



**Automazione industriale
dispense del corso (a.a. 2008/2009)
12. Metodi top-down, bottom-up e ibridi**

Luigi Piroddi
piroddi@elet.polimi.it

Introduzione

Abbiamo visto un esempio di costruzione di un modello di un sistema FMS con reti di Petri.

Esistono varie tecniche sistematiche di modellizzazione che consentono di ottenere un modello completo a partire da sotto-modelli più semplici.

Alcune di esse assicurano determinate proprietà strutturali del modello al termine del progetto.

Noi introdurremo:

- ▶ i metodi *top-down*,
- ▶ i metodi *bottom-up*,
- ▶ i metodi *ibridi*.

Questi metodi di modellizzazione prendono anche il nome di *metodi diretti di progetto*, in quanto un procedimento molto usato per il progetto di un controllore logico consiste nel modellizzare direttamente il comportamento desiderato del sistema, a partire dal quale l'effettivo controllore si ricava in modo semplice.

Metodi top-down

I metodi top-down sono basati su affinamenti successivi di un modello a reti di Petri:

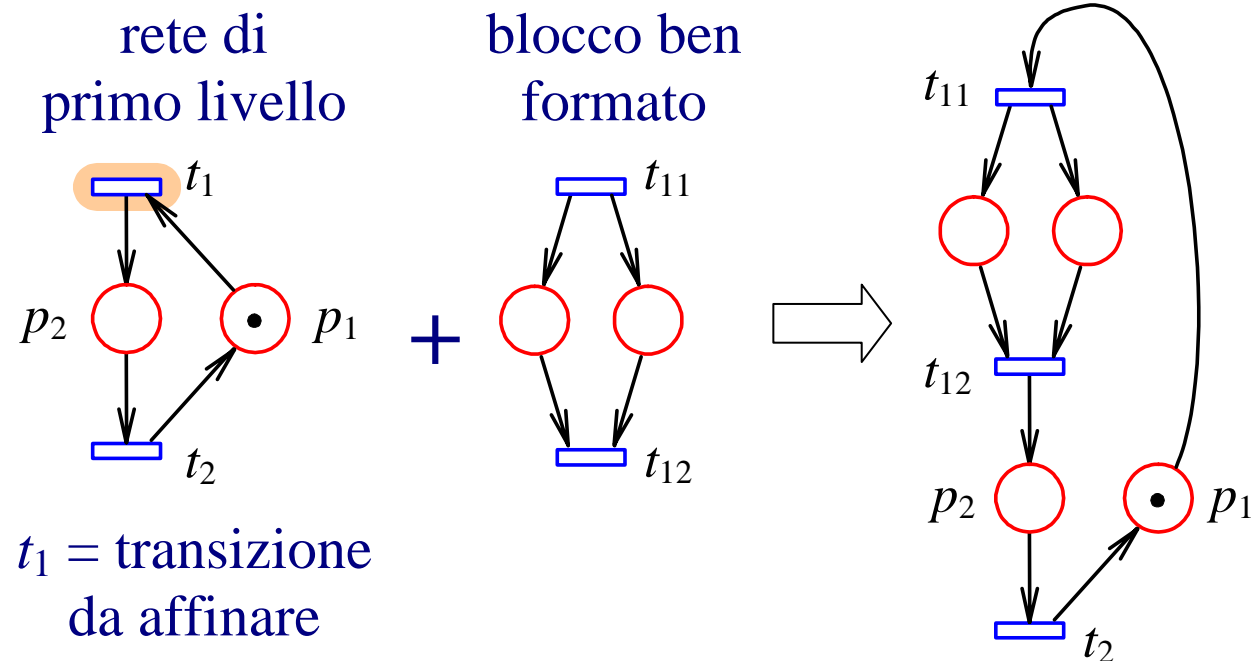
- ❶ Si parte da una rete molto semplice (*rete di primo livello*) che descrive in modo aggregato il funzionamento dell'intero sistema.
La rete di primo livello è molto astratta, ma anche molto semplice, e quindi facile da analizzare.
- ❷ Si affina (si espande) la rete di primo livello, agendo su posti o transizioni, e aumentando il dettaglio descrittivo contenuto nel modello.
Le regole di affinamento conservano proprietà importanti della rete come la limitatezza e la vivezza (l'idea è simile a quella delle regole di riduzione, ma applicata in senso opposto).

Metodo di Valette

Si consideri una rete, detta *rete di primo livello* (N_1), in cui esistano transizioni 1-abilite (cioè tali che il numero di gettoni in ingresso a tali transizioni basti solo per uno scatto, in una data marcatura).

Tali transizioni possono essere sostituite con *blocchi ben formati* (reti di Petri con ben determinate caratteristiche, che vedremo tra breve).

L'affinamento della transizione avviene eliminandola e collegando gli archi in ingresso [uscita] alla transizione iniziale [finale] del blocco ben formato.

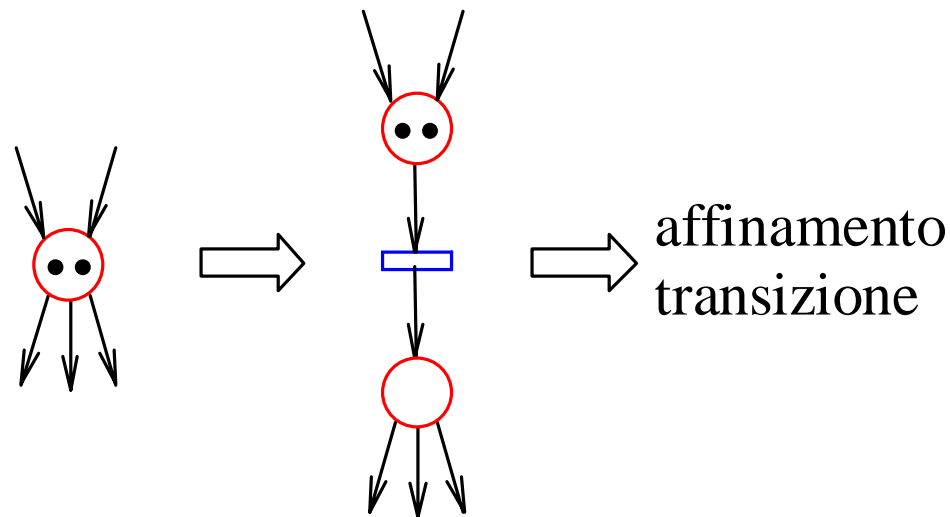


E' possibile operare anche l'affinamento dei posti.

Esso avviene sostituendo prima il posto con una sequenza posto-transizione-posto e poi affinando la transizione intermedia.

Il posto a monte [valle] della transizione intermedia viene collegato con gli archi in ingresso [uscita] al posto da affinare.

I gettoni originariamente presenti nel posto da affinare vengono trasferiti nel posto a monte della transizione intermedia.



Proprietà del metodo di Valette

La rete risultante dopo l'affinamento di una transizione (o un posto) nella rete di primo livello prende il nome di *rete di secondo livello* (N_2).

