Politecnico di Milano

Facoltà di Ingegneria dell'Informazione

SISTEMI INFORMATICI – Esercitazioni

SCHEDULING e STATECHARTS

Indice

Parte I

Scheduling

Capitolo 1

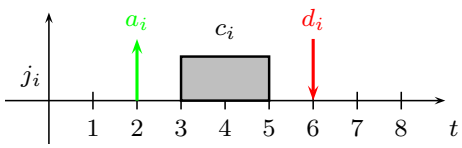
Processi aperiodici

1.1 Modello

1.1.1 Simboli

- $\{j_1, \ldots, j_n\}$ processi aperiodici
- a_i istante di arrivo di un processo
- f_i istante di fine di un processo
- d_i deadline di un processo
- c_i tempo di computazione di un processo
- $L_i = f_i - d_i$ lateness di un processo; se
  - $L_i \leq 0$ la deadline del processo i è rispettata
  - $L_i > 0$ la deadline del processo i non è rispettata
- $L_{max} = \max_i L_i$ lateness massima dei processi; se
  - $L_{max} \leq 0$ i processi sono schedulabili
  - $L_{max} > 0$ i processi non sono schedulabili

1.1.2 Grafico dei processi



1.2 EDF (Preemption, no vincoli di precedenza)

Early deadline first. Sceglie il processo pronto con deadline più piccola.

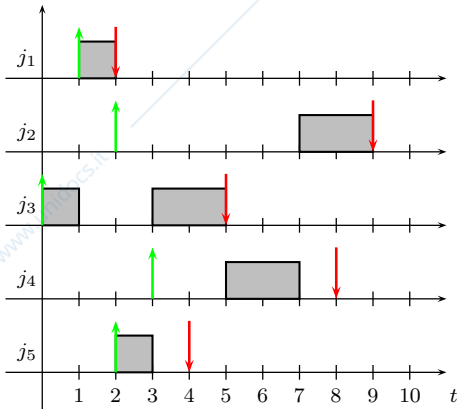
Garanzia di ottimalità: restituisce la soluzione con il minimo L_{max} tra tutte le possibili schedulazioni (esistenti). □

Esercizio 1.1

Schematizzare la schedulazione dei processi, dare la sequenza di completamento, dare L_i e L_{max}.

	j_1	j_2	j_3	j_4	j_5
a_i	1	2	0	3	2
c_i	1	2	3	2	1
d_i	2	9	5	8	4

	j_1	j_5	j_3	j_4	j_2
L_i	0	-1	0	-1	0



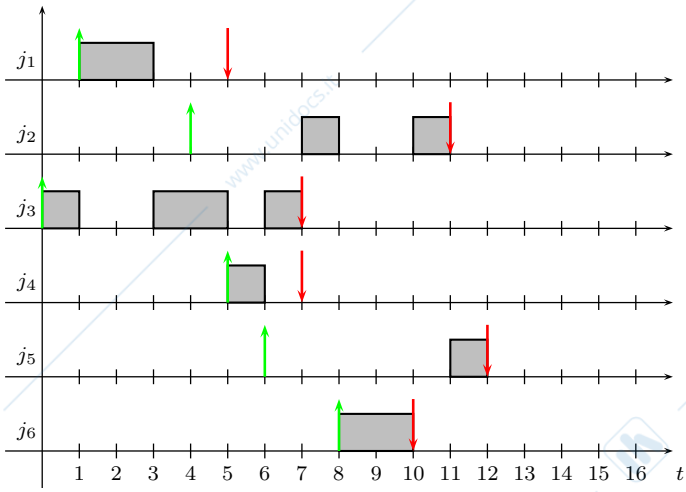
$L_{max} = 0$, quindi i processi sono schedulabili.

Esercizio 1.2

Schematizzare la schedulazione dei processi, dare la sequenza di completamento, dare L_i e L_{max}.

	j_1	j_2	j_3	j_4	j_5	j_6
a_i	1	4	0	5	6	8
c_i	2	2	4	1	1	2
d_i	5	11	7	7	12	10

	j_1	j_4	j_3	j_6	j_2	j_5
L_i	-2	-1	0	0	0	0



$L_{max} = 0$, quindi i processi sono schedulabili.

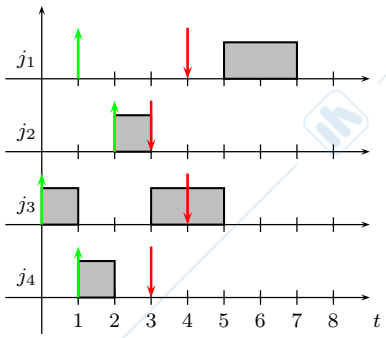
Nota. All'istante 5 si potrebbero schedulare sia j_3 che j_4; la scelta è del tutto arbitraria. □

Esercizio 1.3

Dire se il seguente problema è schedulabile.

	j_1	j_2	j_3	j_4
a_i	1	2	0	1
c_i	2	1	3	1
d_i	4	3	4	3

	j_4	j_2	j_3	j_1
L_i	-1	0	1	3



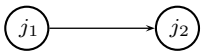
$L_{max} > 0$, quindi i processi non sono schedulabili.

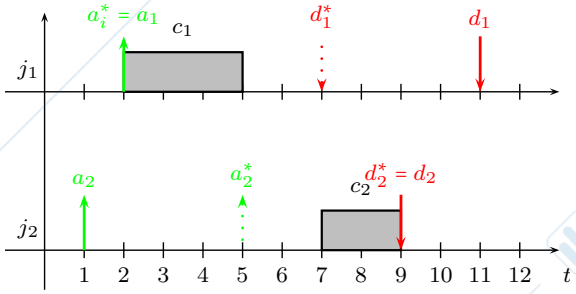
Nota. All'istante 3 si potrebbero schedulare sia j_1 che j_3; indipendentemente dalla scelta effettuata sarebbe risultato $L_{max} = 3$ perché EDF restituisce sempre il valore ottimo. □

1.3 EDF* (Preemption, vincoli di precedenza)

Due fasi:

1. **Costruzione del *problema ausiliario***
   - $a_i^* = \max_{j_k \to j_i}\{a_i, a_k^* + c_k\}$ – un processo *non* deve essere attivato prima che il suo predecessore *non* possa aver finito
   - $d_i^* = \min_{j_i \to j_k}\{d_i, d_k^* - c_k\}$ – devo poter terminare il predecessore e avere tempo a sufficienza per terminare il successore
2. **EDF**



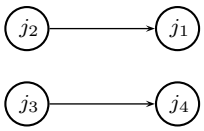


Nota. Il predecessore non può fare preemption del successore perché ogni volta che si dovrà scegliere tra j_1 e j_2 sarà scelto j_1 perché la sua deadline è minore. □

Esercizio 1.4

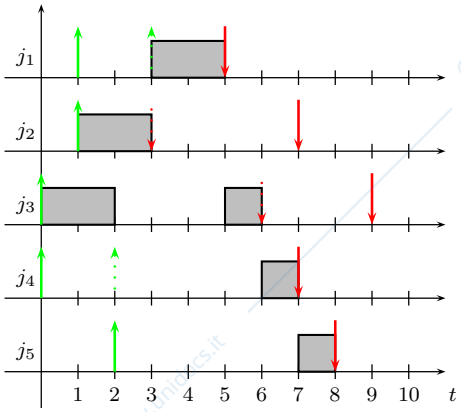
Schedulare i seguenti processi tenendo conto delle precedenze.

	j_1	j_2	j_3	j_4	j_5
a_i	1	1	0	0	2
c_i	2	2	2	1	1
d_i	5	7	9	7	8



$$a_1^* = \max\{1, 1+2\} = 3 \qquad d_1^* = d_1$$
$$a_2^* = a_2 \qquad d_2^* = \min\{7, 5-2\} = 3$$
$$a_3^* = a_3 \qquad d_3^* = \min\{9, 7-1\} = 6$$
$$a_4^* = \max\{0, 0+2\} = 2 \qquad d_4^* = d_4$$
$$a_5^* = a_5 \qquad d_5^* = d_5$$

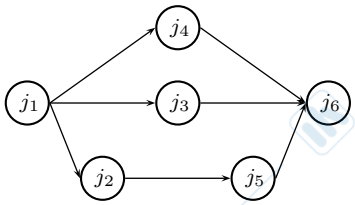
	j_1	j_2	j_3	j_4	j_5
a_i^*	3	1	0	2	2
c_i	2	2	2	1	1
d_i^*	5	3	6	7	8



Esercizio 1.5

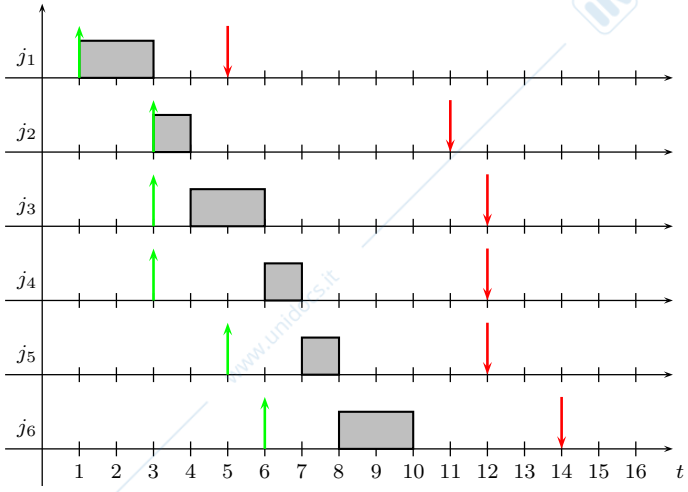
Schedulare i seguenti processi tenendo conto delle precedenze; calcolare L_{max}.

	j_1	j_2	j_3	j_4	j_5	j_6
a_i	1	0	1	3	5	5
c_i	2	1	2	1	1	2
d_i	5	11	12	15	13	14



$$a_1^* = a_1 \qquad d_1^* = \min\{5, 11-1, 12-2, 12-1\} = 5$$
$$a_2^* = \max\{0, 1+2\} = 3 \qquad d_2^* = \min\{11, 12-1\} = 11$$
$$a_3^* = \max\{1, 1+2\} = 3 \qquad d_3^* = \min\{12, 14-2\} = 12$$
$$a_4^* = \max\{3, 1+2\} = 3 \qquad d_4^* = \min\{15, 14-2\} = 12$$
$$a_5^* = \max\{5, 3+1\} = 5 \qquad d_5^* = \min\{13, 14-2\} = 12$$
$$a_6^* = \max\{5, 3+2, 5+1, 3+1\} = 6 \qquad d_6^* = d_6$$

	j_1	j_2	j_3	j_4	j_5	j_6
a_i^*	1	3	3	3	5	6
c_i	2	1	2	1	1	2
d_i^*	5	11	12	12	12	14

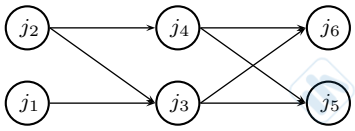


$L_{max} = -2$.

Esercizio 1.6

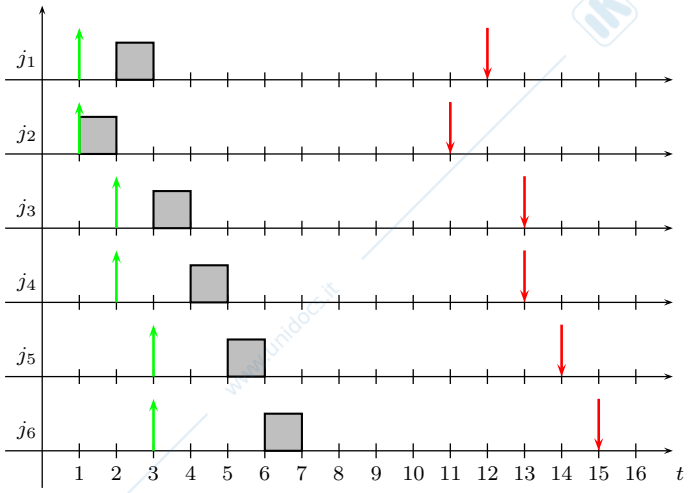
Schedulare i seguenti processi tenendo conto delle precedenze; calcolare L_{max}.

	j_1	j_2	j_3	j_4	j_5	j_6
a_i	1	1	2	2	2	3
c_i	1	1	1	1	1	1
d_i	14	11	13	24	14	15



$$a_1^* = a_1 \qquad d_1^* = \min\{14, 13-1\} = 12$$
$$a_2^* = a_2 \qquad d_2^* = \min\{11, 13-1, 13-1\} = 11$$
$$a_3^* = \max\{2, 1+1, 1+1\} = 2 \qquad d_3^* = \min\{13, 14-1, 15-1\} = 13$$
$$a_4^* = \max\{2, 1+1\} = 2 \qquad d_4^* = \min\{24, 14-1, 15-1\} = 13$$
$$a_5^* = \max\{2, 2+1, 2+1\} = 3 \qquad d_5^* = d_5$$
$$a_6^* = \max\{3, 2+1, 2+1\} = 3 \qquad d_6^* = d_6$$

	j_1	j_2	j_3	j_4	j_5	j_6
$a_i^\star$	1	1	2	2	3	3
c_i	1	1	1	1	1	1
$d_i^\star$	12	11	13	13	14	15

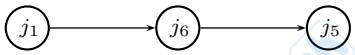


$L_{max} = -8$.

Esercizio 1.7

Valutare EDF considerando la conoscenza del vincolo solo all'arrivo del processo.*

	j_1	j_2	j_3	j_4	j_5	j_6
a_i	2	1	0	3	6	4
c_i	2	2	3	1	2	1
d_i	12	3	7	8	11	12



$t = 0$

$$a_1^* = a_1 \qquad d_1^* = d_1$$
$$a_2^* = a_2 \qquad d_2^* = d_2$$
$$a_3^* = a_3 \qquad d_3^* = d_3$$
$$a_4^* = a_4 \qquad d_4^* = d_4$$
$$a_5^* = a_5 \qquad d_5^* = d_5$$
$$a_6^* = a_6 \qquad d_6^* = d_6$$

$t = 4$

$$a_1^* = a_1 \qquad d_1^* = \min\{12, 12-1\} = 11$$
$$a_2^* = a_2 \qquad d_2^* = d_2$$
$$a_3^* = a_3 \qquad d_3^* = d_3$$
$$a_4^* = a_4 \qquad d_4^* = d_4$$
$$a_5^* = a_5 \qquad d_5^* = d_5$$
$$a_6^* = \max\{4, 2+2\} = 4 \qquad d_6^* = d_6$$

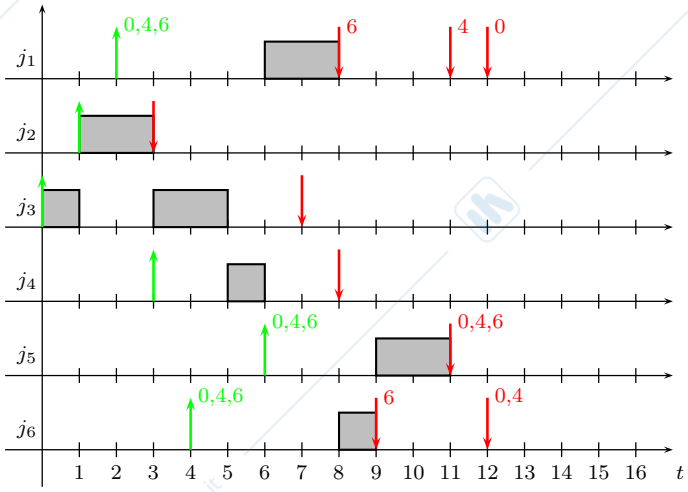
$t = 6$

$$a_1^* = a_1 \qquad d_1^* = \min\{12, 9-1\} = 8$$
$$a_2^* = a_2 \qquad d_2^* = d_2$$
$$a_3^* = a_3 \qquad d_3^* = d_3$$
$$a_4^* = a_4 \qquad d_4^* = d_4$$
$$a_5^* = \max\{6, 4+1\} = 6 \qquad d_5^* = d_5$$
$$a_6^* = \max\{4, 2+2\} = 4 \qquad d_6^* = \min\{12, 11-2\} = 9$$

$t = 0$	j_1	j_2	j_3	j_4	j_5	j_6
a_i^*	2	1	0	3	6	4
c_i	2	2	3	1	2	1
d_i^*	12	3	7	8	11	12

$t = 4$	j_1	j_2	j_3	j_4	j_5	j_6
a_i^*	2	1	0	3	6	4
c_i	2	2	3	1	2	1
d_i^*	11	3	7	8	11	12

$t = 6$	j_1	j_2	j_3	j_4	j_5	j_6
a_i^*	2	1	0	3	6	4
c_i	2	2	3	1	2	1
d_i^*	8	3	7	8	11	9



1.4 SPRING (No preemption)

Senza preemption con EDF *non* si ha garanzia di trovare la soluzione anche se essa esiste.

	j_1	j_2
a_i	1	2
c_i	2	2
d_i	6	4

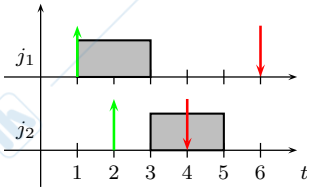


Figura 1.1: EDF

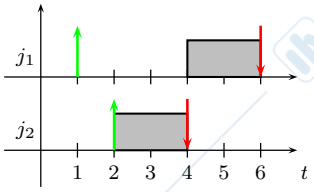


Figura 1.2: Soluzione esistente

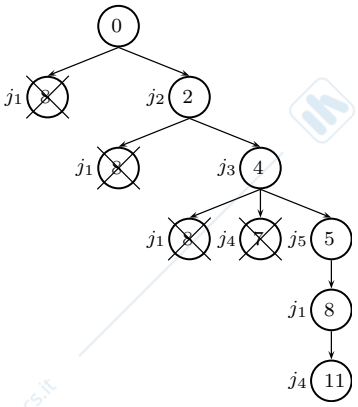
Ricercare quindi tra tutte le possibili schedulazioni, costruzione di un albero di enumerazione, problema $\mathcal{NP}$–completo, non applicabile nella realtà.

