

Esercizi sugli Schedule

Usr1

$$r_1(n) \ r_2(y) \ w_1(z) \ r_2(n) \ r_3(n) \ w_2(z) \ w_2(y) \ r_1(y)$$

legge da $\rightarrow r_1(y)$ legge che $w_2(y)$

Scrittura finali $\rightarrow w_2(y), w_2(r)$

r_1 legge che w_2 se e solo
se w_1 precede r_1 e non vi e'
 w_k true due.

$$\begin{array}{l} n \ r_1(n) \ r_2(n) \ r_3(n) \\ y \ r_2(y) \ w_2(y) \ r_1(y) \\ z \ w_1(z) \ w_2(z) \end{array}$$

Considerazioni

- * 2 deve precedere 1 per mantenere le leggi due
- * Se 2 precede 1 non puo' essere rispettata la scrittura finale.

Lo schedule non e' in Usr1.

2PL

$$r_1(n) \ r_2(y) \ w_1(z) \ r_2(n) \ r_3(n) \ w_2(z) \ w_2(y) \ r_1(y)$$

	n	y	r
rlock1	t1 t2 t3	t2 t1	
wlock1		t2	t1 t2

Non vi sono deadlocks

Verifica se ci sono deadlock

2PL

$w_0(n) \cdot r_1(n) \cdot r_0(y) \cdot w_1(y) \cdot w_1(n) \cdot r_2(n) \cdot w_3(n) \cdot r_2(y)$

	n	y
r_{lock}	t1 t2	t0 t3
w_{lock}	t0 t1 t2	
	t_3	

Non ci sono deadlock

Verifica se ci sono deadlock

2PL

$r_1(n) \cdot r_2(y) \cdot w_1(z) \cdot r_3(n) \cdot r_3(n) \cdot w_2(z) \cdot w_2(y) \cdot r_1(y)$

	n	y	z
r_{lock}	t1 t2 t3	t2 t1	
w_{lock}	t_2	t1 t2	

t_2 attende t_1 su r

t_2 risponde

Non ci sono deadlock.

Convenzione

→ l'ultima operazione

prevede una commit

Subito dopo, quindi il

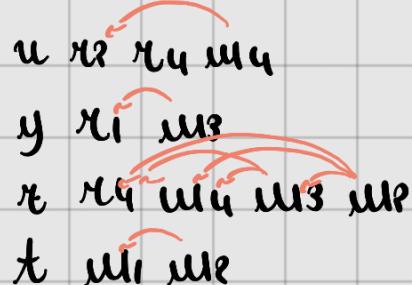
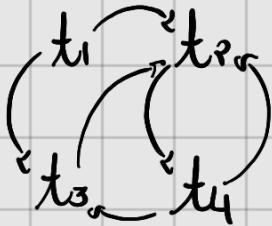
ribescio immediato dei

lock.

