A Formal Proof for FLP Result

Vijay K. Garg

TR-PDS-1999-002 November 1999

Parallel & Distributed Systems group
Department of Electrical & Computer Engineering
University of Texas at Austin
Austin, Texas 78712

A Formal Proof for FLP Result

Vijay K. Garg*
Parallel and Distributed Systems Laboratory,
Electrical and Computer Engineering Department
The University of Texas at Austin,
Austin, TX 78712.
http://maple.ece.utexas.edu

November 29, 1999

Abstract

I present a formal proof of impossibility of consensus in an asynchronous system, a result due to Fischer, Lynch and Paterson.

1 Introduction

In this paper, I present a formal proof of impossibility of consensus in an asynchronous system, a result due to Fischer, Lynch and Paterson [FLP85]. Our proof makes the assumptions required for the impossibility result explicit. It also avoids proof by contradiction, thereby providing an insight into the structure of computation trees for asynchronous systems.

We use the following notation in this paper.

- G, H, J, K: global states of the system
- $G[i]$: State of process i in the global state G
- $G \leq H$: H reachable from G
- s: sequence of events
- $s(G)$: global state resulting from application of s to G.
- procs(s): set of processes on which events in s are defined.

We assume that there are n processes in the system. Each process has a register D which indicates the value committed by the process. The value of 0 in D indicates that the process has agreed on the value 0. The same holds for the value 1. The value $\perp$ indicates that the process has

*supported in part by the NSF Grants ECS-9414780, CCR-9520540, TRW faculty assistantship award, a General Motors Fellowship, and an IBM grant.

not agreed on any value. Initially, we require all processes to have $\perp$ in their register D. We define a predicate *match* for a global state H and a value $v \in \{0, 1\}$ as follows.

$$match(H, v) = \exists i : H[i].D = v$$

We will require that D register be write-once.

$$G[i].D = v \Rightarrow \forall H : G \leq H : H[i].D = v$$

From the write-once nature of D register, and the definition of *match*, we deduce that *match* is a stable property.

$$\forall G, v : match(G, v) \Rightarrow \langle \forall H : G \leq H : match(H, v) \rangle$$

The predicate $match(H)$ will denote the condition that one of the value is matched, i.e.,

$$match(H) = match(H, 0) \vee match(H, 1)$$

2 Requirements and Assumptions

The binary consensus protocol requires:

- **agreement**: Two processes cannot commit on different values.

$$\forall G : \neg match(G, 0) \vee \neg match(G, 1)$$

- **non-triviality**: Both values 0 and 1 should be possible outcomes.

$$\exists G0 : match(G0, 0) \wedge \exists G1 : match(G1, 1)$$

- **termination**: The system reaches agreement in finite time. To formalize this requirement, we first define *runs*. A run is a sequence of global states. An *admissible* run is one in which at most one process crashes and all messages sent to non-faulty processes are eventually reached. We define $future(G)$ as the set of possible admissible sequences of events from G. Now termination can be stated as:

$$\forall G : init(G) : \langle \forall \sigma \in future(G) : (\exists H \in \sigma : match(H)) \rangle$$

We now list the assumptions about the underlying system. These are

- **Initial Independence**: We assume that initial states satisfy the following independence property.

$$init(X) \wedge init(Y) \wedge (X[i] \neq Y[i]) \Rightarrow \exists Z : init(Z) \wedge (\forall j : j \neq i : Z[j] = X[j]) \wedge (Z[i] = Y[i])$$

- **Commute Property of disjoint events:** Let G be anly global state such that two events e and f are enabled in it. If $proc(e) \neq proc(f)$, then these two events commute, i.e.,

$$enabled(G, e) \wedge enabled(G, f) \wedge (proc(e) \neq proc(f)) \Rightarrow ef(G) = fe(G)$$

  Equivalently, for any two sequences of events s and t

$$enabled(G, s) \wedge enabled(G, t) \wedge (proc(s) \cap proc(t) = \phi) \Rightarrow st(G) = ts(G)$$

- **Asynchrony of events**: The asynchrony assumption states that any *receive* event may be arbitrarily delayed. Let $receive(e)$ denote that e is a receive event. Then, the assumption can be stated as

$$receive(e) \wedge enabled(G, e) \wedge enabled(G, s) \wedge e \notin s \Rightarrow enabled(G, se) \quad (1)$$

  Note that the above assumption does not state that $se(G) = es(G)$. The event e commutes with s only when $proc(e) \notin proc(s)$.

