Verifica dei Programmi Concorrenti

Università degli Studi di Torino

Dipartimento di Informatica

Esercizio 2: Reti di Petri

Anno Accademico 2017/2018

Studente:
Matteo Marsala

1 Rete A:

Due master identici e due slave di tipo 1 identici.

1.1 La rete di Petri:

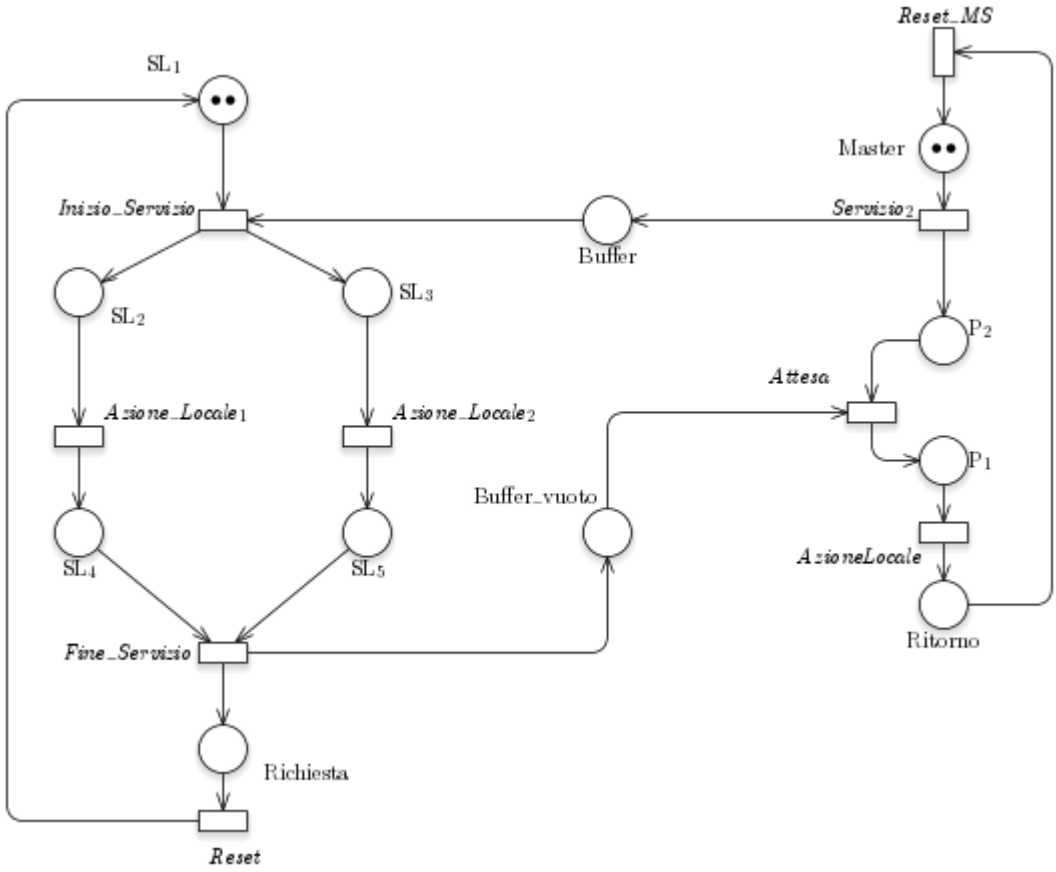

Figure 1: Rete di Petri *A*

La figura 1 rappresenta la rete di Petri P/T di nome *A*, del secondo esercizio. Il master è modellato dai posti *Master, P2, P1, Ritorno* e dalle transizioni *Servizio2, Attesa, AzioneLocale, Reset_MS*.
Gli slave sono modellati dai posti *SL1, SL2, SL2, SL4, SL5, Buffer* e dalle transizioni *Inizio_Servizio, Azione_Locale1, Azione_Locale2, Fine_Servizio, Reset*.

La richiesta del servizio da parte dei master è gestita attraverso un buffer in ingresso di nome *Richiesta_Servizio* allo slave $SL1$ e un buffer in uscita di nome *Buffer_Vuoto*.

1.2 Lo svolgimento:

In quanto primo esercizio, la rete è stata semplice e non si sono riscontrate difficoltà.

