuration C has **decision value** v if some process p is in a decision state with $y_p = v$.

Definition (Partial correctness)

A consensus protocol is **partially correct** if:

1. No accessible configuration has more than one decision value.
2. For each $v \in \{0, 1\}$, some accessible configuration has decision value v.

Total correctness in spite of one fault

A process p is **nonfaulty** in run if it takes infinitely many steps, otherwise it is **faulty**.

A run is **admissible** if at most one process is faulty and all messages sent to nonfaulty processes are eventually received.

A run is **deciding** if some process reaches a decision state.

Definition (Total correctness in spite of one fault)

A consensus protocol P is **totaly correct in spite of one fault** if it is partially correct and every admissibile run is deciding.

Main result

Theorem (Fischer, Lynch, Paterson 1985)

No consensus protocol is totally correct in spite of one fault.

A configuration is **bivalent** if the set of decision values of configurations reachable from it has 2 elements. It is instead 0-**valent** or 1-**valent** according to the corresponding value.

Proof (sketch).

Given an initial bivalent configuration, we construct an admissible run that at each stage results in another bivalent configuration. □

Lemma 1

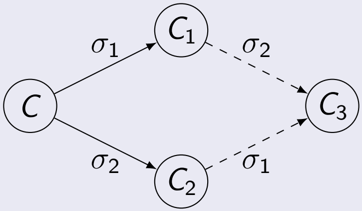
Lemma

Suppose that from some configuration C, the schedules σ_1, σ_2 lead to configurations C_1, C_2 respectively. If the sets of processes taking steps in σ_1 and σ_2, respectively, are disjoint, then σ_2 can be applied to C_1 and σ_1 can be applied to C_2, and both lead to the same configuration C_3.

In other words: **schedules about disjoint processes commute with each other**.

Proof of Lemma 1

Proof (Lemma 1).



Because the sets of processes are disjoint, an event in σ_1 applicable to C is applicable to C_2 as well.

Because of determinism, after all events are processed they must end up in the same state. □

Lemma 2

Lemma

P has a bivalent initial configuration.

Proof (Lemma 2).

$\square$

Lemma 3

Lemma

Let C be a bivalent configuration of P, and let $e = (p, m)$ be an event that is applicable to C. Let $\mathcal{C}$ be the set of configurations reachable from C without applying e, and let $\mathcal{D} = e(\mathcal{C}) = \{e(E) | E \in \mathcal{C}$ and e is applicable to $E\}$. Then, $\mathcal{D}$ contains a bivalent configuration.

In other words: given a bivalent configuration and an event e applicable to it, **we construct another bivalent configuration having e as the last applied event**.

Proof of Lemma 3

Proof (Lemma 3).

$\square$

Proof of main result

Proof (main result).

$\square$