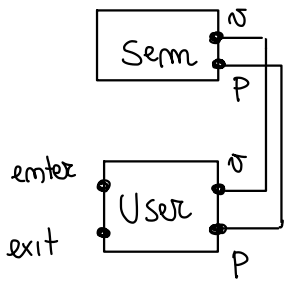
LCD (03/03)

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

$Sem = p.\ v.\ Sem$

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

τ

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

τ

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

$\downarrow enter$

$\downarrow exit$

τ

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

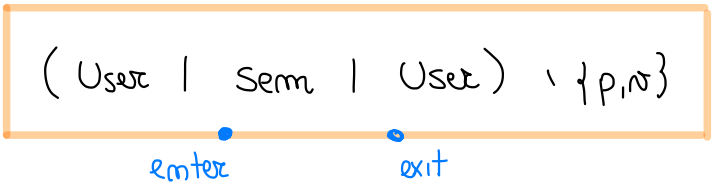
$\downarrow enter$

$\downarrow exit$

$Sem = p . v . Sem$

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

critical code



$\approx$ Spec

enter . exit . enter . exit . . .

enter . enter .
NO!

$Spec = enter . exit . Spec$

EXERCISE: Semaphore which allows K programs in the critical section at the same time

$K = 2$

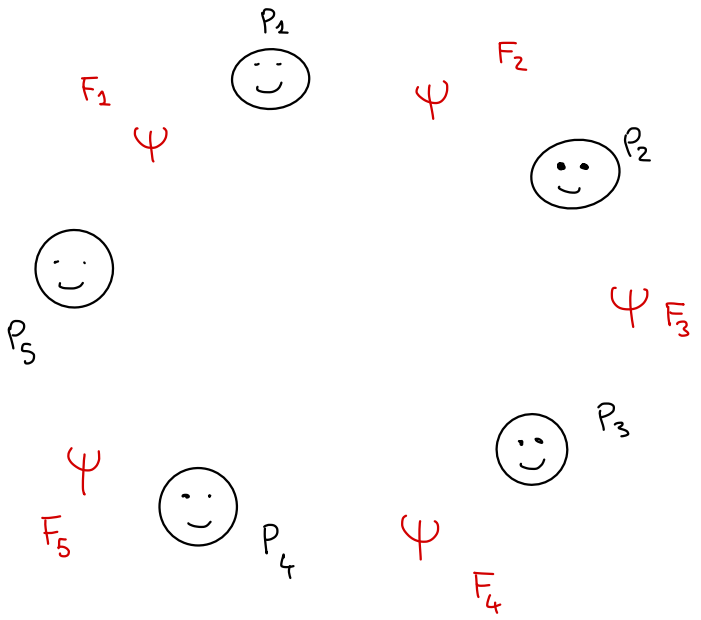
$Sem_2 = p . Sem_1$

$Sem_1 = p . Sem_0 + v . Sem_2$

$Sem_0 = v . Sem_1$

Implement Sem_2 by using Sem as a "module"

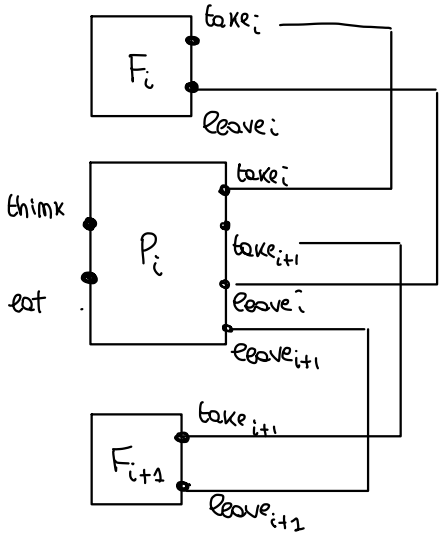
* DINING PHILOSOPHERS



philosophers
- thimk
- eat

$F_i = take_i \,.\, leave_i \,.\, F_i$

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



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

Spec = ?

* Peterson's Mutual Exclusion

mutual exclusion for two processes P_1, P_2

shared memory – b_1, b_2 booleans

b_i = " P_i wants to enter critical section "

– $k \in \{1,2\}$

turn variable

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

while true do

< non critical code >

b_i = true

$k = j$

while (b_j and $k = j$) do skip

< critical code >

b_i = false

end while

Is it working? Does it ensure mutual exclusion?

Encoding in CCS

process P_i

while true do

$P_i = \tau.$ ---------- < non critical code >

$\overline{b_i}\omega t.$ ---------- b_i = true

$\overline{k}\omega j.$ ---------- $k = j$

P_{i1} while (b_j and $k = j$) do skip

$P_{i1} = b_j \& f.\ P_{i2} + b_j \& t.$

$(k?i.\ P_{i2} + k?j.\ P_{i1})$ < critical code >

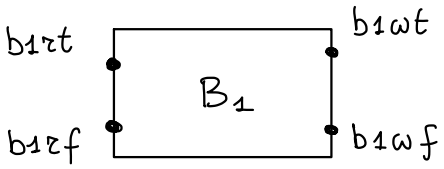
$P_{i2} = enter_i.\ \tau.\ exit_i$ ---------- b_i = false

$\overline{b_i}\omega f.\ P_i$ ----------

end while

How to encode variables ?

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



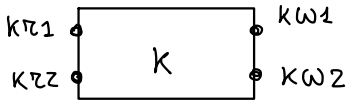
$$B_{1t} = \overline{b_{1r}t}.\ B_{1t} + b_{1w}t.\ B_{1t} + b_{1w}f.\ B_{1f}$$

$$B_{1f} = \overline{b_{1r}f}.\ B_{1f} + b_{1w}t.\ B_{1t} + b_{1w}f.\ B_{1f}$$

(value passing

$$B_1(x) = \overline{b_{1r}}(x).\ B_1(x) + b_{1w}(y).\ B_1(y) \quad)$$

variable K



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

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

Exercises

2.7 Office derive the transition by using rules

2.8 showing that some transitions exist, some do not exist.