Esercizio sugli schedule

$r_2(u) r_u(u) m_4(u) r_u(y) r_u(t) m_4(t) m_3(y) m_3(t) m_1(t) m_2(t)$

Verifico se in CSPI

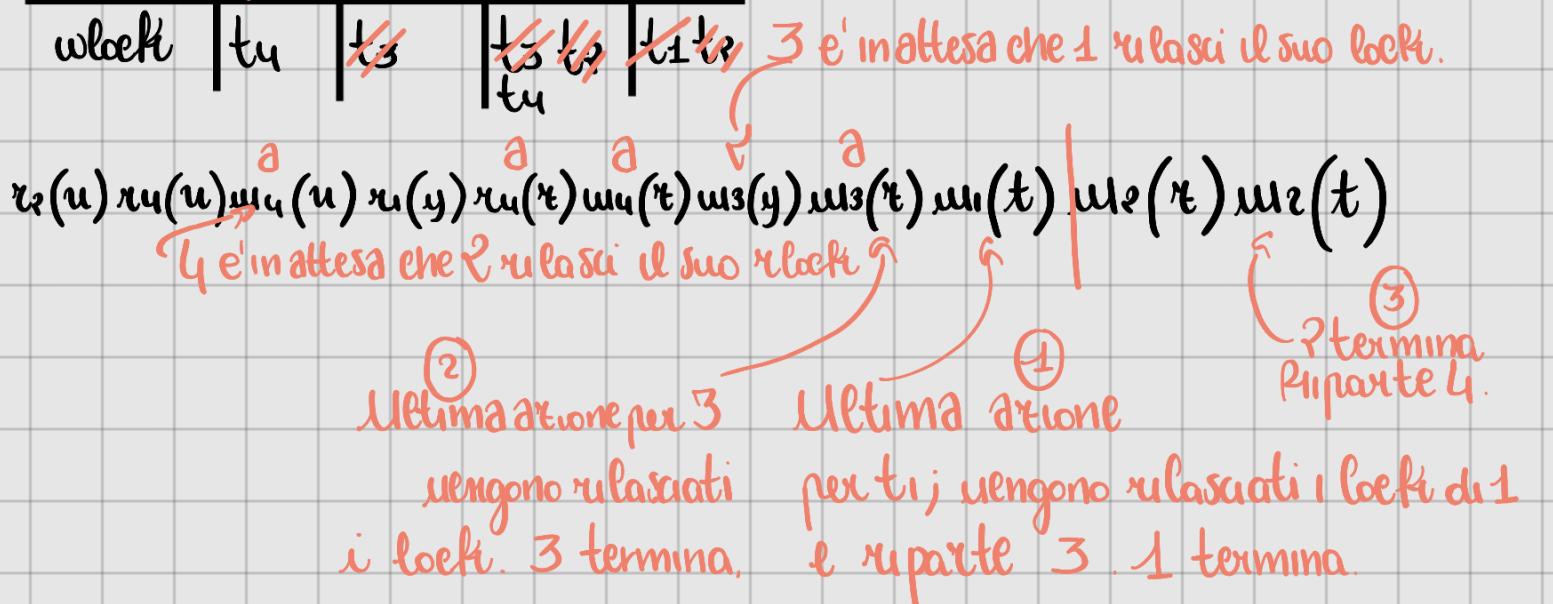


Lo schedule non e' in CSPI poche' il suo grafo dei conflitti presenta cicli.

Verifico se lo schedule presenta dead lock se passato ad uno scheduler PPL.

Costruisco la tabella dei locki.

	u	y	r	t
r locki	t2, tu	t1	tu	
w locki	tu	t3	t3, t2	t1, t3
			tu	



Non ci sono dead locki

Verifica che la schedule sia in USP

$$w_0(u) \quad r_2(u) \quad r_1(u) \quad w_2(u) \quad w_1(r)$$

legge da : $r_2(u)$ legge da $w_0(u)$
 $r_1(u)$ legge da $w_0(u)$
scritture finali : $w_2(r)$, $w_1(u)$

Confrontiamo con uno schedule seriale.

$$w_0(u) \quad r_1(u) \quad r_2(u) \quad w_2(u) \quad w_1(r)$$

legge due : $r_1(u)$ legge due $w_0(u)$
 $r_2(u)$ legge due $w_0(u)$
scritture finali : $w_2(u)$
 $w_1(r)$

La schedule e' in USP.

Verifica che la schedule sia in USP

$$r_1(u) \quad r_2(u) \quad w_2(u) \quad w_1(u)$$

legge da \rightarrow Nessuna
scritture finale \rightarrow $w_1(u)$

Confronto con le sue due permutazioni seriali.

$$r_1(u) \quad w_1(u) \quad r_2(u) \quad w_2(u)$$
$$r_2(u) \quad w_2(u) \quad r_1(u) \quad w_1(u)$$

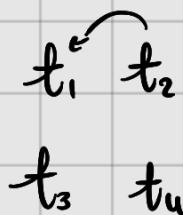
La schedule non e' in USP.