1.3 I risultati:

La tabella 1 elenca la dimensione dello spazio degli stati al variare del numero di M master e S slave. Osserviamo come, al variare del numero di token, il numero di stati del grafo di raggiungibilità aumenta in modo esponenziale.

Dipendenza dalla marcatura iniziale		
Numero di M e S	Stati	Archi
1	14	19
2	94	222
3	426	1334
4	1500	5610
5	4422	18720
6	11418	52998
7	26598	132594
8	57057	301158
9	114400	632775
10	216788	1246960

Table 1: Variazione dello spazio degli stati.

2 Rete B:

Due master identici e due slave, uno di tipo 1 e uno di tipo 2. Ad ogni ciclo il master sceglie in modo indipendente di quale dei due slave servirsi.

2.1 La rete di Petri:

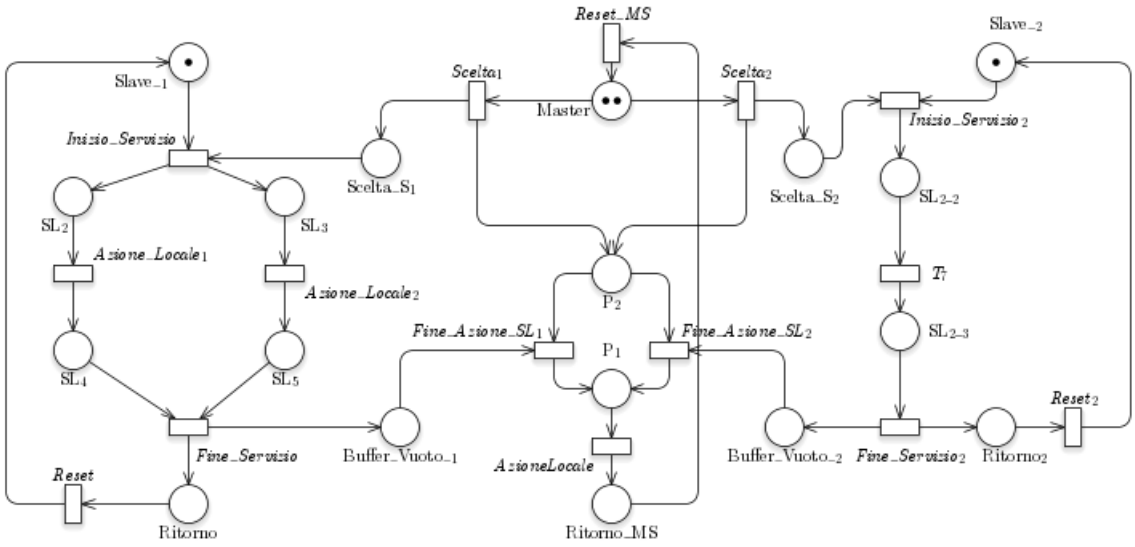

Figure 2: Rete di Petri *B*

La figura 2 rappresenta la rete di Petri P/T di nome *B*, del secondo esercizio. I master sono modellati dai posti *Master, P2, P1, Ritorno_MS* e dalle transizioni *Fine_Azione _SL1, Fine_Azione _SL2, AzioneLocale, Reset_MS*.
Lo slave1 è modellato dai posti *Slave_1, SL2, SL2, SL4, SL5, Ritorno* e dalle transizioni *Inizio_Servizio, Azione_Locale1, Azione_Locale2, Fine_Servizio, Reset*.
Lo slave 2 è modellato dai posti *Slave_2, SL2_2, SL2_3, Ritorno2* e dalle transizioni *Inizio_Servizio2, T7, Fine_Servizio2, Reset2*.
Il Master può scegliere lo slave 1 tramite la transizione *Scelta1* e il posto *Scelta_S1*, oppure può scegliere lo slave 2 tramite la transizione *Scelta2* e il posto *Scelta_S2*.
La fine del servizio dello slave1 è gestita tramite il posto *Buffer_Vuoto_1* e la fine del servizio dello slave2 è gestita dal posto *Buffer_Vuoto_2*.

2.2 Lo svolgimento:

In quanto primo esercizio, la rete è stata semplice e non si sono riscontrate difficoltà.

