Esercizio P/N

Francesco Mecca

May 22, 2020

1 Rete A

M master identici e S slave identici di tipo 1.

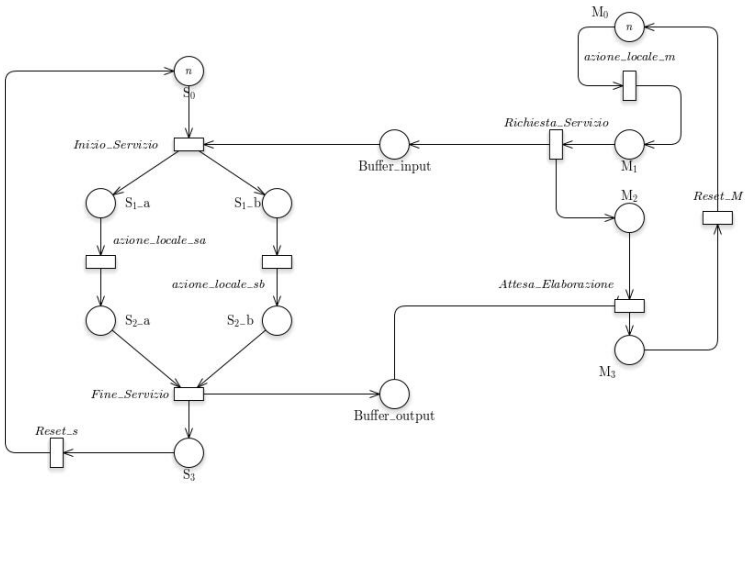


Figure 1: Modello della reteA

La figura rappresenta la rete di Petri P/T dell'esercizio A. Il master è modellato dai posti M0, M1, M2, M3 e dalle transizioni *Azione_Locale*, *Richiesta_Servizio*, *Attesa_Elaborazione* e *Reset_M* Lo slave è modellato dai posti *S0*, *S1_a*, *S1_b*, *S2_a*, *S2_b* e *S3* e dalle transizioni *Inizio_Servizio*,

Azione_Locale_S_a, *Azione_Locale_S_b*, *Fine_Servizio* e *Reset_S*. La richiesta del servizio verso lo slave è gestita attraverso due buffer, posti *Buffer_Input* e posto *Buffer_Output*.

1.1 Risultati

Nella tabella vengono mostrate il numero di archi e di nodi al variare dei parametri M e S. Le cifre sono indicative dell'aumentare della dimensione dello spazio degli stati proporzionalmente al numero di marcature.

master, slaves	Nodi	Archi
1, 1	14	19
2, 2	94	222
3, 3	426	334
4, 4	1500	5610
5, 5	4422	18720
6, 6	11418	52998
7, 7	26598	132594
8, 8	57057	301158
9, 9	114400	632775
10, 10	216788	1246960
11, 11	391612	2328612
12, 12	678912	4153916
13, 13	1135668	7123272
14, 14	1841100	11802420
15, 15	2903124	18973020

1.2 Considerazioni su Fork/Join

Il modello non garantisce che avvenga il join di due processi dello stesso padre quando la marcatura degli slave è maggiore di 2. Si può garantire che avvenga il join di due processi forkati dallo stesso padre attraverso differenti strutture slaves o usando reti WN.

1.3 Riduzione

Una rete di petri può essere ridotta usando le seguendi tecniche:

- fusione
- eliminazione
- rimozione dei loop

Nelle figure vengono mostrate alcune fasi di riduzione della rete in analisi.