Essa conserva le proprietà fondamentali se queste erano possedute dalla rete di partenza (N_1):

- ▶ N_1 limitata $\Rightarrow N_2$ limitata (N_1 binaria $\Rightarrow N_2$ binaria)
- ▶ N_1 viva $\Rightarrow N_2$ viva
- ▶ N_1 reversibile $\Rightarrow N_2$ reversibile

Pertanto, anche operando più affinamenti successivi si mantengono tali proprietà, se queste erano possedute da N_1 .

Caratteristiche di un blocco ben formato

Un blocco ben formato è una rete con le seguenti caratteristiche:

- ❶ Esistono una sola transizione di ingresso (*transizione iniziale*, t_{iniz}) e una sola transizione di uscita (*transizione finale*, t_{fin}). Quindi,
 - ▼ solo lo scatto di t_{iniz} immette gettoni nel blocco e solo lo scatto di t_{fin} ne toglie;
 - ▼ il pre-set ed il post-set del blocco sono il pre-set ed il post-set di queste due transizioni, ed è chiaro come mettere il blocco al posto di una transizione;
 - ▼ l'evoluzione del blocco “inizia” quando scatta t_{iniz} e “finisce” quando scatta t_{fin} ; il blocco è interpretabile come un (macro) evento che rappresenta la descrizione dettagliata dell'azione simboleggiata dalla transizione affinata.
- ❷ La rete (*blocco aggiunto*) che si ottiene aggiungendo al blocco un posto (p_a) marcato con un gettone e con solo t_{iniz} [t_{fin}] nel suo post-set [pre-set] è viva. Quindi,
 - ▼ l'evoluzione del blocco può avvenire un numero arbitrario di volte, coerentemente col fatto che, se al livello di astrazione superiore una transizione non può più scattare, non è per colpa di quel che avviene “al suo interno”.

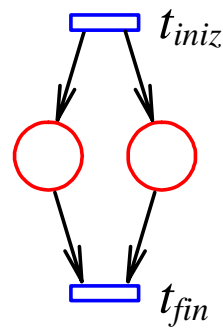
- ③ La marcatura iniziale della rete aggiunta è l'unica marcatura raggiungibile in cui p_a è marcato (con un gettone). Quindi,
- ▼ t_{iniz} non può essere abilitata nuovamente, prima che l'evoluzione del blocco ben formato non sia esaurita, facendo scattare t_{fin} e marcando nuovamente p_a , coerentemente col fatto che, ogni scatto della transizione affinata determina una sola occorrenza del (macro) evento da essa rappresentato;
 - ▼ ogni volta che il blocco inizia ad evolvere, lo fa a partire dalla stessa condizione (il (macro) evento è sempre uguale a sé stesso).
- ④ L'unica transizione del blocco aggiunto abilitata nella sua marcatura iniziale è t_{iniz} . Quindi,
- ▼ quando il blocco non evolve, al suo interno non succede nulla; è evidente che se questo non è vero il blocco non può essere rappresentato al livello di astrazione superiore da una sola transizione.

NB. Il metodo di Valette può essere esteso a transizioni k -abilite se le condizioni 2 e 3 valgono anche marcando il posto aggiuntivo con k gettoni.

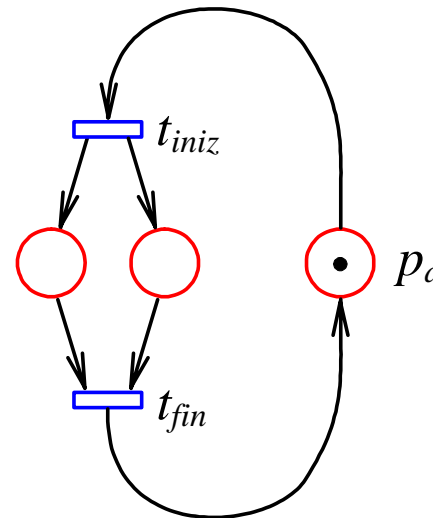
Esempio di blocco ben formato

Un blocco ben formato è per esempio quello raffigurato in figura.

blocco ben
formato



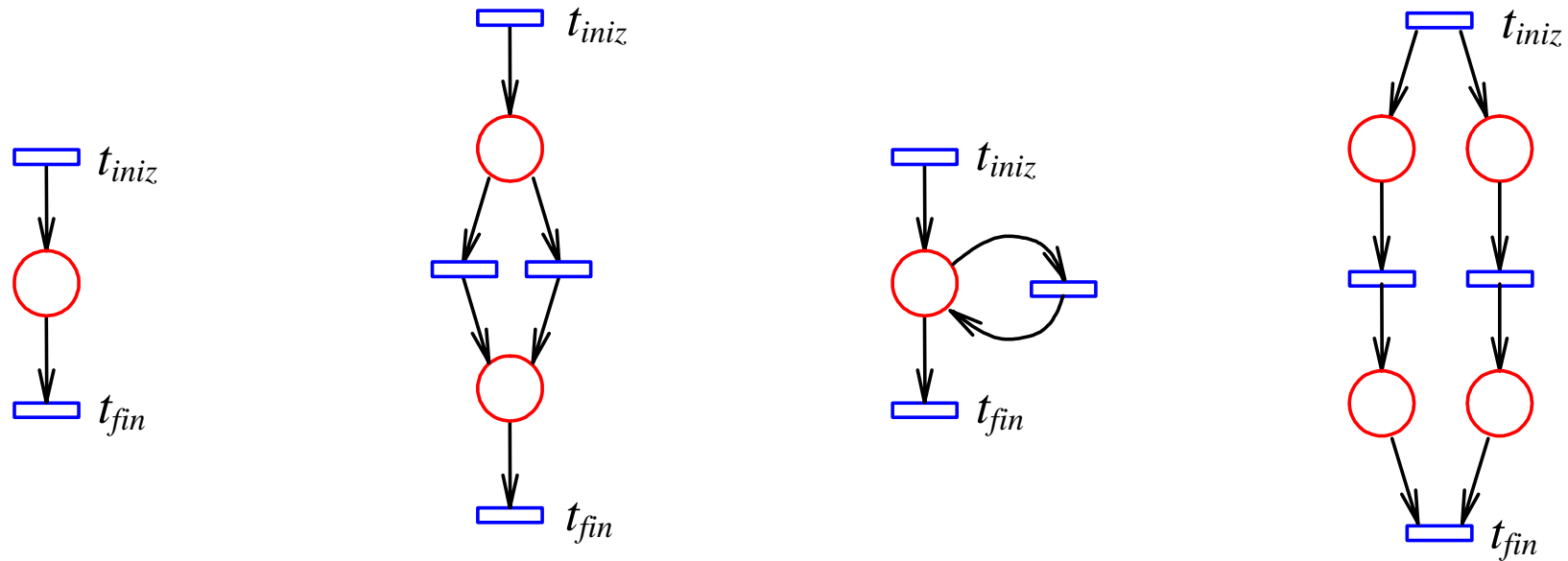
blocco aggiunto



Infatti:

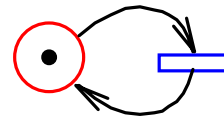
- ❶ il blocco ha una sola transizione di ingresso (t_{iniz}) e una sola transizione di uscita (t_{fin}),
- ❷ il blocco aggiunto è una rete viva,
- ❸ la marcatura iniziale del blocco aggiunto è l'unica in cui il p_a è marcato,
- ❹ l'unica transizione del blocco aggiunto abilitata nella sua marcatura iniziale è t_{iniz} .

