Metodi Formali

UniShare

Davide Cozzi
@dlcgold

Indice

Capitolo 1

Introduzione

Questi appunti sono presi a lezione. Per quanto sia stata fatta una revisione è altamente probabile (praticamente certo) che possano contenere errori, sia di stampa che di vero e proprio contenuto. Per eventuali proposte di correzione effettuare una pull request. Link: `https://github.com/dlcgold/Appunti`.
Grazie mille e buono studio!

1.1 Contenuti del Corso

Il corso tratta di metodi e tecniche formali per specificare, disegnare e analizzare sistemi complessi, in particolare sistemi concorrenti e distribuiti costituiti da componenti che operano in modo indipendente e che interagiscono tra loro.

Si usa un linguaggio logico che spiega il comportamento di tali sistemi e fa riferimento alla **logica temporale** di tali sistemi, in quanto le proprietà di tali sistemi sono tali per cui evolvono con il cambiamento di stato del sistema e quindi serve una logica che descriva le proprietà dell'evoluzione del comportamento.
Si parlerà delle **Reti di Petri**, ovvero uno strumento per modellare tali sistemi concorrenti e distribuiti. Questo modello ha intrinsechi dei teoremi matematici atti a studiare il comportamento di tali sistemi.
In laboratorio si studieranno algoritmi e strumenti software per la modellazione e l'analisi di tali sistemi.
Si introducono a che sistemi dinamici a tempi discreti, come gli **automi cellulari**.

Capitolo 2

Sviluppo di Modelli e Sistemi

Si hanno diverse fasi di sviluppo di *sistemi complessi* (nel nostro caso **concorrenti** e **distribuiti**). Si hanno 4 grandi fasi (che riprendono le generiche fasi dello sviluppo software), che non seguono una rigida sequenza cronologica tra di loro:

1. specifica del problema e delle proprietà della soluzione
2. modellazione della soluzione
3. implementazione
4. verifica, validazione e collaudo, sia sul modello che implementazione (con eventuali modifiche)

Queste fasi possono alternarsi a vicenda.
I metodi formali possono svolgere una parte rilevante in tutte queste 4 fasi e hanno la prerogativa di sviluppare questi sistemi in maniera corretta e persistente.
Ci si focalizza sulla modellazione e sulla specifica delle proprietà. Si studia inoltre la verifica delle proprietà sul modello costruito. *In questo corso si lascia un attimo da parte l'aspetto implementativo, che comunque seguirebbe alla verifica e alla validazione del metodo.*
Si hanno diversi modelli di sistemi concorrenti e distribuiti, presenti in letteratura:

- **Algebre di Processi**, ovvero una miriade di diversi linguaggi, studiate inizialmente da Milner, che introdusse il calcolo dei sistemi comunicanti, un calcolo algebrico utile alla semantica della concorrenza. Inoltre Hoare ha introdotto i **processi sequenziali comunicanti** come un nucleo di linguaggio di programmazione,

usato come linguaggio macchina per le prime macchine parallele. Queste algebre si basano sul paradigma di avere un forte aspetto della **composizionalità**, in quanto un sistema viene visto come costituito da diverse componenti autonome (sia hardware, che software, che umane) che interagiscono tra loro sincronizzandosi (in modo sincrono, *handshaking*, sfruttando un "canale di comunicazione" che viene modellato come un processo) e scambiandosi messaggi. Questo paradigma è anche alla base dello sviluppo di molti linguaggi di programmazione specificatamente dedicati alla concorrenza.

- **Automi a Stati Finiti**. Un modello concorrente e distribuito viene spesso rappresentato attraverso **sistemi di transizioni etichettati**, che sono una derivazione del modello degli automi a stati finiti, già usati in letteratura per modellare reti neurali, progettare circuiti asincroni, modellare macchine a stati finiti, riconoscere linguaggi regolari (il teorema di Kleene ci ricorda che *ad un automa a stati finiti è possibile associare un'espressione regolare*) e per la modellazione di protocolli di comunicazione.

