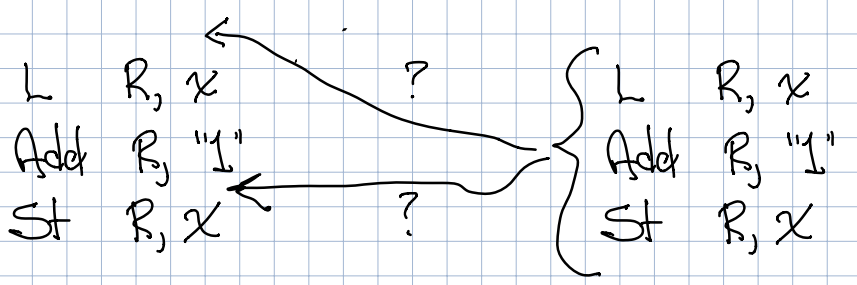
process ← abstraction of: core, processor, process, thread...
- sequential execution of atomic actions
- access to shared memory
- private set of registers
  - PC
  - general purpose regs

Arbitrary interleaving: models time multiplexing with context switches

Shared memory: models any shared resource

atomicity: Instruction that makes at most 1 reference to at most 1 shared memory location appears to be atomic

```
X := X+1   ⟹    L   R1, X
                 Add R1, "1"
                 St  R1, X
```



Critical Section Problem

P_1: do forever
entry$_1$
CS_1
exit$_1$
NCS_1
end

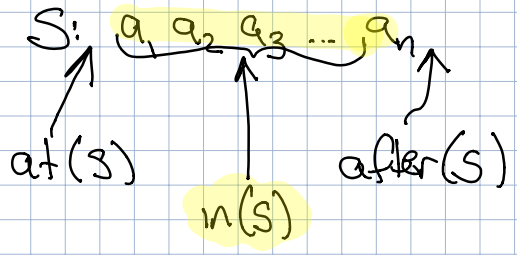
//

P_2: do forever
entry$_2$
CS_2
exit$_2$
NCS_2
end

// means parallel or concurrent execution

Terminology:

Pgm counter



Assumption:

If in(CS_i) then eventually after(CS_i)
= CS_i is terminating

Requirements

- Mutual Exclusion: $\neg(in(CS_1) \wedge in(CS_2))$
- Non Blocking:
  - If at(entry$_i$) then eventually at(CS_i).
  - If in(NCS_2) then process P_1 not blocked
    " in(NCS_1) " " P_2 "

Solution

(3)

Def await B : delays execution until B is true

Boolean expr

implementation: while ¬B do skip end

Solution Attempt 1 Introduce variables: in_1 in_2

P_1: process
do forever
$in_1 := true$ $\{in_1\}$
await $\neg in_2$
$\{in_1 \wedge \neg in_2\}$
CS_1
$in_1 := false$ $\{\neg in_1\}$
NCS_1
end

P_2: process
do forever
$in_2 := true$ $\{in_2\}$
await $\neg in_1$
$\{in_2 \wedge \neg in_1\}$
CS_2
$in_2 := false$ $\{\neg in_2\}$
NCS_2
end

Solution attempt 1 Mutual Exclusion

$in(CS_1) \Longrightarrow in_1 \wedge \neg in_2$

$in(CS_2) \Longrightarrow in_2 \wedge \neg in_1$

$\therefore \quad in(CS_1) \wedge in(CS_2) \Longrightarrow in_1 \wedge \neg in_2 \wedge in_2 \wedge \neg in_1$

$\Longrightarrow$ false

$\therefore$ Mutual Exclusion satisfied.

Solution attempt 1: Non-Blocking

Blocked $in(entry_1) \Longrightarrow in_1 \wedge \neg(\neg in_2)$

$= in_1 \wedge in_2$

Blocked $in(entry_2) \Longrightarrow in_2 \wedge \neg(\neg in_1)$

$= in_2 \wedge in_1$

$\therefore$ Blocked $in(entry_1) \wedge$ Blocked $in(entry_2)$

$\Longrightarrow in_1 \wedge in_2$

(Needed to be "false") ☹

Non-Blocking not satisfied.

Solution attempt 2: Weaken condition is await

(5)

var turn: integer init(1) ... could be 2!

Invariant I: turn=1 ∨ turn=2 Add

P_1: process
do forever
$m_1 := true$ $\{m_1 \wedge I\}$
await $\neg m_2 \vee turn=1$
$\{m_1 \wedge (\neg m_2 \vee turn=1) \wedge I$
CS_1
$m_1 := false$ $\{\neg m_1 \wedge I\}$
NCS_1
end

P_2: process
do forever
$m_2 := true$ $\{m_2 \wedge I\}$
await $\neg m_1 \vee turn=2$
$\{m_2 \wedge (\neg m_1 \vee turn=2 \wedge I\}$
CS_2
$m_2 := false$ $\{\neg m_2 \wedge I\}$
NCS_2
end

How to check soundness of assertions?