Altri esempi di blocchi ben formati

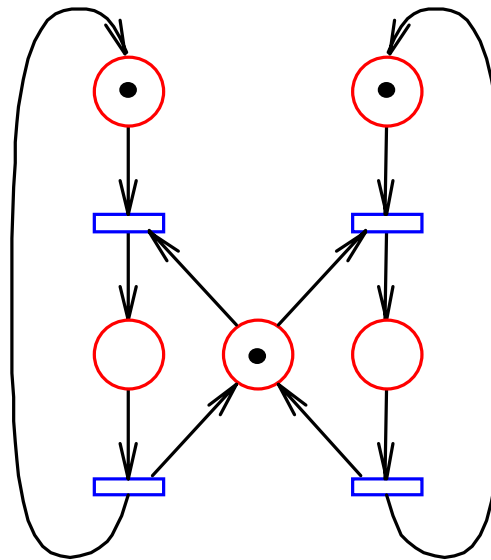


Esempi di reti di primo livello

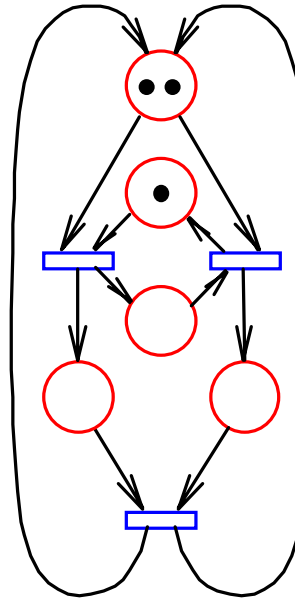
Rete più semplice da
affinare (ciclo di
operazioni)



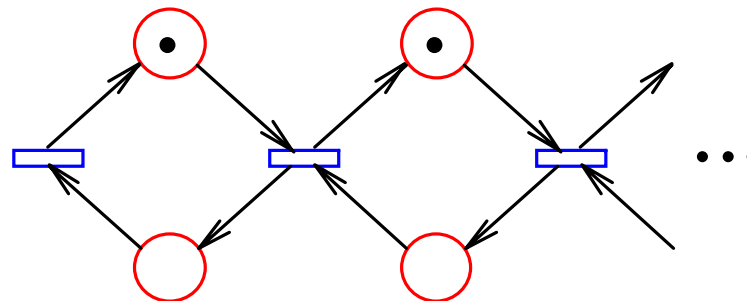
Rete che modella due
sequenze cicliche con una
risorsa condivisa



Rete con struttura a scelta-sincronizzazione, dove due sequenze di lavoro devono essere svolte in alternanza



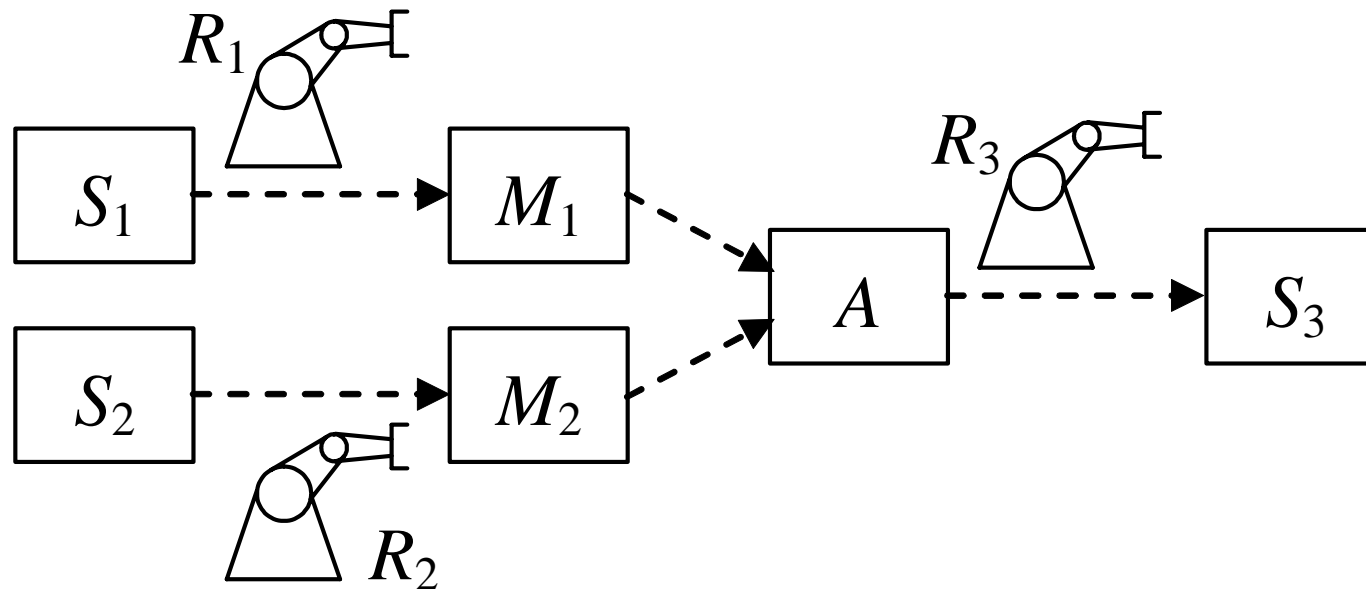
Rete con sincronizzazione tra vari cicli di lavorazione



Esempio di progetto top-down

Si consideri il sistema manifatturiero automatizzato rappresentato in figura, dove:

- ▶ S_1 e S_2 sono magazzini di pezzi grezzi,
- ▶ M_1 e M_2 sono machine,
- ▶ R_1 , R_2 e R_3 sono robot manipolatori di trasporto,
- ▶ A è una cella di assemblaggio,
- ▶ S_3 è il magazzino dei prodotti finiti.



La procedura di lavorazione è la seguente:

- ❶ R_1 [R_2] preleva un pezzo da S_1 [S_2] e lo carica su M_1 [M_2].
- ❷ M_1 ed M_2 effettuano le rispettive lavorazioni.
- ❸ Al termine della lavorazione R_3 preleva il semilavorato da M_1 e lo porta nella cella A e poi preleva il semilavorato da M_2 e lo porta nella cella A_1 .
- ❹ Il manipolatore R_3 effettua l'assemblaggio.
- ❺ Al termine dell'assemblaggio R_3 svuota la cella A e pone il prodotto finito in S_3 .