- **Reti di Petri**, introdotte da Petri con la **teoria generale delle reti di Petri** nella sua tesi di dottorato. Questa teoria parte da una critica al modello a stati finiti dove il focus è su stati globali e trasformazione di stati globali. Petri cercava invece una teoria matematica (fondata sui principi della fisica moderna della relatività e della quantistica) che fosse una teoria dei sistemi in grado di descrivere sistemi complessi in cui mettere al centro il flusso di informazione e che potesse permettere di analizzare l'organizzazione dal punto di vista del flusso di informazione che passa da una componente all'altra. Non si ha il focus, quindi, su "macchine calcolatrici" ma come supporto alla comunicazione in organizzazioni complesse. Si hanno quindi diversi elementi chiave:

  - la comunicazione
  - la sincronizzazione tra componenti
  - il flusso di informazione che passa tra le varie componenti
  - la relazione di concorrenza e l'indipendenza causale tra i vari eventi che comportano i cambiamenti di stato. Ci si concentra su stati locali e non sulla visione di una sequenza di azioni e di uno stato globale

La teoria delle reti di Petri è stata poi sviluppata e ha avuto diverse applicazioni. Sono stati sviluppati diversi linguaggi, ovvero diverse **classi di reti di Petri** per descrivere un sistema complesso a livelli differenti di astrazione.
Sono state anche sviluppate tecniche formali di analisi e di verifica del modello (disegnato mediante reti di Petri), basate sulla teoria dei grafi e sull'algebra lineare.
Le reti di Petri hanno avuto un notevole utilizzo in diversi ambiti applicativi anche estranei all'informatica pura e allo studio della concorrenze, come la modellazione di sistemi biologici o la modellazione di reazioni chimiche. Mediante una classe di reti particolare, le **reti stocastiche** si può valutare le prestazioni di un determinato modello.

Sistemi di Transizioni Etichettati

Definizione 1. *I sistemi di transizione etichettati sono definiti come gli automi a stati finiti ma senza essere visti come riconoscitori di linguaggi infatti un sistema è formato da un insieme, solitamente finito, di stati globali S. Si ha poi un alfabeto delle possibili azioni che può eseguire il sistema. Si hanno anche delle relazioni di transizioni, ovvero delle transizioni che permettono di specificare come, attraverso un'azione, si passa da uno stato ad un altro. Le transizioni si rappresentano con archi etichettati tra i nodi, che rappresentano gli stati. Le etichette degli archi rappresentano le azioni necessarie alla trasformazione. L'insieme delle azioni viene chiamato E mentre $T \subseteq S \times E \times S$ è l'insieme degli archi etichettati. Può essere, opzionalmente, individuato uno stato iniziale s_0. Un sistema non è obbligato a "terminare", quindi non si ha obbligatoriamente uno stato finale.*
Riassumendo quindi un sistema di transizione etichettato è un quadrupla:

$$A = (S, E, T, s_0)$$

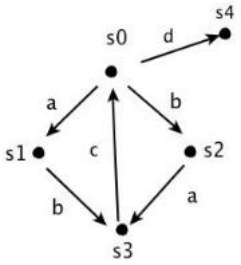


Figura 2.1: Esempio di sistema di transizione etichettato

La critica di Petri è che in un sistema distribuito non sia individuabile uno **stato globale**, che in un sistema distribuito le trasformazioni di stato siano **localizzate** e non globali, che non esista un sistema di riferimento temporale unico (si possono avere più assi temporali in un sistema distribuito). Quindi la simulazione sequenziale non deterministica (emantica a "interleaving") dei sistemi distribuiti è una forzatura e non rappresenta le reali caratteristiche del comportamento del sistema, ovvero la località, la distribuzione degli eventi e la relazione di dipendenza causale e non causale tra gli eventi.

2.1 Sistemi Elementari

Per introdurre i sistemi elementari delle reti di Petri, ovvero una classe molto semplice e astratta partiamo da un esempio:

Esempio 1. *Vediamo l'esempio del Produttore e del Consumatore.*
Si ha un sistema con una componente Produttore che produce elementi e li deposita in un buffer che ha un'unica posizione (quindi o è pieno o è vuoto) e con un consumatore che preleva dal buffer un elemento per poi consumarlo ed essere pronto a prelevare un altro elemento. Si ha un comportamento ciclico. Usiamo quindi le reti di Petri, col modello dei sistemi elementari, per rappresentare questo modello. Bisogna quindi individuare le proprietà fondamentali locali del sistema.
Partiamo dal produttore, che può avere 2 stati locali:

1. *pronto per produrre*
2. *pronto per depositare*

*Usiamo i **cerchi** per rappresentare condizioni locali che sono associabili a delle proposizioni della logica che possono essere vere o false. Queste preposizioni sono quindi stati locali. Gli eventi locali vengono invece rappresentati con un **rettangolo**. Un evento ha un arco entrante da uno stato che rappresenta le precondizioni di quell'evento (che devono essere vere per permettere l'occorrenza dell'evento). L'occorrenza dell'evento rende false le precondizioni e rende vere le postcondizioni (che sono stati raggiungibili con un arco uscente da un evento). Si ha quindi che il produttore può depositare solo se il buffer non è pieno, quindi le postcondizioni di un evento devono essere false affinché l'evento possa occorrere (oltre alle precondizioni vere).*
Passiamo al consumatore che estrae solo se il buffer è pieno ed è pronto a prelevare. Si procede poi con la stessa logica del produttore di cambiamento tra vero e falso delle varie condizioni locali.

In questo esempio si hanno quindi condizioni che sono preposizioni booleane e rappresentano stati locali.

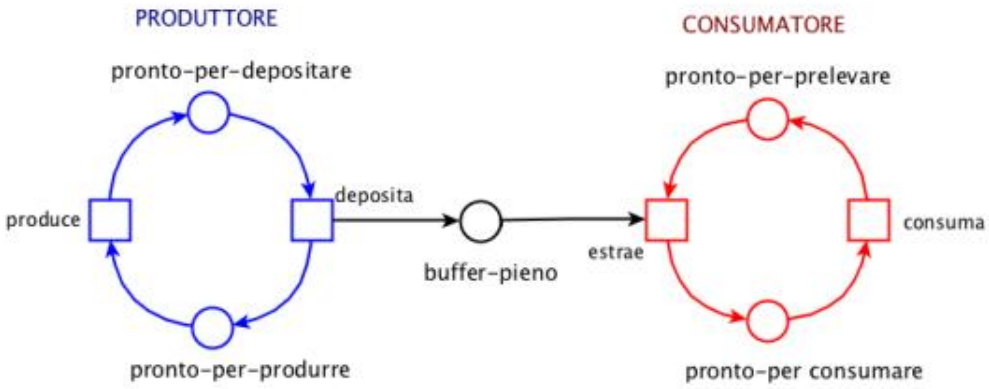


Figura 2.2: Produttore e Consumatore

Lo stato globale del sistema è dato da una collezione di stati locali. Per segnare tali condizioni mettiamo un punto pieno dentro il cerchio e queste condizioni "abilitano" i vari eventi: Si può arrivare ad una configurazione

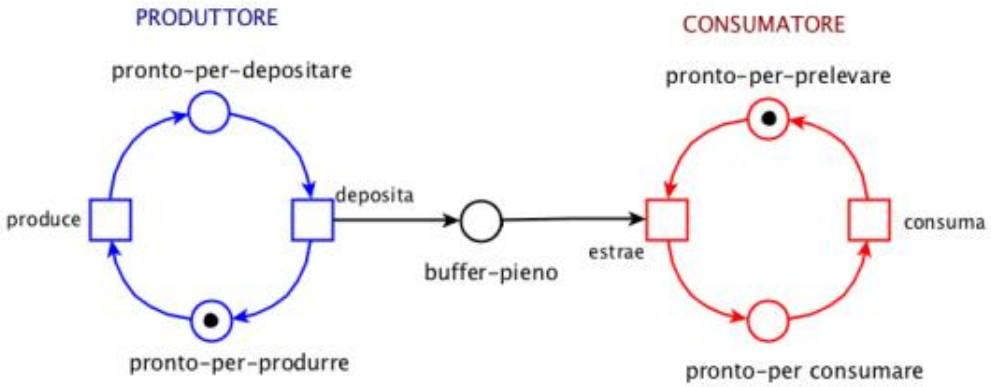


Figura 2.3: Uno stato globale Produttore e Consumatore dove l'evento *produce* è l'unico abilitato

dove, per esempio, sia l'evento produce, del produttore, che l'evento preleva, del consumatore, sono abilitati. Si ha quindi che i due eventi possono