- Sequential Correctness

```
  ⋮
 {p}
  S
 {q}
  ⋮
```

Exec S in any state satisfying P & if S terminates then Q holds of the resulting state

- "Interference freedom" ← invented by Cornell's David Gries & S. Owicki

```
Process                    Process
  ⋮                          ⋮
 {P}                      {pre(S)}
  ⋮                          S
end                        end
```

Must show

$\{pre(S) \wedge P\}\ S\ \{P\}$ holds

for all S in one process & P in the other

Compare: N stmts × N+1 assertions } for N action trace
vs 2^N interleavings

Example of sound/unsound assertions:

$x := 1$
$\{x = 1\}$
$y := 2$
$\{y = 2\}$
$z := x + y$
$\{z = 3\}$

unsound!
If $z := x+y$ exec in a state satisfying $pre(z := x+y)$ which is "$y = 2$" then no guarantee that $x = 1$ holds

$x := 1$
$\{x = 1\}$
$y := 2$
$\{y = 2 \wedge x = 1\}$
$z := x + y$
$\{z = 3\}$

sound

Showing interference-freedom for (5)

P_1 P_2

$m_1 := \text{tt}$ vs $\{m_2 \wedge I\}$

$\{m_2 \wedge (\neg m_1 \vee m = 2) \wedge I\}$

$\{\neg m_2 \wedge I\}$

P_1 P_2

$m_1 := \text{ff}$ vs $\{m_2 \wedge I\}$

$\{m_2 \wedge (\neg m_1 \vee m = 2) \wedge I\}$

$\{\neg m_2 \wedge I\}$

problem!

Soln: exec

$n_1 := true$.

$turn := 2$ together

Use $\langle S \rangle$ notation

to signify that S is exec

as an indivisible operation

E-g

$\langle n_1 := true;\ turn := 2 \rangle$

Add

P_1: process
do forever
$\langle m_1 := true \ \{m_1 \wedge I\} \ turn := 2 \ \{m_1 \wedge I\} \rangle$
await $\neg m_2 \vee turn = 1$
$\{m_1 \wedge (\neg m_2 \vee turn = 1) \wedge I\}$
CS_1
$m_1 := false \ \{\neg m_1 \wedge I\}$
NCS_1
end

P_2: process
do forever
$\langle m_2 := true \ \{m_2 \wedge I\} \ turn := 1 \ \{m_2 \wedge I\} \rangle$
await $\neg m_1 \vee turn = 2$
$\{m_2 \wedge (\neg m_1 \vee turn = 2) \wedge I$
CS_2
$m_2 := false \ \{\neg m_2 \wedge I\}$
NCS_2
end

Interference-freedom for (10) if < > deleted.

(11)

(process 1) vs (process 2)

$m_1 := true$

$\{in_1\}\ turn := 2$

$\{in_1\}\ m_1 := false$

$\{in_2\}$

$\{in_2 \wedge (\neg in_1 \vee turn = 2)\}$

$\{\neg in_2\}$

Soln

$\{in_2 \wedge (\neg in_1 \vee at(turn := 2) \vee turn = 2)\}$

Add

true when pgm counter in process 1 is about to execute "$turn := 2$"

Add ⟨ ⟩ to repair interference after remainder

12

P_1: process
do forever
$m_1 := true$ $\{m_1 \wedge I\}$ $turn := 2$ $\{m_1 \wedge I\}$
await $\neg m_2 \vee turn = 1$
$\{m_1 \wedge (\neg m_2 \vee turn = 1 \vee at(turn := 1)) \wedge I\}$
CS_1
$m_1 := false$ $\{\neg m_1 \wedge I\}$
NCS_1
end

P_2: process
do forever
$m_2 := true$ $\{m_2 \wedge I\}$ $turn := 1$ $\{m_2 \wedge I\}$
await $\neg m_1 \vee turn = 2$
$\{m_2 \wedge (\neg m_1 \vee turn = 2 \vee at(turn := 2)) \wedge I$
CS_2
$m_2 := false$ $\{\neg m_2 \wedge I\}$
NCS_2
end

Rev 9/24

13

Mutex satisfied?

$in(CS_1) \wedge in(CS_2) \Longrightarrow I \wedge$

$in_1 \wedge (\neg in_2 \vee turn=1 \vee at(turn:=1)) \wedge I$

$\wedge\ in_2 \wedge (\neg in_1 \vee turn=2 \vee at(turn:=2)) \wedge I$

Note: $in(CS_1) \Longrightarrow \neg at(turn:=2)$.

$in(CS_2) \Longrightarrow \neg at(turn:=1)$

$\Longrightarrow \quad in_1 \wedge in_2 \wedge \underbrace{turn \neq 2 \wedge turn \neq 1}_{= \text{false}}$

☺

Mutual Blocking satisfied

$$in_1 \wedge \neg(\neg in_2 \vee turn = 1) \wedge I$$
$$\wedge\ in_2 \wedge \neg(\neg in_1 \vee turn = 2) \wedge I$$

$\Rightarrow$

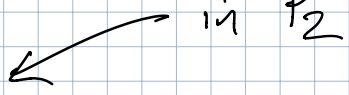
$$in_1 \wedge in_2 \wedge turn \neq 1 \wedge I \wedge$$
$$in_2 \wedge in_1 \wedge turn \neq 2 \wedge I$$

$\Rightarrow$

$$turn \neq 1 \wedge turn \neq 2 \wedge I$$

$\Rightarrow$ false

But....



await $\neg m_1 \vee turn = 2$

involves 2 shared variable references!

$\equiv$

Violates defn of "atomic"

while $\neg(\neg m_1 \vee turn = 2)$ do skip end

$\Downarrow$

while $m_1 \wedge turn \neq 2$ do skip end

$\Downarrow$

$l_2 := false$

$\{I: l_2 \Rightarrow (\neg m_1 \vee turn = 2 \vee at(turn := 1))\}$

while $\neg l_2$ do

$l_2 := l_2 \vee \neg m_1$

$l_2 := l_2 \vee turn = 1$

end

Code involves only 1 shared var reference