

## Esercizio 2

1. 1. "P Q"

2. Nella sol "P Q" si può osservare

3. In quel caso si può osservare "PQ" o "QP"

2. 1. Si può scrivere nella forma semplificata  
perché lo schema:

[ S. wait()  
S. signal()

funkziona come un mutex e quindi garantisce la  
mutua esclusione!

2. Bisogna a dimostrare:

(a) l'invariante è vero nello stato iniziale

(b) Se l'invariante è vero ad un istante t  
è vero dopo un passo

Al inizio abbiamo:

#q3 = 0 e gate = 1 e count = k > 0

Quindi l'invariante è valida.

Supponiamo ora che l'invariante è vero ad  
un momento e vediamo se è vero ancora al istante  
successivo t+1.

Tutto dipende dall'istruzione eseguita fra t e t+1.

- Se è q1, allora come il valore di  $\#q_3$ , count e gate non è cambiato, l'invariante è vero in t+1.

- Se è q4, lo stesso ragionamento vale

- Se è q5, count e gate sono cambiati.

Verifichiamo che (a) (b) ... (f) sono vere a t+1.

(a) Se  $(\#q_3 > 0)$  a t+1 allora  
 $\#q_3 > 0$  a t.

Dunque grazie a (b) abbiamo  $\text{count} \geq 0$  a t.

Da allora q5 non cambia gate

Dunque  $\text{gate} = 0$  in t e t+1

(b) Come per (a) mostriamo che (b) è vero a t+1

(c) rimane vero perché  $\#q_3$  non è cambiato

(d) Se  $\text{gate} = 0 \wedge \#q_3 = 0$  a t+1

Allora abbiamo anche  $\text{gate} = 0, \#q_3 = 0$  a t

(perché q5 può solo aumentare gate)

Da in questo caso a t abbiamo  $\text{count} = 0$

e dunque a t+1  $\text{gate} = 1$ .

Quindi non possiamo avere  $\text{gate} = 0 \wedge \#q_3 = 0$  a t+1

e (d) è falso

(e) Se  $\text{count} < 0$  a t+1 allora  $\text{count} < 0$  a t

e quindi il valore di gate non è cambiato da q5

(f)  $gate=0 \vee gate=1$  potrebbe essere falso se a t abbiamo  $gate=1$  e  $count=0$  ma grazie a (e) non è possibile.

• Se è q2 allora

(a) Se in  $t+1$  ( $\#q_3 > 0$ ). Allora come faciamo q2  
Supponiamo che a t abbiamo  $gate > 0$  ma grazie a (f) abbiamo quindi in t  $gate=1$  e q2 lo metti a 0, quindi a  $t+1$   $gate=0$ .

(b) q2 non cambia count quindi è vera a t e  $t+1$

(c) Se a t abbiamo  $\#q_3 = 1$  (grazie a (c) abbiamo  $q_3 < 1$  a t), allora a  $t+1$  potremo avere  $\#q_3 = 2$ , ma se  $\#q_3 = 1$  a t allora grazie a (a) abbiamo  $gate=0$  a t e non possiamo fare q2, quindi c'è un'contradizione e dunque a t abbiamo  $\#q_3 = 0$

(d) a  $t+1$ ,  $\#q_3 > 0$ , quindi (d) è vero

(e) a  $t+1$ ,  $gate=0$  quindi (e) è vero

(f) Se fra t e  $t+1$  faciamo q2 allora a t abbiamo  $gate=1$  (perché a t abbiamo  $gate=0 \vee gate=1$ ) e a  $t+1$   $gate=0$ .

• Se è q<sub>3</sub>.

Come in t abbiamo  $\#q_3 \leq 1$ , abbiamo che in t

$\#q_3 = 1$  e in t+1  $\#q_3 = 0$ .

Quindi (a), (b) e (c) sono vere a t+1.

(d) Se gate=0 a t+1 allora a t abbiamo count < 1. e come abbiamo (b) a t,  
allora a t abbiamo count=1.  
e quindi a t+1, abbiamo count=0  
e (d) è vero

(e) Come a t abbiamo  $\#q_3 > 0$ , allora a t  
abbiamo count > 0.

Se count < 0 a t+1, nondico che a t  
abbiamo count=1

In più abbiamo anche in t griglia a (a) gate=0

Qui di a t+1, se count < 0, abbiamo gate=0

(f) A t abbiamo niente che gate=0, quindi a (t+1)  
abbiamo gate=0 v gate=1

3. Al inizio abbiamo count=k o  $\#q_5 \#q_4 = 0$  quindi

la formula è vera

Eseguire q<sub>1</sub> q<sub>2</sub> non cambia nulla.

Eseguire q<sub>3</sub> ~~non~~ diminuisce count di 1 e aumenta  $\#q_4$  di uno  
— a<sub>5</sub> aumenta count di 1 e diminuisce  $\#q_5$  di uno.

Quindi la formula è un invarianto.

- 2.4. Supponiamo che a un momento il numero di processi in Session critica è  $> k$ .

Abbiamo allora  $\#q_h + \#q_S > k$

Per avere questo, c'è stato un istante  $t$  con  $\#q_h + \#q_S = k$  e a  $t+1$   $\#q_h + \#q_S = k+1$ .

Ma allora a  $t$  abbiamo  $\text{cont} = 0$  grazie alla domanda 3. E in più fra  $t$  e  $t+1$  si esegue per forza  $q_3$  quindi a  $t+1$  abbiamo  $\#q_3 > 0$ .

Ma questo è una contraddizione con 2. (b)!

### 3. Uso 4 semafori

semaphore sedie =  $n - \blacksquare$

semaphore poltrona = 1

semaphore beginbarb = 0

semaphore endbarb = 0

Client

sedie.wait()

poltrona.wait()

sedie.signal()

beginbarb.signal()

endbarb.wait()

poltrona.signal()

multithread Barber

beginbarb.wait()

endbarb.signal()