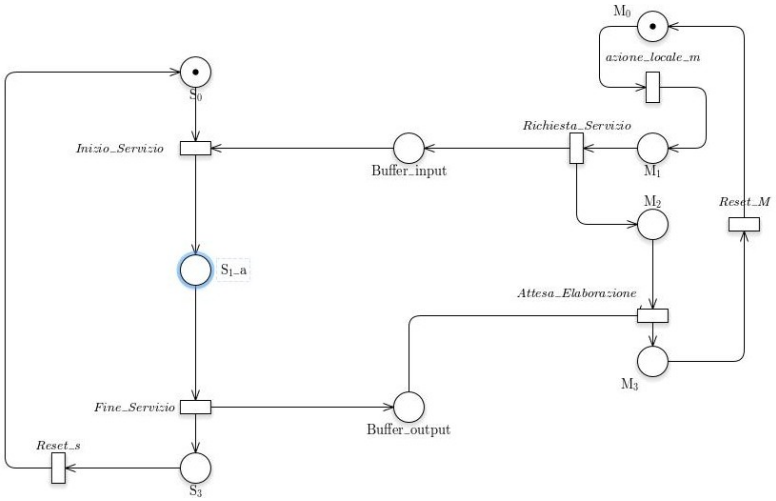


Figure 2: eliminazione di posti identici

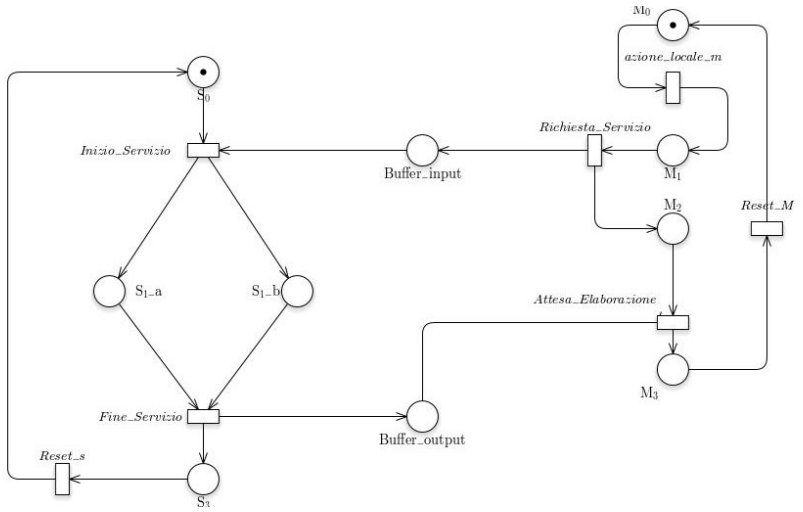


Figure 3: fusione di posti

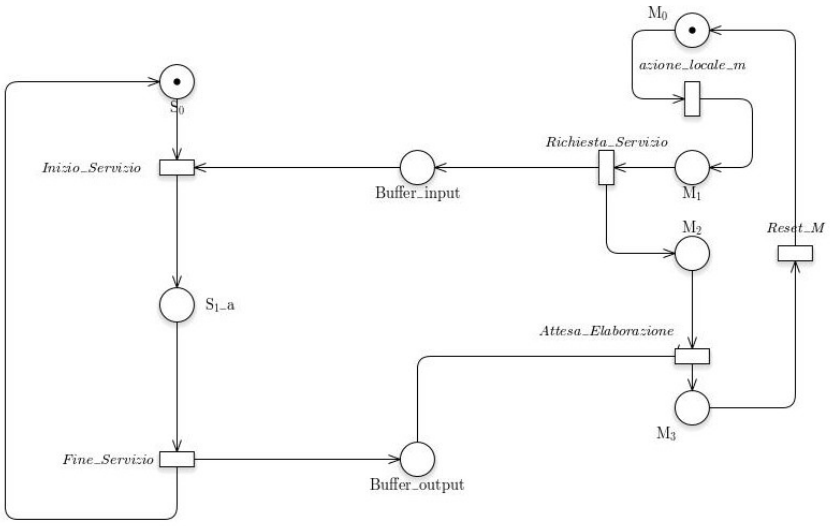


Figure 4: fusione di transizioni

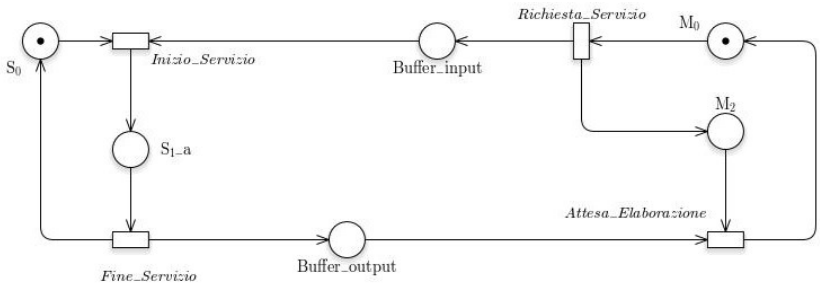


Figure 5: fusione di transizioni e posti

1.4 P e T invarianti

Tramite GreatSPN possiamo calcolare gli T- e P- semiflussi

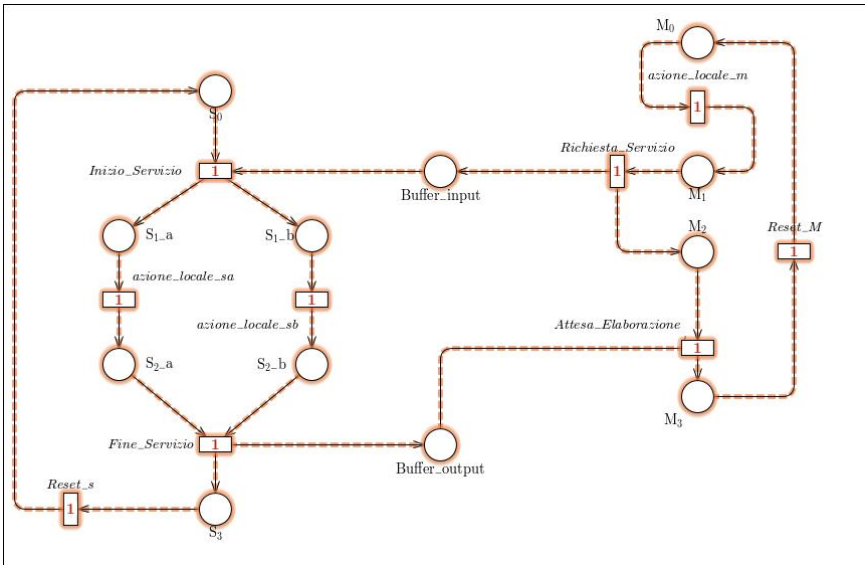


Figure 6: T-semiflows

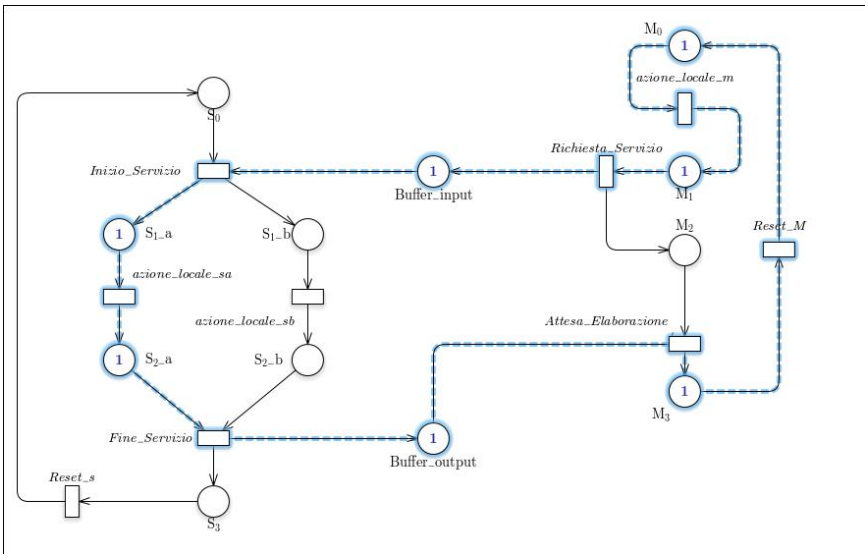


Figure 7: P-semiflows

Gli P-semiflussi sono i seguenti:

- S0 + $S1_a$ + $S2_a$ + S3
- S0 + $S1_b$ + $S2_b$ + S3
- M0 + M1 + M2 + M3
- $S1_a$ + $S2_a$ + Buffer_output + Buffer_input + M0 + M1 + M3
- $S1_b$ + $S2_b$ + Buffer_output + Buffer_input + M0 + M1 + M3

Il T-semiflusso è il seguente:

- Inizio_servizio + azione_locale_sa + azione_locale_sb + Fine_servizio + azione_locale_m + Richiesta_servizio + Attesa_elaborazione + Reset_M + Reset_S

e dato che comprende tutte le transizioni, il sistema rispetta la proprietà di liveness. Dato che la reteA è interamente coperta dagli P-semiflussi, possiamo affermare che la rete sia bounded. Gli P-semiflussi ci permettono di ricavare i seguenti invarianti lineari relativi ai marking m:

- m[S0] + m[$S1_a$] + m[$S2_a$] + m[S3] = 1
- m[S0] + m[$S1_b$] + m[$S2_b$] + m[S3] = 1
- m[M0] + m[M1] + m[M2] + m[M3] = 1
- m[$S1_a$] + m[$S2_a$] + m[Buffer_output] + m[Buffer_input] + m[M0] + m[M1] + m[M3] = 1
- m[$S1_b$] + m[$S2_b$] + m[Buffer_output] + m[Buffer_input] + m[M0] + m[M1] + m[M3] = 1

Dato che $\forall p \in P$, $m[p] \geq 0$ possiamo affermare, a partire dalle precedenti uguaglianze che:

- ogni posto nei seguenti insieme è in mutua esclusione con gli elementi dello stesso insieme:

  {S0, $S1_a$, $S2_a$, S3}
  {S0, $S1_b$, $S2_b$, S3}
  {M0, M1, M2, M3}
  {$S1_a$, $S2_a$, Buffer_output, Buffer_input, M0, M1, M3}
  {$S1_b$, $S2_b$, Buffer_output, Buffer_input, M0, M1, M3}