2.3 I risultati:

La tabella 2 elenca la dimensione dello spazio degli stati al variare del numero di M master e S slave. Osserviamo come, al variare del numero di token, il numero di stati del grafo di raggiungibilità aumenta in modo esponenziale.

Dipendenza dalla marcatura iniziale		
Numero di M e S	Stati	Archi
1	40	76
2	572	1832
3	4796	20224
4	28651	144662
5	134848	775656
6	530880	3374798
7	1817776	12514656
8	5561958	40878472
9	15509168	120399476
10	39997736	325317848

Table 2: Variazione dello spazio degli stati.

3 Rete C:

Due master distinti (seppur di uguale struttura). Ad ogni ciclo il master sceglie in modo indipendente di quale dei due slave servirsi.

3.1 La rete di Petri:

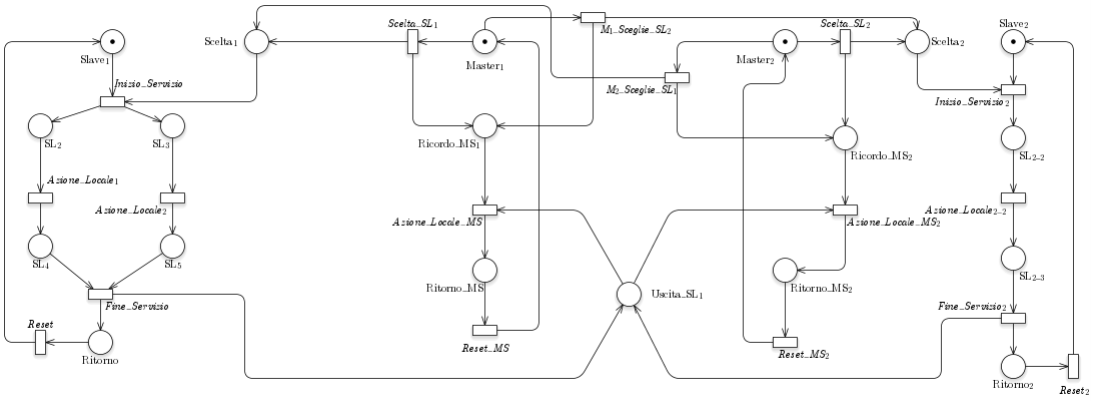

Figure 3: Rete di Petri *C*

La figura 3 rappresenta la rete di Petri P/T di nome *C*, del secondo esercizio. Il master 1 è modellato dai posti *Master1, Ricordo_MS1, Ritorno_MS* e dalle transizioni *Azione_Locale_MS, Reset_MS*.
Il master 2 è modellato dai posti *Master2, Ricordo_MS2, Ritorno_MS2* e dalle transizioni *Azione_Locale_MS2, Reset_MS2*.
Lo slave1 è modellato dai posti *Slave_1, SL2, SL2, SL4, SL5, Ritorno* e dalle transizioni *Inizio_Servizio, Azione_Locale1, Azione_Locale2, Fine_Servizio, Reset*.
Lo slave 2 è modellato dai posti *Slave_2, SL2_2, SL2_3, Ritorno2* e dalle transizioni *Inizio_Servizio2, Azione_Locale2_2, Fine_Servizio2, Reset2*.
Il master 1 può scegliere lo slave 1 tramite la transizione *Scelta_SL1* e il posto *Scelta1*, oppure può scegliere lo slave 2 tramite la transizione *M1_Sceglie _SL2* e il posto *Scelta2*.
Il master 2 può scegliere lo slave 1 tramite la transizione *M2_Sceglie_SL1* e il posto *Scelta1*, oppure può scegliere lo slave 2 tramite la transizione *Scelta _SL2* e il posto *Scelta2*.
La fine del servizio degli slave è gestita tramite il posto *Uscita_SL1*.

3.2 Lo svolgimento:

La rete è abbastanza semplice da costruire, ad esclusione delle uscite dei master. Infatti, inizialmente avevo considerato una duplice uscita per gli slave 1 e 2, rendendomi conto grazie al token game, che questa scelta avrebbe portato erroneamente al deadlock.

3.3 I risultati:

La tabella 3 elenca la dimensione dello spazio degli stati al variare del numero di M master e S slave. Osserviamo come, al variare del numero di token, il numero di stati del grafo di raggiungibilità aumenta in modo esponenziale.

