- Implementing
- Client Server (synchronous)
no equipments/epil ulcer

C, S ->

integrate critical sections to exclude

F' (G, 0)

Dished out, further exclusion.
It reduces if.

One else can get it.

Do you like it too?

Go do it.
\[ \{ r_i \} \subseteq \mathbb{R} \]  

Each element is an \( r_i \), \( i = 1, 2, \ldots \).

Assume there is a local extremum for each \( r_i \).

If \( r_i \) is a local extremum, then "local center" of \( \{ r_i \} \).

Central authority measures read.
Dr. do not leave.
If you do not leave, please bring me my conference back.

Look on Fig. 4.
For all of Fig. 4, it looks

Check does not match.

Well-behaved check.
null([0..n-1])

if - exit ()

send (msg, inst)

Class

Child

loc (p) msg)

Seal (p)

loc "Resource

Clock ID

Clock

loc
loop
recv (msg, msg)

{ check/check, resource

lock (R)

while (R)

reply
\begin{align*}
\text{else} & \quad \text{do not send msg} \\
& \quad \text{Fail} \quad \text{something}, \text{something} \\
& \quad \text{else} \\
& \quad \text{queue} \\
& \quad \text{neg Ack} \\
& \quad \text{seq} (P_4, m_{94}) \\
& \quad \text{change to L, agg chose (L)} \\
& \quad \text{if} \ (L[T] \neq \text{Stop} \Rightarrow \text{L}) \\
& \quad \text{else (R)} \\
& \quad \text{vector function}
\end{align*}
Local response from外交

Lack (L2)

Local (L1)

Lack (L2)

Local (L1)

Effector response (DC, or AIC)

Additional background (no feature)

Temperature balance (no feature)
declarative

- declare (procedure)

| def | let | guard | type | name | def | let | guard | type | name |

```let
seal (rep, "of")
```

no
```let
yes, change during 12 in L[c]
```

```let
\( r = \text{def}(L[c]) \)
```

```let
\text{vnl}\text{assess } \text{f in } L[c] \\
\text{end}\text{vnl}
```

```let
\text{check } \text{(r)}
```
Make it more complex

\[
\rightarrow \text{ Read/Write locks.}
\]

\[
\downarrow \text{ Data structure}
\]
LT
→ R₁

status - U, R, W → (locks) modes

# of pending
→ 2 queues

# of readers holding the lock (if mode = R)
... could we be sure:

1. Should we save

2. Could we save
```plaintext
else count++

if count != 0
    process

else if (mode == m)
    process

else if (mode == g)
    process

if (random() % 100) >= 70
    process

if (k == value)
    process
```
\begin{align*}
\text{else Add to } \mathcal{W}_g \text{ (verify peak 4)} \\
\text{end} & \quad \text{mode} = \text{ mode } \leftarrow \text{ greatest } L_c \\
\text{end-local(R)}
\end{align*}
$\text{send to other core}$

$\text{mode = u lumber}$

if $p$ is empty (assumed) then

else $\text{send (rep, oe)}$

if $\neg p \neq \emptyset$ (W4) then

if $\neg (W4 \text{ not empty})$ then

if $\text{count} = 0$ then

$\text{count -- in local thread}$

$\text{R - unload (e) if of readers}$
3

\begin{align*}
\text{send (client, ok)} & \\
\text{else mode = update} & \\
\text{\quad if (msg not empty)} & \rightarrow \text{send to server} \\
\text{\quad else} & \\
\text{\quad \quad msg = \langle} \\
\text{\quad \quad \quad send to reader} & \\
\text{\quad \quad \quad if (msg not empty) \rightarrow do multiplication} & \\
\text{\quad \quad } & \\
\text{\quad \quad \quad \quad M \cdot \text{update}(R)} & \\
\end{align*}
we balanced?

yes

"no awkward of it"

it looks fine in a week
\[ G \rightarrow \text{merge up, a churn} \rightarrow \text{merge up} \]

\[
\text{if } \text{new version} \}
\text{else}
\]

\[ \text{if} \text{ (local count} = 1) \rightarrow \text{Churn} \]

\[ \text{update } (x \rightarrow y) \rightarrow \text{check} x \]