- $\forall p_i \in P$, $m[p_i] \leq 1$ (bounds)
- dato che i posti che sono gli unici *enablers* di una transizione sono i seguenti:

  $$S1_a,\ S1_b,\ S3,\ M0,\ M1,\ M3$$

  e quindi possiamo provare a dimostrare l'assenza di deadlock partendo dagli invarianti lineari relativi ai marking:

  - $m[S0] + m[S2_a] = 1$
  - $m[S0] + m[S2_b] = 1$
  - $m[M2] = 1$
  - $m[S2_a] + m[Buffer\_output] + m[Buffer\_input] = 1$
  - $m[S2_b] + m[Buffer\_output] + m[Buffer\_input] = 1$

  Dato che M2 è marcata, per far sì che *attesa_ elaborazione* non venga abilitata:

  $$m[Buffer\_output] = 0$$

  Inoltre per far sì che Inizio_Servizio e Fine_Servizio non vengano abilitate:

  - $m[Buffer\_input] + M[S0] \leq 1$
  - $m[S2_a] + m[S2_b] \leq 1$

  Riassumendo, il sistema è il seguente:

  - $m[S0] + m[S2_a] = 1$
  - $m[S0] + m[S2_b] = 1$
  - $m[S2_a] + m[Buffer\_input] = 1$
  - $m[S2_b] + m[Buffer\_input] = 1$
  - $m[Buffer\_input] + M[S0] \leq 1$
  - $m[S2_a] + m[S2_b] \leq 1$

  che per la legge di conservazione dei token, non può essere soddisfatto. Quindi nel sistema non vi è la possibilità di deadlock.

2 Rete B

M master identici, uno slave di tipo 1 e uno slave di tipo 1 scelti liberamente dai master.

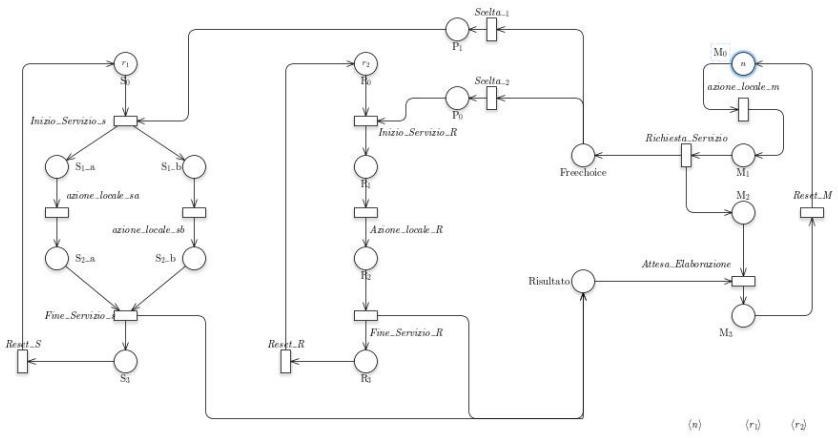


Figure 8: Modello della reteB

La figura rappresenta la rete di Petri P/T dell'esercizio B. Il master è modellato dai posti *M0*, *M1*, *M2*, *M3* e dalle transizioni *Azione_ Locale*, *Richiesta_ Servizio*, *Attesa_ Elaborazione* e *Reset_ M* Lo slave di tipo 1 è modellato dai posti *S0*, *S1_ a*, *S1_ b*, *S2_ a*, *S2_ b* e S3 e dalle transizioni *Inizio_ Servizio*, *Azione_ Locale_ S_ a*, *Azione_ Locale_ S_ b*, *Fine_ Servizio* e *Reset_ S*. Lo slave di tipo 2 è modellato dai posti *R0*, *R1_ a*, *R1_ b*, *R2_ a*, *R2_ b* e *R3* e dalle transizioni *Inizio_ Servizio_ R*, *Azione_ Locale_ R*, *Fine_ Servizio* e *Reset_ R*. La richiesta del servizio verso lo slave scelto è gestita attraverso due buffer, *FreeChoice* e *Risultato*.

2.1 Risultati

Al variare del marking del master:

master, slaves	Stati	Archi
1, 2	40	76
2, 2	204	544
3, 2	728	2400
4, 2	2072	7896
5, 2	5040	21336
6, 2	10920	50064
7, 2	21648	105648
8, 2	39996	205260
9, 2	69784	373252
10, 2	116116	642928

Parametrizzando anche il marking sugli slaves (R+S):

master, slaves	Stati	Archi
1, 2	40	76
2, 2	204	544
4, 4	7265	32674
6, 6	113464	664234
8, 8	1073226	7405654
10, 10	7212128	55762000

2.2 Considerazioni su Fork/Join

Lo slave di tipo 1 processa una sola richiesta alla volta. Il master in attesa del risultato (M2) potrebbe ricevere il risultato di un lavoro richiesto da un altro master.

2.3 P e T invarianti

Tramite GreatSPN possiamo calcolare gli T- e P- semiflussi

Gli P-invarianti sono i seguenti:

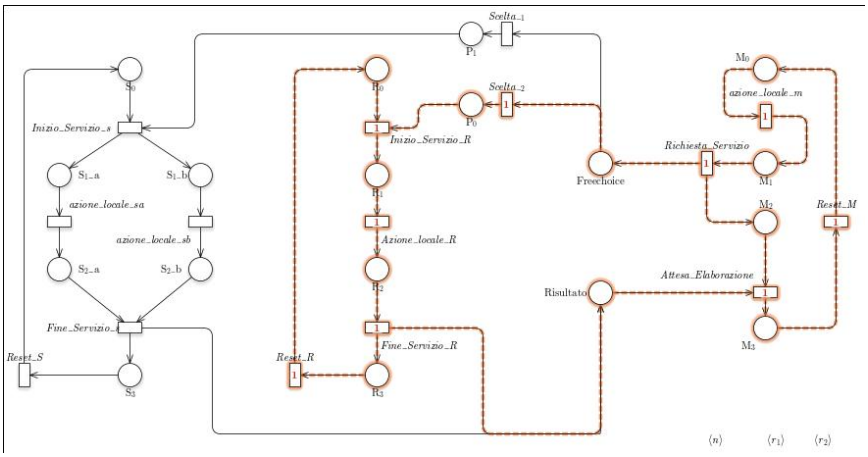


Figure 9: T-semiflows

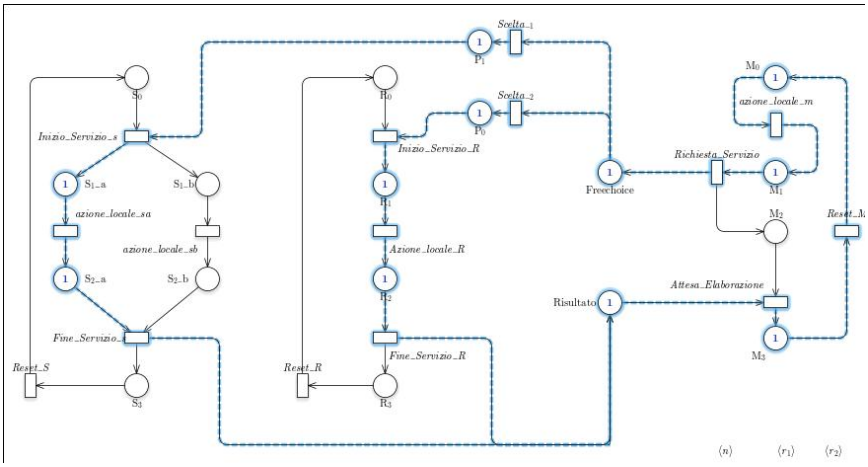


Figure 10: P-semiflows

- S0 + S1_a + S2_a + S3
- S0 + S1_b + S2_b + S3
- R0 + R1 + R2 + R3
- M0 + M1 + M2 + M3
- S1_a + S2_a + R1 + R2 + M0 + M1 + M3 + Freechoice + P0 + P1 + Risultato
- S1_b + S2_b + R1 + R2 + M0 + M1 + M3 + Freechoice + P0 + P1 + Risultato

Gli T-invarianti sono i seguenti:

- Inizio_servizio_R + azione_locale_R +
  Fine_servizio_R + Reset_R + azione_locale_m + Richiesta_servizio +
  Attesa_elaborazione + Reset_M + Scelta_2
- Inizio_servizio_S + azione_locale_sa + azione_locale_sb +
  Fine_servizio_S + Reset_s + azione_locale_m + Richiesta_servizio +
  Attesa_elaborazione + Reset_m + Scelta_1

Dato che ci sono due semiflussi, ognuno relativo alle transizioni dei due diversi slaves, c'è possibilità di starvation. Possiamo infatti immaginare una traccia di esecuzione in cui il master in seguito a FreeChoice sceglie sempre il primo slave. Questo non succederebbe in un sistema fair, ovvero se si obbliga un'automa che entra in uno stato infinite volte ad eseguire tutte le possibili transizioni da quello stato. In tal caso non avremmo starvation e la proprietà di liveness sarebbe rispettata.

Dato che la reteB è interamente coperta dagli P-semiflussi, possiamo affermare che la rete sia bounded. Dimostriamo invece che la rete non ha possibilità di deadlock.

- m[S0] + m[S1_a] + m[S2_a] + m[S3] = 1
- m[S0] + m[$S1_b$] + m[$S2_b$] + m[S3] = 1
- m[R0] + m[R1] + m[R2] + m[R3] = 1
- m[M0] + m[M1] + m[M2] + m[M3] = 1
- m[S1_a] + m[S2_a] + m[R1] + m[R2] + m[M0] + m[M1] + m[M3] + m[Freechoice] + m[P0] + m[P1] + m[Risultato] = 1
- m[$S1_b$] + m[$S2_b$] + m[R1] + m[R2] + m[M0] + m[M1] + m[M3] + m[Freechoice] + m[P0] + m[P1] + m[Risultato] = 1

I posti che sono gli unici enablers di una sola transizione sono:

M0, M1, M3, R1, R2, R3, FreeChoice, $S1_a$, $S1_b$, S3

Gli invarianti lineari dei marking diventano:

- m[S0] + m[S2_a] = 1
- m[S0] + m[$S2_b$] = 1
- m[R0] = 1
- m[M2] = 1
- m[S2_a] + m[P0] + m[P1] + m[Risultato] = 1
- m[$S2_b$] m[P0] + m[P1] + m[Risultato] = 1

Dati i marking in R0 e M2, per far sì che */Inizio_ Servizio/_ R*, Attesa_Elaborazione, */Fine_ Servizio/$_s$* e */Inizio_ Servizio/$_s$* non vengano abilitati:

- m[P0] = 0
- m[Risultato] = 0
- m[$S2_a$] + m[$S2_b$] $\leq$ 1
- m[P1] + m[S0] $\leq$ 1

Il sistema si riduce a:

- m[S0] + m[S2_a]= 1
- m[S0] + m[$S2_b$] = 1
- m[S2_a] + m[P1] = 1
- m[$S2_b$] + m[P1] = 1
- m[$S2_a$] + m[$S2_b$] $\leq$ 1
- m[P1] + m[S0] $\leq$ 1

che non può essere soddisfatto per la legge di conservazione dei token.

3 Rete C

Due master identici, uno slave di tipo 1 e uno slave di tipo 1 scelti liberamente dai master.

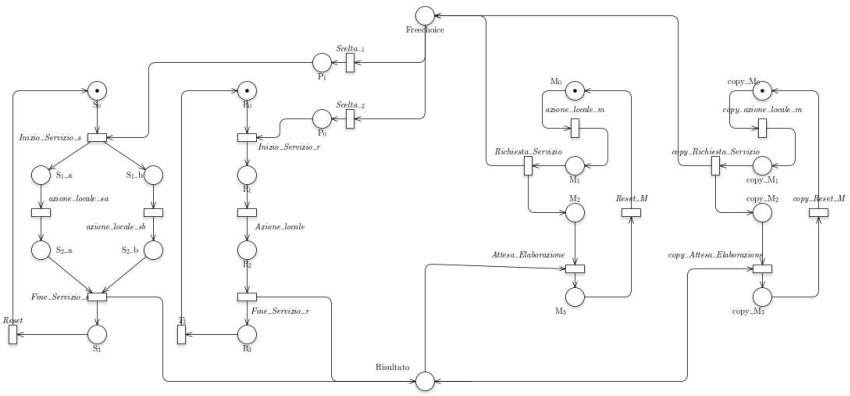


Figure 11: Modello della reteC

La figura rappresenta la rete di Petri P/T dell'esercizio C. Il master è modellato dai posti M0, M1, M2, M3 e dalle transizioni Azione_Locale, *Richiesta_Servizio*, *Attesa_Elaborazione* e *Reset_M* Lo slave di tipo 1 è modellato dai posti S0, S1_a, S1_b, S2_a, S2_b e S3 e dalle transizioni *Inizio_Servizio*, *Azione_Locale_S_a*, *Azione_Locale_S_b*, *Fine_Servizio* e *Reset_S* (il secondo master è una copia del primo). Lo slave di tipo 2 è modellato dai posti R0, R1_a, R1_b, R2_a, R2_b e R3 e dalle transizioni *Inizio_Servizio/_R, Azione_Locale_R, /Fine_Servizio* e Reset_R. La richiesta del servizio verso lo slave scelto è gestita attraverso due buffer, posti FreeChoice e Risultato.