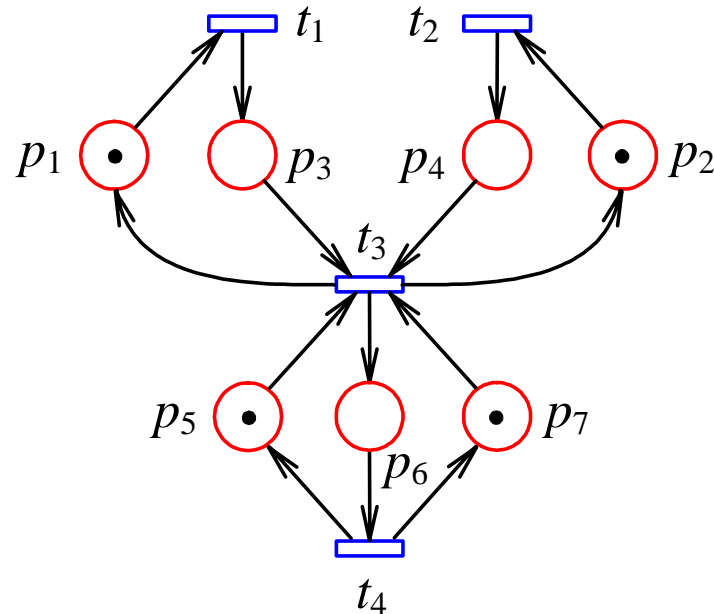
Specifichiamo le attività come transizioni, mentre i posti rappresentano solo condizioni. Specifichiamo le seguenti 4 attività principali:

- t_1) macchina M_1 in lavorazione
- t_2) macchina M_2 in lavorazione
- t_3) scarico di M_1 e M_2 e carico di A
- t_4) cella A in lavorazione

Le condizioni sono:

- p_1) macchina M_1 disponibile
- p_2) macchina M_2 disponibile
- p_3) semilavorato 1 pronto
- p_4) semilavorato 2 pronto
- p_5) cella A disponibile
- p_6) semilavorati 1 e 2 nella cella A
- p_7) robot R_3 disponibile

La rete di primo livello che rappresenta nel modo più astratto la procedura di lavorazione è la seguente:

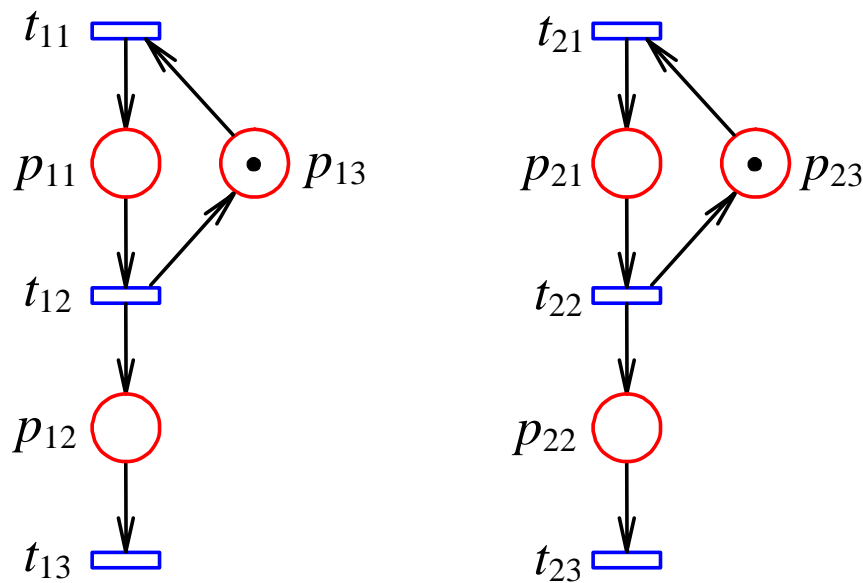


Si notano quattro strutture cicliche, a 2 posti (“libero”, “occupato”), che modellizzano gli stati delle due macchine, della cella A e del robot R_3 , e una transizione di sincronizzazione.

E' facile verificare che la rete di primo livello è viva e binaria.

Per dettagliare le attività del sistema le transizioni della rete di primo livello possono essere affinate.

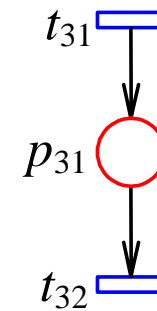
Ad esempio, la lavorazione di una macchina può essere dettagliata in due operazioni: il carico della macchina e la vera e propria lavorazione.



Significato nuovi posti:

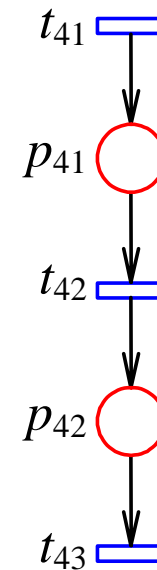
- p_{11}) R_1 carica M_1
- p_{12}) M_1 in lavorazione
- p_{13}) R_1 disponibile
- p_{21}) R_2 carica M_2
- p_{22}) M_2 in lavorazione
- p_{23}) R_2 disponibile

L'operazione di scarico delle macchine e carico della cella può essere rappresentata nel modo seguente:



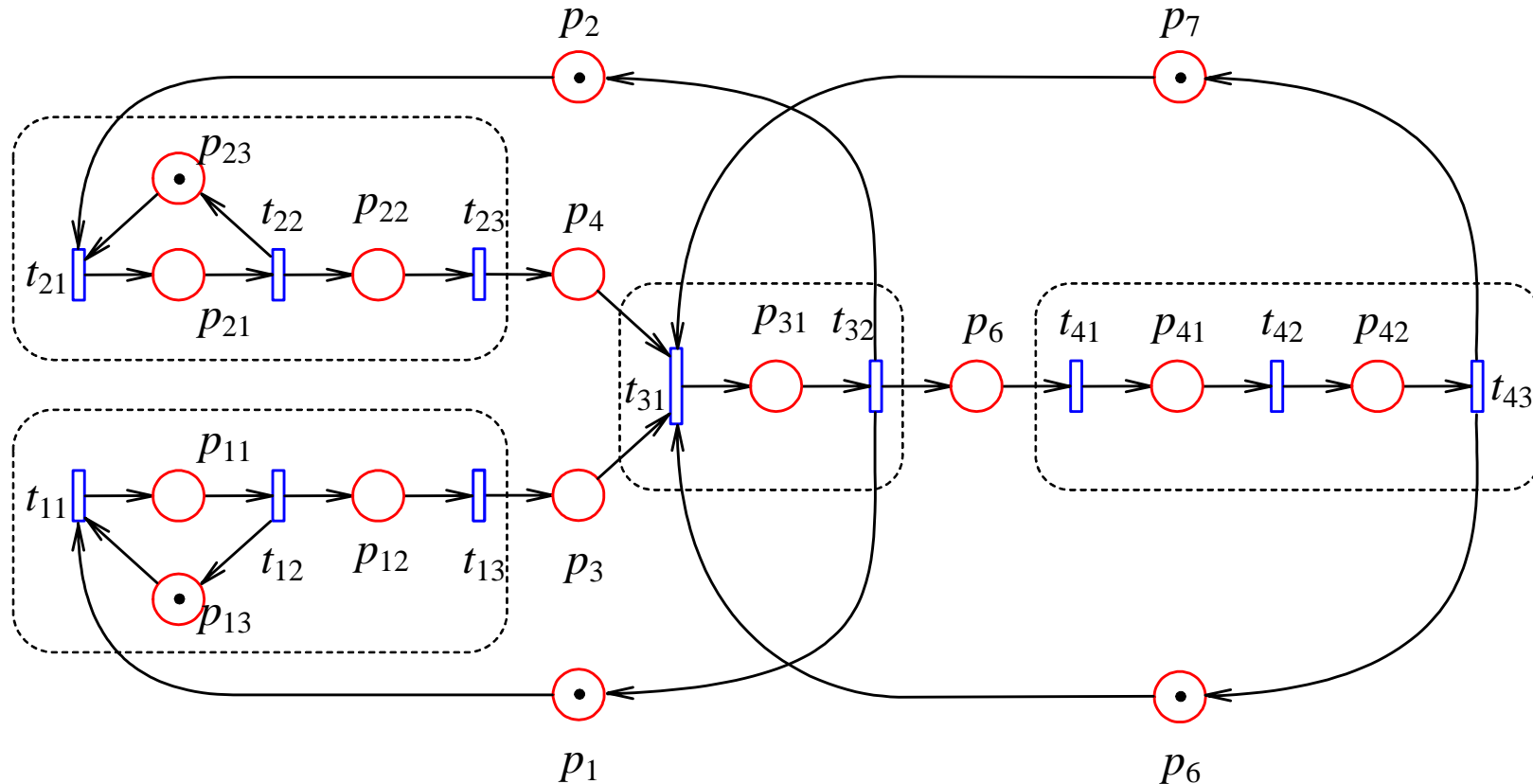
Significato nuovo posto:
 p_{31}) scarico di M_1 e M_2 e carico di A