Spring. L'istante temporale del nodo di arrivo deve essere minore di $d_i - c_i$, per tutti i processi non ancora schedulati. Utilizzo di euristiche (di I e II grado) e limiti di backtracking (numero massimo di nodi) per migliorare il tempo di restituzione di una soluzione. □

Esercizio 1.8

Schedulare i seguenti processi.

	j_1	j_2	j_3	j_4	j_5
a_i	5	0	3	0	1
c_i	3	2	1	3	1
d_i	8	4	5	11	7

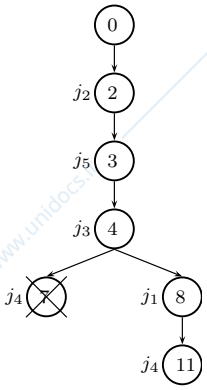


Esercizio 1.9

Schedulare i seguenti processi utilizzando le seguenti euristiche:

- $h_i = \max\{a_i - t, 0\}$
- $h'_i = d_i$.

	j_1	j_2	j_3	j_4	j_5
a_i	5	0	3	0	1
c_i	3	2	1	3	1
d_i	8	4	5	11	7



$t = 0$

$h_1 = 5,\ h'_1 = 8$

$h_2 = 0,\ h'_2 = 4$

$h_3 = 3,\ h'_3 = 5$

$h_4 = 0,\ h'_4 = 11$

$h_5 = 1,\ h'_5 = 7$

$t = 2$

$h_1 = 3,\ h'_1 = 8$

...

$h_3 = 1,\ h'_3 = 5$

$h_4 = 0,\ h'_4 = 11$

$h_5 = 0,\ h'_5 = 7$

$t = 3$

$h_1 = 2,\ h'_1 = 8$

...

$h_3 = 0,\ h'_3 = 5$

$h_4 = 0,\ h'_4 = 11$

...

$t = 4$	$t = 8$
$h_1 = 1,\ h_1' = 8$	...
...	...
...	...
$h_4 = 0,\ h_4' = 11$	$h_4 = 0,\ h_4' = 11$
...	...

Nota. La ricerca nel presente esercizio è computazionalmente più efficiente di quella dell'Esercizio 1.8. □

Capitolo 2

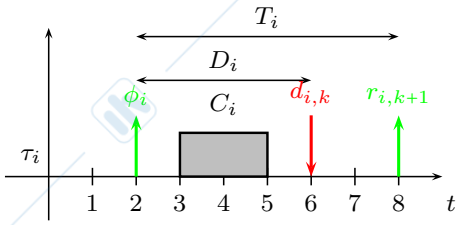
Processi periodici

2.1 Modello

2.1.1 Simboli

- $\{\tau_1, \ldots, \tau_n\}$ processi periodici
- $\tau_{i,k}$ istanza k-esima del processo i
- ϕ_i fase di un processo (primo tempo di attivazione)
- T_i periodo del processo
- $r_{i,k} = \phi_i + (k-1) \cdot T_i$ k-esima attivazione
- $D_i = d_{i,k} - a_{i,k}$ deadline relativa di un processo
- C_i tempo di computazione di un processo
- $U = \sum_i U_i = \sum_i \frac{C_i}{T_i}$ fattore di utilizzazione

2.1.2 Grafico dei processi



2.1.3 Risultati notevoli di schedulabilità

Se $U > 1$ il sistema non è schedulabile indipendentemente dall'algoritmo utilizzato.

Rate monotonic

- Lyu–Layland (condizione sufficiente)

$$\overline{U}_{th} = n \cdot \left(\sqrt[n]{2} - 1 \right) \text{ dove } n \text{ è il numero dei processi}$$

  - $U \leq \overline{U}_{th}$ il sistema è schedulabile
  - $\overline{U}_{th} < U \leq 1$ non ci si può esprimere sulla schedulabilità del sistema

- Bini–Buttazzo (condizione sufficiente)

  - $\prod_i (U_i + 1) \leq 2$ il sistema è schedulabile
  - $\prod_i (U_i + 1) > 2$ non ci si può esprimere sulla schedulabilità del sistema

- Response time analysis (condizione necessaria e sufficiente – computazionalmente pesante)

$$R_i^0 = C_i \tag{2.1}$$

$$I_i^k = \sum_{j|T_j<T_i} C_j \cdot \left\lceil \frac{R_i^k}{T_j} \right\rceil \tag{2.2}$$

$$R_i^k = C_i + I_i^{(k-1)} \tag{2.3}$$

  - $R_i \leq D_i$, $\forall i$ il sistema è schedulabile

Nota. La verificata schedulazione con Lyu–Layland implica la verificata schedulazione con Bini–Buttazzo. □

Deadline monotonic

- Response time analysis (condizione sufficiente – computazionalmente pesante)

Early deadline first

- $U \leq 1$ il sistema è schedulabile

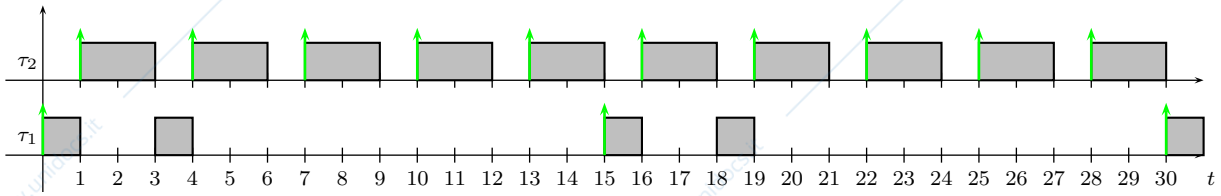
2.2 RM (Statico, $D_i = T_i$)

Rate monotonic. Le priorità sono fisssate all'istante $t = 0$ e non possono cambiare durante la schedulazione dei processi: più piccolo è il periodo, più alta la priorità. □

Esercizio 2.1

Schedulare i seguenti processi.

	τ_1	τ_2
ϕ_i	0	1
C_i	2	2
T_i	15	3



Esercizio 2.2

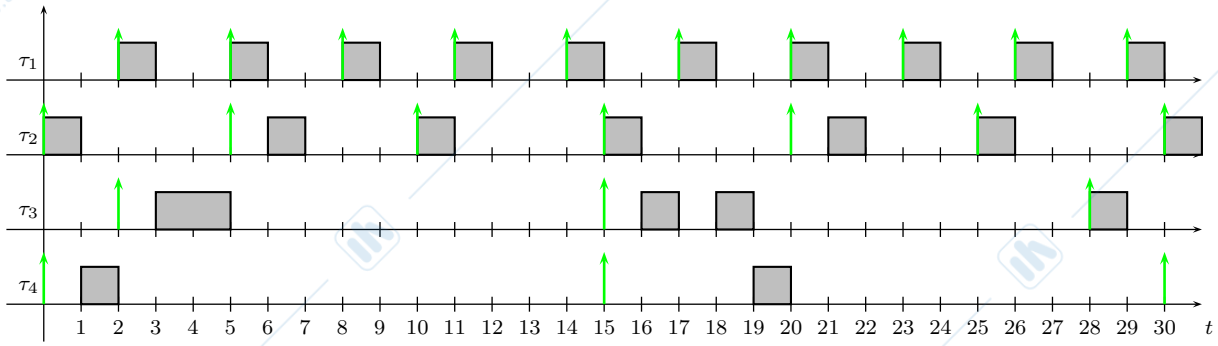
Dire se il seguente sistema è schedulabile con RM.

	τ_1	τ_2	τ_3	τ_4
ϕ_i	2	0	2	0
C_i	1	1	2	1
T_i	3	5	13	15

Lyu–Layland test.

$$U = \frac{1}{3} + \frac{1}{5} + \frac{2}{13} + \frac{1}{15} \cong 0,754$$
$$\overline{U}_{th} = 4 \cdot \left(\sqrt[4]{2} - 1\right) \cong 0,7568$$
$$U < \overline{U}_{th} \Rightarrow \text{ schedulabile con RM}$$

Nota. Essendo $U < 1$ il sistema risulta schedulabile anche con EDF. □



Esercizio 2.3

Dire se il seguente sistema è schedulabile con RM.

	τ_1	τ_2	τ_3	τ_4
ϕ_i	0	1	2	3
C_i	2	2	1	2
T_i	6	10	4	11

Lyu–Layland test.

$$U = \frac{2}{6} + \frac{2}{10} + \frac{1}{4} + \frac{2}{11} \cong 0,965$$
$$\overline{U}_{th} = 4 \cdot \left(\sqrt[4]{2} - 1\right) \cong 0,757$$
$$U > \overline{U}_{th} \Rightarrow \text{ non ho garanzia di schedulabilità}$$

Bini–Buttazzo test.

$$\prod_i (U_i + 1) = \left(\frac{2}{6} + 1\right) \cdot \left(\frac{2}{10} + 1\right) \cdot \left(\frac{1}{4} + 1\right) \cdot \left(\frac{2}{11} + 1\right) \cong 2,36$$
$$\prod_i (U_i + 1) > 2 \Rightarrow \text{ non ho garanzia di schedulabilità}$$

RTA test.

Ordino in modo crescente i processi per T_i: τ_3, τ_1, τ_2, τ_4.

$$R_3^0 = C_3 = 1$$
$$I_3^0 = 0$$
$$R_3^1 = C_3 + I_3^0 = 1 = R_3^0$$
$$R_3 < D_3 = T_3 \Rightarrow \tau_3 \text{ è schedulabile}$$

$$R_1^0 = C_1 = 2$$
$$I_1^0 = C_3 \cdot \left\lceil \frac{R_1^0}{T_3} \right\rceil = 1 \cdot \left\lceil \frac{2}{4} \right\rceil = 1$$
$$R_1^1 = C_1 + I_1^0 = 2 + 1 = 3 \neq R_1^0$$
$$I_1^1 = C_3 \cdot \left\lceil \frac{R_1^1}{T_3} \right\rceil = 1 \cdot \left\lceil \frac{3}{4} \right\rceil = 1$$
$$R_1^2 = C_1 + I_1^1 = 2 + 1 = 3 = R_1^1 < D_1 = T_1 \Rightarrow \ \tau_1 \text{ è schedulabile}$$

$$R_2^0 = C_2 = 2$$
$$I_2^0 = C_1 \cdot \left\lceil \frac{R_2^0}{T_1} \right\rceil + C_3 \cdot \left\lceil \frac{R_2^0}{T_3} \right\rceil = 2 \cdot \left\lceil \frac{2}{6} \right\rceil + 1 \cdot \left\lceil \frac{2}{4} \right\rceil = 3$$
$$R_2^1 = C_2 + I_2^0 = 2 + 3 = 5 \neq R_2^0$$
$$I_2^1 = C_1 \cdot \left\lceil \frac{R_2^1}{T_1} \right\rceil + C_3 \cdot \left\lceil \frac{R_2^1}{T_3} \right\rceil = 2 \cdot \left\lceil \frac{5}{6} \right\rceil + 1 \cdot \left\lceil \frac{5}{4} \right\rceil = 4$$
$$R_2^2 = C_2 + I_2^1 = 2 + 4 = 6 \neq R_2^1$$
$$I_2^2 = C_1 \cdot \left\lceil \frac{R_2^2}{T_1} \right\rceil + C_3 \cdot \left\lceil \frac{R_2^2}{T_3} \right\rceil = 2 \cdot \left\lceil \frac{6}{6} \right\rceil + 1 \cdot \left\lceil \frac{6}{4} \right\rceil = 4$$
$$R_2^3 = C_2 + I_2^2 = 2 + 4 = 6 = R_2^2 < D_2 = T_2 \Rightarrow \ \tau_2 \text{ è schedulabile}$$

$$R_4^0 = C_4 = 2$$
$$I_4^0 = C_2 \cdot \left\lceil \frac{R_4^0}{T_2} \right\rceil + C_1 \cdot \left\lceil \frac{R_4^0}{T_1} \right\rceil + C_3 \cdot \left\lceil \frac{R_4^0}{T_3} \right\rceil = 2 \cdot \left\lceil \frac{2}{10} \right\rceil + 2 \cdot \left\lceil \frac{2}{6} \right\rceil + 1 \cdot \left\lceil \frac{2}{4} \right\rceil = 5$$
$$R_4^1 = C_4 + I_4^0 = 2 + 5 = 7 \neq R_4^0$$
$$I_4^1 = C_2 \cdot \left\lceil \frac{R_4^1}{T_2} \right\rceil + C_1 \cdot \left\lceil \frac{R_4^1}{T_1} \right\rceil + C_3 \cdot \left\lceil \frac{R_4^1}{T_3} \right\rceil = 2 \cdot \left\lceil \frac{7}{10} \right\rceil + 2 \cdot \left\lceil \frac{7}{6} \right\rceil + 1 \cdot \left\lceil \frac{7}{4} \right\rceil = 8$$
$$R_4^2 = C_4 + I_4^1 = 2 + 8 = 10 \neq R_4^1$$
$$I_4^2 = C_2 \cdot \left\lceil \frac{R_4^2}{T_2} \right\rceil + C_1 \cdot \left\lceil \frac{R_4^2}{T_1} \right\rceil + C_3 \cdot \left\lceil \frac{R_4^2}{T_3} \right\rceil = 2 \cdot \left\lceil \frac{10}{10} \right\rceil + 2 \cdot \left\lceil \frac{10}{6} \right\rceil + 1 \cdot \left\lceil \frac{10}{4} \right\rceil = 9$$
$$R_4^3 = C_4 + I_4^2 = 2 + 9 = 11 \neq R_4^2$$
$$I_4^3 = C_2 \cdot \left\lceil \frac{R_4^3}{T_2} \right\rceil + C_1 \cdot \left\lceil \frac{R_4^3}{T_1} \right\rceil + C_3 \cdot \left\lceil \frac{R_4^3}{T_3} \right\rceil = 2 \cdot \left\lceil \frac{11}{10} \right\rceil + 2 \cdot \left\lceil \frac{11}{6} \right\rceil + 1 \cdot \left\lceil \frac{11}{4} \right\rceil = 11$$
$$R_4^4 = C_4 + I_4^3 = 2 + 11 = 13 \neq R_4^4$$
$$R_4 > D_4 = T_4 \Rightarrow \ \tau_4 \text{ non è schedulabile}$$

Esercizio 2.4

Trovare valore di C_4 tale che:

1. *sia massimo ma schedulabile con Lyu–Layland,*
2. *sia massimo ma schedulabile con Bini–Buttazzo,*
3. *sia massimo ma schedulabile in sezione critica.*

	τ_1	τ_2	τ_3	τ_4
ϕ_i	0	0	0	0
C_i	2	3	1	?
T_i	8	12	5	20

1. Lyu–Layland test.

$$\overline{U}_{th} = 4 \cdot \left(\sqrt[4]{2} - 1\right) \cong 0,757$$

$$Hp:\ C_4 = 1 \qquad U = \frac{2}{8} + \frac{3}{12} + \frac{1}{5} + \frac{1}{20} = 0,75$$

$$U < \overline{U}_{th} \Rightarrow\ \tau_4 \text{ è schedulabile}$$

$$Hp:\ C_4 = 2 \qquad U = \frac{2}{8} + \frac{3}{12} + \frac{1}{5} + \frac{2}{20} = 0,8$$

$$U > \overline{U}_{th} \Rightarrow\ \text{non ho garanzia di schedulabilità}$$

$C_4 = 1$.

2. Bini–Buttazzo test.

$$Hp:\ C_4 = 1 \qquad \prod_i (U_i + 1) = \left(\frac{1}{4} + 1\right) \cdot \left(\frac{1}{4} + 1\right) \cdot \left(\frac{1}{5} + 1\right) \cdot \left(\frac{1}{20} + 1\right) \cong 1,968$$

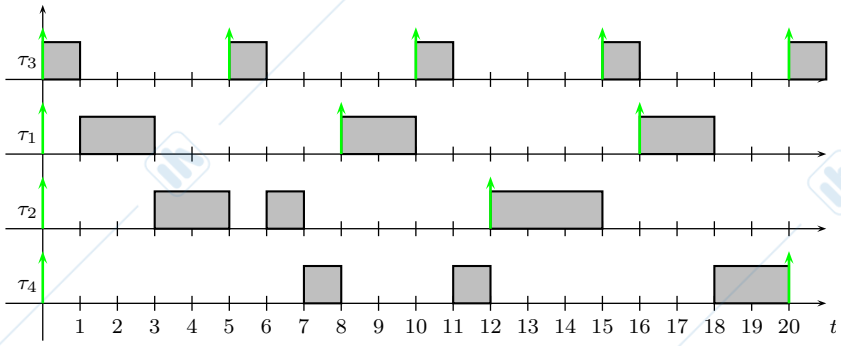
$$\prod_i (U_i + 1) > 2 \Rightarrow\ \tau_4 \text{ è schedulabile}$$

$$Hp:\ C_4 = 2 \qquad \prod_i (U_i + 1) = \left(\frac{1}{4} + 1\right) \cdot \left(\frac{1}{4} + 1\right) \cdot \left(\frac{1}{5} + 1\right) \cdot \left(\frac{2}{20} + 1\right) \cong 2,0625$$

$$\prod_i (U_i + 1) > 2 \Rightarrow\ \text{non ho garanzia di schedulabilità}$$

$C_4 = 1$.

3. Sezione critica test. $C_4 = 4$.



Nota. Nel caso $C_4 = 4$ cosa ci direbbero i test Lyu–Layland e Bini–Buttazzo?

Lyu–Layland

$$U = \frac{2}{8} + \frac{3}{12} + \frac{1}{5} + \frac{4}{20} = 0,9$$
$$U > \overline{U}_{th} \Rightarrow \text{ non ho garanzia di schedulabilità}$$

Bini–Buttazzo

$$\prod_i (U_i + 1) = \left(\frac{1}{4} + 1\right) \cdot \left(\frac{1}{4} + 1\right) \cdot \left(\frac{1}{5} + 1\right) \cdot \left(\frac{4}{20} + 1\right) = 3$$
$$\prod_i (U_i + 1) > 2 \Rightarrow \text{ non ho garanzia di schedulabilità}$$

□

Nota. Con EDF sarebbe schedulabile? Essendo $U < 1$ la risposta è affermativa. □

2.3 DM (Statico, $D_i < T_i$)

Deadline monotonic. Le priorità sono fisssate all'istante $t = 0$ e non possono cambiare durante la schedulazione dei processi: più piccola è la deadline, più alta la priorità. □

Esercizio 2.5

Dire se il seguente sistema è schedulabile.