3.1 P e T invarianti

Tramite GreatSPN possiamo calcolare gli T- e P- semiflussi

Gli P-invarianti sono i seguenti:

- S0 + $S1_a$ + $S2_a$ + S3

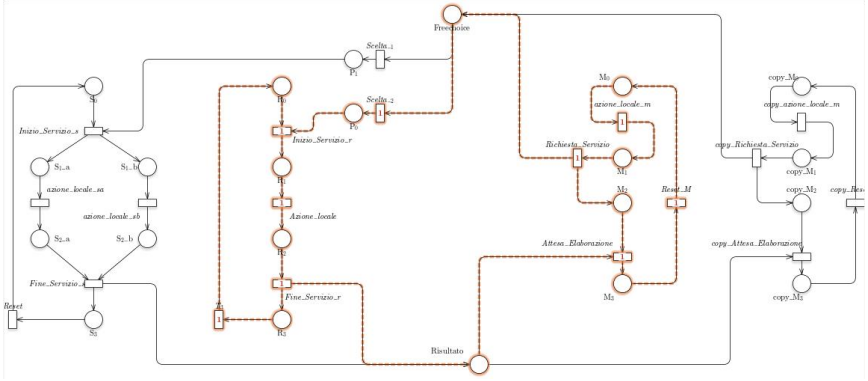


Figure 12: T-semiflows

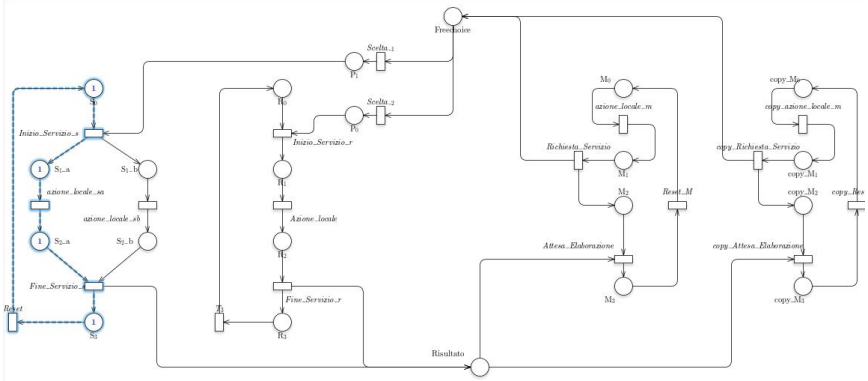


Figure 13: P-semiflows

- S0 + $S1_b$ + $S2_b$ + S3
- R0 + R1 + R2 + R3
- M0 + M1 + M2 + M3
- copy_M0 + copy_M1 + copy_M2 + copy_M3
- $S1_a$ + $S2_a$ + R1 + R2 + M0 + M1 + M3 + Freechoice + P0 + P1 + Risultato + copy_M0 + copy_M1 + copy_M3
- $S1_b$ + $S2_b$ + R1 + R2 + M0 + M1 + M3 + Freechoice + P0 + P1 + Risultato + copy_M0 + copy_M1 + copy_M3

Dato che la reteC è interamente coperta dagli P-semiflussi, possiamo affermare che la rete sia bounded. Gli P-semiflussi ci permettono di ricavare i seguenti invarianti lineari relativi ai marking m:

- m[S0] + m[$S1_a$] + m[$S2_a$] + m[S3] = 1
- m[S0] + m[$S1_b$] + m[$S2_b$] + m[S3] = 1
- m[R0] + m[R1] + m[R2] + m[R3] = 1
- m[M0] + m[M1] + m[M2] + m[M3] = 1
- m[copy_M0] + m[copy_M1] + m[copy_M2] + m[copy_M3] = 1
- m[$S1_a$] + m[$S2_a$] + m[R1] + m[R2] + m[M0] + m[M1] + m[M3] + m[Freechoice] + m[P0] + m[P1] + m[Risultato] + m[copy_M0] + m[copy_M1] + m[copy_M3] = 1
- m[$S1_b$] + m[$S2_b$] + m[R1] + m[R2] + m[M0] + m[M1] + m[M3] + m[Freechoice] + m[P0] + m[P1] + m[Risultato] + m[copy_M0] + m[copy_M1] + m[copy_M3] = 1

Gli spazi *enablers* di una sola transizione sono i seguenti:

R1, R2, R3, $S1_a$, $S1_b$, S3, Risultato, M0, M1, M3, copy_M0, copy_M1, copy_M3, FreeChoice

il sistema precedente diventa:

- m[S0] + m[S2$_a$] = 1
- m[S0] + m[S2$_\mathrm{b}$] = 1
- m[R0] = 1
- m[M2] = 1
- m[copy_M2] = 1
- m[S2$_\mathrm{b}$] + m[P0] + m[P1] = 1
- m[S2$_\mathrm{a}$] + m[P0] + m[P1] = 1

Dati i marking in R0 e M2 e copy_M2, per far sì che */Inizio_Servizio/_R*, Attesa_Elaborazione, *copy_/Attesa_Elaborazione/*, */Fine_Servizio/$_s$* e */Inizio_Servizio/$_s$* non vengano abilitati:

- m[P0] = 0
- m[Risultato] = 0
- m[S2$_a$] + m[S2$_\mathrm{b}$] $\leq$ 1
- m[P1] + m[S0] $\leq$ 1

Il sistema si riduce allo stesso della precedente rete B:

- m[S0] + m[S2$_a$] = 1
- m[S0] + m[S2$_\mathrm{b}$] = 1
- m[S2$_\mathrm{b}$] + m[P1] = 1
- m[S2$_\mathrm{a}$] + m[P1] = 1
- m[S2$_a$] + m[S2$_\mathrm{b}$] $\leq$ 1
- m[P1] + m[S0] $\leq$ 1

e non può essere soddisfatto per la legge di conservazione dei token.

Gli T-invarianti sono i seguenti:

- Inizio_Servizio$_r$ + Azione_Locale + Fine_Servizio$_r$ + T3 + azione_locale$_m$ + Richiesta_Servizio + Attesa_Elaborazione + Reset_M + Scelta$_1$
- Inizio_Servizio$_s$ + Azione_Locale$_\mathrm{sa}$ + Azione_Locale$_\mathrm{sb}$ + Fine_Servizio$_s$ + T3 + azione_locale$_m$ + Richiesta_Servizio + Attesa_Elaborazione + Reset_M + Scelta$_1$

- Inizio_Servizio$_r$ + Azione_Locale + Fine_Servizio$_r$ + T3 + Scelta$_2$ + copy$_a$zione_locale$_m$ + copy_Richiesta_Servizio + copy_Attesa_Elaborazione + copy_Reset$_m$

- Inizio_Servizio$_s$ + Azione_Locale$_{sa}$ + Azione_Locale$_{sb}$ + Fine_Servizio$_s$ + Reset + Scelta$_1$ + copy_azione_locale$_m$ + copy_Richiesta_Servizio + copy_Attesa_Elaborazione + copy_Reset$_m$

Come nella rete B, in assenza di fairness non possiamo rispettare la condizione di liveness e c'è possibilità di starvation.

4 Rete D

Due master identici, uno slave di tipo 1 e uno slave di tipo 1 scelti associati ciascuno ad un master diverso.