Infine, l'operazione di assemblaggio può essere dettagliata nell'effettiva lavorazione e il successivo scarico della cella.



Significato nuovi posti:
 p_{41}) R_1 opera l'assemblaggio in A
 p_{42}) R_1 scarica A

Nel modello finale le attività sono diventate 9 e sono modellizzate da posti.



Considerazioni conclusive sul metodo top-down

Vantaggi:

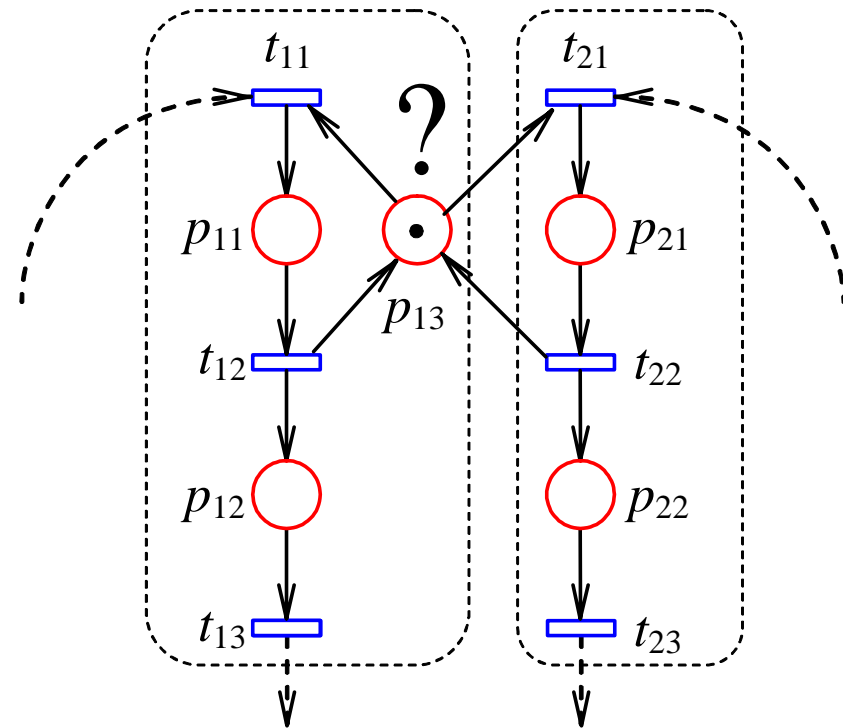
- ▶ garanzia che la rete di Petri finale soddisfi importanti proprietà strutturali
- ▶ semplicità → facile implementazione in ambienti CAD dedicati

Svantaggi:

- ▶ non è semplice garantire la coerenza interpretativa del modello (dipende dal livello di espansione delle varie parti)
- ▶ impossibilità di inserire blocchi non indipendenti

Se, per esempio, nell'esempio precedente ci fosse stato un robot solo per le operazioni di carico macchina (risorsa condivisa), il metodo non si sarebbe potuto applicare.

Questo è un limite notevole, perchè la presenza di risorse condivise è assai comune nei sistemi manifatturieri e non è ovvio il livello al quale modellarle.



Allora, o si modellizzano subito le risorse condivise nella rete di primo livello, oppure occorre che tutte le risorse condivise siano confinate all'interno di blocchi. In ogni caso, il metodo top-down è molto rigido e si adatta bene a sistemi essenzialmente sequenziali, ma non altrettanto a sistemi concorrenti con elevati accoppiamenti dovuti a condivisione delle risorse.

Metodi bottom-up

Nei metodi bottom-up si sviluppano dei sotto-modelli (*moduli*) indipendenti, che poi vengono connessi, condividendo (per *fusione*) alcuni elementi della rete.

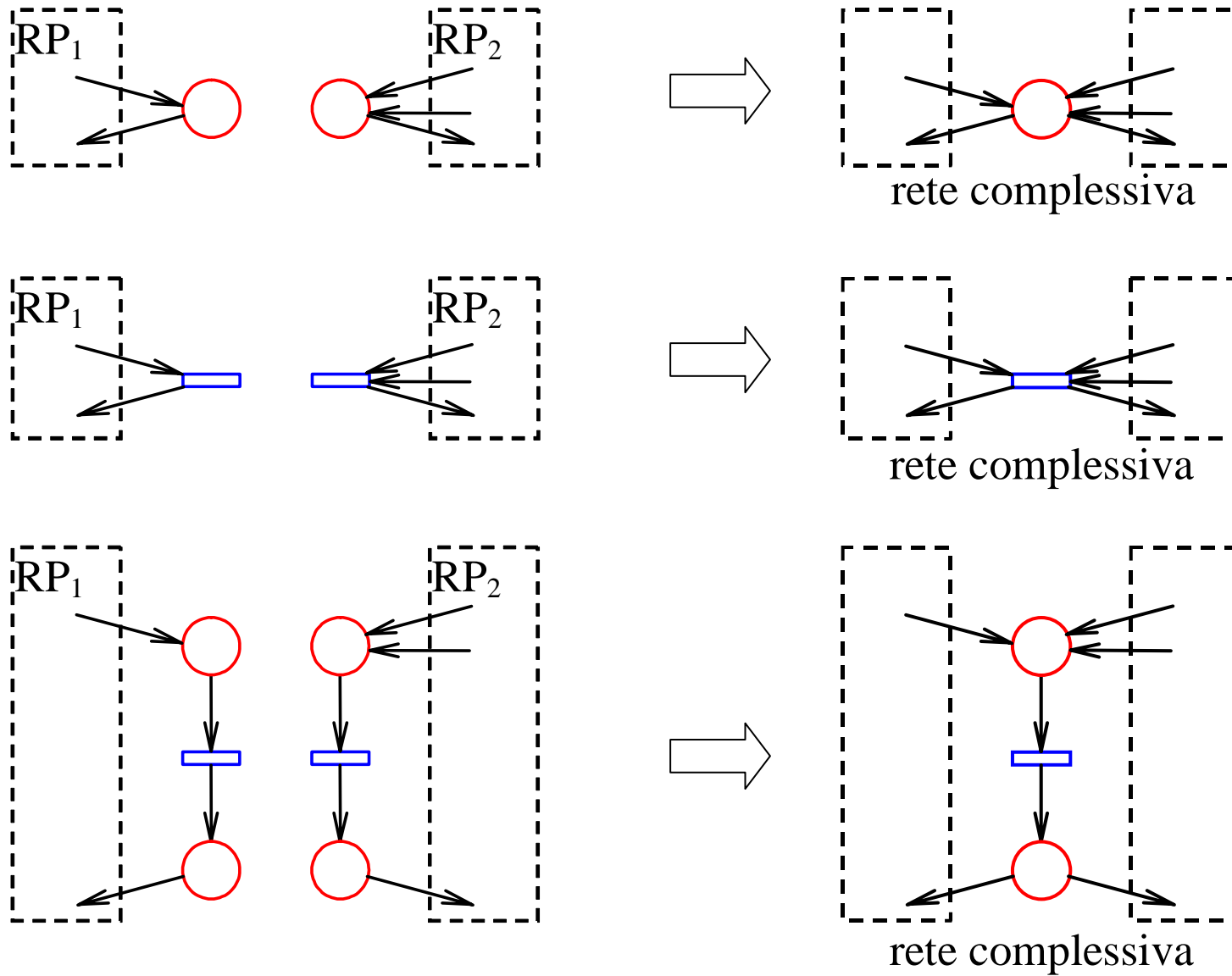
Si possono fondere posti, transizioni, percorsi (*path*) particolari (sequenze, alternative, fork-join, ecc.)