Dipendenza dalla marcatura iniziale		
Numero di M e S	Stati	Archi
1	224	624
2	9435	45498
3	170000	1077208
4	1817986	13651612
5	13607904	114933168
6	78471206	722931404

Table 3: Variazione dello spazio degli stati.

4 Rete D:

Due master distinti (seppur di uguale struttura) con uno slave associato al singolo master (il primo master usa sempre lo slave di tipo 1 e il secondo usa sempre quello di tipo 2).

4.1 La rete di Petri:

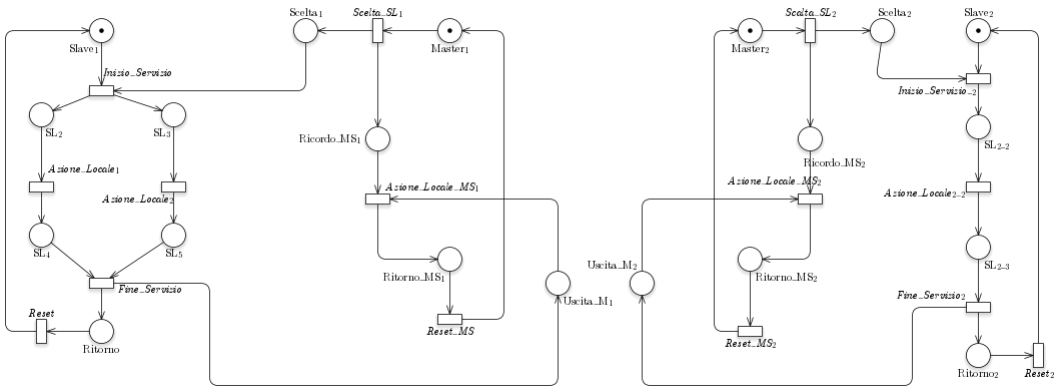

Figure 4: Rete di Petri *D*

La figura 4 rappresenta la rete di Petri P/T di nome *D*, del secondo esercizio. Il master 1 è modellato dai posti *Master1, Ricordo_MS1, Ritorno_MS1* e dalle transizioni *Azione_Locale_MS1, Reset_MS*.
Il master 2 è modellato dai posti *Master2, Ricordo_MS2, Ritorno_MS2* e dalle transizioni *Azione_Locale_MS2, Reset_MS2*.
Lo slave1 è modellato dai posti *Slave_1, SL2, SL2, SL4, SL5, Ritorno* e dalle transizioni *Inizio_Servizio, Azione_Locale1, Azione_Locale2, Fine_Servizio, Reset*.
Lo slave 2 è modellato dai posti *Slave_2, SL2_2, SL2_3, Ritorno2* e dalle transizioni *Inizio_Servizio2, Azione_Locale2_2, Fine_Servizio2, Reset2*.
Il master 1 può scegliere lo slave 1 tramite la transizione *Scelta_SL1* e il posto *Scelta1*. Seguendo le direttive dell'esercizio, questa è l'unica scelta che può fare.
Il master 2 può scegliere lo slave 2 tramite la transizione *Scelta_SL2* e il posto *Scelta2*. Seguendo le direttive dell'esercizio, questa è l'unica scelta che può

fare.
La fine del servizio dello slave1 è gestita tramite il posto *Uscita_M1* e la fine del servizio dello slave2 è gestita dal posto *Uscita_M2*.

4.2 Lo svolgimento:

In quanto molto simile alla rete *C*, la rete è stata semplice e non si sono riscontrate difficoltà.

4.3 I risultati:

La tabella 4 elenca la dimensione dello spazio degli stati al variare del numero di M master e S slave. Osserviamo come, al variare dei numeri di token, il numero di stati del grafo di raggiungibilità aumenta in modo esponenziale.

Dipendenza dalla marcatura iniziale		
Numero di M e S	Stati	Archi
1	120	316
2	3479	15562
3	48384	282240
4	424116	2918160
5	2699424	20818224
6	13568940	113823864

Table 4: Variazione dello spazio degli stati.