occorrere in modo ***concorrente*** *infatti i due eventi sono* ***indipendenti*** *in quanto condizionati da* ***precondizioni e postcondizioni completamente disgiunte****. Due eventi che occorrono in maniera concorrente lo possono fare in qualsiasi ordine, non si ha infatti una sequenza temporale specifica tra i due.*
In questo sistema quindi siano solo stati locali ed eventi localizzati e non stati ed eventi globali. Un evento dipende solo dalle sue precondizioni e dalle sue postcondizioni.
Se rappresentiamo con delle marche le condizioni vere possiamo simulare il comportamento del sistema con il gioco delle marche che mostra come l'evoluzione delle condizioni avviene all'occorrenza degli eventi.
La simula di un tale sistema può comunque avvenire con un sistema di transizioni etichettato, ovvero con un automa a stati finiti, che rappresenta gli stati globali corrispondenti alle diverse combinazioni di stati locali che di volta in volta sono veri. Gli archi vengono etichettati con gli eventi che comportano un cambiamento di stato globale:

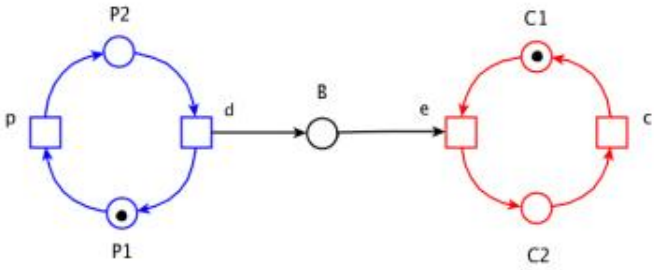


Figura 2.4: Semplificazione della nomenclatura del sistema per praticità

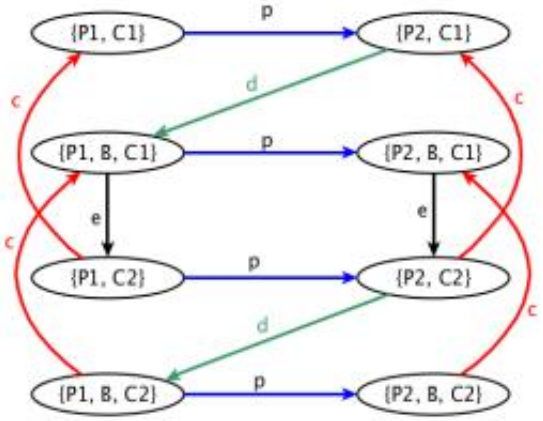


Figura 2.5: Rappresentazione del sistema con un automa a stati finiti che rappresenta stati globali

Passiamo ora alla formalizzazione di questi aspetti.

Definizione 2. *Una **rete elementare** è definita come una tripla:*

$$N = (B, E, F)$$

dove:

- *B è un insieme finito di **condizioni**, ovvero stati locali, preposizioni booleane etc.... Vengono rappresentate con un cerchio*
- *E è un insieme finito di **eventi**, ovvero trasformazioni locali di stato e transizioni locali. Vengono rappresentate con un quadrato*
- *F è una **relazione di flusso** che connette condizioni ad eventi ed eventi a condizioni. Si ha quindi che:*

  $$F \subseteq (B \times E) \cup (E \times B)$$

  *Le relazioni di flusso sono rappresentate da archi orientati. Inoltre la relazione di flusso è tale per cui non esistano **elementi isolati**, in quanto non avrebbero senso, in un tale sistema, eventi isolati (che non modificherebbero mai una condizione) o condizioni isolate (che non verrebbero mai modificate da un evento). Si ha, formalmente, che:*

  $$dom(F) \cup ran(F) = B \cup E$$

  chiedere per formula sopra

Si ha che:

$$B \cap E = \emptyset$$

$$B \cup E \neq \emptyset$$

Ovvero gli insiemi delle condizioni e degli eventi sono tra loro disgiunti e non vuoti.
Sia ora x un elemento qualsiasi della rete, ovvero x può essere o una condizione o un evento, formalmente:

$$x \in B \cup E$$

si ha che:

- ${}^\bullet x = \{y \in X : (y, x)\}$ *rappresenta l'insieme di tutti gli elementi* y *che sono connessi dalla relazione di flusso ad* x*, ovvero si ha un arco da* y *a* x*. Sono quindi i* ***pre-elementi*** *di* x*, ovvero le precondizioni, se* x *è un evento, o i pre-eventi, se* x *è una condizione*