In generale i metodi bottom-up forniscono risultati poco potenti:

- ▶ non garantiscono il mantenimento delle proprietà fondamentali nel corso del progetto;
- ▶ i metodi che garantiscono qualche proprietà ci riescono a prezzo di semplificazioni estreme del modello;

Tuttavia,

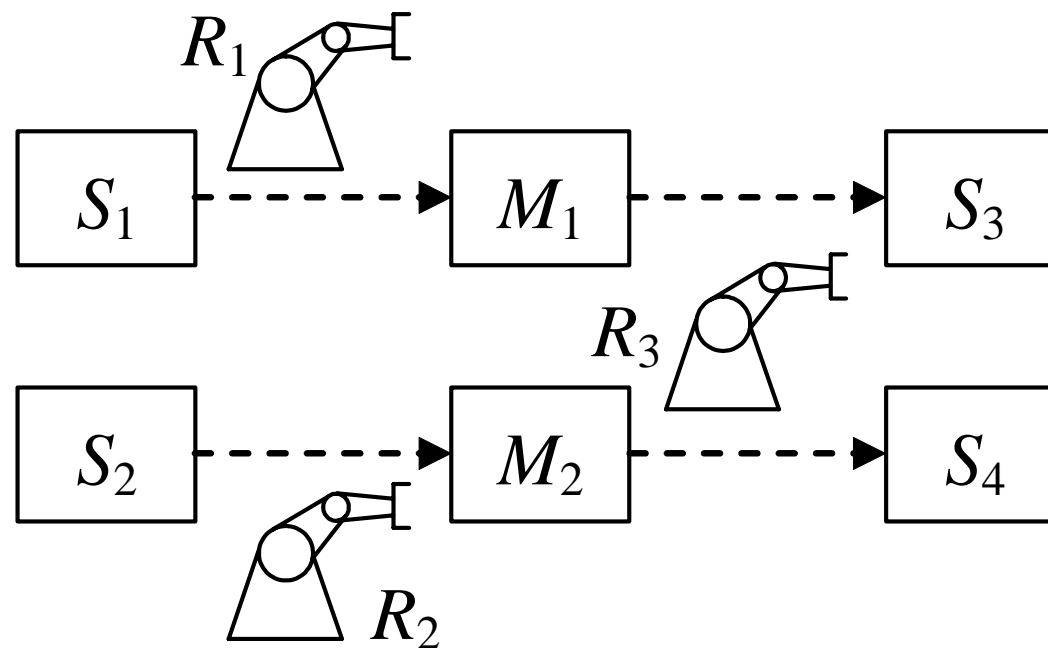
- ▶ sono intrinsecamente adatti al riutilizzo dei modelli;
- ▶ se si parte da moduli “ben formati” (p.es. coperti da P- e da T-invarianti, vivi e reversibili), è facile individuare le cause del mancato rispetto di qualche proprietà fondamentale.



Esempio di progetto bottom-up

Si consideri il sistema manifatturiero automatizzato rappresentato in figura, dove:

- ▶ S_1 e S_2 sono magazzini di pezzi grezzi,
- ▶ M_1 e M_2 sono machine,
- ▶ R_1 , R_2 e R_3 sono robot manipolatori di trasporto,
- ▶ S_3 e S_4 sono i magazzini dei prodotti finiti.

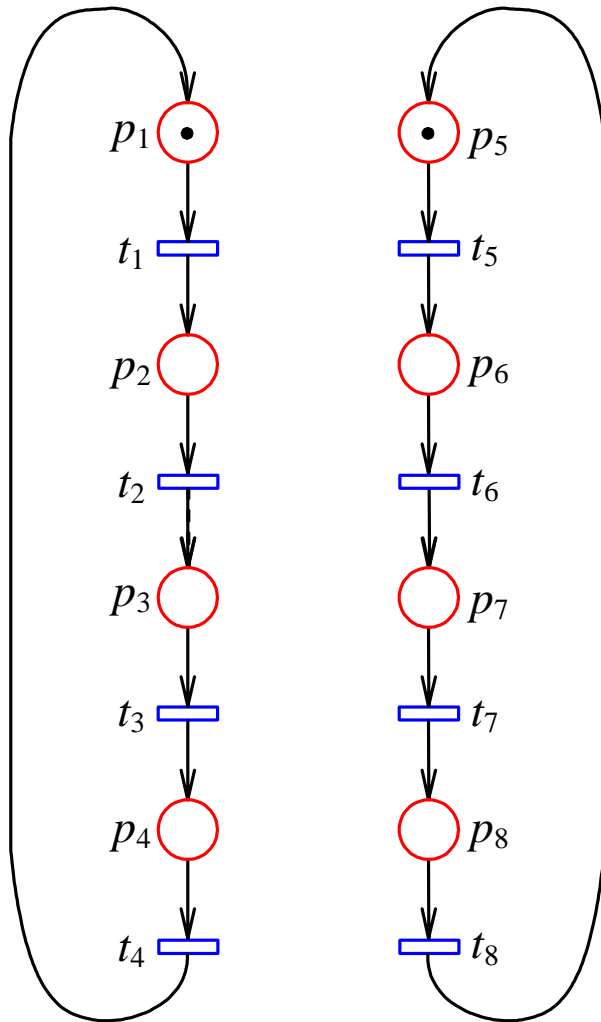
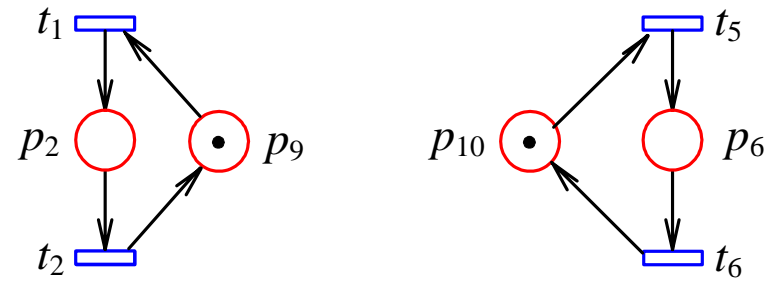
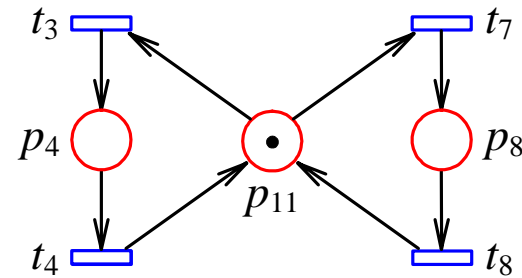