5 Domande sulle reti PT:

Sperimentate la dipendenza dalla marcatura iniziale provando ad aumentare il numero di master e di slave delle reti. Sino a che numero di master e di slave potete arrivare con la memoria a vostra disposizione (indicare quanta [free -h]) e tempi di esecuzione che non superino i 10 minuti?

Dipendenza dalla marcatura iniziale - Memoria: 2GB		
	Master	Slave
Rete A	16	16
Rete B	7	7
Rete C	4	4
Rete D	5	5

Table 5: Dati dell'esercizio precedente.

Interessante come, nella rete C, con 3 Master e 3 Slave, il grafo sia stato costuito in 16 secondi. Con 4 Master e 4 Slave in 259 secondi. Invece, con 5 Master e 5 Slave, il tempo è stato superiore ai 15 minuti.

In quali casi siamo certi che il join avvenga fra due sottoprocessi creati dalla stessa fork? E, in caso di risposta negativa, come potremmo fare in modo che questo succeda? In tutte le reti il join avviene sempre tra due sottoprocessi creati dalla stessa rete, ad esclusione della Rete C, poichè se i due master scelgono lo stesso slave, non si sa quale dei due ha finito.

6 Rete E:

Master di tre tipi (seppur di uguale struttura) che chiamiamo m1, m2, m3 e due slave (uno di struttura 1 e uno di struttura 2). Ad ogni ciclo il master sceglie in modo indipendente di quale dei due slave servirsi. Vi siano N master per ognuno dei tre tipi e R1 slave per la struttura 1 e R2 per la struttura 2.

6.1 La rete di Petri colorata:

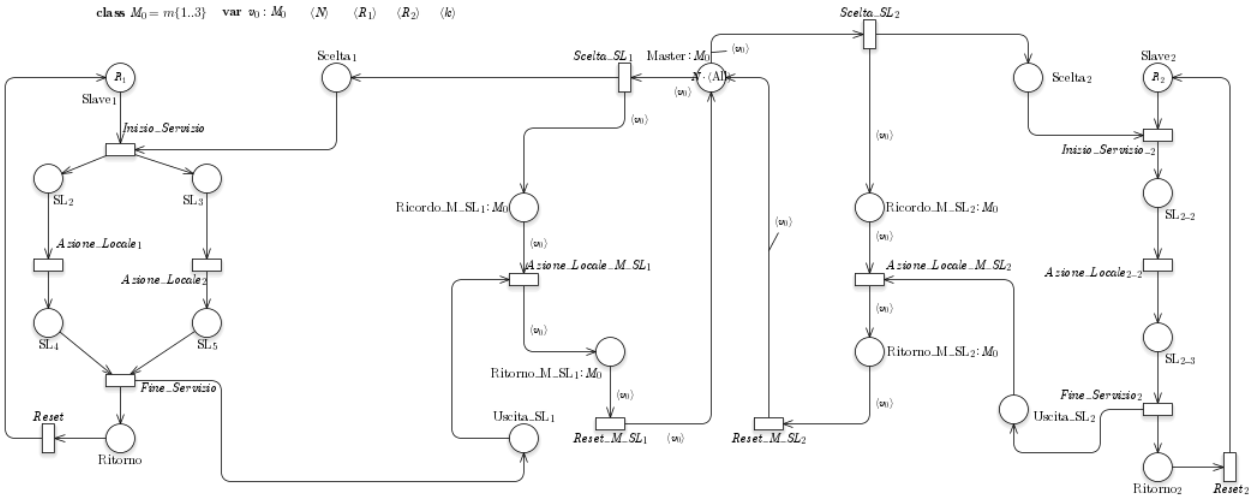

Figure 5: Rete di Petri *E*

La figura 5 rappresenta la rete di Petri P/T di nome *E*, del secondo esercizio. I master sono modellati dai posti *Master, Ricordo_M_SL1, Ritorno_M_SL1* e dalle transizioni *Azione_Locale_M_SL1, Reset_M_SL1.*
Lo slave1 è modellato dai posti *Slave_1, SL2, SL2, SL4, SL5, Ritorno* e dalle transizioni *Inizio_Servizio, Azione_Locale1, Azione_Locale2, Fine_Servizio, Reset.*
Lo slave 2 è modellato dai posti *Slave_2, SL2_2, SL2_3, Ritorno2* e dalle transizioni *Inizio_Servizio2, Azione_Locale2_2, Fine_Servizio2, Reset2.*
I master possono scegliere lo slave 1 tramite la transizione *Scelta_SL1* e il posto *Scelta1*, oppure possono scegliere lo slave 2 tramite la transizione *Scelta_SL2* e il posto *Scelta2.*
La fine del servizio dello slave1 è gestita tramite il posto *Uscita_SL1* e la fine

del servizio dello slave2 è gestita dal posto *Uscita_SL2*. Vi sono delle transizioni per ricordarsi quale slave ha scelto un determinato master, in modo che esso attenda la fine del servizio dello slave, prima di proseguire.