- $x^\bullet = \{y \in X : (x, y)\}$ *rappresenta l'insieme di tutti gli elementi* y *che sono connessi dalla relazione di flusso a partire da* x*, ovvero si ha un arco da* x *a* y*. Sono quindi i* ***post-elementi*** *di* x*, ovvero le postcondizioni, se* x *è un evento, o i post-eventi, se* x *è una condizione*

Posso estendere questa notazione ad insiemi di elementi. Sia A *un insieme qualsiasi di elementi, che possono quindi essere sia condizioni che eventi:*

$$A \subseteq B \cup E$$

Si ha quindi che i pre-elementi dell'insieme A *sono rappresentati con:*

$${}^\bullet A = \cup^\bullet_{x \in A} x$$

ovvero l'unione dei pre-elementi di ogni singolo elemento dell'insieme A*.*
Analogamente si ha che i post-elementi dell'insieme A *sono rappresentati con:*

$$A^\bullet = \cup_{x \in A} x^\bullet$$

ovvero l'unione dei post-elementi di ogni singolo elemento dell'insieme A*. Nelle reti c'è sempre una relazione di* ***dualità*** *tra due elementi, per esempio tra condizioni ed eventi, tra pre-eventi e post-eventi, tra pre-condizioni e post-condizioni. Inoltre si ha la caratteristica della* ***località****, quindi si hanno stati locali e trasformazioni di stato locali*

La rete $N = (B, E, F)$ descrive la *struttura statica del sistema*, il comportamento é definito attraverso le nozioni di **caso (o configurazione)** e di **regola di scatto (o di transizione)**.
Una rete può anche essere suddivisa in sotto-reti, seguendo l'esempio sopra si potrebbe avere una sotto-rete per il produttore, una per il consumatore e anche una per il buffer.

Definizione 3. *Un* ***caso*** *(o* ***configurazione****) é un insieme di condizioni* $c \subseteq B$ *che rappresentano l'insieme di condizioni vere in una certa configurazione del sistema, un insieme di* ***stati locali*** *che collettivamente individuano lo* ***stato globale*** *del sistema.*
Graficamente le condizioni vere presentano un puntino in mezzo al cerchio mentre le condizioni false solo un cerchio vuoto

Definizione 4. *Sia $N = (B, E, F)$ una rete elementare e sia $c \subseteq B$ una certa configurazione (non serve quindi necessariamente conoscere tutto lo stato del sistema). La* ***regola di scatto*** *mi permette di stabilire quando un evento $e \in E$ è abilitato, ovvero può occorrere, in c sse:*

$$^\bullet e \subseteq c \ e \ e^\bullet \cap c = \emptyset$$

ovvero sse tutte le precondizioni dell'evento sono vere (e quindi sono contenute nella configurazione c) e sse tutte le postcondizioni sono false (quindi non si hanno intersezioni tra le postcondizioni e la configurazione).
L'occorrenza (l'abilitazione) di e in c si denota con la scrittura:

$$c[e >$$

Se un evento e è abilitato in c, ovvero $c[e >$, si ha che quando e occorre in c genera un nuovo caso c' e si usa la notazione:

$$c[e > c'$$

Si ha quindi che c' è così calcolabile:

$$c' = (c -^\bullet e) \cup e^\bullet$$

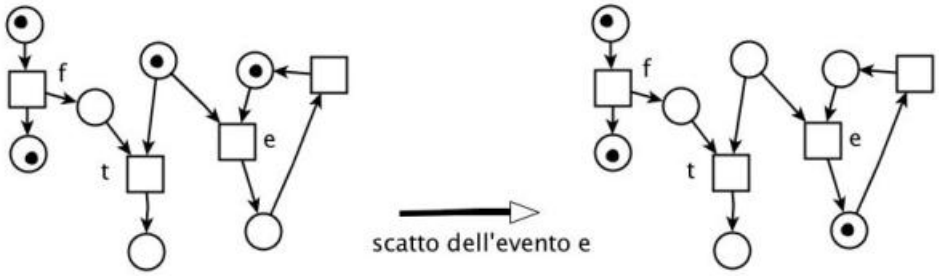
Ovvero togliendo da c tutte le precondizioni dell'evento e e aggiungendo quindi tutte le postcondizioni di e

Le reti si basano sul **principio di estensionalità**, ovvero sul fatto che il cambiamento di stato è locale:

un evento è completamente caratterizzato dai cambiamenti che produce negli stati locali, tali cambiamenti sono indipendenti dalla particolare configurazione in cui l'evento occorre.

