Recap of Critical Section

Requirements: Mutual exclusion (mutex)
Absence of deadlocks
Absence of starvation

Abbreviating problems

Global: int turn ← 1

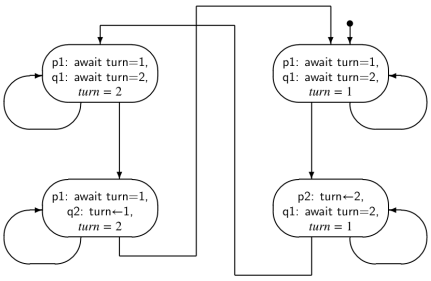
Processes: P

loop forever
1: await turn = 1
2: turn ← 2

Q

loop forever
await turn = 2
turn ← 1

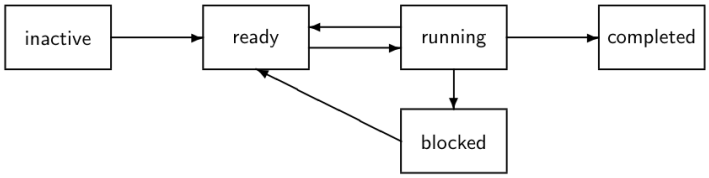
State Diagram for the Abbreviated First Attempt



One should avoid splited requests i.e: 1: Is the knife free?
2: Take the knife.

Since the knife could have been taken before you reach for it. One could solve this atomically: atomic (request grant) ← mail, block by testing and seting in one operation.

State Changes of a Process



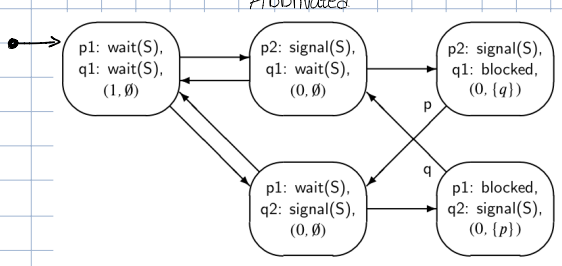
If "blocked" the process cannot countinue until "un blocked". If this is implemented (often in the kernel) we can use semaphores

Algorithm 6.2: Critical section with semaphores (two proc., abbrev.)

binary semaphore S ← (1, ∅)

p		q	
	loop forever		loop forever
p1:	wait(S)	q1:	wait(S)
p2:	signal(S)	q2:	signal(S)

Abbriviated



States for the semaphore
0, {P}
0, {Q}