Ho scelto un'unica classe di colore poichè, da consegna, abbiamo 3 tipi di master diversi. Questa classe, denominata *M0*, va quindi a descrivere tali master. Inoltre, di tale classe esiste una variabile *v0*. Tutta la struttura di ricordo, che permette al master di attendere che lo slave scelto finisca il suo servizio, ha un dominio di colore di classe *M0*, in modo da far tornare allo stato iniziale sempre e solo i master che ne sono usciti, senza scegliere il colore liberamente. Le strutture degli slave, tuttavia, non hanno colore, poichè non necessario per il corretto funzionamento dell'esercizio. Infatti, ogni slave ha un unico tipo, per cui non è necessario usare un colore.

6.2 Lo svolgimento:

Nella rete colorata, i colori non hanno dato problemi. Inizialmente la rete è stata realizzata con una sola uscita, poichè, non portando al deadlock, sembrava una buona scelta per risparmiare un posto. Tuttavia, i master non tornavano sempre al loro posto iniziale, e grazie al token game è stato trovato un caso in cui il master 1 poteva completare la sua esecuzione grazie al token prelevato dal master 2 (e viceversa), cambiando quindi il master 1 con il master 2. Per cui, ho diviso le due uscite in modo da eliminare il problema.

6.3 I risultati:

La tabella 6 elenca la dimensione dello spazio degli stati al variare del numero di M master e S slave. Osserviamo come, al variare del numero di token, il numero di stati del grafo di raggiungibilità ordinario aumenta in modo esponenziale.

La tabella 7 elenca la dimensione dello spazio degli stati al variare del numero di M master e S slave. Osserviamo come, al variare di token, il numero di stati del grafo di raggiungibilità simbolico aumenta in modo esponenziale.

Confrontando i dati del grafo ordinario e quello simbolico, possiamo dire che l'aumento del primo è di tipo esponenziale, mentre per quanto riguarda il grafo simbolico abbiamo un aumento sempre esponenziale, ma di vari gradi più basso. Inoltre, è da notare come, nel grafo simbolico, con Master e Slave pari a 3, l'esecuzione sia comunque superiore ai 30 minuti.
Notiamo inoltre, come il grafo simbolico sia più piccolo, rispetto all'ordinario.

Dipendenza dalla marcatura iniziale		
Numero di M e S	Stati	K - Tipi di master
1	2720	40
2	752832	348
3	>30 min	2720
4	>30 min	19700
5	>30 min	1350000
6	>30 min	887500

Table 6: Variazione dello spazio degli stati - grafo ordinario.

Dipendenza dalla marcatura iniziale		
Numero di M e S	Stati	K - Tipi di master
1	728	40
2	151092	204
3	>30 min	728
4	>30 min	2072
5	>30 min	5040
6	>30 min	10920

Table 7: Variazione dello spazio degli stati - grafo simbolico.

Passando invece al variare del numero di tipi di master (con gli altri parametri pari a 1), vediamo come il numero di stati aumenta in modo esponenziale se usiamo un grafo ordinario, quadratico per un grafo simbolico.

Sotto quali condizioni, per i valori dei parametri, il join avviene tra due sotto processi creati dalla stessa fork? Solo nel caso in cui i master effettuino la scelta sullo stesso slave uno alla volta. Se tutti i master (o anche solo 2) scelgono lo stesso slave. quando quest'ultimo avrà finito, non si sa quale master dovrà terminare la sua esecuzione. Lo stesso vale per il secondo slave: se un master sceglie lo slave 1 e un'altro master sceglie lo slave 2, il terzo master deve attendere che gli altri due finiscano, o il join potrebbe non avvenire tra sottoprocessi creati dalla stessa fork. Inoltre, nel caso in cui ci siano più Master e Slave dello stesso tipo, il join tra due sotto processi creati dalla stessa fork non può essere assicurato. Con un solo master per tipo (N

= 1) il join avverrà sempre tra due sotto processi creati dalla stessa fork.