Indica quali schedule sono USR_i o CSR_i
Inoltre, se questi venissero presentati ad uno scheduler
a due fasi, quali transazioni verrebbero messe in attesa?

$\Sigma_1(u) w_1(u) r_2(z) r_1(y) w_1(y) r_2(u) w_2(u) w_2(z)$

Verifico se in CSR_i. Se cosi' non fosse allora sicuramente non sarebbe
in USR_i (poiche' CSR_i c' e USR_i).

Costruisco il grafo dei conflitti. Ricordiamo che a_i e' in conflitto
con a_j se i due operano sullo stesso oggetto ed almeno
uno dei due oggetti e' una scrittura. Disegniamo, quindi,
un nodo per ogni transazione ed un arco da i a j , per
ogni conflitto tra a_i ed a_j tale che a_i precede a_j .



Nel grafo non esistono cicli, quindi lo schedule e' in CSR_i,
e di conseguenza anche USR_i.

Quali transazioni verrebbero messe in attesa in 2PL?

$\Sigma_1(u) w_1(u) r_2(z) r_1(y) w_1(y) r_2(u) w_2(u) w_2(z)$

$$u = r_1 w_1 r_2 w_2$$

$$y = r_1 w_1$$

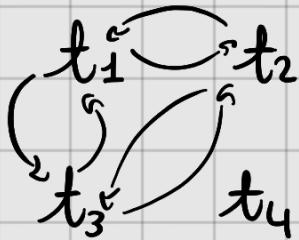
$$z = r_2 w_2$$

Nessuna transazione e' in attesa e non si hanno deadlock.

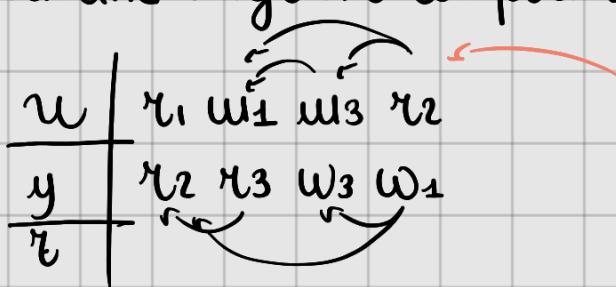
Indica quali schedule sono USP_i o CS_{P_i}
Inoltre, se questi venissero presentati ad uno scheduler
a due fasi, quali transazioni verrebbero messe in attesa?

$\tau_1(u) w_1(u) w_2(u) \tau_2(y) \tau_3(y) w_3(y) w_1(y) \tau_2(u)$

Verifico se in CS_{P_i}. Se così non fosse allora sicuramente non sarebbe in USP_i (poiché CS_{P_i} c'è USP_i).



Per una migliore comprensione dividio le azioni per tabella.



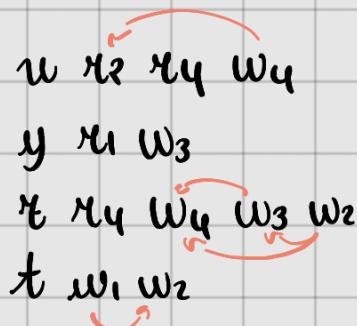
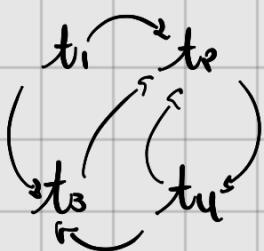
Ogni riga parte da un vertice v_i per cui genera un arco i,j .

Il grafo è circolare, quindi non è in CS_{P_i}.

Esercizio sugli schedule

$$r_2(u) r_4(u) w_4(u) r_1(y) r_4(t) w_4(t) w_3(y) w_3(t) w_1(t) w_2(t) w_2(t)$$

Verifico se lo schedule è in CSP1.



Lo schedule non è in CSP1. Potrebbe essere in USP1.