3 Classification of Global States

In this section we classify global states of any protocol depending upon *match* predicate. Informally, a $0valent$ state is one in which the protocol will eventually decide 0. Similarly, a $1valent$ state will eventually result in decision of 1. If a state is *undecided*, then there exists an admissible execution which will result in neither 0 nor 1 being committed. A *bivalent* state is neither a $0valent$ state nor a $1valent$ state. Note that, every undecided state is a bivalent state, but not vice-versa. Formally,

Definition 1

$$0valent(G) = \forall \sigma \in future(G) : \exists H \in \sigma : match(H, 0).$$

$$1valent(G) = \forall \sigma \in future(G) : \exists H \in \sigma : match(H, 1).$$

$$bivalent(G) = \neg 0valent(G) \wedge \neg 1valent(G)$$

$$undecided(G) = \exists \sigma \in future(G) : \forall H \in \sigma : \neg match(H).$$

The following lemma shows that a 1valent state G can never reach a state H in which $match(H, 0)$ is true. It uses *agreement* and stability of *match*.

Lemma 1
$\forall G, H : G \leq H : match(H, 0) \Rightarrow \neg 1valent(G)$

Proof:
$G \leq H$
$\equiv$ { definition of $\leq$ }
$\exists \sigma \in future(G) : H \in \sigma.$

To show $\neg 1valent(G)$, we prove, for this σ,

$$\forall K \in \sigma : \neg match(K, 1).$$

We do a case analysis.
<u>*Case 1*</u>: $H \leq K$.
$match(H, 0)$
$\Rightarrow$ { $H \leq K$ and stability of $match$ }
$match(K, 0)$
$\Rightarrow$ { agreement }
$\neg match(K, 1)$.

<u>*Case 2*</u>: $K \leq H$.
$match(H, 0)$
$\Rightarrow$ {agreement}
$\neg match(H, 1)$
$\Rightarrow$ { stability of $match$, $K \leq H$ }
$\neg match(K, 1)$.

■

Similarly, the following results can be established.

Lemma 2 *a)* $(G \leq H) \wedge match(H, 1) \Rightarrow \neg 0valent(G)$.
b) $(G \leq H) \wedge 0valent(G) \Rightarrow 0valent(H)$
c) $(G \leq H) \wedge 1valent(G) \Rightarrow 1valent(H)$
d) $0valent(G) \Rightarrow \neg 1valent(G)$

4 Impossibility Result

Lemma 3 *Assume non-triviality and initial independence.*

$$\exists X : init(X) \wedge bivalent(X)$$

Proof:
non-triviality
$\Rightarrow$ { $G0$ and $G1$ are reachable global states }
$\exists X0 : init(X0) \wedge X0 \leq G0 : match(G0, 0)$
$\wedge \exists X1 : init(X1) \wedge X1 \leq G1 : match(G1, 1)$.

$X0 \leq G0 \wedge match(G0, 0)$
$\Rightarrow$ { Lemma 1 }
$\neg 1valent(X0)$
$\Rightarrow$ { definition of $bivalent$ }

$bivalent(X0) \vee 0valent(X0)$

Similarly, $bivalent(X1) \vee 1valent(X1)$. If either $X0$ or $X1$ is bivalent we are done. Therefore, we consider the case $0valent(X0) \wedge 1valent(X1)$. We do a case analysis on k equal to the number of components in which $X0$ and $X1$ differ.

Case 1: $(k = 1)$.
Let $X0$ and $X1$ differ in only one component - i.

Consider an admissible sequence of moves σ from $X0$ in which process i does not move.

$0valent(X0)$
$\Rightarrow$ { σ admissible, definition of 0valent }
$\exists s : match(s(X0), 0) \wedge i \notin procs(s)$
$\Rightarrow$ { $\forall j : j \neq i : X0[j] = X1[j]$ and $X0[i].D = X1[i].D = \perp$ }
$match(s(X1), 0)$
$\Rightarrow$ { Lemma 1, $X1 \leq s(X1)$ }
$\neg 1valent(X1)$

Case 2: $(k > 1)$.
Let i be first component in which $X0$ and $X1$ differ. From initial independence, there exists an initial state Z such that Z and $X1$ differ in $k-1$ components.
and Z and $X0$ differ in 1 component. We now do a case analysis on Z.

Case 2.1: $bivalent(Z)$

We are done because Z is also an inital state.

Case 2.2: $0valent(Z)$.

We have two initial states Z and $X1$ such that $0valent(Z)$ and $1valent(X1)$ which differ in $k-1$ components. Hence, the induction is applicable.

Case 2.3: $1valent(Z)$.

We have two initial states $X0$ and Z such that $0valent(X0)$ and $1valent(Z)$. Further, $X0$ and Z differ in exactly one component. Therefore, the base case is applicable.

■

In the following Lemma, we will exploit the asynchronous nature of computation. We show

Lemma 4
For all receive events e,

$$enabled(G, e) \wedge match(t(G), 1) \Rightarrow \exists s : \neg 0valent(se(G))$$

Proof: We do a case analysis.
Case 1: $e \in t$

$match(t(G), 1))$
$\Rightarrow$ { let t be rewritten as seu }
$\exists s, u : match(seu(G), 1)$
$\Rightarrow$ { Lemma 1 }
$\exists s : \neg 0valent(se(G))$

Case 2: $e \notin t$

$match(t(G), 1))$
$\Rightarrow$ { $enabled(G, e)$, Assumption of asynchrony 1}
$match(t(G), 1) \wedge enabled(G, te)$
$\Rightarrow$ { stability of $match$ }
$match(te(G), 1)$
$\Rightarrow$ { Lemma 1 }
$\exists s : \neg 0valent(se(G))$

■

The following Lemma gives a forbidden structure in the state transition graph of any protocol.