La procedura di lavorazione è la seguente:

- ❶ R_1 [R_2] preleva un pezzo da S_1 [S_2] e lo carica su M_1 [M_2].
- ❷ M_1 ed M_2 effettuano le rispettive lavorazioni.
- ❸ Al termine della lavorazione R_3 preleva il prodotto da M_1 [M_2] e lo pone in S_3 [S_4].

Per ognuno dei dispositivi dell'impianto è possibile individuare un ciclo di lavorazione, come nelle figure seguenti, dove il significato dei posti è il seguente:

- p_1 [p_5] – macchina M_1 [M_2] disponibile
 p_2 [p_6] – muovi un pezzo da S_1 [S_2] a M_1 [M_2]
 p_3 [p_7] – M_1 [M_2] lavora
 p_4 [p_8] – muovi un pezzo da M_1 [M_2] a S_3 [S_4]
 p_9 [p_{10}] – robot R_1 [R_2] disponibile
 p_{11} – robot R_3 disponibile

cicli di lavorazione di M_1 e M_2 cicli di lavorazione di R_1 e R_2 cicli di lavorazione di R_3

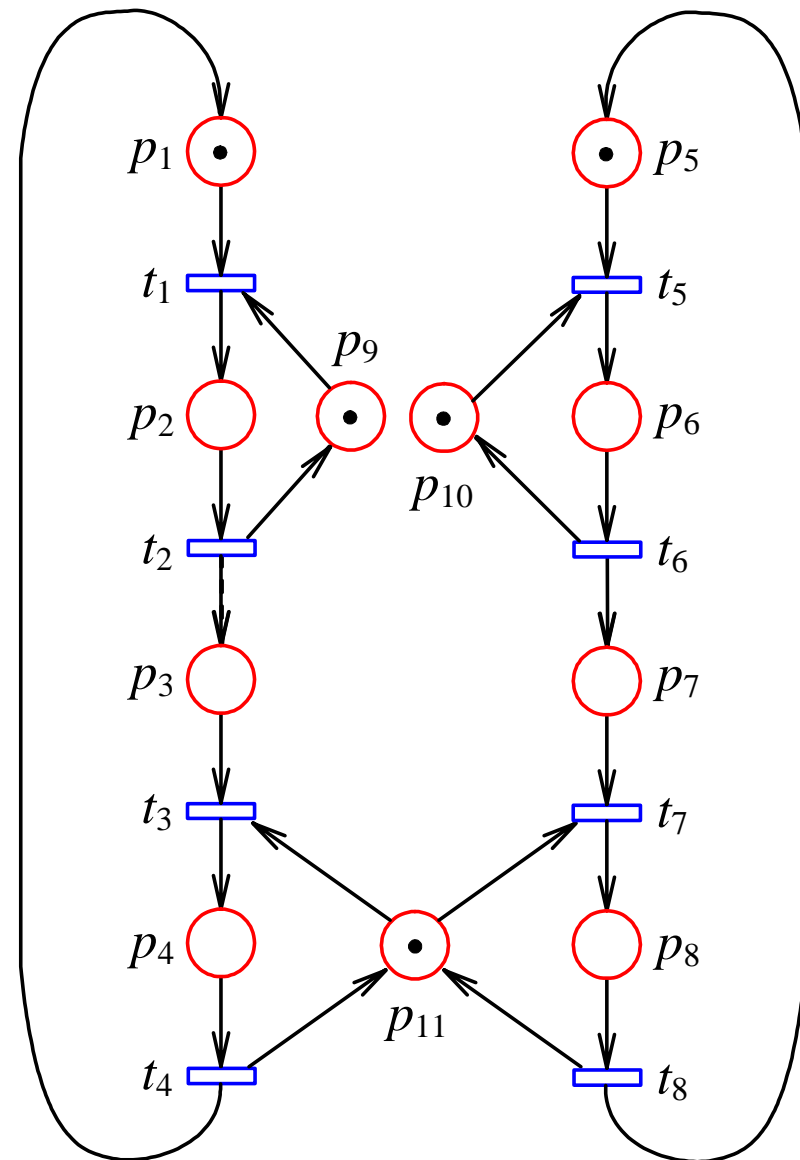
Le operazioni di fusione si riferiscono tutte a sequenze di 3 elementi (transizione-posto-transizione):

- ▶ $t_1-p_2-t_2 \rightarrow$ il carico del pezzo richiede sia R_1 che M_1
- ▶ $t_5-p_6-t_6 \rightarrow$ il carico del pezzo richiede sia R_2 che M_2
- ▶ $t_3-p_4-t_4 \rightarrow$ lo scarico del pezzo richiede sia R_3 che M_1
- ▶ $t_7-p_8-t_8 \rightarrow$ lo scarico del pezzo richiede sia R_3 che M_1

La rete risultante è rappresentata in figura.
Essa contiene una sola risorsa condivisa, è viva e binaria.

In generale non c'è garanzia che la rete finale sia viva (anche se lo sono i singoli moduli).

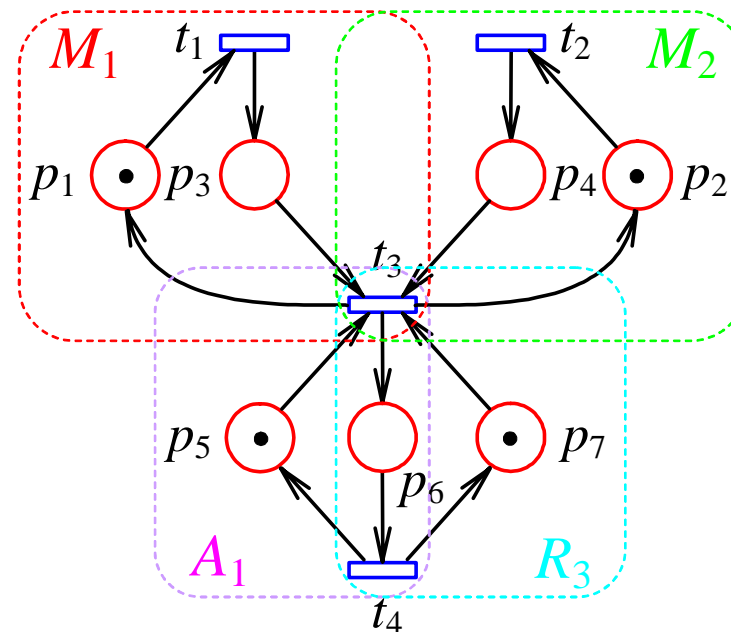
Ad esempio, se ci fossero 2 risorse condivise (R_{12} e R_3) e i due flussi produttivi fossero in senso opposto (uno utilizza prima R_{12} e poi R_3 , l'altro prima R_3 e poi R_{12}), il sistema può finire in deadlock.