	τ_1	τ_2
ϕ_i	0	3
C_i	2	2
T_i	6	6
D_i	3	3

$$R_2^0 = C_2 = 2$$
$$I_2^0 = 0$$
$$R_2^1 = C_2 + I_2^0 = 1 = R_2^0$$
$$R_2 < D_2 \Rightarrow \ \tau_2 \text{ è schedulabile}$$

$$R_1^0 = C_1 = 2$$
$$I_1^0 = C_2 \cdot \left\lceil \frac{R_1^0}{T_2} \right\rceil = 2 \cdot \left\lceil \frac{2}{6} \right\rceil = 2$$
$$R_1^1 = C_1 + I_1^0 = 2 + 2 = 4 > D_1 \Rightarrow \ \tau_1 \text{ non è schedulabile}$$

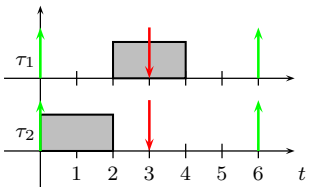


Figura 2.1: Sezione critica

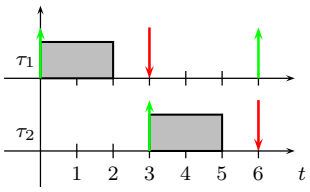


Figura 2.2: DM

Nota. L'esercizio è una dimostrazione che RTA è una condizione sufficiente e non necessaria per la schedulabilità di un sistema con DM. □

2.4 EDF (Dinamico, $D_i = T_i$)

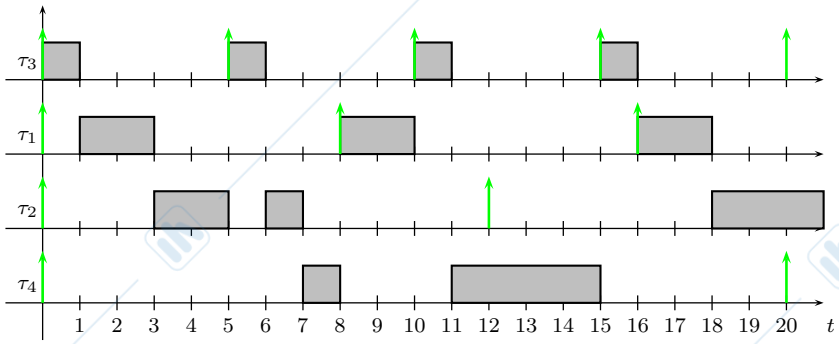
Early deadline first. Le priorità cambiano dinamicamente al variare degli istanti di tempo: deadline più stringente, priorità più alta. □

Esercizio 2.6

Schedulare i seguenti processi.

	τ_1	τ_2	τ_3	τ_4
ϕ_i	0	0	0	0
C_i	2	3	1	5
T_i	8	12	5	20

$$U = \frac{2}{8} + \frac{3}{12} + \frac{1}{5} + \frac{5}{20} = 0,95$$
$$U < 1 \Rightarrow \text{ il sistema è schedulabile}$$

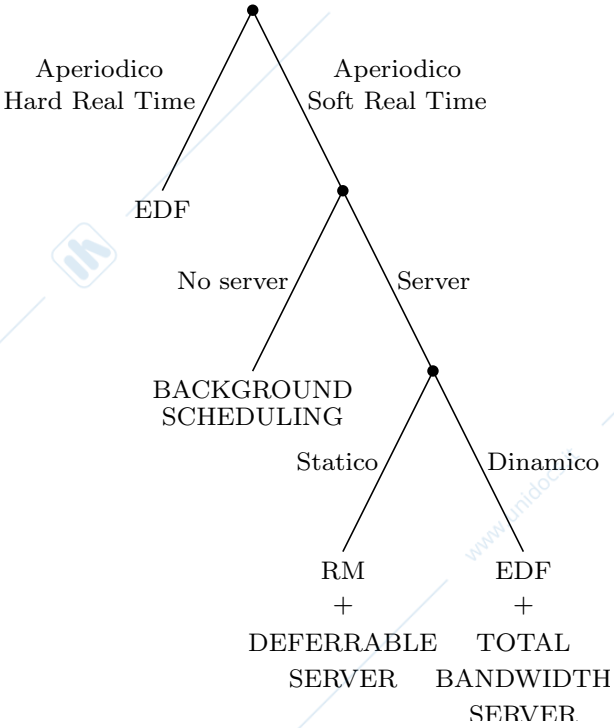


Nota. Nell'Esercizio 2.4 il sistema risultava schedulabile con algoritmo RM e al più $C_4 = 4$ (analisi RTA); con $C_4 = 5$ il sistema risulta schedulabile utilizzando EDF, ma non con RM. □

Capitolo 3

Processi misti

3.1 Scenario



3.2 Modello

3.2.1 Simboli

- $r_i = f_i - a_i$ tempo di risposta del processo
- $C_i(t)$: tempo di computazione rimanente al processo i all'istante t
- U_p fattore di utilizzazione dei processi periodici (definito in modo analogo ad U del Cap. 2)

- U_s fattore di utilizzazione del processo server τ_s

3.2.2 Risultati notevoli di schedulabilità

Se $U_p + U_s > 1$ il sistema non è schedulabile indipendentemente dall'algoritmo utilizzato.

Deferrable server

$$\overline{U}_{th} = n \cdot \left(\sqrt[n]{\frac{U_s + 2}{2 \cdot U_s + 1}} - 1 \right) \text{ dove } n \text{ è il numero dei processi}$$

- $U_p \leq \overline{U}_{th} \Leftrightarrow U_p \leq n \cdot \left(\sqrt[n]{\frac{U_s + 2}{2 \cdot U_s + 1}} - 1 \right)$ il sistema è schedulabile

3.3 Aperiodici Hard Real Time

Esercizio 3.1

Schedulare i seguenti processi.

	τ_1	τ_2	τ_3	τ_4
ϕ_i	1	0	2	0
C_i	2	2	1	1
T_i	4	8	16	32

	j_1	j_2	j_3
a_i	3	5	7
c_i	2	2	2
d_i	8	17	22

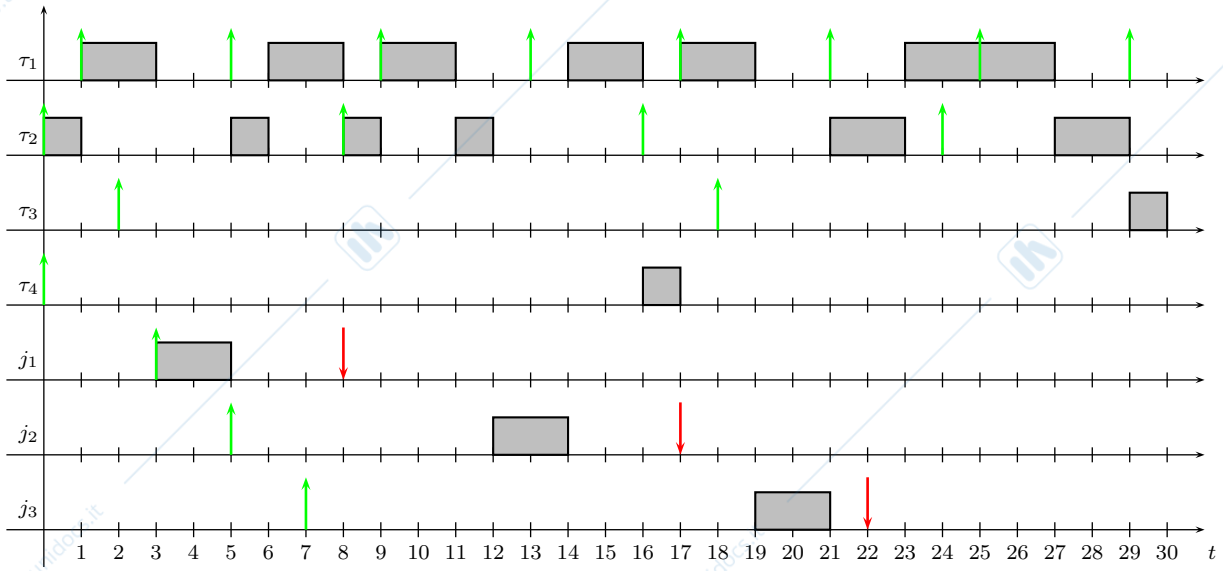
$$U_p = \frac{2}{4} + \frac{2}{8} + \frac{1}{16} + \frac{1}{32} = 0,84375$$

$U_p < 1 \Rightarrow$ l'insieme dei processi periodici è schedulabile

Nota. All'istante $t = 3$ è possibile mandare in esecuzione sia τ_2 che j_1 ed all'istante $t = 13$ sia τ_1 che j_2; in questi casi hanno precedenza i processi aperiodici. □

3.4 RM + DS

Deferrable server. Laddove non diversamente specificato manda in esecuzione i processi aperiodici con politica FIFO. □



Esercizio 3.2

Schedulare i seguenti processi.

	τ_1	τ_s
ϕ_i	0	1
C_i	2	2
T_i	15	3

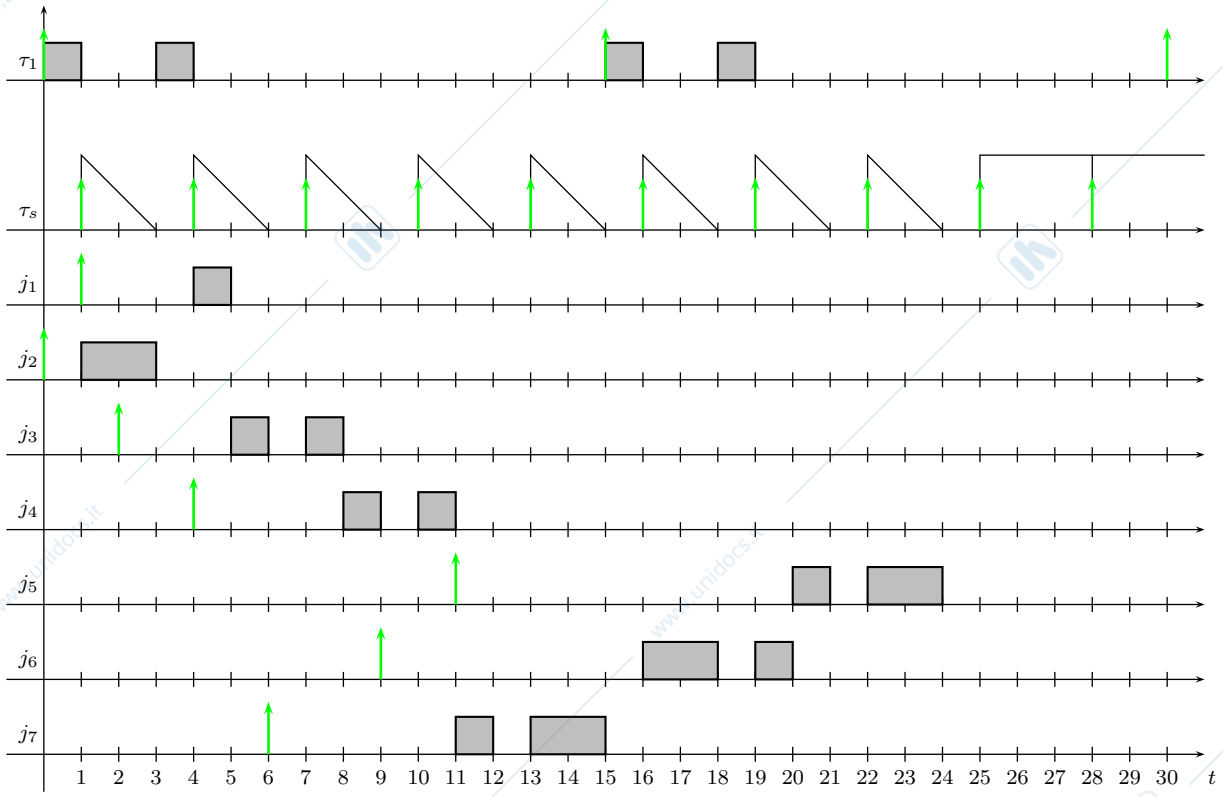
	j_1	j_2	j_3	j_4	j_5	j_6	j_7
a_i	1	0	2	4	11	9	6
c_i	1	2	2	2	3	3	3

$$U_p = \frac{2}{15} = 0,1\overline{3}$$

$$U_s = \frac{2}{3} = 0,\overline{6}$$

$$\overline{U}_{th} = 1 \cdot \left(\sqrt[1]{\frac{0,\overline{6}+2}{2 \cdot 0,\overline{6}+1}} - 1 \right) \cong 0,143$$

$U_p < \overline{U}_{th} \Rightarrow$ schedulabile con RM



Esercizio 3.3

Calcolare la capacità massima del DS affinché il sistema sia schedulabile. Mostrare la schedulazione.

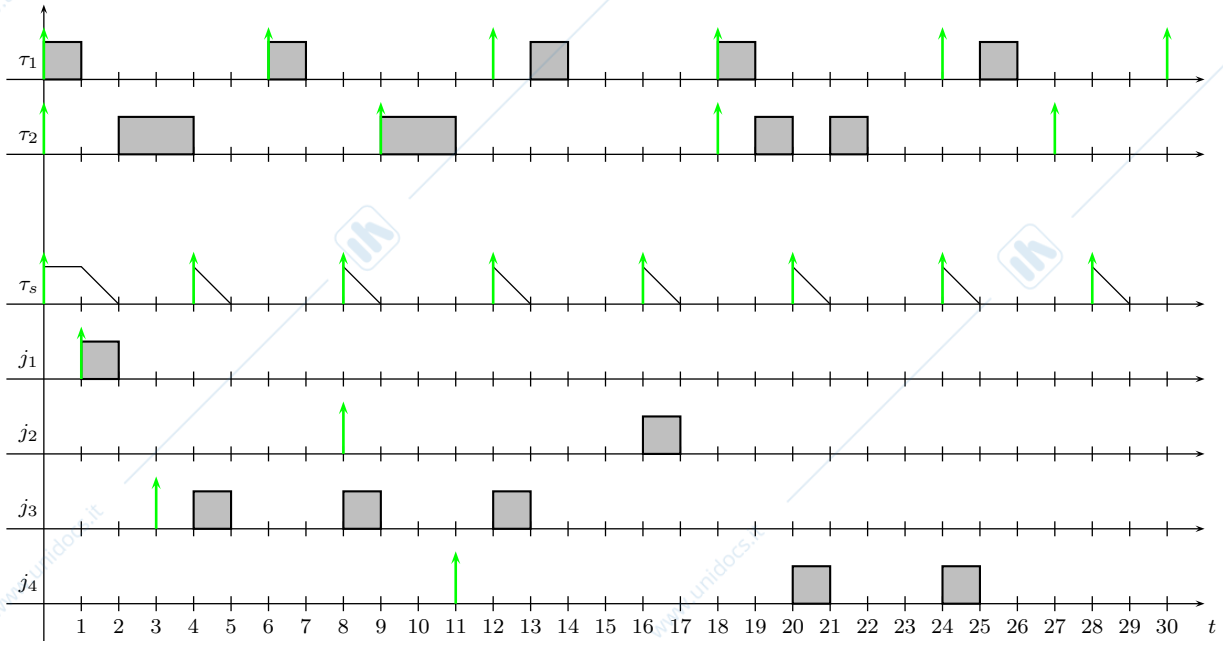
	τ_1	τ_2	τ_s
ϕ_i	0	0	0
C_i	1	2	?
T_i	6	9	4

	j_1	j_2	j_3	j_4
a_i	1	8	3	11
c_i	1	1	3	2

$$U_p = \frac{1}{6} + \frac{2}{9} \cong 0,39$$

$$0,39 \leq 2 \cdot \left(\sqrt[2]{\frac{U_s + 2}{2 \cdot U_s + 1}} - 1 \right) \Leftrightarrow U_s \leq 0,308$$

$$U_s = \frac{C_s}{4} \Leftrightarrow C_s = \lfloor 4 \cdot U_s \rfloor = \lfloor 4 \cdot 0,308 \rfloor = 1$$



3.5 EDF + TBS

Total bandwidth server. È necessario introdurre delle soft deadline per i processi aperiodici che consentano la schedulazione con EDF. I processi aperiodici sono ordinati in base all'istante di attivazione. □

$$d_k = \max\{a_k, d_{k-1}\} + \left\lceil \frac{c_k}{U_s} \right\rceil$$

$$d_0 = 0$$

Nota. Per la condizione necessaria e sufficiente di schedulabilità si ha che $U_s + U_p \leq 1$, conseguentemente al più $U_s = 1 - U_p$. □

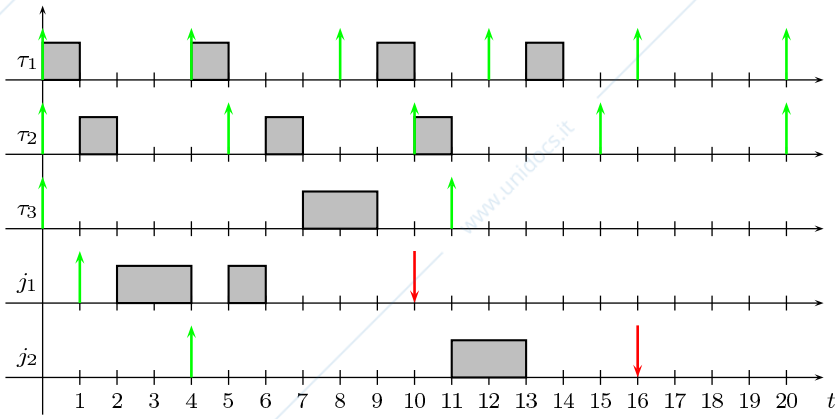
Esercizio 3.4

Schedulare i seguenti processi.

	τ_1	τ_2	τ_3
ϕ_i	0	0	0
C_i	1	1	2
T_i	4	5	11

	j_1	j_2
a_i	1	4
c_i	3	2

$$U_p = \frac{1}{4} + \frac{1}{5} + \frac{2}{11} \cong 0,63$$
$$U_s = 1 - 0,63 = 0,37$$
$$d_1 = \max\{1,0\} + \left\lceil \frac{3}{0,37} \right\rceil = 1 + 9 = 10$$
$$d_2 = \max\{4,10\} + \left\lceil \frac{2}{0,37} \right\rceil = 10 + 6 = 16$$



3.6 EDF + TBS*

È possibile migliorare le soft deadline per rendere la schedulazione computazionalmente più efficiente attraverso stime iterative degli istanti di fine dei processi.

$$d_k^0 = \max\{a_k, d_{k-1}^0\} + \left\lceil \frac{c_k}{U_s} \right\rceil$$
$$d_0^0 = 0$$
$$\tilde{f}_k^s = a_k + c_k + I_p(a_k, d_k^s)$$
$$d_k^{s+1} = \tilde{f}_k^s$$

dove

$$I_p(a_k, d_k^s) = I_a(a_k, d_k^s) + I_f(a_k, d_k^s)$$
$$I_a(a_k, d_k^s) = \sum_{i|\tau_i \text{ attivo e } d_i < d_k^s} C_i(a_k)$$
$$I_f(a_k, d_k^s) = \sum_i \max\left\{0, \left\lceil \frac{d_k^s - next\_r_i(a_k)}{T_i} \right\rceil - 1\right\} \cdot C_i$$
$$next\_r_i(a_k) = \left\lceil \frac{a_k + 1}{T_i} \right\rceil \cdot T_i$$

Significato.

