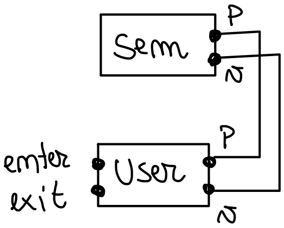
LCD (11/03/2024)

CCS → Syntax

→ Operational behaviour (via syntax driven rules)

* Example   two processes



$$Sem = p.v.Sem$$

$$User = \bar{p}.enter.exit.\bar{v}.User$$

$$(Sem \mid User) \setminus \{p, v\}$$
$$\downarrow \tau$$
$$(v.Sem \mid enter.exit.\bar{v}.User) \setminus \{p, v\}$$
$$\downarrow enter$$
$$(v.Sem \mid exit.\bar{v}.User) \setminus \{p, v\}$$
$$\downarrow exit$$
$$(v.Sem \mid \bar{v}.User) \setminus \{p, v\}$$

τ

$$Sem = p.v.Sem$$

$$User = \bar{p}.enter.exit.\bar{v}.User$$

$$(User \mid Sem \mid User) \setminus \{p, v\}$$

$$(User \mid Sem \mid User) \setminus \{p, v\} \xrightarrow{\tau} (enter.exit.\bar{v}.User \mid v.Sem \mid User) \setminus \{p, v\} \xrightarrow{enter} \ \xrightarrow{exit}$$

$$(User \mid Sem \mid User) \setminus \{p, v\} \xrightarrow{\tau} (User \mid v.Sem \mid enter.exit.\bar{v}.User) \setminus \{p, v\} \xrightarrow{enter} \ \xrightarrow{exit}$$

τ τ

$$Sem = p.v.Sem$$

$$User = \tau.\bar{p}.\underbrace{enter.\ \tau\ .exit}_{\text{critical section}}.\bar{v}.User$$

$$\boxed{(User \mid Sem \mid User) \setminus \{p, v\}} \approx Spec$$

enter exit

enter exit enter exit ....

$$Spec = enter.exit.Spec$$

$$(\,enter\ .\cancel{enter}\ exit\ .\ exit\,)$$

<u>Exercise</u> : Semaphore which allows at most k processes at the same time in the critical section

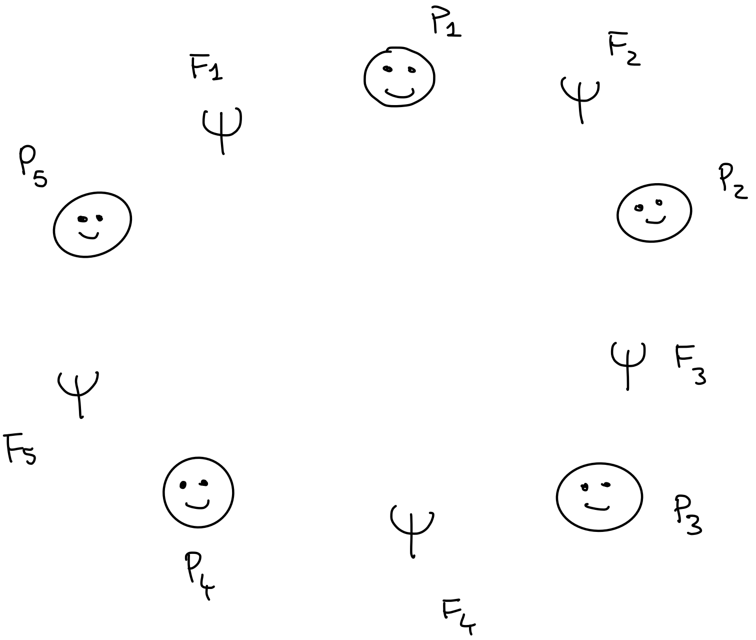
$K = 2$

$$\begin{cases} Sem_2 = p\ .\ Sem_1 \\ Sem_1 = p\ .\ Sem_0 + v.\ Sem_2 \\ Sem_0 = v.\ Sem_1 \end{cases}$$

Implement the above behaviour by using Sem as a module

$$\boxed{Sem \mid Sem} \sim \boxed{Sem_2}$$

* Dining Philosophers



philosophers

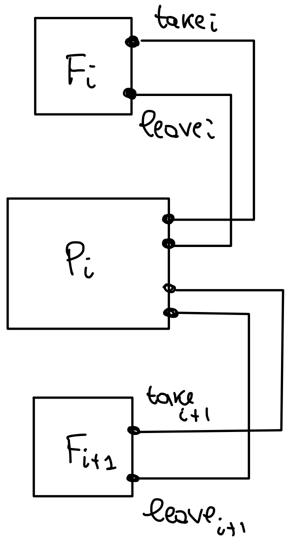
- think
- eat

P_i needs

F_i & F_{i+1}

$$F_i = take_i . leave_i . F_i$$

$$P_i = think . \overline{take_i} . \overline{take_{i+1}} . eat . \overline{leave_i} . \overline{leave_{i+1}} . P_i$$



$$Sys = (F_1 \mid P_1 \mid F_2 \mid P_2 \mid F_3 \mid P_3 \mid F_4 \mid P_4 \mid F_5 \mid P_5) \setminus \{ take_i , leave_i \mid i = 1,..,5 \}$$

Spec = ?

* Peterson's mutual exclusion

for two processes P_1 and P_2

shared variables
- b_1, b_2 boolean

  $b_i \equiv P_i$ wants to enter the critical section

- k with values 1 or 2

  "turn variable"

process P_i ($i \in \{1,2\}$, I use j for "the other value")

while true do

begin

<non critical code>

b_i = true

$k = j$

while (b_j and $k = j$) skip

<critical code>

b_i = false

end

Is it really working? Does it ensure mutual exclusion?

CCS encoding?

$$P_i = \tau . \overline{b_i w t} . \overline{k w j} . P_{i1}$$

$$P_{i1} = b_j r f . P_{i2} + b_j r t . (k r j . P_{i1} + k r i . P_{i2})$$

$$P_{i2} = enter_i . \tau . exit_i . \overline{b_i w f} . P_i$$

process P_i

while true do

begin

<non critical code>

b_i = true

$k = j$

while (b_j and $k = j$) skip

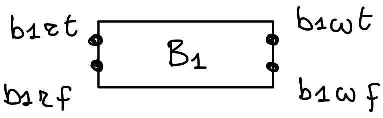
<critical code>

b_i = false

end

How do we represent a variable? As a process!

b_1 states B_{1t}, B_{1f}



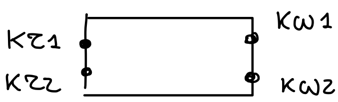
$$B_{1f} = \overline{b_1rf}.B_{1f} + b_1wt.B_{1t} + b_1wf.B_{1f}$$

$$B_{1t} = \overline{b_1rt}.B_{1t} + b_1wt.B_{1t} + b_1wf.B_{1f}$$

(if we had value passing

$$B1(x) = \overline{b_1r}(x).B1(x) + b_1w(y).B_1(y)$$
)

for K: K1, K2



<u>System</u>

$$Sys = (P_1 \mid P_2 \mid B1 \mid B2 \mid K) \setminus \{\text{all channels apart from } enter_i,\ exit_i\}$$

$B1 = \tau.B_{1t} + \tau B_{1f}$

$B2 =$

$K = \ldots$ ↖ initial values

(Is $B1 = B_{1t} + B_{1f}$ the same?)

$\boxed{2.7}$ Office: derive transition by using the rules

$\boxed{2.8}$ show that transitions exist or do not exist ...