Metodi di progetto ibridi

Si utilizzano in modo combinato tecniche top-down e bottom-up per avere i vantaggi di entrambi, combinando la flessibilità di utilizzo con la possibilità di analisi.

Ad esempio la rete di primo livello usata in precedenza può essere ricavata aggregando 4 semplici moduli che rappresentano M_1 , M_2 , R_3 e A .



Reti gerarchiche

Nella modellizzazione di sistemi complessi è importante poter scomporre il modello in sotto-modelli, sfruttando una qualche struttura gerarchica.

Un metodo (ibrido) di questo genere è basato sul metodo di affinamento di posti di Valette, ma aggrega reti.

Non si *sostituiscono* elementi della rete, ma si *sincronizza* la rete contenente il posto da affinare (*posto radice*) con la sottorete che lo affina.

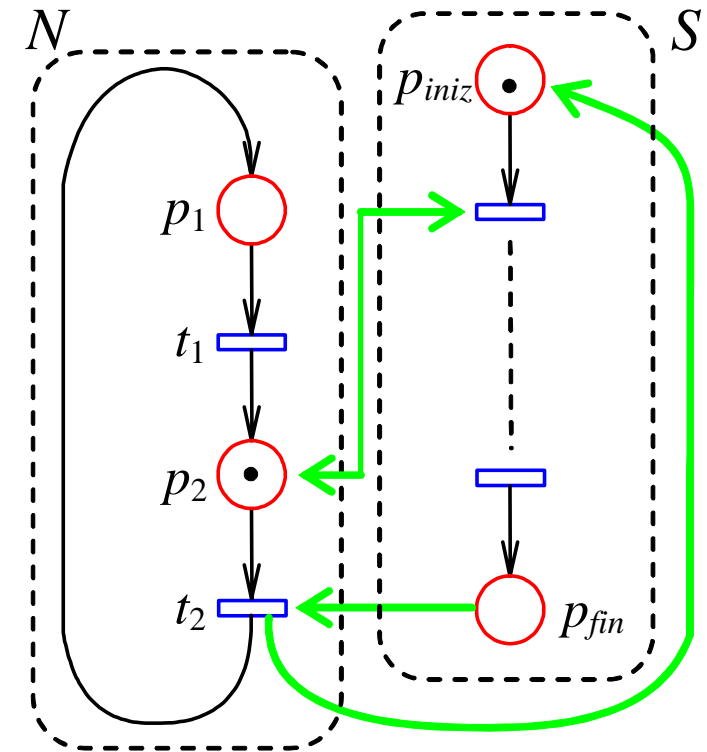
In tal modo il posto radice e la sottorete coesistono, coordinati opportunamente.

Si consideri la rete di primo livello N riportata in figura, dove p_1 rappresenta un posto di attesa e p_2 (il posto radice) l'esecuzione di un programma A .

Il programma A è rappresentato da una rete di secondo livello S , caratterizzata dall'avere un unico posto di partenza p_{iniz} ($\bullet p_{iniz} = \emptyset$) e un unico posto di finale p_{fin} ($p_{fin} \bullet = \emptyset$).

Una rete di questo tipo è detta *P-blocco*.

Si vuole connettere opportunamente N ad S in modo che A venga eseguito quando viene posto un gettone in p_2 e che quest'ultimo non possa essere rimosso finché A non sia terminato.



Questa connessione, detta *P-connessione* è ottenuta nel modo seguente:

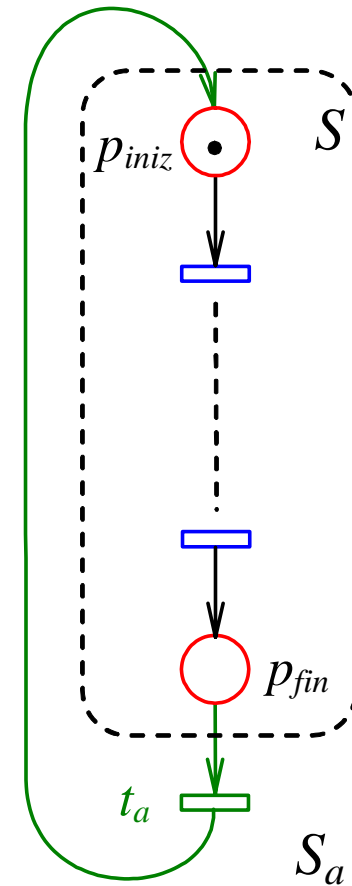
- ▶ si inserisce un autoanello tra p_2 e ogni transizione in uscita da p_{iniz} ;
- ▶ si inserisce un arco da p_{fin} ad ogni transizione in uscita da p_2 ;
- ▶ si inserisce un arco da ogni transizione in uscita da p_2 a p_{iniz} .

Si chiama *P-blocco aggiunto* la rete ottenuta aggiungendo al P-blocco un transizione che sia in uscita da p_{fin} e in ingresso a p_{iniz} , e marcata inizialmente con un gettone in p_{iniz} .

Il P-blocco si dice ben definito se e solo se il corrispondente P-blocco aggiunto è vivo e binario.

Allora la rete F ottenuta operando una P-connes-
sione tra la rete N e il P-blocco ben definito S ha le
seguenti proprietà:

- ▶ F viva $\Leftrightarrow N$ viva
- ▶ F binaria $\Leftrightarrow N$ binaria
- ▶ F reversibile $\Leftrightarrow N$ reversibile



Evoluzione della rete F :

- ▶ la sottorete S non può essere eseguita finché p_2 non è marcato;
- ▶ se p_2 è marcato, t_2 non è abilitata e può evolvere solo la sottorete S ;
- ▶ quando, durante l'evoluzione della sottorete S , si marca p_{fin} (ciò avviene sicuramente poiché il P-blocco S è ben definito), t_2 risulta abilitata, mentre S non può evolvere ulteriormente;
- ▶ quando scatta t_2 , la rete N può continuare ad evolvere liberamente e il P-blocco è nuovamente marcato con un gettone in p_{iniz} .

In altre parole, quando scatta t_1 , il gettone si congela nel posto p_2 per consentire l'esecuzione della sottorete.

Alla fine dell'esecuzione di quest'ultima, l'evoluzione della rete N riprende.

L'evoluzione della rete N non è influenzata dalla P-connessione, ma solo ritardata.