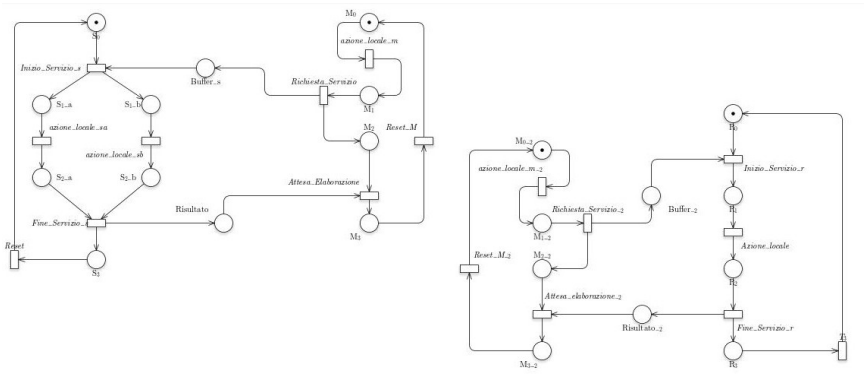


Figure 14: Modello della reteD

4.1 P e T invarianti

Tramite GreatSPN possiamo calcolare gli T- e P- semiflussi

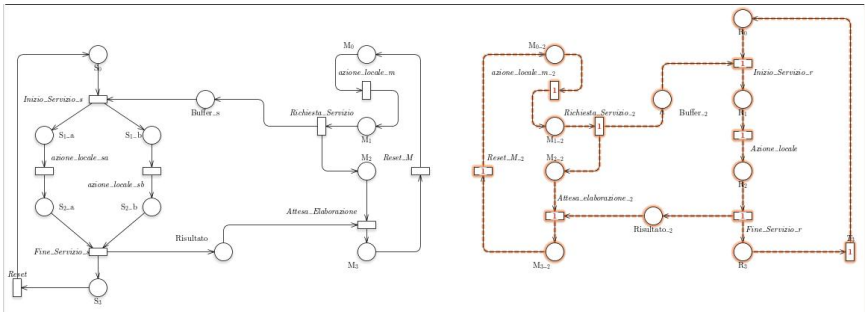


Figure 15: T-semiflows

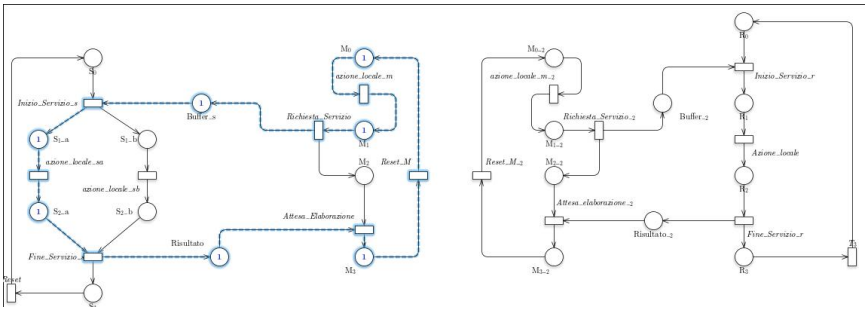


Figure 16: P-semiflows

Gli P-invarianti sono i seguenti:

- S0 + $S1_a$ + $S2_a$ + S3
- S0 + $S1_b$ + $S2_b$ + S3
- R0 + R1 + R2 + R3
- M0 + M1 + M2 + M3
- $S1_a$ + $S2_a$ + M0 + M1 + M3 + $Buffer_s$ + Risultato
- $S1_b$ + $S2_b$ + M0 + M1 + M3 + $Buffer_s$ + Risultato
- $M0_2$ + $M1_2$ + $M3_2$
- R1 + R2 + $M0_2$ + $M1_2$ + $M3_2$ + $Buffer_2$ + $Risultato_2$

Ai fini della dimostrazione dell'assenza di deadlock, possiamo notare che lo slave di tipo 2 è equivalente allo slave di tipo 1 se si applicano due riduzioni alla rete (vengono fusi in un unico posto $S1_a$-$S2_a$ e $S1_b$-$S2_b$, poi eliminata la fork). Inoltre i master sono indipendenti fra di loro e ciascuno rispetta l'assenza di deadlock come già dimostrato nella rete A. Gli T-invarianti sono i seguenti:

- $Inizio\_Servizio_s$ + $azione\_locale_{sa}$ + $azione\_locale_{sb}$ + $Fine\_Servizio_s$ + Reset + $azione\_locale_m$ + Richiesta_Servizio + Attesa_Elaborazione + $Reset_m$
- $Inizio\_Servizio_r$ + Azione_locale + $Fine\_Servizio_r$ + T3 $azione\_locale_{m2}$ + $Richiesta\_Servizio_2$ + $Attesa\_Elaborazione_2$ + $Reset_{m2}$

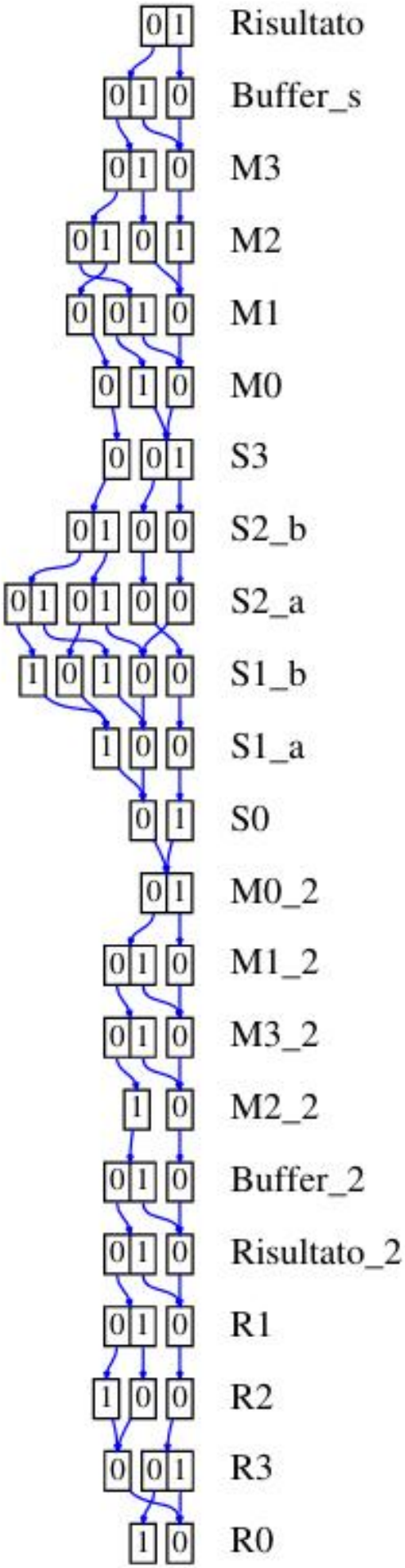
Come nella rete B, in assenza di fairness non possiamo rispettare la condizione di liveness e c'è possibilità di starvation.

