1. Check if \( a \leq b \) then \( c(a) > c(b) \)

2. Total order = extension \( ightarrow \) (partial order)

3. Continue if any even \( a, b \)

4. Every relation \( \rightarrow \)

5. Compare the clock processions before and after.

6. Distributed mutual exclusion with a lamp.
A process which has been granted the resource

\[ P > P' \]

And for distributed shared exclusive for

\[ \text{if } (a_1 = c_2) \text{ and } P' > P \]

\[ \exists x \text{ such that } P' > P \]

\[ \exists x \text{ and only if } P' > P \]

And only if

Is an event in process P' then a \leq 0

Def: if a is an event in process P' and 0 <= t
Not a Kanji problem?

prefix

(Known)

unprocessed

unprocessed

Becomes

V2+V3
ack

T: Request release resource

Other necessary things:

System:

- Any number in R
- Value of lock
- Time in init
- S: it is less than stamp

Init: It: 1o, request released

R0: Key: Assume for process i;

Data Structure: For each process i:

Alg:

(no permanent failure)
2. Every process is eventually delivered
between any two processes.

1. There is a FIFO, in order necessary delivery
Algorithm Rules:

AI: To request resource. Pi sends request to every other process.

A2: When process Pi receives request from Rj, it places it on Rj's request. (broadcast)

To release resource, process Pi releases resource.  
To release resource, process Pi removes it from Rj.  
To release resource, process Pi releases resource broadcast (broadcast)

VCi: \[ C_0, c_1, \ldots, c_n \]
A9: When process $P_j$ receives $P_i$ release resource message then it removes the corresponding entry $T_{m_j}: P_i$ request resource from $R_{q_j}$

A5: Process $P_i$ is granted resource when:
(i) there is $T_{m_j}: P_i$ request resource msg in $R_{q_j}$ which is earliest according to $\Rightarrow$
(ii) $P_i$ has received a msg from every other process with a higher timestamp than $T_{m_j}$

$\downarrow$ Assumption $\Rightarrow$ then process $P_i$ has seen all the previous requests.

Generalized as states $M/c$ approach to any distributed co-ordination problem such replica system.
- What happens if any process fails?

- The command is not executed, so no relation.

- The command is achieved because all processes proceed.

- All the processes of the system command issued by the process independently simulate the command.

- In a partition x (e.g.,)

  - \( E \cap S = \emptyset \)
  - \( E \cap S = \{5, \ldots, 5\} \)
  - \( E \cap S = \{5\} \)
  - \( E \cap S = \emptyset \)

- Set of possible commands (e.g., commands)
Consistency preserved.

Example: Use lock in disk buffer cache for a limited period of time.

Specific file writes over protected (or accessed) resources:

Lease: Is a contract that gives the holder...

Leasing v.s. Locks

- In monolithic, per resource usage.
- In distributed systems, exclusive.
- Performance & distributed mutual exclusion
Adv of Leases over Locks:

- Fault tolerance: if a process fails while holding a lease, upon expiration of that lease, server is free to grant the resource to other process.
- we will see its use in GFS.

Ref. Leaser: An Efficient Faulttolerant Mechanism for distributed File Cache Consistency