Lemma 5

$$\forall G, e, f : \neg 0valent(e(G)) \vee \neg 1valent(fe(G)) \vee undecided(G)$$

Proof:

Case 1: $proc(e) \neq proc(f)$

$0valent(e(G))$
$\Rightarrow$ { 0valency is stable }
$0valent(ef(G))$
$\Rightarrow$ { $fe(G) = ef(G)$ from the commuting property}
$0valent(fe(G))$
$\Rightarrow$
$\neg 1valent(fe(G))$.

Case 2:$proc(e) = proc(f)$.

true
$\Rightarrow$ { definition of admissible }
$\exists \sigma \in future(G) : i \notin procs(\sigma)$.

We have the following three sub-cases.
Case 2.1: $\forall H \in \sigma : H \geq G : \neg match(H)$.
$\Rightarrow$ { definition }

$undecided(G)$.

Case 2.2: $\exists H \in \sigma : H \geq G : match(H, 0)$
$\Rightarrow$ { definition of σ }
$\exists s : match(s(G), 0) \wedge proc(e) \notin proc(s)$
$\Rightarrow$ { fe disjoint from s }
$\exists s : match(sfe(G), 0)$
$\Rightarrow$ { Lemma 1 }
$\neg 1valent(fe(G))$

Case 2.3: $\exists H \in \sigma : H \geq G : match(H, 1)$
$\Rightarrow$ { definition of σ }
$\exists s : match(s(G), 1) \wedge i \notin proc(s)$
$\Rightarrow$ { e disjoint from s }
$\exists s : match(se(G), 1)$
$\Rightarrow$ { Lemma 1 }
$\neg 0valent(e(G))$

■

Lemma 6 *For any state G and any receive event e,*

$$bivalent(G) \wedge enabled(G, e) \Rightarrow undecided(G) \vee \exists s : bivalent(se(G))$$

Proof:
If $e(G)$ is bivalent, we are done with s as empty sequence. If not let $0valent(e(G))$ without any loss of generality.

$bivalent(G)$
$\Rightarrow$
$\neg 0valent(G)$
$\Rightarrow$
$\exists \sigma \in future(G) : \langle \forall H \in \sigma : \neg match(H, 0) \rangle$.

Case 1: $\forall H \in \sigma : \neg match(H, 1)$.
$\Rightarrow$
$\forall H \in \sigma : \neg match(H, 0) \wedge \neg match(H, 1)$.
$\equiv$ { definition of undecided }
$undecided(G)$

Case 2: $\exists H \in \sigma : match(H, 1)$.
$\Rightarrow$
$\exists t : match(t(G), 1)$

$\Rightarrow$ { $enabled(G, e)$, Lemma 4 }
$\exists s : \neg 0valent(se(G)$
$\equiv$ { definition }
$\exists s : 1valent(se(G)) \vee bivalent(se(G))$.

If $bivalent(se(G))$, then we are done. Therefore, we have $bivalent(G), 0valent(e(G))$ and $1valent(se(G))$. We now claim that

$$\exists C, f : 0valent(e(C)) \wedge 1valent(fe(C))$$

We use induction on the length of $k = length(s)$.

Base Case: $(k = 1)$
s has only one event. Define $C = G$ and $f = s$.

Induction Case: $(k > 1)$
Let $s = dt$. Consider $d(G)$. It cannot be 0valent because $dte(G)$ is 1valent

<u>Case 1</u>: $d(G)$ is bivalent

If $de(G)$ is 1-valent, let $f = d$.
If $de(G)$ is 0-valent, we have $bivalent(d(G)), 0valent(e(d(G)))$ and $1valent(te(d(G)))$.
If $de(G)$ is bivalent, we have the required result.

<u>Case 2:</u> $d(G)$ is 1valent

This implies $de(G)$ is 1valent. Let $f = d$.
Thus we have
$\exists C, f : 0valent(e(C)) \wedge 1valent(fe(C))$
$\Rightarrow$ {from Lemma 5 }
$undecided(C)$

■

We can now state the impossibility result.

Theorem 1 *Any protocol that satisfies non-triviality and agreement has an initial state G such that $undecided(G)$ holds.*

Proof: From Lemma 3, there exists an inital global state G such that $bivalent(G)$ holds. We will construct an admissible sequence from G in which no decision is reached. We will run processes in a round robin fashion. Each process will receive the first message from its buffer. Let e be the event that is enabled next according to this schedule. From Lemma 6, either G is undecided or there exists a sequence s such that on executing se, the system stays in a bivalent state. By indefinitely repeating this process, we get the desired result.

■

References

[FLP85] M. J. Fischer, N. Lynch, and M. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2), April 1985.