4.2 Decision Diagram

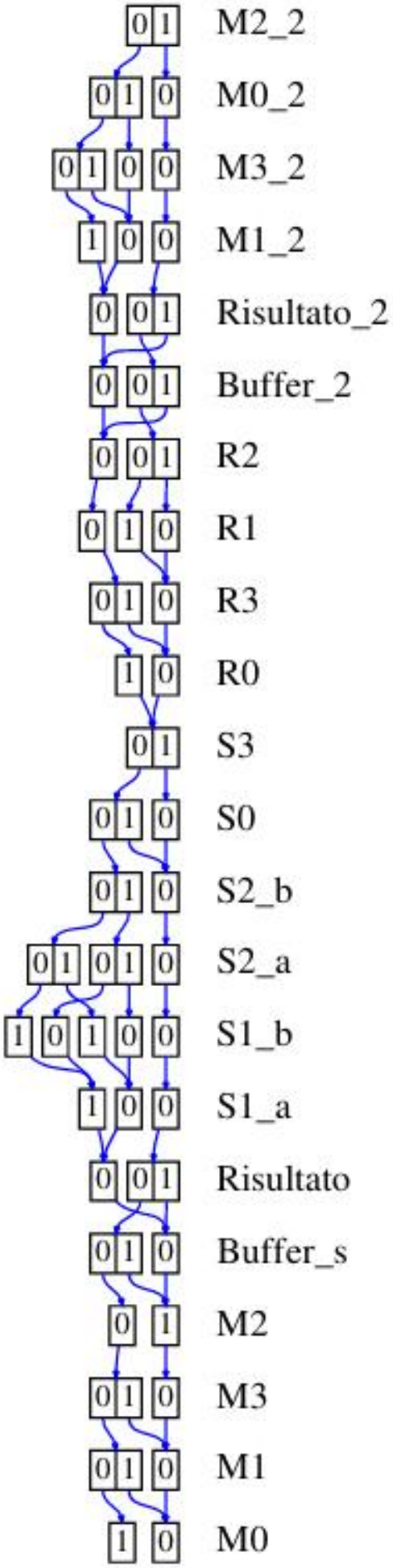
L'efficacia dei decision diagram sulla generazione dello stato degli spazi dipende fortemente dall'ordine delle variabili. Di seguito vengono mostrati i decision diagram usando per le assegnazioni i seguenti algoritmi:

- Sloan: un algoritmo di riduzione della banda di matrici sparse con una buona performance

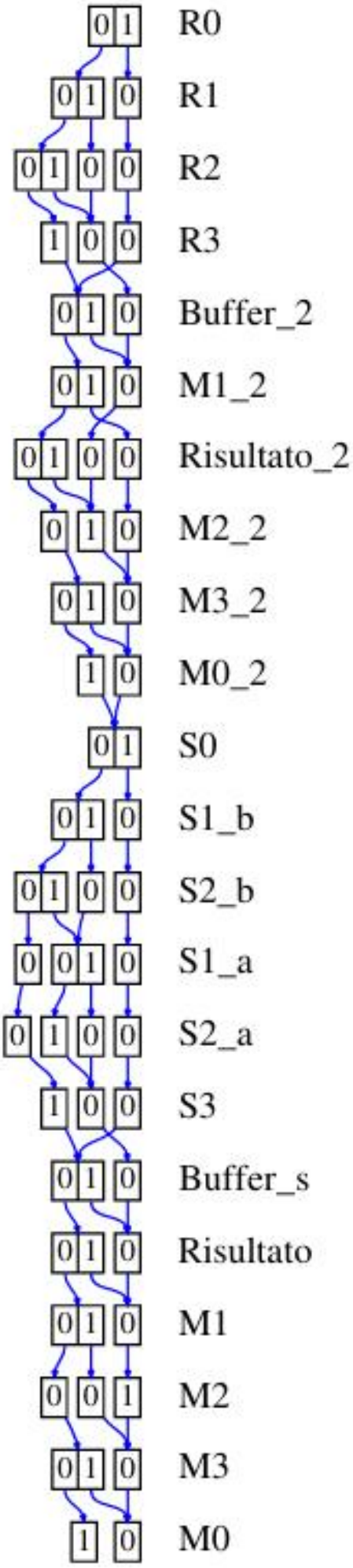
- Cuthill-McKee: un altro algoritmo di riduzione della banda di matrici sparse
- Tovchigrechko e Noack: due algoritmo appositamente ideati per le reti di Petri, anch'essi con una buona performance
- P-chaining: un algoritmo che sfrutta le informazioni strutturali della rete ma ha una bassa performance
- Gradient-P
- Gibbs-Poole-Stockmeier: un altro algoritmo matriciale che nella rete in analisi ha restituito il risultato peggiore