- $I_p(a_k, d_k^s)$: interferenza (stimata) su j_k da parte di processi periodici in $[a_k, d_k^s)$
- $I_a(a_k, d_k^s)$: interferenza causata da istanze periodiche attivate prima di a_k con deadline minore di d_k^s
- $I_f(a_k, d_k^s)$: interferenza causata da istanze periodiche che si attiveranno dopo a_k con deadline minore di d_k^s
- $C_i(a_k)$: tempo di computazione rimanente al processo i all'istante $t = a_k$
- $\left\lceil \frac{d_k^s - next\_r_i(a_k)}{T_i} \right\rceil$: numero di volte che il processo i si riattiva dopo a_k (è minore di 0 se si attiva dopo d_k^s)
- $\left\lceil \frac{a_k + 1}{T_i} \right\rceil$: istante della prima riattivazione del processo i dopo a_k (sul libro questa formula non è corretta, manca il termine "+1" al numeratore)

□

Esercizio 3.5

Schedulare i seguenti processi.

	τ_1	τ_2	τ_3	τ_4
ϕ_i	0	0	0	0
C_i	1	1	2	2
T_i	4	5	11	16

	j_1
a_i	0
c_i	3

$$U_p = \frac{1}{4} + \frac{1}{5} + \frac{2}{11} + \frac{2}{16} \cong 0,757$$
$$U_s = 1 - 0,757 = 0,243$$
$$d_1^0 = \max\{0,0\} + \left\lceil \frac{3}{0,243} \right\rceil = 0 + 13 = 13$$

$$next\_r_1(0) = \left\lceil \frac{0+1}{4} \right\rceil \cdot 4 = 4 \qquad next\_r_2(0) = \left\lceil \frac{0+1}{5} \right\rceil \cdot 5 = 5$$
$$next\_r_3(0) = \left\lceil \frac{0+1}{11} \right\rceil \cdot 11 = 11 \qquad next\_r_4(0) = \left\lceil \frac{0+1}{16} \right\rceil \cdot 16 = 16$$

$$\tilde{f}_1^0 = a_1 + c_1 + I_a(0,13) + I_f(0,13) = 0 + 3 + 4 + 3 = 10 = d_1^1 \neq d_1^0$$
$$I_a(0,13) = 1 + 1 + 2 = 4$$
$$I_f(0,13) = \max\left\{0, \left\lceil \frac{13-4}{4} \right\rceil - 1\right\} \cdot 1 + \max\left\{0, \left\lceil \frac{13-5}{5} \right\rceil - 1\right\} \cdot 1 +$$
$$+ \max\left\{0, \left\lceil \frac{13-11}{11} \right\rceil - 1\right\} \cdot 2 + \max\left\{0, \left\lceil \frac{13-16}{16} \right\rceil - 1\right\} \cdot 2 = 2 + 1 + 0 + 0 = 3$$

$$\tilde{f}_1^1 = a_1 + c_1 + I_a(0,10) + I_f(0,10) = 0 + 3 + 2 + 1 = 6 = d_1^2 \neq d_1^1$$
$$I_a(0,10) = 1 + 1 = 2$$
$$I_f(0,10) = \max\left\{0, \left\lceil \frac{10-4}{4} \right\rceil - 1\right\} \cdot 1 + \max\left\{0, \left\lceil \frac{10-5}{5} \right\rceil - 1\right\} \cdot 1 +$$
$$+ \max\left\{0, \left\lceil \frac{10-11}{11} \right\rceil - 1\right\} \cdot 2 + \max\left\{0, \left\lceil \frac{10-16}{16} \right\rceil - 1\right\} \cdot 2 = 1 + 0 + 0 + 0 = 1$$

$$\tilde{f}_1^2 = a_1 + c_1 + I_a(0,6) + I_f(0,6) = 0 + 3 + 2 + 0 = 5 = d_1^3 \neq d_1^2$$
$$I_a(0,6) = 1 + 1 = 2$$
$$I_f(0,6) = \max\left\{0, \left\lceil \frac{6-4}{4} \right\rceil - 1\right\} \cdot 1 + \max\left\{0, \left\lceil \frac{6-5}{5} \right\rceil - 1\right\} \cdot 1 +$$
$$+ \max\left\{0, \left\lceil \frac{6-11}{11} \right\rceil - 1\right\} \cdot 2 + \max\left\{0, \left\lceil \frac{6-16}{16} \right\rceil - 1\right\} \cdot 2 = 0 + 0 + 0 + 0 = 0$$

$$\tilde{f}_1^3 = a_1 + c_1 + I_a(0,5) + I_f(0,5) = 0 + 3 + 1 + 0 = 4 = d_1^4 \neq d_1^3$$

$$I_a(0,5) = 1$$

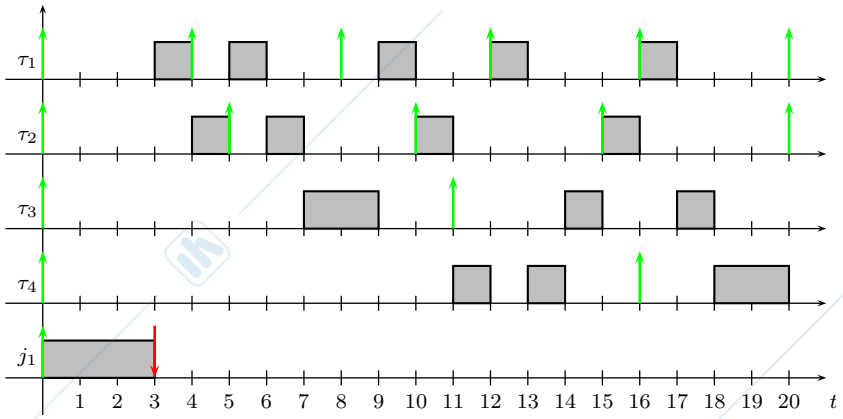
$$I_f(0,5) = \max\left\{0, \left\lceil \frac{5-4}{4} \right\rceil - 1\right\} \cdot 1 + \max\left\{0, \left\lceil \frac{5-5}{5} \right\rceil - 1\right\} \cdot 1 +$$
$$+ \max\left\{0, \left\lceil \frac{5-11}{11} \right\rceil - 1\right\} \cdot 2 + \max\left\{0, \left\lceil \frac{5-16}{16} \right\rceil - 1\right\} \cdot 2 = 0 + 0 + 0 + 0 = 0$$

$$\tilde{f}_1^4 = a_1 + c_1 + I_a(0,4) + I_f(0,4) = 0 + 3 + 0 + 0 = 3 = d_1^5 \neq d_1^4$$

$$I_a(0,4) = 0$$

$$I_f(0,4) = \max\left\{0, \left\lceil \frac{4-4}{4} \right\rceil - 1\right\} \cdot 1 + \max\left\{0, \left\lceil \frac{4-5}{5} \right\rceil - 1\right\} \cdot 1 +$$
$$+ \max\left\{0, \left\lceil \frac{4-11}{11} \right\rceil - 1\right\} \cdot 2 + \max\left\{0, \left\lceil \frac{4-16}{16} \right\rceil - 1\right\} \cdot 2 = 0 + 0 + 0 + 0 = 0$$

$d_1 = 3$.



Nota. Anche se $d_1^5 \neq d_1^4$ è inutile reiterare il sistema perché per la schedulabilità non potrà mai essere $d_1 < a_1 + c_1$ e $a_1 + c_1 = 3$. □

Esercizio 3.6

Schedulare i seguenti processi.

	τ_1	τ_2
ϕ_i	0	0
C_i	1	2
T_i	6	9

	j_1	j_2
a_i	1	8
c_i	2	1

$$U_p = \frac{1}{6} + \frac{2}{9} \cong 0,39$$

$$U_s = 1 - 0,39 = 0,61$$

$$d_1^0 = \max\{1, 0\} + \left\lceil \frac{2}{0,61} \right\rceil = 1 + 4 = 5$$

$$next\_r_1(1) = \left\lceil \frac{1+1}{6} \right\rceil \cdot 6 = 6 \qquad next\_r_2(1) = \left\lceil \frac{1+1}{9} \right\rceil \cdot 9 = 9$$

$$\tilde{f}_1^0 = a_1 + c_1 + I_a(1,5) + I_f(1,5) = 1 + 2 + 0 + 0 = 3 = d_1^1 \neq d_1^0$$

$I_a(1,5) = 0$ non ci sono processi attivi con $d_i < d_k^s$

$$I_f(1,5) = \max\left\{0, \left\lceil \frac{5-6}{6} \right\rceil - 1\right\} \cdot 1 + \max\left\{0, \left\lceil \frac{5-9}{9} \right\rceil - 1\right\} \cdot 2 = 0 + 0 = 0$$

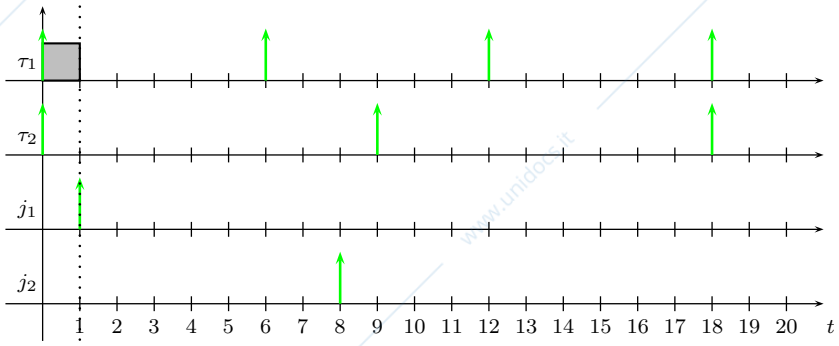


Figura 3.1: Valutazione processi attivi all'istante $t = 1$

$$d_2^0 = \max\{8, 5\} + \left\lceil \frac{1}{0,61} \right\rceil = 8 + 2 = 10$$

$$next\_r_1(8) = \left\lceil \frac{8+1}{6} \right\rceil \cdot 6 = 12 \qquad next\_r_2(8) = \left\lceil \frac{8+1}{9} \right\rceil \cdot 9 = 9$$

$$\tilde{f}_2^0 = a_2 + c_2 + I_a(8,10) + I_f(8,10) = 8 + 1 + 0 + 0 = 9 = d_2^1 \neq d_2^0$$

$I_a(8,10) = 0$ non ci sono processi attivi all'istante $t = 8$

$$I_f(8,10) = \max\left\{0, \left\lceil \frac{10-12}{6} \right\rceil - 1\right\} \cdot 1 + \max\left\{0, \left\lceil \frac{10-9}{9} \right\rceil - 1\right\} \cdot 2 = 0 + 0 = 0$$

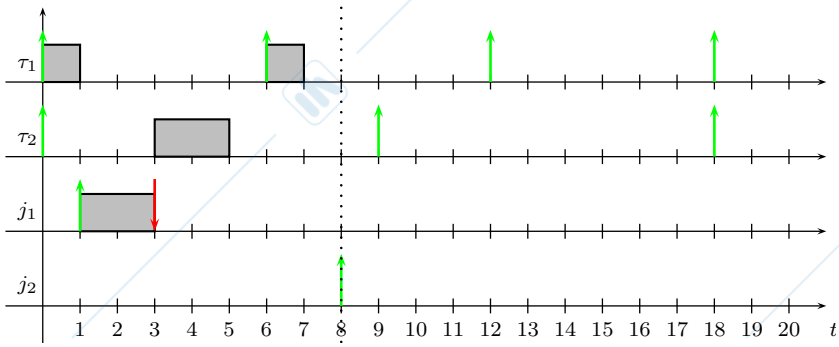
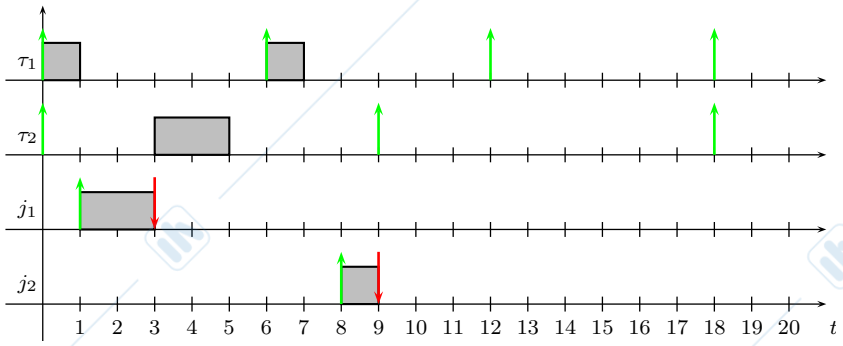


Figura 3.2: Valutazione processi attivi all'istante $t = 8$



Esercizio 3.7

Schedulare i seguenti processi.

	τ_1	τ_2
ϕ_i	0	0
C_i	2	1
T_i	5	3

	j_1	j_2
a_i	1	5
c_i	2	3

$$U_p = \frac{2}{5} + \frac{1}{3} \cong 0,73$$

$$U_s = 1 - 0,73 = 0,27$$

$$d_1^0 = \max\{1,0\} + \left\lceil \frac{2}{0,27} \right\rceil = 1 + 8 = 9$$

$$next\_r_1(1) = \left\lceil \frac{1+1}{5} \right\rceil \cdot 5 = 5 \qquad next\_r_2(1) = \left\lceil \frac{1+1}{3} \right\rceil \cdot 3 = 3$$

$$\tilde{f}_1^0 = a_1 + c_1 + I_a(1,9) + I_f(1,9) = 1 + 2 + 2 + 1 = 6 = d_1^1 \neq d_1^0$$

$I_a(1,9) = 2$ è attivo solo τ_1 all'istante $t = 1$

$$I_f(1,9) = \max\left\{0, \left\lceil \frac{9-5}{5} \right\rceil - 1\right\} \cdot 2 + \max\left\{0, \left\lceil \frac{9-3}{3} \right\rceil - 1\right\} \cdot 1 = 0 + 1 = 1$$

$$\tilde{f}_1^1 = a_1 + c_1 + I_a(1,6) + I_f(1,6) = 1 + 2 + 2 + 0 = 5 = d_1^2 \neq d_1^1$$

$I_a(1,6) = 2$ è attivo solo τ_1 all'istante $t = 1$

$$I_f(1,6) = \max\left\{0, \left\lceil \frac{6-5}{5} \right\rceil - 1\right\} \cdot 2 + \max\left\{0, \left\lceil \frac{6-3}{3} \right\rceil - 1\right\} \cdot 1 = 0 + 0 = 0$$

$$\tilde{f}_1^2 = a_1 + c_1 + I_a(1,5) + I_f(1,5) = 1 + 2 + 0 + 0 = 3 = d_1^3 \neq d_1^2$$

$I_a(1,5) = 0$ non ci sono processi attivi all'istante $t = 1$

$$I_f(1,5) = \max\left\{0, \left\lceil \frac{5-5}{5} \right\rceil - 1\right\} \cdot 2 + \max\left\{0, \left\lceil \frac{5-3}{3} \right\rceil - 1\right\} \cdot 1 = 0 + 0 = 0$$

$$d_2^0 = \max\{5,9\} + \left\lceil \frac{3}{0,27} \right\rceil = 9 + 12 = 21$$

$$next\_r_1(5) = \left\lceil \frac{5+1}{5} \right\rceil \cdot 5 = 10 \qquad next\_r_2(5) = \left\lceil \frac{5+1}{3} \right\rceil \cdot 3 = 6$$

$$\tilde{f}_2^0 = a_2 + c_2 + I_a(5,21) + I_f(5,21) = 5 + 3 + 3 + 8 = 19 = d_2^1 \neq d_2^0$$

$I_a(5,21) = 3$ entrambi i processi sono attivi all'istante $t = 5$

$$I_f(5,21) = \max\left\{0, \left\lceil \frac{21-10}{5} \right\rceil - 1\right\} \cdot 2 + \max\left\{0, \left\lceil \frac{21-6}{3} \right\rceil - 1\right\} \cdot 1 = 4 + 4 = 8$$

$$\tilde{f}_2^1 = a_2 + c_2 + I_a(5,19) + I_f(5,19) = 5 + 3 + 3 + 6 = 17 = d_2^2 \neq d_2^1$$

$I_a(5,19) = 3$ entrambi i processi sono attivi all'istante $t = 5$

$$I_f(5,19) = \max\left\{0, \left\lceil \frac{19-10}{5} \right\rceil - 1\right\} \cdot 2 + \max\left\{0, \left\lceil \frac{19-6}{3} \right\rceil - 1\right\} \cdot 1 = 2 + 4 = 6$$

$$\tilde{f}_2^2 = a_2 + c_2 + I_a(5,17) + I_f(5,17) = 5 + 3 + 3 + 5 = 16 = d_2^3 \neq d_2^2$$

$I_a(5,17) = 3$ entrambi i processi sono attivi all'istante $t = 5$

$$I_f(5,17) = \max\left\{0, \left\lceil \frac{17-10}{5} \right\rceil - 1\right\} \cdot 2 + \max\left\{0, \left\lceil \frac{17-6}{3} \right\rceil - 1\right\} \cdot 1 = 2 + 3 = 5$$

$$\tilde{f}_2^3 = a_2 + c_2 + I_a(5,17) + I_f(5,17) = 5 + 3 + 3 + 5 = 16 = d_2^4 = d_2^3$$

$I_a(5,16) = 3$ entrambi i processi sono attivi all'istante $t = 5$

$$I_f(5,16) = \max\left\{0, \left\lceil \frac{16-10}{5} \right\rceil - 1\right\} \cdot 2 + \max\left\{0, \left\lceil \frac{16-6}{3} \right\rceil - 1\right\} \cdot 1 = 2 + 3 = 5$$