7 Rete F:

Master di tre tipi distinti (seppur di uguale struttura) che chiameremo m1, m2, m3 e due slave, di distinto tipo ID1 e ID2, ma di uguale struttura (useremo quella di tipo 1). I master di tipo m1 e m3 richiedono servizio solo agli slave di tipo ID1 e i master di tipo m2 richiedono servizio solo agli slave di tipo ID2. Vi siano N master per ognuno dei tre tipi e R1 slave per il tipo ID1 e R2 per il tipo ID2.

7.1 La rete di Petri colorata:

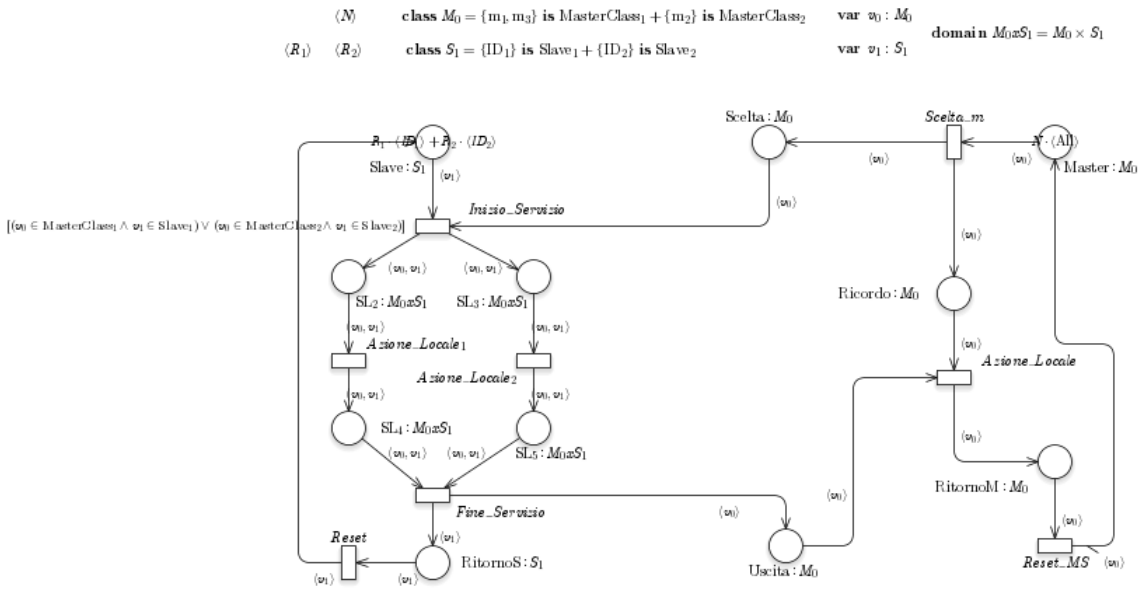

Figure 6: Rete di Petri F

La figura 6 rappresenta la rete di Petri P/T di nome F, del secondo esercizio. I master sono modellati dai posti *Master, Ricordo, RitornoM* e dalle transizioni *Azione_Locale, Reset_MS*.
Gli slave sono modellati dai posti *Slave_, SL2, SL2, SL4, SL5, RitornoS* e dalle transizioni *Inizio_Servizio, Azione_Locale1, Azione_Locale2, Fine_Servizio, Reset*.
I master possono scegliere gli slave tramite la transizione *Scelta_m* e il posto *Scelta*.