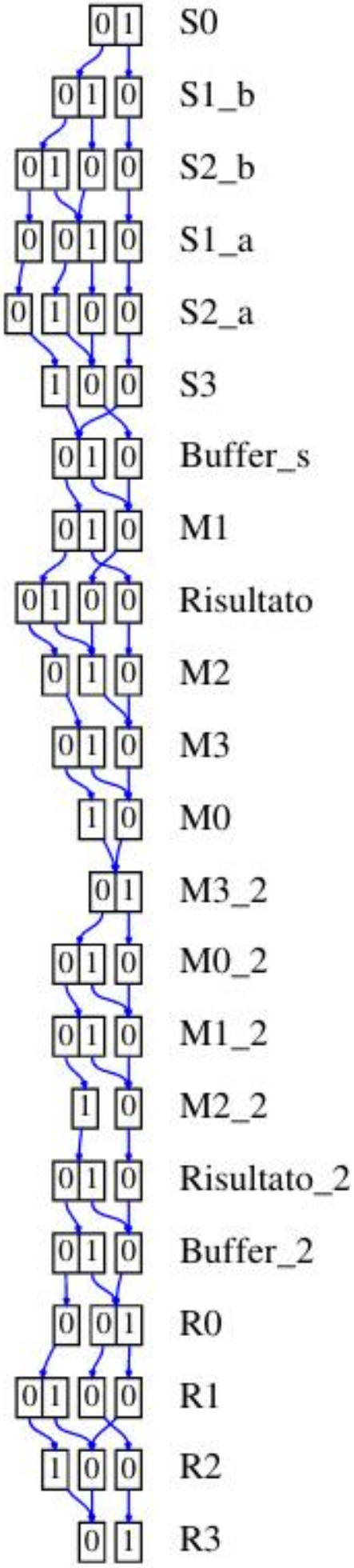
Algoritmo di Sloan



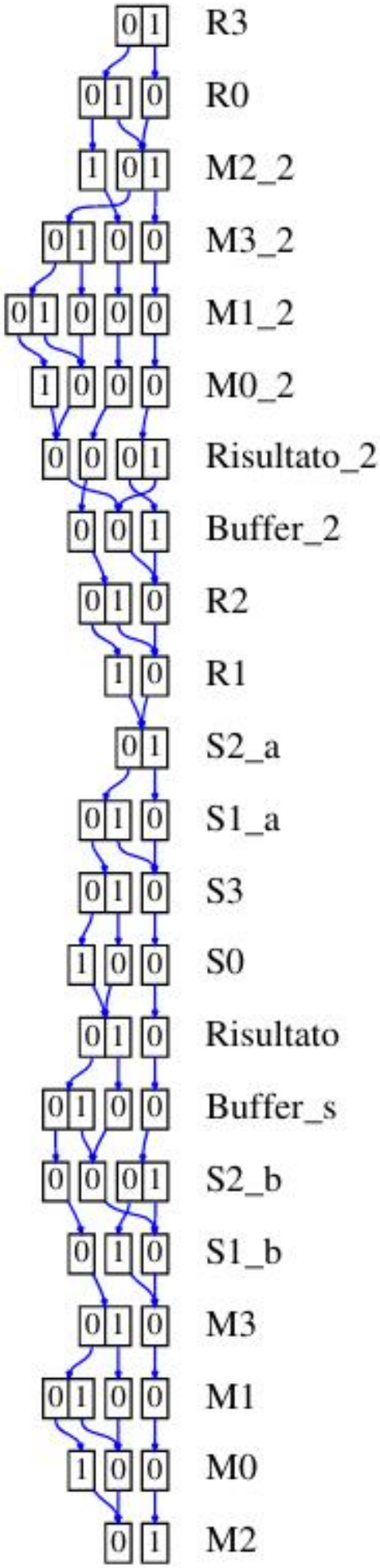
Algoritmo di Cuthull-McKee



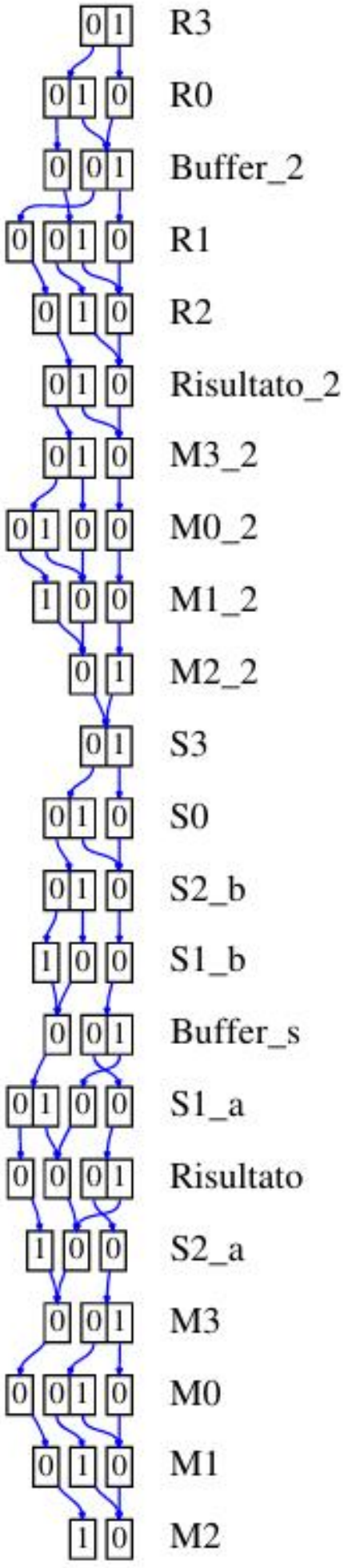
Algoritmo di Tovchigrechko



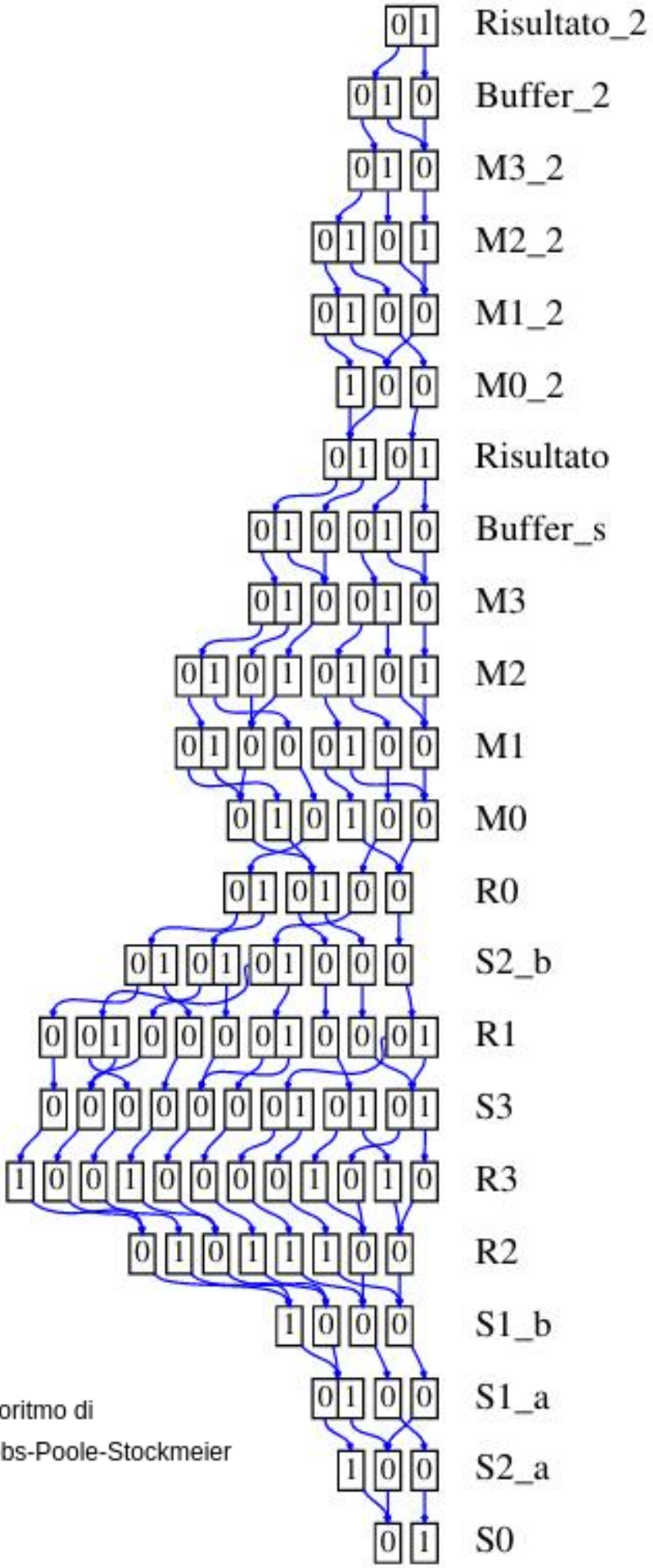
Algoritmo di Noack



Algoritmo P-Chain



Algoritmo Gradient-P



Algoritmo di
Gibbs-Poole-Stockmeier