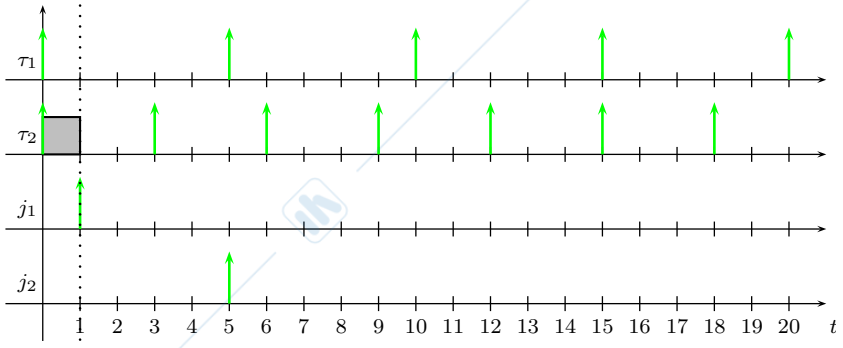


Figura 3.3: Valutazione processi attivi all'istante $t = 1$

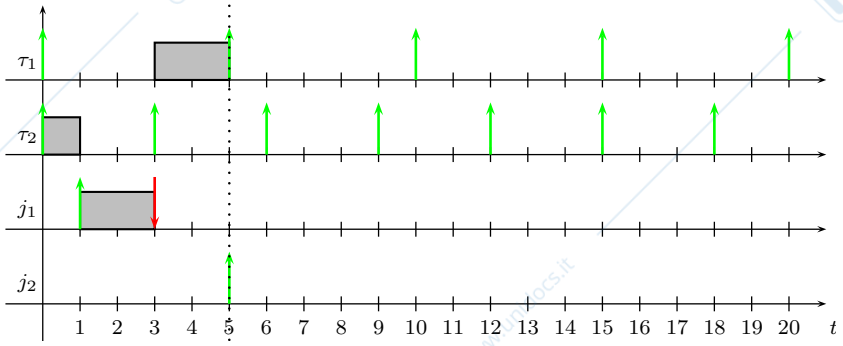
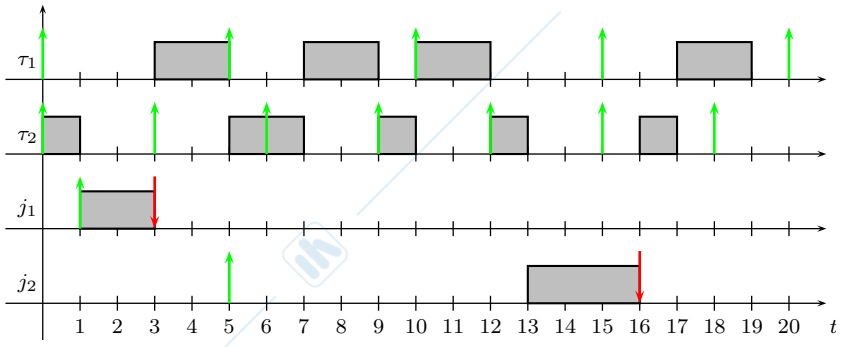


Figura 3.4: Valutazione processi attivi all'istante $t = 5$



Capitolo 4

Gestione dell'overload

4.1 Modello

4.1.1 Simboli

- λ frequenza media di arrivo delle richieste
- c carico medio della richiesta
- $\rho = \lambda \cdot c$ carico
- $V_i(f_i)$ valore del processo
- tipi di processo:

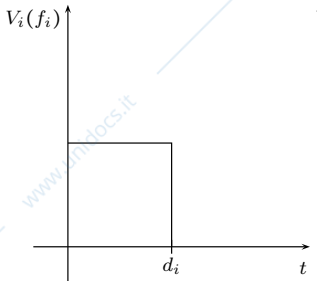


Figura 4.1: Firm

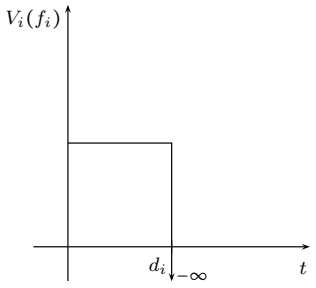


Figura 4.2: Critico

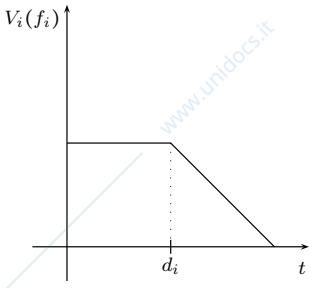


Figura 4.3: SRT

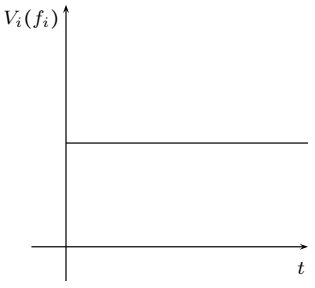


Figura 4.4: Non RT

4.2 Workload

Fattore di *load* da calcolare negli istanti t di attivazione dei processi presenti nel sistema.

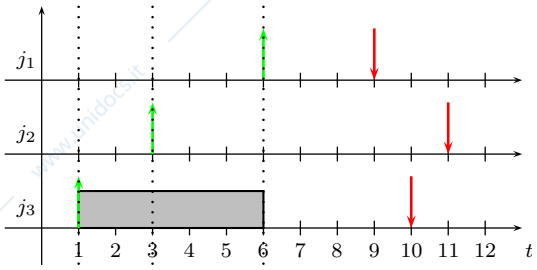
$$\rho(t) = \max_{k|j_k \text{ attivo}} \left\{ \frac{\sum_{i|d_i \leq d_k} c_i(t)}{d_k - t} \right\}$$

se $\rho(t) > 1$ siamo in presenza di overload.

Esercizio 4.1

Calcolare il fattore di load negli istanti $t = 1$, $t = 3$ e $t = 6$.

	j_1	j_2	j_3
a_i	6	3	1
c_i	2	3	6
d_i	9	11	10

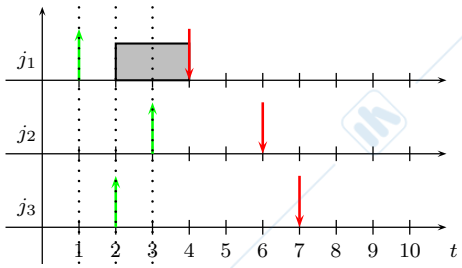


$$\rho(1) = \max\left\{\frac{c_3(1)}{d_3 - 1}\right\}$$
$$= \frac{6}{9} < 1 \quad \Rightarrow \text{ no overload}$$
$$\rho(3) = \max\left\{\frac{c_2(3) + c_3(3)}{d_2 - 3}, \frac{c_3(3)}{d_3 - 3}\right\}$$
$$= \max\left\{\frac{7}{8}, \frac{4}{7}\right\} = \frac{7}{8} < 1 \quad \Rightarrow \text{ no overload}$$
$$\rho(6) = \max\left\{\frac{c_1(6)}{d_1 - 6}, \frac{c_1(6) + c_2(6) + c_3(6)}{d_2 - 6}, \frac{c_1(6) + c_3(6)}{d_3 - 6}\right\}$$
$$= \max\left\{\frac{2}{3}, \frac{6}{5}, \frac{3}{4}\right\} = \frac{6}{5} > 1 \quad \Rightarrow \text{ overload}$$

Esercizio 4.2

Calcolare il fattore di load negli istanti $t = 1$, $t = 2$ e $t = 3$.

	j_1	j_2	j_3
a_i	1	3	2
c_i	2	2	2
d_i	4	6	7



$$\rho(1) = \frac{2}{3} < 1 \quad \Rightarrow \text{ no overload}$$

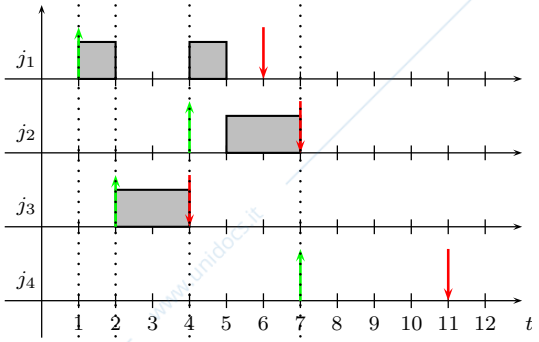
$$\rho(2) = \max\left\{\frac{1}{2}, \frac{3}{5}\right\} = \frac{3}{5} < 1 \quad \Rightarrow \text{ no overload}$$

$$\rho(3) = \max\left\{0, \frac{2}{3}, \frac{4}{4}\right\} = 1 \quad \Rightarrow \text{ no overload}$$

Esercizio 4.3

Calcolare il fattore di load negli istanti $t = 1$, $t = 2$, $t = 4$ e $t = 7$.

	j_1	j_2	j_3	j_4
a_i	1	4	2	7
c_i	4	2	2	3
d_i	6	7	4	11



$$\rho(1) = \frac{4}{5} < 1 \quad \Rightarrow \text{ no overload}$$

$$\rho(2) = \max\left\{\frac{5}{4}, \frac{2}{2}\right\} = \frac{5}{4} > 1 \quad \Rightarrow \text{ overload}$$

4.3 D_{OVER}

D_{OVER}. Massimizza $\Gamma_A = \sum_i V_i$.
Dato Γ^* (valore ottimo della schedulazione qualora l'algoritmo fosse chiaroveggente), sia

$$\max_{\phi_A} \frac{\Gamma_A}{\Gamma^*} \geq \phi_A$$

$\mathrm{D_{OVER}}$ è un algoritmo on–line ottimo in situazioni di overload, cioè $\phi_{\mathrm{D_{OVER}}} = 0,25$.
Come EDF tranne quando un professo j_z reggiunge LST_z (last start time), cioè il suo tempo rimanente di computazione è uguale al tempo rimanente fino alla sua deadline.
Due set di task

- *privileged*: ogni volta che viene fatta preemption su un task questo diventa privilegiato
- *waiting*: ogni volta che un task è schedulato a causa di un LST tutti i task ready diventano waiting

All'istante $t = \mathrm{LST}_z$ se z non è il processo che EDF schedulerebbe

- EDF $\rightarrow j_{\mathrm{curr}}$
- LST $\rightarrow j_z$

si verifica la seguente condizione

$$V_z > (1 + \sqrt{k}) \cdot (V_{\mathrm{curr}} + V_{\mathrm{priv}})$$

- se è soddisfatta eseguo z, metto *curr* tra i pronti
  - se non ho ancora terminato z e $t = \mathrm{LST}_x$ verifico la seguente condizione

$$V_x > (1 + \sqrt{k}) \cdot V_z$$

    * se è verificata scarto z
    * se non è verificata scarto x
- se non è verificata scarto z

dove

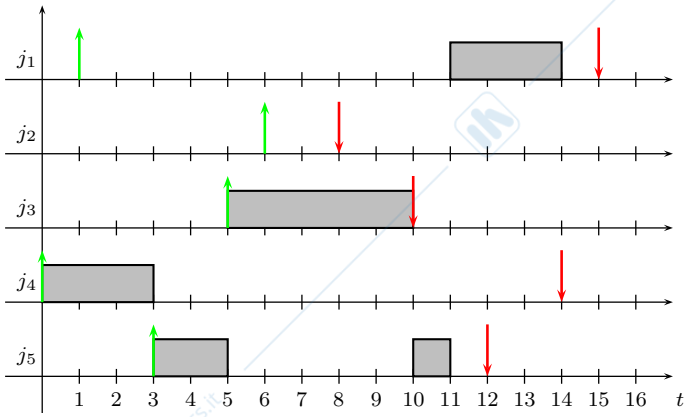
$$k = \frac{\overline{V}}{\underline{V}} \qquad \overline{V} = \max_{i|i \text{ è attivo a } t} \left\{ \frac{V_i}{c_i} \right\} \qquad \underline{V} = \min_{i|i \text{ è attivo a } t} \left\{ \frac{V_i}{c_i} \right\}$$

□

Esercizio 4.4

Calcolare il fattore di load negli istanti notevoli e riportare la schedulazione dei seguenti processi.

	j_1	j_2	j_3	j_4	j_5
a_i	1	6	5	0	3
c_i	3	2	5	3	3
d_i	15	8	10	14	12
V_i	1	1	17	1	1



$$\rho(0) = \frac{3}{14} < 1 \quad \Rightarrow \text{ no overload}$$

$$\rho(1) = \max\left\{\frac{5}{14}, \frac{2}{13}\right\} = \frac{5}{14} < 1 \quad \Rightarrow \text{ no overload}$$

$$\rho(3) = \max\left\{\frac{6}{12}, \frac{3}{9}\right\} = \frac{6}{12} < 1 \quad \Rightarrow \text{ no overload}$$

$$\rho(5) = \max\left\{\frac{9}{10}, \frac{5}{5}, \frac{6}{7}\right\} = 1 \quad \Rightarrow \text{ no overload}$$

$$\rho(6) = \max\left\{\frac{10}{9}, \frac{2}{2}, \frac{6}{4}, \frac{7}{5}\right\} = \frac{6}{4} > 1 \quad \Rightarrow \text{ overload}$$

$t = 6$ curr $= j_2$ priv $= \{j_5\}$ $j_z = j_3$

$$\overline{V} = \max\left\{\frac{1}{3}, \frac{1}{2}, \frac{17}{5}, \frac{1}{3}\right\} \qquad \underline{V} = \min\left\{\frac{1}{3}, \frac{1}{2}, \frac{17}{5}, \frac{1}{3}\right\}$$

$$k = \frac{3 \cdot 17}{5} \qquad \overline{V} = \frac{17}{5} \qquad \underline{V} = \frac{1}{3}$$

$$17 > \left(1 + \sqrt{\frac{3 \cdot 17}{5}}\right) \cdot (1 + 1) \cong 8,39 \Rightarrow \text{ verificato}$$

A $t = 6$ viene eseguito j_3 e scartato j_2 (in quanto era anche quest'ultimo in LST).

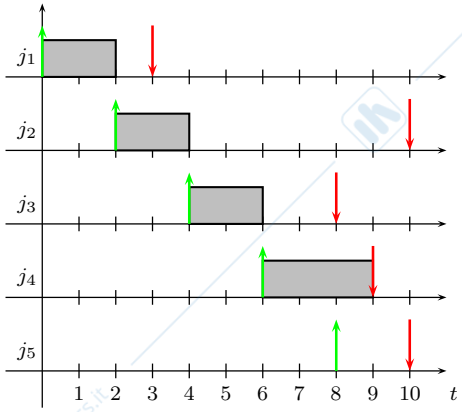
Nota. All'istante $t = 5$ il processo j_3 era in LST, ma non è stato necessario verificare nessuna condizione in quanto

1. non c'era overload
2. il processo j_3 era quello che EDF avrebbe mandato in esecuzione □

Esercizio 4.5

Calcolare il fattore di load negli istanti notevoli e riportare la schedulazione dei seguenti processi.

	j_1	j_2	j_3	j_4	j_5
a_i	0	2	4	6	8
c_i	2	3	3	3	2
d_i	3	10	8	9	10
V_i	3	1	2	15	7



$$\rho(0) = \frac{2}{3} < 1 \quad \Rightarrow \text{ no overload}$$

$$\rho(2) = \frac{3}{8} < 1 \quad \Rightarrow \text{ no overload}$$

$$\rho(4) = \max\left\{\frac{4}{6}, \frac{3}{4}\right\} = \frac{3}{4} < 1 \quad \Rightarrow \text{ no overload}$$

$$\rho(6) = \max\left\{\frac{5}{4}, \frac{1}{2}, \frac{4}{3}\right\} = 2 > 1 \quad \Rightarrow \text{ overload}$$

$t = 6$ curr $= j_3$ priv $= \{j_2\}$ $j_z = j_4$

$$\overline{V} = \max\left\{\frac{1}{3}, \frac{2}{3}, \frac{15}{3}\right\} \qquad \underline{V} = \min\left\{\frac{1}{3}, \frac{2}{3}, \frac{15}{3}\right\}$$

$$k = \frac{3 \cdot 15}{3} = 15 \qquad \overline{V} = \frac{15}{3} \qquad \underline{V} = \frac{1}{3}$$

$$15 > \left(1 + \sqrt{15}\right) \cdot (2 + 1) \cong 14,6 \Rightarrow \text{ verificato}$$

A $t = 6$ viene eseguito j_4 e j_3 è inserito tra i processi pronti.

$t = 7$ curr $= j_4$ $j_x = j_3$

$$\overline{V} = \max\left\{\frac{1}{3}, \frac{2}{3}, \frac{15}{3}\right\} \qquad \underline{V} = \min\left\{\frac{1}{3}, \frac{2}{3}, \frac{15}{3}\right\}$$

$$k = \frac{3 \cdot 15}{3} = 15 \qquad \overline{V} = \frac{15}{3} \qquad \underline{V} = \frac{1}{3}$$

$$2 > \left(1 + \sqrt{15}\right) \cdot 15 \cong 73 \Rightarrow \text{ non verificato}$$

A $t = 7$ viene eseguito j_4 e j_3 è scartato.

$$\rho(8) = \max\left\{\frac{4}{2}, \frac{1}{1}, \frac{4}{2}\right\} = 2 > 1 \qquad \Rightarrow \text{ overload}$$

$$t = 8 \qquad \text{curr} = j_4 \qquad j_x = j_5$$

$$\overline{V} = \max\left\{\frac{1}{3}, \frac{15}{3}, \frac{7}{2}\right\} \qquad \underline{V} = \min\left\{\frac{1}{3}, \frac{15}{3}, \frac{7}{2}\right\}$$

$$k = \frac{3 \cdot 15}{3} = 15 \qquad \overline{V} = \frac{15}{3} \qquad \underline{V} = \frac{1}{3}$$

$$7 > \left(1 + \sqrt{15}\right) \cdot 15 \cong 73 \Rightarrow \text{ non verificato}$$

A $t = 8$ viene eseguito j_4 e j_5 è scartato.

