Sharing a meal

Two persons, one set of cutlery

proctype P{grab knife; grab fork; eat}
proctype Q{grab fork; grab knife; eat}

If you: {run P; run P} everything will work fine. You will eat one after another.
{run P; run Q} will eventually result in a deadlock.

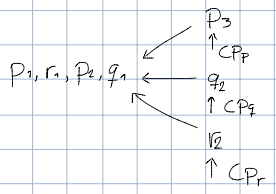
Shared a bank acc

proctype W{loc = bal; loc--; out!1; bal = loc}
- bal is a shared global balance
- loc is local register
- out is payout

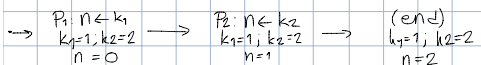
Then {run W; run W} could result in both succeeding withdrawing but with the acc being debited only once.

- loc1 = bal; loc2 = bal; loc1--; loc2--; out!1; out!1; bal?

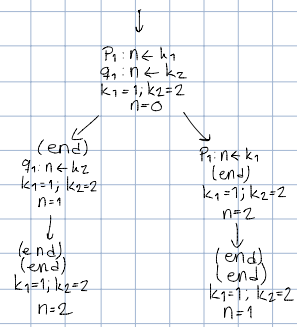
Interleaving - Slides from book



Seq



concurrent



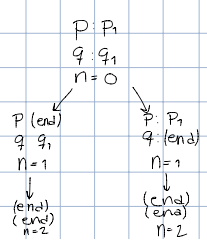
One of the possible run of the state diagram above.

P	Q	n	k_1	k_2
P1: n ← k1	q1: n ← k2	0	1	2
(end)	**q1: n ← k2**	1	1	2
(end)	(end)	2	1	2

Atomic assignment

Global: n ← 0
Processes: P, q
Local: P1: n ← n+1 : q1: n ← n+1

Will always give two



Scenarios: Slide 2:13

Atomic w/ one global

Global: $n \leftarrow 0$

Processes: P

int tmp

P_1: tmp $\leftarrow n$

P_2: $n \leftarrow$ tmp+1

Q

int tmp

q_1: tmp $\leftarrow n$

q_2: $n \leftarrow$ tmp+1

Scenarious

$P_1, P_2, q_1, q_2 \rightarrow n = 2$

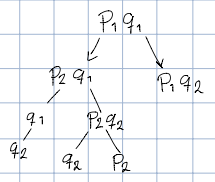
$P_1, q_1, P_2, q_2 \rightarrow n = 1$

$P_1, q_1, q_2, P_2 \rightarrow n = 1$

$q_1, q_2, P_1, P_2 \rightarrow n = 2$

$q_1, P_1, q_2, P_2 \rightarrow n = 1$

$q_1, P_1, P_2, q_2 \rightarrow n = 1$



Concurrent counting

Global: $n = 0$

Processes: P Q

Local: tmp tmp

P1: do 10 times q2: do 10 times

P2: tmp $\leftarrow n$ q2 tmp $\leftarrow n$

P3: $n \leftarrow$ tmp+1 q3 $n \leftarrow$ tmp+1

Small codes: much thought required

Critical Section

Global: —

Processes: P Q

P	Q
local	local
loop ∞	loop ∞
non-crit	non-crit
preprotocol	preprotocol
critical	critical
Post protocol	Post protocol

Library

All atomic	Sections
Close the whole	Look left
library as soon	
as one enters	

Requirements and assumptions: slides from Prasad

- Correctness reqs
  - Both p and q cannot run their Critical Section (CS) at once. Mutex
  - If p and q both wish to enter CS one must eventually succeed. Deadlock
  - If p tries to enter CS it will eventually succeed. Starvation

- Assumptions
  - A process in its CS will leave eventually
  - Progress in non CS is optional.

Back to slides from Ben Ari (Slide 3.1)

Solving CS

Global: turn = 1

Processes: P Q

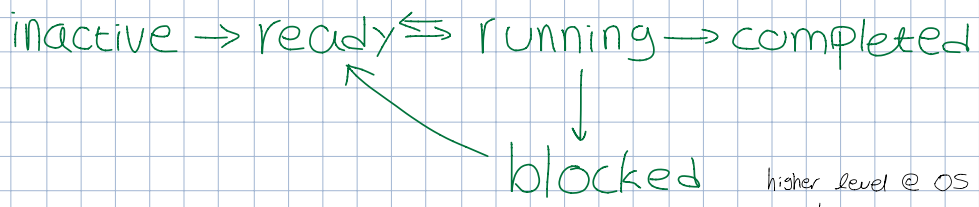
P	Q
loop forever	loop forever
P1: n-cs	q1: n-cs
P2: await turn=1	q2: await turn=2
P3: CS	q3: CS
P4: turn $\leftarrow$ 2	q4: turn $\leftarrow$ 1

Issues

Q terminates during n-CS => turn is stuck at 2 => Starvation for p

Slides from BA 61

Stage changes for a process



higher level @ OS level

How do we "program" the mail from the library, at a software level?