La fine del servizio degli slave è gestita tramite il posto *Uscita*.
Anche qui troviamo una struttura di ricordo per il master, utile per sapere quale slave ha scelto un determinato master, in modo che esso attenda la fine del servizio dello slave, prima di proseguire.
Ho scelto due classi di colore: una per i master, di nome *M0* e una per gli slave, di nome *S1*. La classe *M0* gestisce i tre diversi tipi di master ed è divisa in due sottoclassi, che serviranno poi per impostare la guardia della transizione *Inizio_Servizio*. Invece, la classe *S1* gestisce i due diversi tipi di slave. Anch'essa è formata da due sottoclassi.
Tutta la struttura di ricordo, che permette al master di attendere che lo slave scelto finisca il suo servizio, ha un dominio di colore di classe *M0*, in modo da far tornare allo stato iniziale sempre e solo i master che ne sono usciti, senza scegliere il colore liberamente. La struttura degli slave ha invece un dominio di colore $M0xS1$. Finito il servizio, l'uscita prenderà il dominio della classe *M0* e il reset degli slave prenderà il dominio della classe *S1*.
Occorre fare riferimento alla transizione *Inizio_Servizio*, poichè è l'unica a usare una **guardia**. Questo serve per completare correttamente l'esercizio, in modo che i master *m1* e *m3* scelgano solo uno degli slave di tipo *ID1* e i master di tipo *m2* scelgano uno slave di tipo *ID2*. Tale guardia infatti, specifica questo vincolo, grazie alle sottoclassi. // Questa guardia poteva essere costruita anche senza l'utilizzo delle sottoclassi, ma poi avremmo avuto problemi con la costruzione di uno dei grafi utili per l'analisi.

7.2 Lo svolgimento:

Essendo la rete finale, l'esercizio è risultato più difficile a causa della sintassi della guardia. Infatti, ho notato che il tool usato non consente l'uso del connettivo OR dentro una parentesi annidata. Infatti, inizialmente, la guardia era impostata come segue:

$$(ID1 \wedge (m1 \vee m3)) \vee (ID2 \wedge m2)$$

L'errore era infatti segnalato nell'OR interno, nella prima parentesi. In ogni caso, ho dovuto inserire le sottoclassi per risolvere il problema del grafo, eliminando il problema.
Inoltre, sempre parlando della guardia, inizialmente non ero riuscito a capire che nel codice della transizione, la guardia non voleva valori delle classi ma direttamente la sottoclasse. Per cui i grafi davano errori.

7.3 I risultati:

La tabella 8 elenca la dimensione dello spazio degli stati al variare del numero di M master e S slave. Osserviamo come, al variare del numero di token, il numero di stati del grafo di raggiungibilità ordinario aumenta in modo esponenziale.

Dipendenza dalla marcatura iniziale		
Numero di M e S	Stati	K - Tipi di master
1	768	144
2	97696	1296
3	>30 min	6400
4	>30 min	22500
5	>30 min	63504
6	>30 min	153664

Table 8: Variazione dello spazio degli stati - grafo ordinario.

La tabella 9 elenca la dimensione dello spazio degli stati al variare del numero di M master e S slave. Osserviamo come, al variare del numero di token, il numero di stati del grafo di raggiungibilità simbolico aumenta in modo esponenziale.

Dipendenza dalla marcatura iniziale		
Numero di M e S	Stati	K - Tipi di master
1	432	144
2	50481	1296
3	>30 min	6400
4	>30 min	22500
5	>30 min	63504
6	>30 min	153664

Table 9: Variazione dello spazio degli stati - grafo simbolico.

Confrontando i dati del grafo ordinario e quello simbolico, possiamo dire che l'aumento del primo è di tipo esponenziale, mentre per quanto riguarda il grafo simbolico abbiamo un aumento sempre esponenziale, ma di vari gradi

più basso. Inoltre, è da notare come, nel grafo simbolico, con Master e Slave pari a 3, l'esecuzione sia comunque superiore ai 30 minuti.
Notiamo inoltre, come il grafo simbolico sia più piccolo, rispetto all'ordinario. Passando invece al variare del numero di tipi di master (con gli altri parametri pari a 1), vediamo come il numero di stati aumenta in modo esponenziale se usiamo un grafo ordinario, quadratico per un grafo simbolico.

Sotto quali condizioni, per i valori dei parametri, il join avviene tra due sotto processi creati dalla stessa fork? Nel caso in cui ci siano più Master e Slave dello stesso tipo, il join tra due sotto processi creati dalla stessa fork non può essere assicurato. Con un solo master per tipo (N = 1) e un solo slave per tipo (R1 o R2 = 1), il join avverrà sempre tra due sotto processi creati dalla stessa fork.