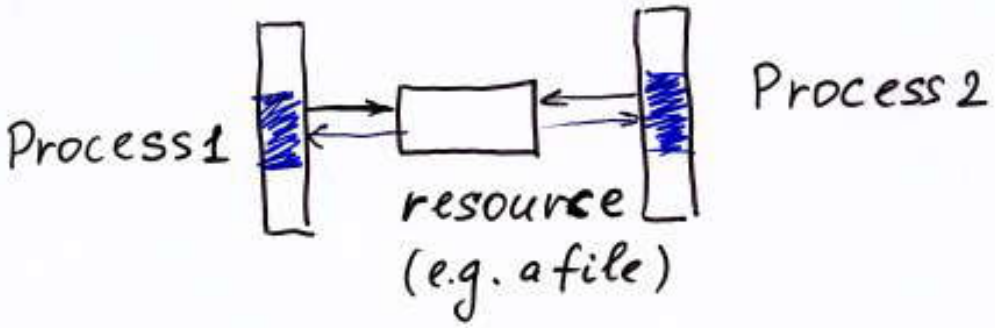
Mutual Exclusion



Critical sections of code include all the access to the shared resource

(only one process can be in its critical section at a time)

Task: find a protocol for determining which process is allowed to enter its critical section at which time

The protocol must have the following properties:

Safety: only one process is in its critical section at a time

Problem: a protocol, where none of the processes ever enters its critical section is safe!

also, need:

Liveness: Whenever any process wants to enter its critical section, it will eventually be permitted to do so

non-blocking: A process can always request to enter its critical section (when in non-critical one)

no strict sequencing: Processes need not enter their critical section in strict sequence

(access of Process 1 and access of Process 2 to their critical sections don't have to alternate)

C_1 can be after C_1

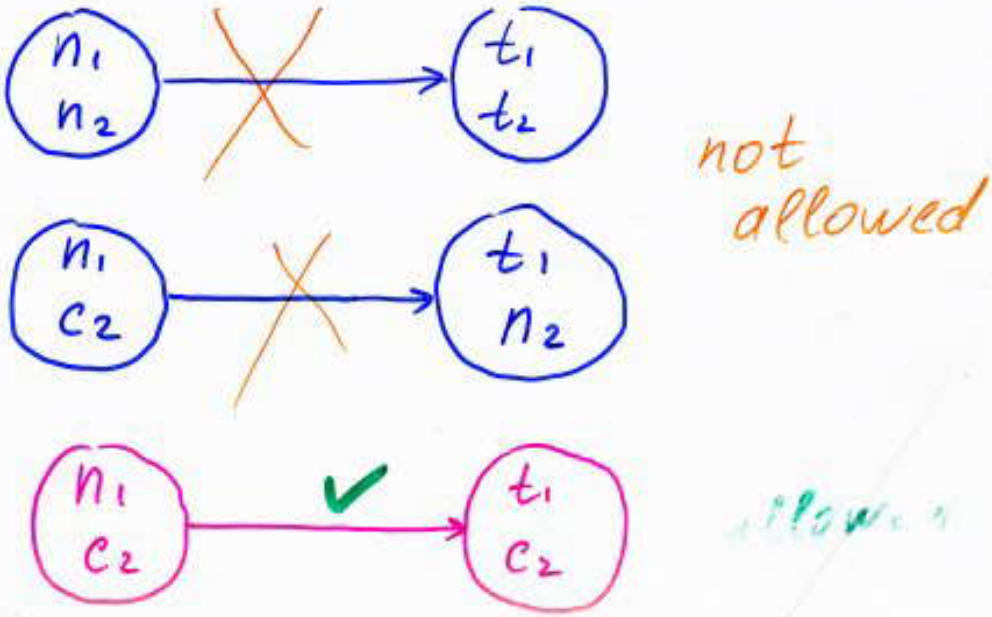
n_i process i is in its non-critical state

t_i process i is trying to enter its critical state

c_i process i is in its critical state

$$n_i \to t_i \to c_i \to n_i \to t_i \to c_i \to n_i \ldots$$

only one process can make a transion (say, from n to t) at a time



LTL	CTL
Safety	
$G \neg (c_1 \wedge c_2)$	$AG \neg (c_1 \wedge c_2)$
Liveness	
$G(t_1 \rightarrow F c_1)$	$AG(t_1 \rightarrow AF c_1$
$\wedge$	$\wedge$
$G(t_2 \rightarrow F c_2)$	$AG(t_2 \rightarrow AF c$
Non-blocking	
—	$AG(t_1 \rightarrow EX$
	$\wedge$
	$AG(t_2 \rightarrow EX t_2$

For the first modelling attempt:

Safety: ✓

Liveness: ✗ because

there is a path, where t_1 is true in a state, but from that state on c_0 is false.

$s_0 \to s_1 \to s_3 \to s_7 \to s_1 \to s_3 \to s_7 \to \ldots$

($s_1 \to s_3 \to s_7$: loop; $s_1 \to s_3 \to s_7$: loop)

(similar for the second process)

non-blocking: ✓

for every state sat. n_1, there is a successor sat. t_1

(similar for the second process)

No strict sequencing: ✓

there is a path where

$\ldots \underbrace{c_2 c_2 c_2}_{c_2} \_ \underbrace{c_1 c_1 c_1 \ldots c_1}_{c_1} \_ \underbrace{c_2 c_2 c_2}_{c_2} \_ \underbrace{c_1 c_1}_{c_1} \_ \underbrace{c_2 c_2 \ldots c_2}_{c_2}$

such alternation does not occur (for each process)