Parte II

Statecharts

Capitolo 5

Statecharts

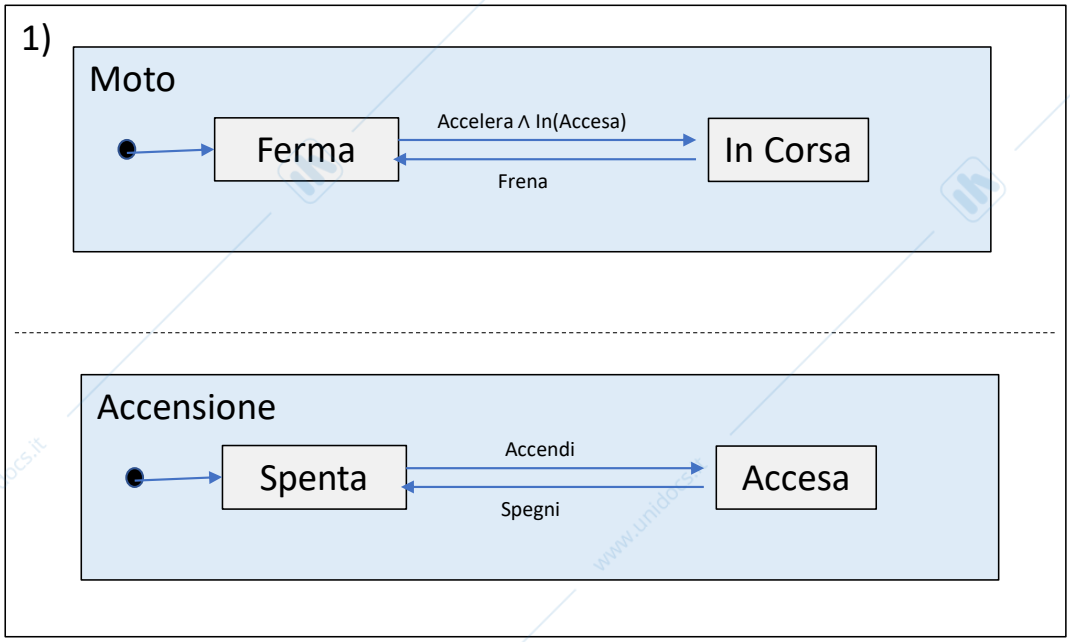
5.1 Esercizi

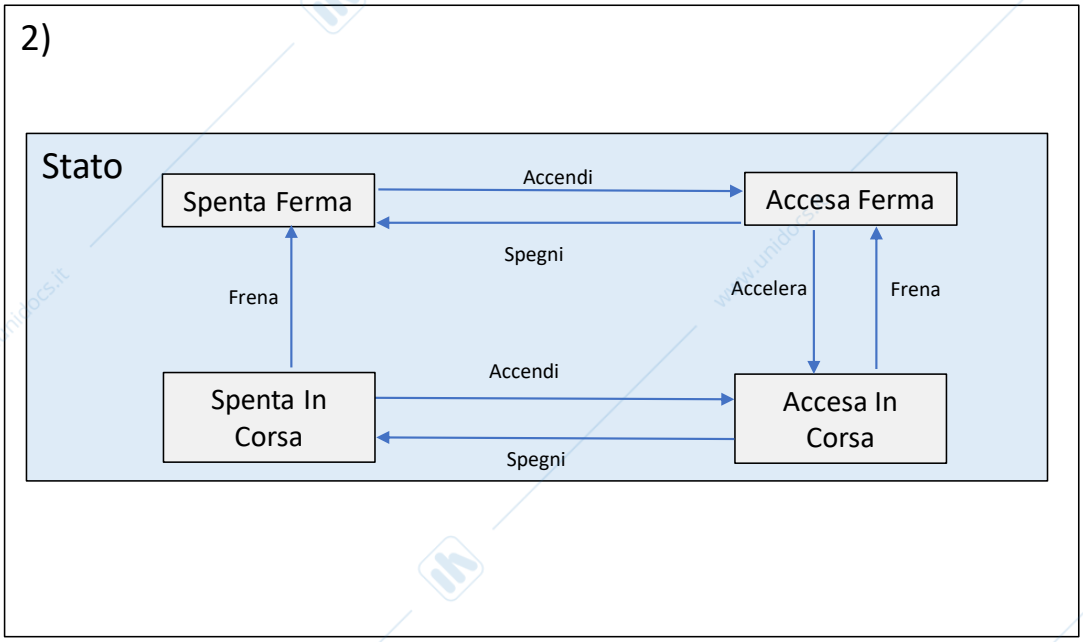
Esercizio 5.1

Specificare mediante il formalismo degli statecharts il seguente problema. Si ha una macchina che si muove su un rettilineo e può essere accesa o spenta. Allo stesso tempo, la macchina può essere ferma o in corsa a secondo che si verifichi "accelera" *o* "frena"*. Se spenta, la macchina non può accelerare.*

1 Modellizzare il problema definendo degli stati concorrenti.

2 Modellizzare poi il problema senza usare concorrenza tra stati.

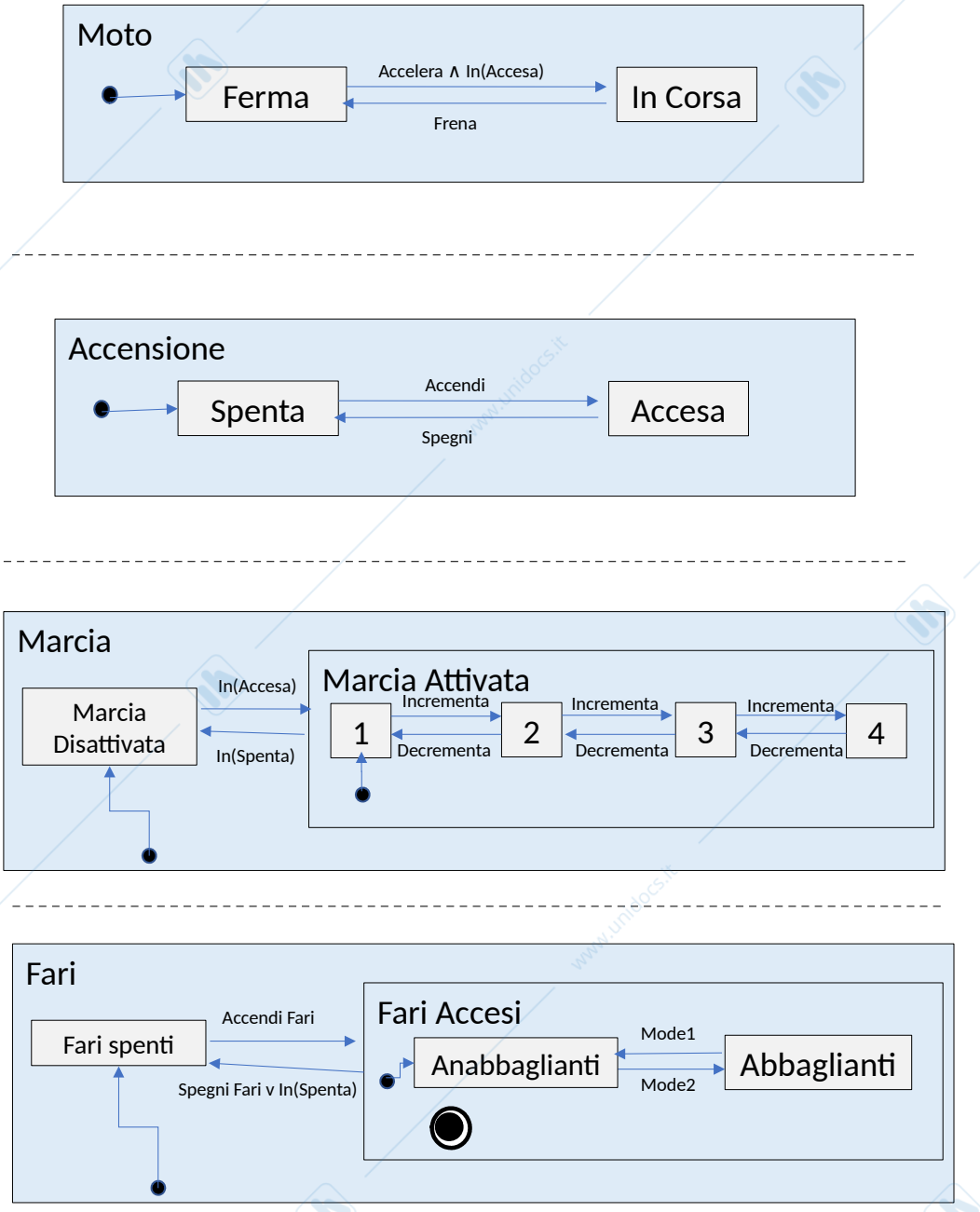




Esercizio 5.2

Partendo dalla soluzione del primo punto dell'esercizio precedente, aggiungere la modellizzazione

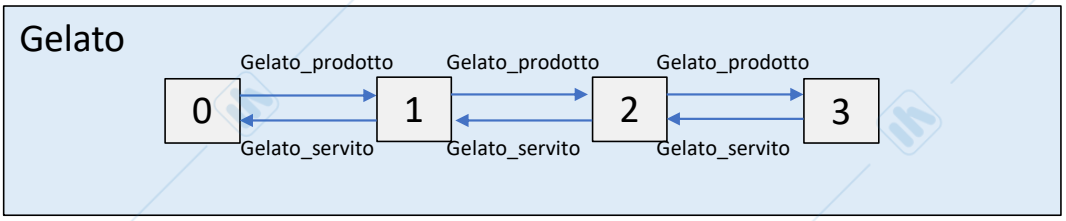
del cambio della marcia. La macchina può trovarsi negli stati "marcia attivata" *o* "marcia disattivata". *Assumere che l'automobile può procedere su 4 marce e che si può cambiare marcia solo decrementando e incrementando di 1 la marcia corrente. Nel caso in cui la macchina proceda sulla prima o quarta marcia, si potrà solo incrementare o decrementare la marcia. Ogni volta che la macchina viene spenta la marcia viene disattivata. Ogni volta che la macchina viene accesa la prima marcia viene attivata. Aggiungere poi la modellizzazione del funzionamento dell'illuminazione dell'automobile. I fari possono essere accesi o spenti. Quando la macchina è spenta i fari sono spenti, mentre quando la macchina è accesa i fari possono essere accesi o spenti a secondo che si verifichino gli eventi "accendi fari"* e "spegni fari". *I fari possono essere accesi in due modalità: abbaglianti e anabbaglianti. Il cambio di modalità avviene ogni volta che si verifica l'evento* "mode1" e "mode2". *Quando si accendono i fari, questi si troveranno nell'ultima modalità utilizzata prima di essere stati spenti.*

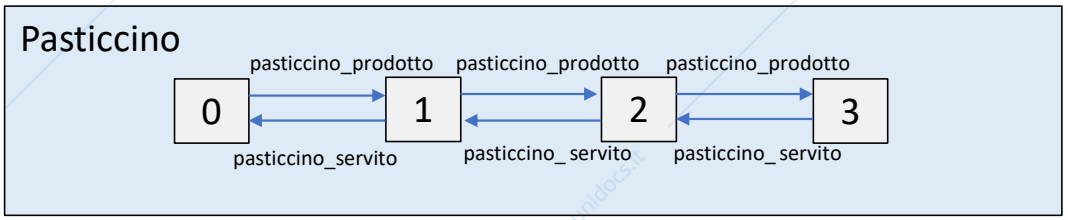


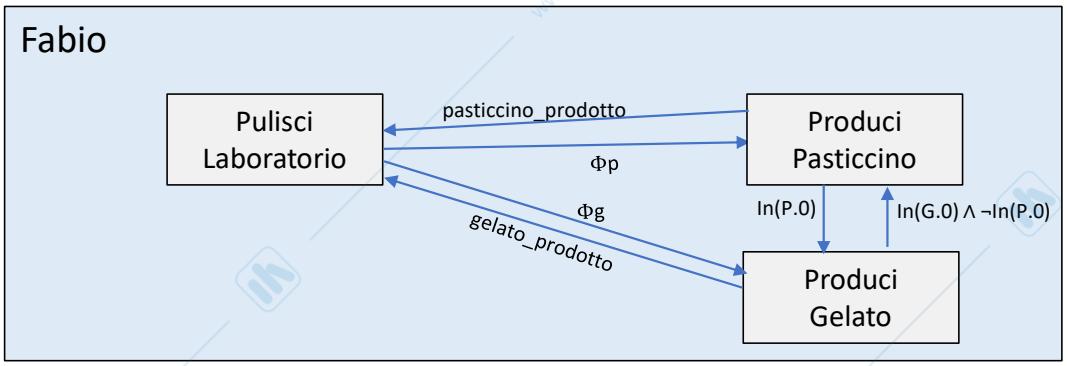
Esercizio 5.3

(6 punti) Si vuole modellare tramite il formalismo degli StateCharts la produzione e la vendita in

una delle migliori gelaterie e pasticcerie italiane. Ci sono due persone: Fabio, addetto alla produzione e Cecilia, addetta alle vendite; ci sono due prodotti: gelato e pasticcino. Per ogni prodotto si deve tenere traccia della scorta in magazzino che può essere da 0 a 3 pezzi. Se il magazzino è pieno (contiene 3 pezzi per ciascun prodotto), Fabio si trova nello stato pulisce_laboratorio, altrimenti si trova nello stato produce_pasticcino o produce_gelato. Nella produzione viene prediletto l'articolo di cui è presente meno scorta in magazzino, a parità vengono prediletti i pasticcini. Una volta terminata la produzione di un pasticcino si verifica l'evento pasticcino pasticcino_prodotto. Analogamente per la produzione del gelato. Qualora stia producendo un prodotto (p.e. gelato) e termina la scorta dell'altro prodotto (p.e. pasticcino), il lavoro viene interrotto e Fabio cambia prodotto in produzione (rispettando le precedenze introdotte prima). Se non vi sono presenti clienti, Cecilia si trova nello stato pulisce_negozio, ma all'arrivo di un potenziale cliente (evento cliente_entra) si ferma ed entra nello stato attende_ordine. Da qui se si verifica l'evento ordina_pasticcino e in magazzino sono presenti pasticcini, lei si sposta nello stato serve_pasticcino; una volta servito il pasticcino (evento pasticcino_servito) si riporta nello stato attende_ordine. Gli ordini di gelato sono gestiti analogamente. Qualora il cliente faccia un ordine di cui non si dispone merce in magazzino, egli può attendere che venga prodotto in laboratorio o uscire dal negozio (evento cliente_esce). Non è richiesto di modellare il pagamento della merce (si supponga che dopo essere stato servito il cliente possa uscire dal negozio).

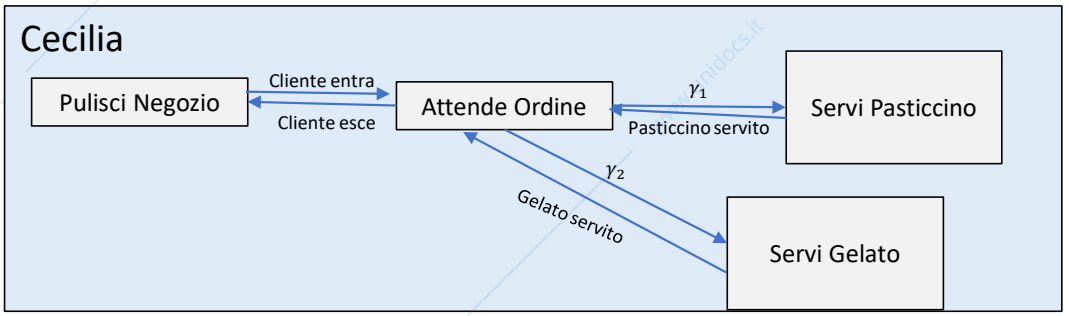






Φp = In(P.0) ∨ (In(P.1) ∧ ¬In(G.0)) ∨ (In(P.2) ∧ (¬(In(G.1) ∧ ¬(G.0))

Φg = ¬ Φp ∧ ¬In(G.3)



γ_1 = ordina pasticcino ∧ ¬In(P.0)

γ_2 = ordina gelato ∧ ¬In(G.0)

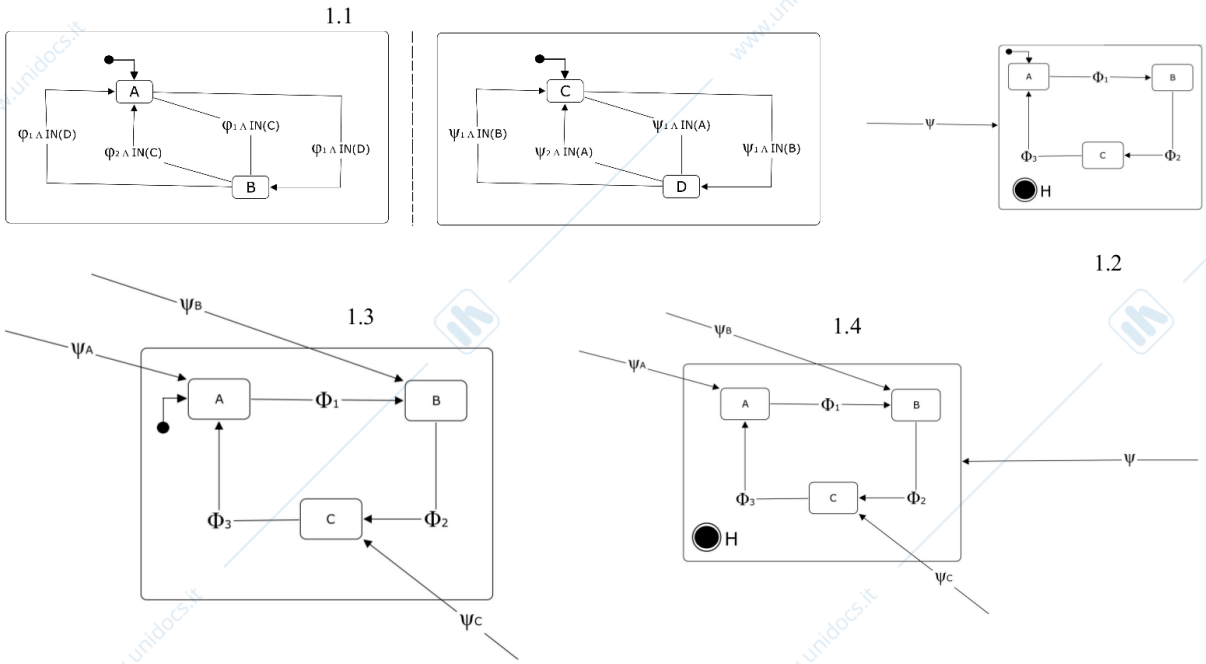
Esercizio 5.4

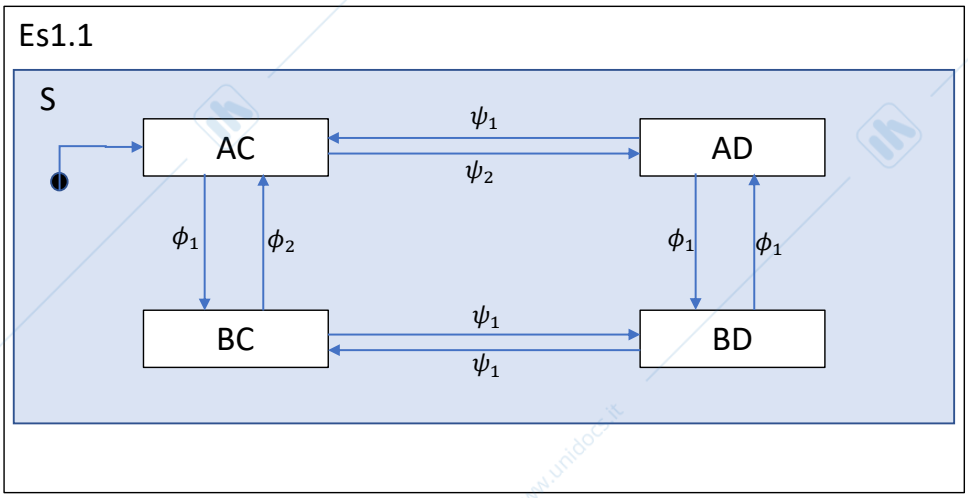
Per ognuna delle porzioni di Statechart riportate qui sotto si vuole produrre una porzione equivalente (anche tramite l'aggiunta di stati e transizioni se necessario) ponendo vincoli alla sintassi del linguaggio. 1.1 In questo caso non è possibile avere concorrenza tra stati.

