DHCP-201@caltech.edu
Lynch, Ch. 12

**Goal:** "Impossibility" Result

**Conclusion:** Some timing may be required or settle for probabilistic success

[Diagram showing synch and async transitions]
Problem: Agreement

Fairness: one task/process
- processes take \( \infty \) steps
  unless stopped
- each user sends exactly one init.

Execution \( \varphi \) is successful if:
1. well-formedness
   \( \text{track}(x) \cup \{v_i \} \in \varphi \), \( \text{init}(v), \text{init}(v_i), \text{decide}(v_i) \)
2. agreement
   all decisions are the same
3. validity
   if all inits are \( v \), then all decides are \( v \)
4. termination

\[ \text{Diag. 17.1} \]

\( f \)-failure termination
- if \( \text{init} \) on all ports
- if ports stopped
  then unstopped
  ports decide

\( 4a \) Wait-free \( (f=n) \)
\( 4b \) Failure-free \( (f=0) \)
\( 4c \) Single-failure \( (f=1) \)
Restrict to Read/Write Memory
Enabling depends only on process state

Read
\[ P_i := f(x_j) \]

Write
\[ x_j := f(P_i) \]

(not) Modify
\[ (P_i, x_j) := f(P_i, x_j) \]

Proof outline:
① Simplify assumptions, defns
② Impossibility for Wait-free alg
③ " Single-failure alg
Assumptions (WLOG)
- \( V \subseteq \{0, 1\} \)
- users generate exactly one \( \text{init} \)
- determinism
- every non-failed process has locally exactly one enabled step
  (add dummy read steps)

```
defn: initialization = init(v_1), init(v_2), ..., init(v_n)
```

Assume all executions begin with initialization

Valence

Final execution of \( \delta \) is:

- 0-valent: for \( \delta \), all extensions
  - all decisions are 0
  - and some decision occurs

- 1-valent: some

- univalent

- bivalent: no decision
  - has \( \leq \) \( \{ \text{extensions} \} \)

- zero decisions

Lemma: For a working algorithm, excess are unique
Suppose $A$ is a working single-failure alg.

**Lemma**: $A$ has bivalent initialization.

Suppose all initializations univalent.

$$\begin{align*}
\text{init}_1(0) \text{ init}_2(0) \ldots \text{ init}_n(0) & \rightarrow 0 \\
\text{init}_1(1) & \rightarrow 0 \\
\alpha & \rightarrow 0 \\
\beta & \rightarrow 1 \\
\text{indistinguishable} & \rightarrow \\
\text{init}_1(1) \text{ init}_2(1) \ldots \text{ init}_n(1) & \rightarrow 1
\end{align*}$$

Consider

$$\begin{align*}
\alpha & \rightarrow 0 \\
\beta & \rightarrow 1
\end{align*}$$
Defn: Finite failure free exec $\alpha$ is a decider if
- $\alpha$ is bivalent
- $\forall i \exists i$ univalent
  - unique locally enabled step of $P_i$

Lemma: $A$ is a working wait-free alg, $A$ has a decider exec $\beta$.

Pf: Suppose no such $\alpha$.

Start with bivalent initialization $\alpha_0$.

Extend to infinite bivalent exec $\alpha$.

If $\exists$ st. $P_i$ takes $\infty$ steps

Consider $\exists \alpha_1$ with all $P_i$ taking finite # of steps

Step 1 after last step

$\alpha_1 \in \{1, 2, ..., \infty\}$

$X$'s fair $\alpha_1$ decides

$\alpha_1, \alpha_2$ are indistinguishable $\Rightarrow \alpha_1$ decides

$\Rightarrow$ at some point $\alpha$ must become univalent in any extension $\Rightarrow$
Thm: No wait-free writing alg.

Pf: Suppose a waiting wait-free alg.

\[ \Rightarrow \exists \text{ decider exec } \alpha. \]

\[ \alpha \text{ is } 0\text{-valent, } \alpha_i \text{ is } 1\text{-valent} \]

Case 1: Suppose \( i \) is a read step

\[ \text{in init: } \{ x_i \text{ stop; } j \rightarrow 0 \} \]

\[ \text{Case 2: } j \text{ sen, same.} \]

\[ \text{Case 3a: } i, j \text{ write to different vars.} \]

\[ \text{inin: } \{ x_{ij} \rightarrow 0 \} \]

\[ \{ x_{ij} \rightarrow 1 \} \]

\[ \text{Case 3b: } i, j \text{ write to some var} \]

\[ \text{inin: } \{ x_{ij} \text{ stop; } j \rightarrow 0 \} \]

\[ \{ x_{ij} \text{ stop; } i \rightarrow 0 \} \]

\[ \therefore A \text{ does not exist.} \]
\( \text{With Read/Modify/Write:} \)

\( A_{5,5}: \) Each process waits for init

\( \text{shared variable } x:= \text{"unknown"} \)

\( \text{init}(v), \Rightarrow \)

1) \( P_i:= v \)

2) \( \text{Modify:} \)

\[
(P_i, x):= \begin{cases} 
(P_i, P_c) & \text{if } x=\text{"unknown"}, \\
(x, x) & \text{otherwise}
\end{cases}
\]

3) \( \text{decide}_i(P_c) \)