Verifico se in USP1.

legge da: Nessuna

Soutture finali: $w_2(t), w_2(t), w_3(y), w_4(u)$

Considerazioni

Attraversando le soutture finali possiamo notare che

- t_1 deve precedere t_2 (altrimenti $w_2(t)$ non sarebbe satura finale)
- t_2 deve succedere t_3 (altrimenti $w_2(t)$ non sarebbe satura finale)
- t_2 deve succedere t_4 (altrimenti $w_2(t)$ non sarebbe satura finale)

Capiamo, quindi, che per mantenere le stesse souture finali, t_2 deve essere posta alla fine, d'altra parte così si generano delle leggi che. Lo schedule non può essere più verificabile.

$t_1 \ r_1(y) \ w_1(t)$

$t_2 \ r_2(u) \ w_2(t) \ w_2(t)$

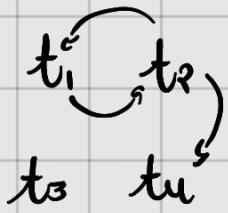
$t_3 \ w_3(y) \ w_3(t)$

$t_4 \ r_4(u) \ w_4(u) \ r_4(t) \ w_4(t)$

Esercizio sugli schedule

$r_1(u) r_2(y) r_1(y) r_3(u) r_1(t) w_1(y) w_2(t) r_4(t) r_1(t) w_2(t)$

Verifico se CSPI.



w r₁ r₃
y r₂ r₁ w₁
r r₁ w₂ r₄
t r₁ w₂

Lo schedule non e' in CSPI. Verifico se in USPI.

legge da : r₁ legge da w₂

soutture finali : w₁(y) w₂(t) w₂(t)

Considerazioni

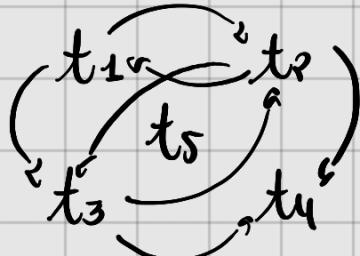
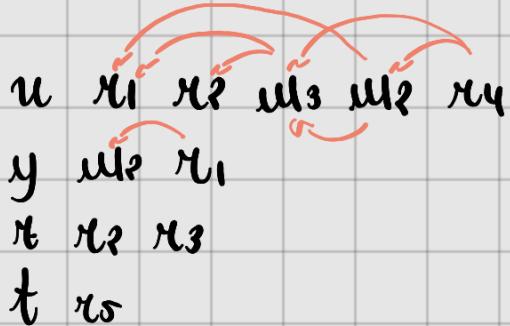
t₂ deve precedere t_u per mantenere la legge da r₁(t) w₂(t)
t₁ deve succedere t₂ per evitare la legge da. r₂(y) w₁(y)

Lo schedule non e' in USPI poiche' in ogni caso ($t_1 \rightarrow t_2$ o $t_2 \rightarrow t_1$) si generano delle legge da.

t₁ r₁(u) r₁(y) r₁(t) w₁(y) r₁(t)
t₂ r₂(y) w₂(t) w₂(t)
t₃ r₃(u)
t_u r_u(t)

Esercizio schedule

$r_1(u) \quad r_2(u) \quad m_3(u) \quad m_2(y) \quad m_1(u) \quad r_1(y) \quad r_2(t) \quad r_3(t) \quad r_4(u) \quad r_5(t)$



Lo schedule non e' in CSPI poiche' il suo grafo dei conflitti presenta cicli.

Verifico se lo schedule e' in USPI.

legge da: $r_1(u)$ legge da $m_2(u)$, $r_1(y)$ legge da $m_2(y)$
scritture finali: $m_2(u)$, $m_2(y)$

Considerazioni

t_4 deve succedere t_2 per mantenere $r_4(u)$ legge da $m_2(u)$

t_1 deve succedere t_2 per mantenere $r_1(y)$ legge da $m_2(y)$

t_2 deve succedere t_3 per mantenere le scritture finali

Non esistono schedule seriali ma equivalenti allo schedule assegnato, dunque lo schedule non e' in USPI.