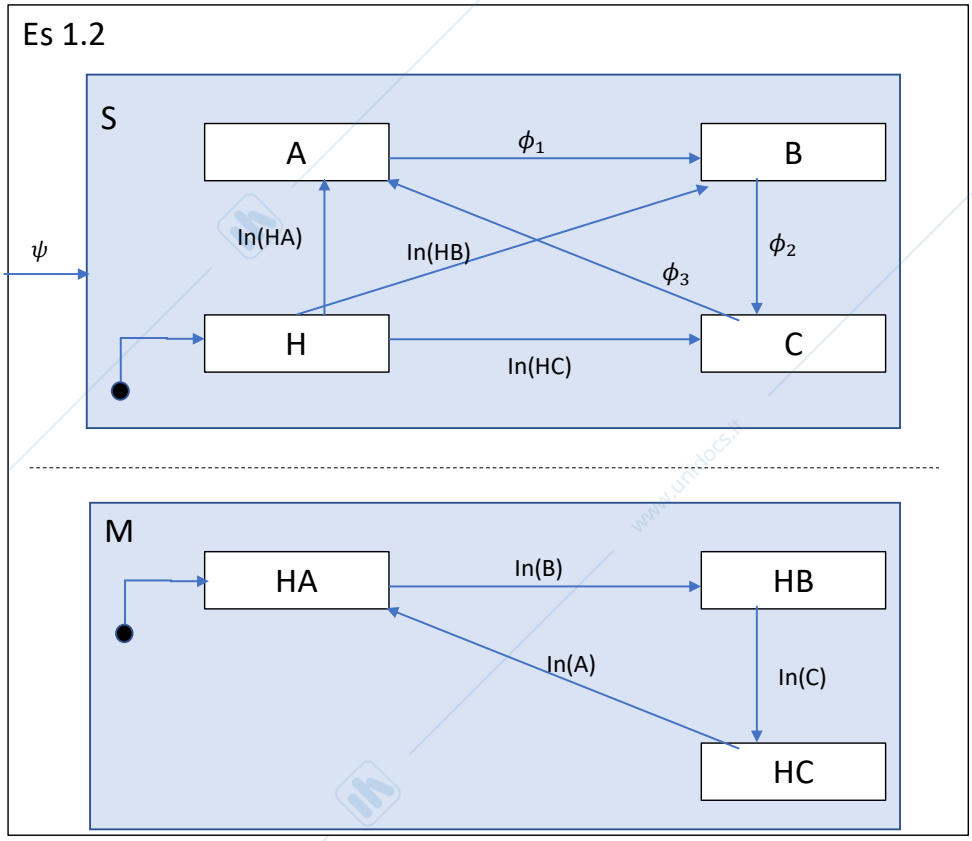
1.2 In questo caso non è possibile avere né stati di storia né transizioni che conducono a stati interni ad un macrostato.

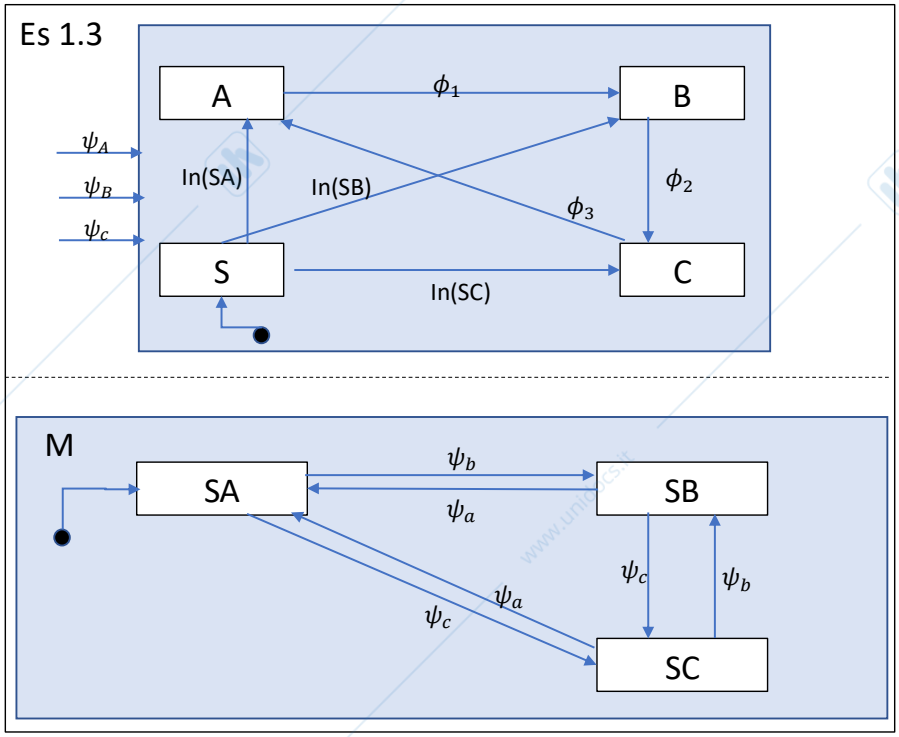
1.3 In questo caso non è possibile avere transizioni che conducono direttamente a stati interni ad un macrostato.

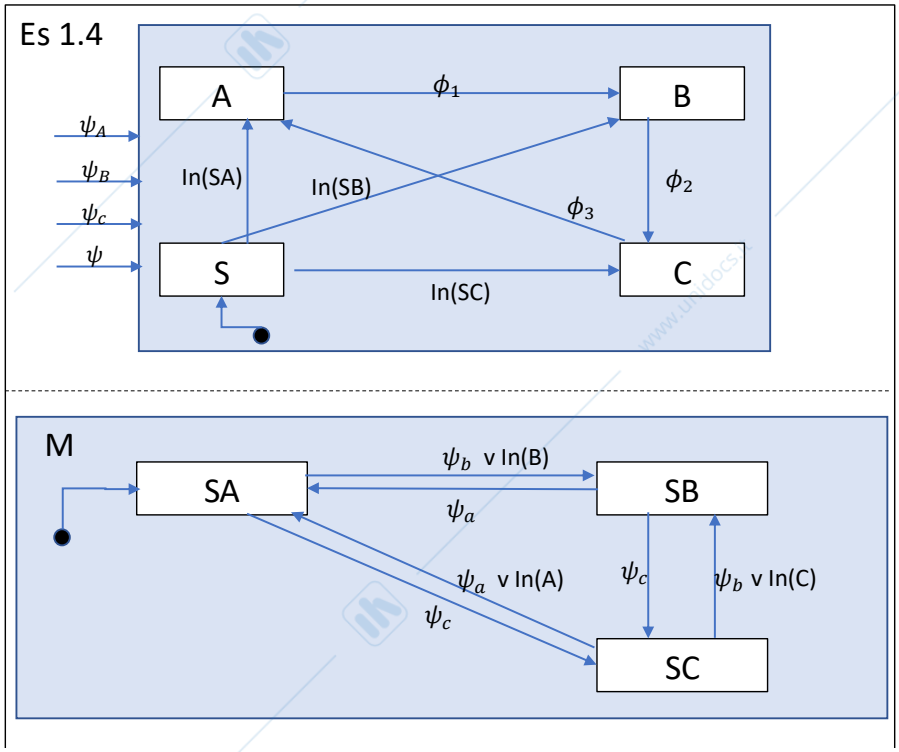
1.4 In questo caso non è possibile avere né stati di storia né transizioni che conducono a stati interni ad un macrostato.











Esercizio 5.5

(9 punti) Specificare mediante il formalismo degli statecharts il seguente problema senza utilizzare variabili. Si ha un ascensore che può muoversi attraverso quattro piani numerati da 0 (piano terra)

a 3. L'ascensore può essere fermo oppure in movimento. Qualora sia fermo, la porta può essere aperta oppure chiusa. Qualora l'ascensore sia in movimento, si può trovare in uno dei quattro piani (considerare il tempo di transizione da un piano a un altro trascurabile dal punto di vista della specifica e non specificare lo stato della porta durante il movimento, in quanto viene assunto chiuso). Il controllo dell'ascensore è basato sull'utilizzo di due pulsanti set e start dove, in linea di principio, il pulsante set permette di selezionare il piano a cui si vuole andare e il pulsante start permette di far partire l'ascensore. Il pulsante set permette di selezionare, solo qualora l'ascensore sia fermo e la porta sia chiusa, il piano desiderato incrementando questo di 1. Ad esempio, premendo set con la porta chiusa e l'ascensore fermo quando il piano selezionato è il piano 1, si seleziona il piano 2. Inoltre, la selezione è ciclica: schiacciando set quando il piano selezionato è il piano 3 si seleziona il piano 0. Esiste una selezione iniziale del piano data di default (scelta dallo studente) e la selezione non viene a cancellarsi a fronte di eventi quali l'apertura o la chiusura delle porte o start. Premendo il pulsante start con la porta chiusa l'ascensore va in movimento. Qualora l'ascensore abbia raggiunto il piano selezionato, l'ascensore diventa fermo. Qualora il piano selezionato sia il piano in cui l'ascensore si trova, l'ascensore va in movimento e si ferma subito. Gli unici eventi conosciuti dal sistema sono: apri_porta, chiudi_porta, set, start. Gli stati di default possono essere assegnati liberamente purché consistenti. (Consiglio: si risolva l'esercizio in modo incrementale, prima con soli 3 piani e poi lo si estenda al caso in cui i piani siano 4.)

Esercizio 5.6

(9 punti)

Specificare mediante il formalismo degli statecharts senza utilizzare variabili il problema di coordinare due robot pulitori (R1 e R2) per lavare il pavimento di una stanza (S). Viene fornito il funzionamento di ogni robot quando questo lava la stanza da solo, e si richiede di modificare tale funzionamento al fine di permettere ai due robot di lavare il pavimento in modo coordinato. Il funzionamento di ogni robot, quando lava da solo la stanza, è il seguente: si attiva dal magazzino a fronte dell'evento start, entra all'interno della stanza, lava la stanza e quando la stanza è completamente pulita torna nel magazzino. Il criterio con cui il robot lava la stanza è il seguente: la stanza (S) viene divisa in quattro aree (A1-A2-A3-A4) contigue e le aree vengono lavate in successione una dopo l'altra (le stanze vengono lavate o nell'ordine A1-A2-A3-A4 o nell'ordine A4-A3-A2-A1). Il passaggio del robot da un'area alla successiva è basato sull'evento pulito: questo viene generato ogni qual volta l'area che il robot sta lavando diventi pulita. In sintesi, il robot pulisce un'area finché non si è presentato l'evento pulito, e, appena si presenta tale evento, il robot si ferma e si sposta nell'area da pulire successiva (qualora si trovi nell'ultima area, il robot si sposta in magazzino). Le aree possono essere sporche in vario modo richiedendo così tempi diversi per poter essere pulite. Si specifichi il funzionamento coordinato dei due robot in modo tale che:

- R1 e R2 agiscano contemporaneamente,

- R1 inizi a lavare la stanza S partendo dall'area A1 e si muova verso l'area A4,
- R2 inizi a lavare la stanza S partendo dall'area A4 e si muova verso l'area A1,
- tutte le aree vengano lavate una sola volta,
- non siano mai presenti due robot nella stessa area contemporaneamente,
- i tempi di passaggio dei robot tra due aree contigue e tra una generica area e il magazzino siano trascurabili ai fini della specifica.

(Suggerimento: si specifichi prima il funzionamento dei robot quando lavano la stanza singolarmente, e poi quando lavano la stanza in modo coordinato.)

Esercizio 5.7

(9 punti)

Specificare mediante il formalismo degli statecharts senza utilizzare variabili il seguente problema. Si ha un sistema di log-in munito di un video, una tastiera e un mouse. A video viene proiettata una schermata costituita da un campo nome, un campo password, e un pulsante di OK. L'utente può inserire i propri dati (nome, password) premere il pulsante di Ok, e il sistema verifica la correttezza dei dati inseriti per regolarne l'accesso. Il sistema può trovarsi o in uno stato in cui il campo nome è attivo (nome_attivo), oppure il campo password è attivo (pw_attivo), oppure in uno stato (controllo) in cui il sistema verifica la correttezza dei dati immessi, oppure, ancora, in un insieme di stati, descritti sotto, in cui vengono segnalati degli avvisi all'utente. All'avvio lo stato di default è nome_attivo. Tramite il mouse è poi possibile:

- qualora sia attivo il campo password, attivare il campo nome (evento click_nome),
- qualora sia attivo il campo nome, attivare il campo password (evento click_pw),
- premere il pulsante di OK (evento click_ok).

Solo nel campo correntemente attivato è possibile inserire caratteri. All'avvio del sistema i campi sono vuoti (per ogni campo esiste lo stato vuoto) e, a fronte dell'inserimento di caratteri (evento flusso_di_caratteri), il campo selezionato diventa non vuoto (non_vuoto). Premendo il pulsante di Ok è possibile:

- qualora sia il campo nome che il campo password siano vuoti, il sistema mostra un'avviso (avviso_dati_mancanti) e attiva l'ultimo stato attivo prima di aver premuto OK,
- qualora uno tra il campo nome e il campo password non sia vuoto, il sistema mostra un'avviso (avviso_nome_mancante oppure avviso_pw_mancante) e attiva lo stato relativo al campo mancante,
- qualora sia il campo nome che il campo password siano non vuoti, il sistema controlla i dati, e genera un evento true se i dati sono corretti e false se i dati sono scorretti. A fronte di true mostra un avviso di login (avviso_login), a fronte di un false mostra un'avviso (avviso_dati_sbagliati) e attiva l'ultimo stato attivo prima di aver premuto OK.

Esercizio 5.8

(10 punti)

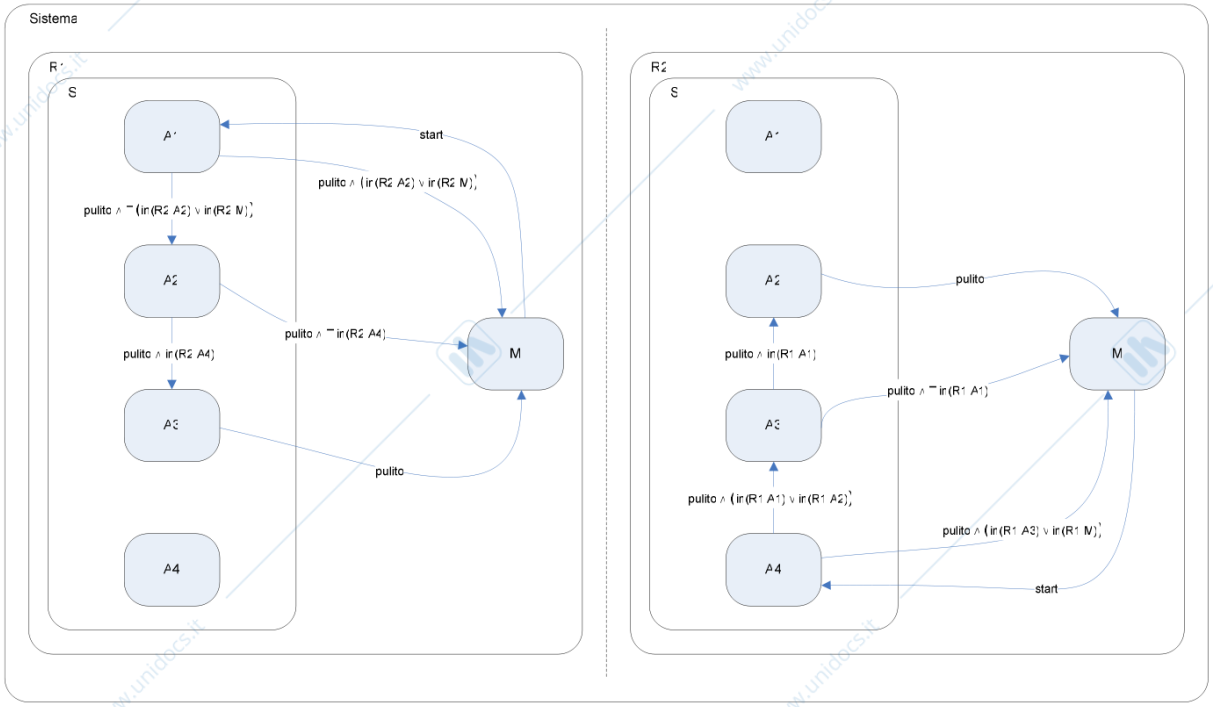
Formalizzare attraverso gli Statecharts il seguente problema. Si ha una catena di montaggio costituita da tre robot, ognuno dei quali dedicato ad un pezzo di uno specifico tipo, ma cooperanti nell'assemblaggio del prodotto finale. Ogni robot è collegato a sua volta a un macchinario che porta, tramite un sistema a nastro, i singoli pezzi al robot. L'obiettivo è coordinare i tre robot nell'assemblaggio del pezzo. Ogni robot i-esimo è schematizzabile considerando il suo stato (attendi_pezzo, prendi_pezzo, impila_pezzo) e lo stato del tipo di pezzo i-esimo portato dal sistema

a nastro al robot i-esimo (pezzo_presente, pezzo_mancante). Ogni volta che arriva un pezzo per il robot i- esimo, viene generato l'evento pezzo-i e lo stato del relativo tipo di pezzo passa da pezzo_mancante a pezzo_presente (il sistema è fatto in modo tale che un pezzo può arrivare solo se nessun pezzo di quel tipo è presente). Se il tipo di pezzo i-esimo è presente, allora il robot i-esimo può prenderlo, e conseguentemente il tipo di pezzo i-esimo diventa mancante. I pezzi dei tre robot devono essere impilati in un particolare ordine per poter assemblare il prodotto finale. Nello specifico, deve prima essere impilato il pezzo1, poi il pezzo2, e, infine, il pezzo3. Per poter coordinare in modo efficiente i tre robot, si consiglia di schematizzare il prodotto in vari stati, ad esempio (vuoto, se nessun pezzo è stato impilato, primo, se il pezzo1 è stato impilato, secondo, se il pezzo2 è stato impilato, e terzo, qualora tutti i pezzi siano stati impilati).

Esercizio 5.9

(10 punti)

Sia data la formalizzazione basata su Statecharts riportata in figura dove si descrive la coordinazione tra due robot pulitori (R1 e R2). Nello specifico: ci sono due robot uguali che alloggiano in un magazzino e che devono periodicamente pulire una stanza S. La stanza è divisa, ai fine del lavaggio, in quattro aeree (A1-A4) e i robot sono coordinati in modo tale che il robot R1 inizia a pulire la stanza a partire dall'area A1 e prosegue scendendo (rispetto alla figura) verso l'area A4 e il robot R2 inizia dall'area A2 e prosegue verso l'area A1. Gli unici eventi che i robot riconoscono sono start e pulito (generato quando l'area che il robot sta pulendo è diventata pulita). La coordinazione è tale che nessuna area che è stata pulita viene pulita nuovamente e che non ci possono essere due robot nella stessa area.



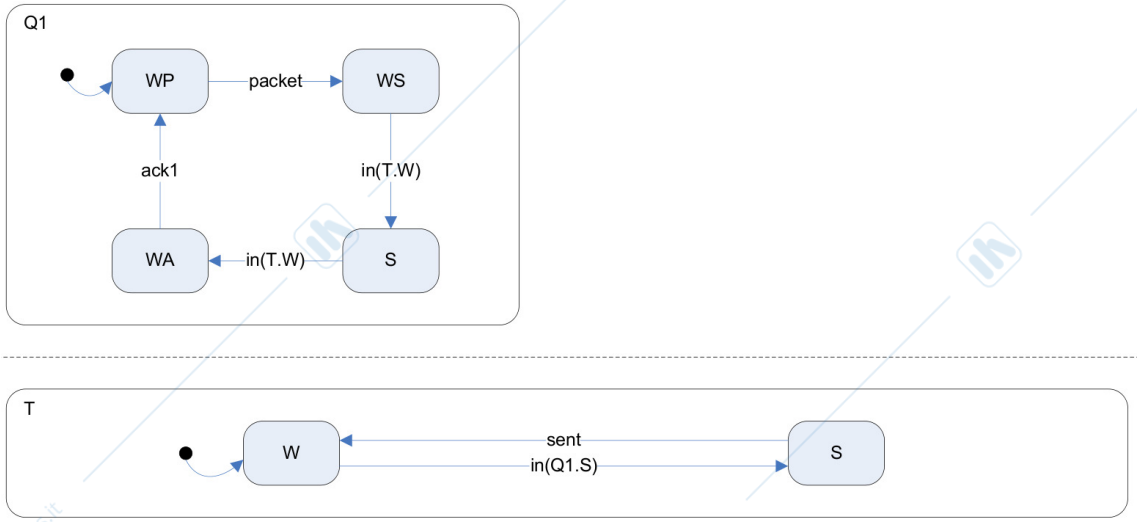
Si pone il problema che R1 ha un guasto al serbatoio d'acqua e non può immagazzinare la quantità d'acqua necessaria per pulire (si assume che la quantità immagazzinata dal serbatoio può non essere sufficiente neanche per pulire una singola area). Così R1 deve, quando necessario (cioè appena si presenta l'evento acqua_sporca), andare a cambiare l'acqua (in una stanza denominata L). Durante il cambio dell'acqua, R1 si accorgerà che l'acqua è pulita in quanto viene a generarsi l'evento acqua_pulita. E' permesso a R2, al fine di ridurre il tempo di lavaggio della stanza, di andare a pulire un'area che R1 ha cominciato a pulire, ma che non ha concluso di pulire in quanto è dovuto andare a cambiare l'acqua. Così, una volta che l'acqua è stata cambiata, R1 tornerà a pulire l'area che stava pulendo al momento in cui si è generato l'evento acqua_sporca oppure, qualora in quell'area sia presente R2, tornerà in magazzino. Si consiglia di tenere traccia dello stato di R1 prima di andare in L, usando uno stato concorrente con stati: A1, A2, A3, M. Formalizzare tale problema modificando la figura sopra aggiungendo gli stati necessari e modificando le condizioni di transizione.

Esercizio 5.10

(10 punti)

Si vuole specificare tramite il formalismo degli Statechart senza l'utilizzo di variabili la trasmissione di pacchetti TCP su rete da parte un singolo host. Il mezzo trasmissivo è costituito da un trasmettitore T che può essere in attesa (W - waiting) oppure in invio (S - sending). Il trasmettitore può trasmettere un solo pacchetto per volta e, qualora la trasmissione del singolo pacchetto si concluda, viene generato un evento sent. Il mezzo trasmissivo viene utilizzato con mutua esclusività da alcune code a priorità (Q – queue) il cui obiettivo è gestire l'invio di un pacchetto: recuperare il pacchetto, inviarlo al mezzo trasmissivo, aspettare il messaggio che confermi la corretta consegna del pacchetto e, qualora non arrivi tale messaggio, rinviare il pacchetto. In figura è fornito lo statechart che specifica la situazione in cui sia presente una sola coda Q1 e la rete sia affidabile. La coda si può trovare: in attesa dell'arrivo di un pacchetto (WP – waiting for a packet), in attesa di accedere al mezzo trasmissivo (WS – waiting for sending), in invio (S - sending) e in attesa del messaggio di corretta consegna del pacchetto (WA – waiting for acknoledgment). Ogni volta che almeno una coda (in questo caso solo Q1) è in WP ed è presente un pacchetto da inviare, viene generato un evento packet che viene comunicato a tutte le code. Nel caso ci sia una sola coda, il pacchetto viene automaticamente assegnato a Q1 e quindi questa transisce da WP a WS. Una volta in WS, la coda Q1 transisce in S qualora il mezzo trasmissivo sia disponile, cioè quando è in W. Una volta terminata la trasmissione il mezzo trasmissivo ritorna nello stato W e conseguentemente la coda transisce in WA. A fronte dell'evento ack1 la coda transisce da WA a WP.

Si richiede di estendere lo statechart dato per cogliere due aspetti: 1. la possibilità che un pacchetto non venga consegnato correttamente e venga quindi ritrasmesso e 2. la presenza di due code concorrenti.



1. Una volta inviato un pacchetto tramite la coda Q1 e non si abbia ricevuto il messaggio di conferma entro una scadenza prefissata, viene generato un evento timeout1. A fronte di timeout1 il pacchetto viene ritrasmesso. Il numero massimo di ritrasmissioni è 2, dopodichè la coda gestisce un altro pacchetto. Si utilizzi uno stato contatore C1 per tenere traccia del numero di trasmissioni di un pacchetto.

2. Si introduca una nuova coda Q2. Le due code funzionano in modo analogo in parallelo. Viene però data precedenza a Q1: se è disponile un pacchetto ed entrambe le code lo possono gestire viene preferita la coda Q1; analogamente se entrambe le due code sono in attesa di accedere al mezzo trasmissivo viene preferita la coda Q1.

Esercizio 5.11

(10 punti)

Si vuole specificare tramite il formalismo degli Statecharts il meccanismo con cui la tecnologia Ethernet gestisce l'accesso al mezzo trasmissivo. Si considerino tre host (denotati con H1, H2, H3) e il mezzo trasmissivo (denotato con medium). Gli host possono essere in attesa dell'arrivo di pacchetti (stato WP), in trasmissione sul mezzo per poter avere l'accesso (stato MA), in attesa di poter ritentare l'accesso al mezzo trasmissivo (W) e in trasmissione dei dati sul mezzo trasmissivo (PT). Per descrive il funzionamento dell'host si utilizza inoltre una coda di pacchetti capace di contenere al massimo tre pacchetti. Il mezzo trasmissivo può essere in uso da una coppia di host (blocked) o libero (free). Una volta che arriva un pacchetto, questo entra in coda: l'evento che segnala l'arrivo del pacchetto è incomingHi, dove i indica il numero dell'host. Qualora c'è almeno un pacchetto in coda, l'host passa da WP a MA, nel tentativo di accedere al mezzo trasmissivo. Qualora il mezzo trasmissivo sia libero l'host prende possesso di questo e trasmette (quindi passa a PT). Qualora invece il mezzo trasmissivo sia occupato l'host passa a W. La transizione da W a MA è attivata a fronte di un evento timeoutHi. Una volta cominciata la trasmissione del pacchetto, questo viene tolto dalla coda. Finita la trasmissione (evento sentHi) lo stato dell'host può passare

in MA qualora ci siano altri pacchetti in coda, oppure in WP qualora non ci siano pacchetti in coda. Il mezzo trasmissivo transisce da free a blocked quando almeno un host passa a PT e da blocked a free quando nessun host è in PT. Riportare lo statechart di un generico host i e del mezzo trasmissivo.

Esercizio 5.12
(10 punti)

Si vuole specificare tramite il formalismo degli Statecharts, e senza l'utilizzo di variabili, il funzionamento di una macchina distributrice di merendine. La macchina contiene due tipi di merendine: merendina1 e merendina2. Per ogni tipo di merendina la macchina contiene al massimo 2 merendine. Il numero di merendine scende ogni volta che una merendina viene comprata (si veda sotto per le condizioni da considerare) e ritorna al massimo ogni volta che viene ricaricata (evento ricarica). Inizialmente si suppone che il numero di merendine sia pari al massimo. Ogni utente è munito di una chiave magnetica contenente il proprio credito. La chiave può essere inserita (stato inserita) o non inserita (stato non_inserita) nella macchina distributrice. Nella chiave è presente il credito. Questo può essere al massimo di 5 euro e scende ogni qual volta si compra una merendina del costo della merendina comprata (si veda sotto per le condizioni da considerare) e sale di un euro ogni qualvolta la chiave è inserita e si verifica l'evento 1euro. La macchina distributrice può essere in tre soli stati: attesa, merendina1, merendina2. In attesa attende che l'utente scelga la merendina da comprare. In merendina1 la macchina fa prendere all'utente una merendina1. La transizione da attesa a merendina1 è possibile se: la chiave è inserita, l'utente ha premuto il pulsante 1 (evento pulsante1), esiste almeno una merendina1 nella macchina e nella chiave è presente un credito sufficiente per acquistare la merendina1. La merendina1 ha un costo pari a 2 euro. La transizione da merendina1 ad attesa avviene in automatico. In modo analogo vengono gestite le transizioni tra attesa e merendina2, in questo caso però il costo di una merendina2 è di 1 euro.

Esercizio 5.13
(10 punti)

Si vuole specificare tramite il formalismo degli Statecharts, e senza l'utilizzo di variabili, il funzionamento di un forno a microonde. Il forno può essere aperto o chiuso (gli eventi apri e chiudi regolano le corrispettive transizioni). Quando è chiuso, il forno può essere: in funzione, non in funzione, oppure può trovarsi in uno stato (beep) in cui allerta l'utente del forno emettendo un suono. Il sistema di controllo del forno si basa principalmente sulla selezione del programma e sul timer. I pulsanti utilizzabili dall'utente sono: start, reset e set. La particolarità del sistema di controllo è che, una volta selezionato un programma di cottura e fatto partire il forno, il programma non può essere cambiato se non tramite il pulsante reset (ciononostante la cottura può essere sospesa aprendo il forno). Ci sono due possibili programmi (4 e 8) e un programma fittizio (0) che identifica il fatto che non è stato selezionato alcun programma. Il nome del programma fa riferimento alla durata della cottura, ad esempio col programma 4 la cottura dura 4 minuti. E' possibile selezionare

un programma tramite l'uso del pulsante set solo se il forno non è in funzione, è chiuso, e non ci sono programmi di cottura da terminare. L'uso del pulsate set è ciclico: se premuto una volta permette di selezionare il programma 4, se premuto una seconda volta permette di selezionare il programma 8, se premuta un'altra volta il programma 0, e così via. Premendo il pulsante start, qualora sia stato impostato un programma diverso da 0, il forno va in funzione. Per poter gestire una cottura si modellizzi il timer: può essere spento oppure in uno stato da 0 a 8 (si considerino solo i pari). La transizione da 0 a spento è automatica; le transizioni da 8 a 6, da 6 a 4, e così via sono regolate da un evento timer (questo evento si attiva automaticamente ogni 2 minuti di cottura effettiva). Una volta selezionato un programma e attivato il forno, il timer passa da spento allo stato corrispondente al programma. Via via che passa il tempo (evento timer) lo stato del timer viene aggiornato. Si vuole che a metà cottura e a fine cottura il forno passi dallo stato in funzione allo stato beep e successivamente allo stato non in funzione (questa ultima transizione avviene in automatico). La sospensione a metà cottura ha la finalità di permettere all'utente di "girare" l'alimento da cuocere. Infine, se premuto il tasto reset, indipendentemente dallo stato in cui si trova il forno, tutte le funzionalità del forno vengono azzerate.

Esercizio 5.14

(10 punti)

Si vuole specificare tramite il formalismo degli Statecharts, e senza l'utilizzo di variabili, l'uso di due risorse (A e B) da parte di due processi periodici (1 e 2). Un generico processo i può trovarsi nello stato di ready, execution o wait, in quest'ultimo si riconoscono due sottostati: waitingA e waitingB (corrispondenti all'attesa della risorsa A e B rispettivamente). Si presuppone che un processo non possa aspettare di accedere contestualmente a più di una risorsa (un processo può però utilizzare entrambe le risorse). Gli eventi che guidano le transizioni da execution a ready sono generati dallo scheduler e sono: dispatch_i e preempt_i. Ogni qual volta un processo i è in execution e richiede una risorsa X, viene generato un evento i.requireX. Qualora la risorsa è occupata (vedi sotto) il processo passa a waitingX. È necessario modellizzare lo stato delle risorse e l'uso delle risorse da parte di ogni singolo processo. Una risorsa X può trovarsi o libera (free) o occupata (hold). Le transizioni da free a hold sono guidate dagli eventi i.requireX e i.freeX (quet'ultimo evento viene generato quando un processo i in esecuzione libera la risorsa X). L'uso delle risorse da parte di un processo i viene modellizzata con uno statechart costituito da un numero di stati pari al numero di combinazioni di risorse utilizzate dal processo.

Esercizio 5.15

(10 punti)

Si vuole specificare tramite il formalismo degli Statecharts il funzionamento di un ascensore in un palazzo in Israele, dove al sabato l'ascensore si muove autonomamente senza nessun intervento da parte degli utenti. L'ascensore può essere fermo o in movimento (le condizioni che guidano le transizioni tra questi stati sono descritti in seguito). Inoltre, l'ascensore può trovarsi su 3 piani (1, 2, 3)

più il piano terra (0). È presente inoltre un sistema di impostazione del piano ad uso dell'utente. L'utente può, quando l'ascensore è fermo, selezionare tramite il pulsante set (da considerare come evento) il piano desiderato. Precisamente, il pulsante set permette di passare ciclicamente da 0 a 1, da 1 a 2, da 2 a 3, e infine da 3 a 0. Una volta impostato il piano desiderato e premuto il tasto start (da considerare come evento), se l'ascensore non si trova al piano impostato, allora va in movimento. La transizione da un piano all'altro è possibile solo se l'ascensore è in movimento. In questo caso l'ascensore si muoverà ovviamente verso il piano impostato.

Si consideri ora il problema di modificare il sistema di controllo per permettere di gestire il sabato. Il funzionamento dell'ascensore in questo caso è: si muove piano per piano in modo tale che, una volta che l'ascensore ha raggiunto un piano e si è fermato, allora autonomamente il sistema di impostazione seleziona il prossimo piano in cui andare. Qualora sia a 0 seleziona 1, qualora sia a 1 seleziona 2, fino a 3, a quel punto seleziona 2 e riscende piano per piano fino a 0. Si introduca un macrostato giorno per gestire il giorno della settimana dove un evento (Event_day) fa cambiare il giorno della settimana. Si introduca un macrostato direzione che descriva la direzione dell'ascensore (Up e Down) e tale che quando in discesa l'ascensore raggiunge il piano 0 la direzione diventa Up e quando invece in salita raggiunge il piano 3 la direzione diventa Down. Al sabato l'ascensore passa da fermo a in movimento appena il piano impostato è diverso da quello corrente. Al sabato inoltre il pulsante set e il pulsante start non hanno effetto.

Esercizio 5.16

(10 punti)

Si vuole modellizzare tramite il formalismo degli StateCharts il funzionamento di un distributore di videocassette. Ogni utente registrato ha una tessera, questa può essere in oppure out. Gli eventi che attivano queste due transizioni sono inserisci (da out a in) ed estrai (da in a out). Nella transizione da in ad out è necessario inoltre che non ci sia alcuna transazione in atto (vedi sotto). Ogni tessera ha caricato un certo credito che può essere 0,1,2,3. La transizione verso un credito maggiore è attivata dall'evento gettone e dal fatto che la tessera sia inserita. La transizione verso un credito negativo è attivata dal verificarsi di una transazione. Il distributore può contenere diversi film e per ognuno di questi più copie. Si supponga che per ogni film possa contenere un massimo di tre copie. La transizione verso un numero maggiore di copie è attivata dall'evento restituito. Si specifichino gli stati relativi a un film generico i, dove i è un parametro. Il distributore offre una interfaccia la cui schermata principale è la home. Da questa un utente, anche senza aver inserito la propria tessera, può passare alla schermata scelta (evento scegli) da cui scegliere il film da prendere. Una volta scelto un film i (evento sel_film_i) si possono verificare alcuni casi: se non ci sono copie disponibili non avvengono transizioni, se la tessera non è inserita viene visualizzata la schermata in cui si richiede di inserire la tessera e si ritorna a scelta, se il credito non è sufficiente viene visualizzata la schermata dicendo che il credito è insufficiente e si ritorna a scelta, altrimenti viene

attivata la transazione e si ritorna alla home. In un qualunque istante di tempo venga estratta la tessera, si ritorna alla home.

Esercizio 5.17

(10 punti)

Si vuole modellare tramite il formalismo degli StateCharts il coordinamento di un gruppo di meccanici durante il pit-stop di un'auto da corsa. Un'auto può essere ferma o in_movimento (l'evento pit_stop attiva la transizione da in_movimento a ferma, la transizione inversa è attivata quando tutte le gomme sono state cambiate). Una volta che una macchina si ferma, per ogni gomma si attiva un meccanico (meccanicoi) che prima svita e poi avvita la gomma. Una volta svitata la gomma, un ulteriore meccanico (il cui comportamento non è da modellare) si occupa di rimuovere la gomma che è stata svitata e montare la gomma nuova. Ogni gomma può trovarsi in: da_svitare, da_rimuovere, da_avvitare, pronta. Le transizioni sono attivate dai seguenti eventi: svitata, rimossa, avvitata. La particolarità del problema è che ogni meccanico addetto a svitare/avvitare una specifica gomma può aiutare i meccanici addetti a svitare/avvitare le altre gomme. (L'assunzione è che meccanici diversi possono impiegare tempi diversi per avvitare/svitare e rimuovere le gomme.) Nello specifico: un meccanico può trovarsi in svita, aspetta, e avvita. In aspetta il meccanico sta aspettando che la propria gomma venga rimossa oppure ha finito di avvitare la propria gomma e sta aspettando che tutte le altre gomme siano state cambiate. In questo stato il meccanicoi può aiutare il meccanicoj che in quell'istante sta svitando o avvitando la propria gomma j (indicare gli stati come: aiutaj, aiutak, aiutal, dove j, k, e l sono indici). Inoltre, se il meccanicoi sta aiutando un altro meccanico e la gomma i è stata rimossa, allora il meccanicoi va subito ad avvitare quest'ultima gomma, indipendentemente da cosa stia facendo. La macchina ripartirà quando tutte le gomme sono state cambiate.