L'importante è che le precondizioni di un evento siano vere e le postcondizioni false (siamo comunque interessati solo alla validità delle condizioni che riguardano l'evento).

Esempio 2. *Vediamo un esempio esplicativo dove l'evento e è l'unico abilitato, ovvero le sue precondizioni sono vere e le sue postcondizioni sono false. Lo scatto di e rende le precondizioni false e le postcondizioni vere, mentre le altre condizioni rimangono inalterate:*



Si nota quindi che lo scatto dell'evento e riguarda solo le precondizioni e le postcondizioni di quel dato evento, come ci ricorda il principio di estensionalità

Definizione 5. *Sia $N = (B, E, F)$ una rete elementare. Possiamo definire due tipologie di rete:*

1. *N è definita **semplice** sse:*

$$\forall x, y \in B \cup E,\ ({}^\bullet x = {}^\bullet y) \wedge (x^\bullet = y^\bullet) \Rightarrow x = y$$

Ovvero per ogni coppia di elementi (che siano quindi eventi o condizioni) se i loro pre-elementi e i loro post-elementi coincidono allora non ha senso distinguere x e y.

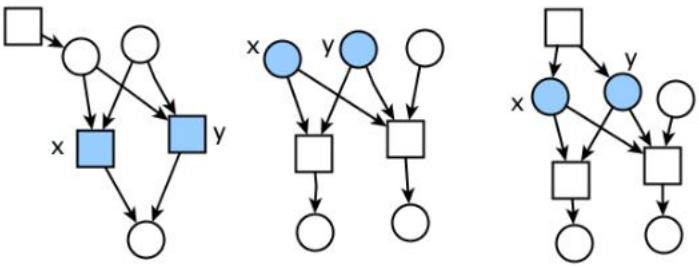


Figura 2.6: Esempi di reti **non** semplici

2. *N è definita **pura** sse:*

$$\forall e \in E : \ {}^\bullet e \cap e^\bullet = \emptyset$$

*Ovvero se per ogni evento non esiste una precondizione che sia anche postcondizioni. Si ha quindi un **cappio** (detto anche **side condition**) tra un evento e una condizione. Avere questa situazione comporta che l'evento non può scattare in quanto la condizione che per lui è sia una precondizione che una postcondizioni non può essere contemporaneamente vera e falsa, l'evento non potrà mai scattare e quindi non potrà mai essere osservato. Non avrebbe quindi senso modellarlo*

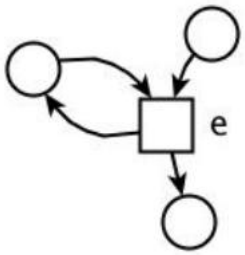


Figura 2.7: Esempio di rete **non** pura

Definizione 6. *Data una rete elementare $N = (B, E, F)$ e sia $U \subseteq E$ un sottoinsieme di eventi e siano $c, c_1, c_2 \in B$ tre configurazioni. Si ha che:*

- *U è un **insieme di eventi indipendenti** sse:*

$$\forall e_1, e_2 \in U : \ e_1 \neq e_2 \Rightarrow ({}^\bullet e_1 \cup e_1^\bullet) \cap ({}^\bullet e_2 \cup e_2^\bullet) = \emptyset$$

ovvero per ogni coppia distinta di eventi nell'insieme U si ha che le precondizioni e le postcondizioni dei due eventi sono completamente disgiunte.

- *U è un **passo abilitato**, ovvero un insieme di eventi concorrenti in una certa configurazione c, che si indica con:*

$$c[U >$$

sse:

$$U \text{ è un insieme di eventi indipendenti } \wedge \ \forall e \in U : \ c[e >$$

U quindi deve essere un insieme di eventi indipendenti e ogni evento in U è abilitato in c, quindi le sue precondizioni sono vere e le sue postcondizioni sono false. Si ha quindi che U è un insieme di eventi abitati in maniera concorrente in c

- *U è un **passo** dalla configurazione c_1 alla configurazione c_2, che si indica con:*

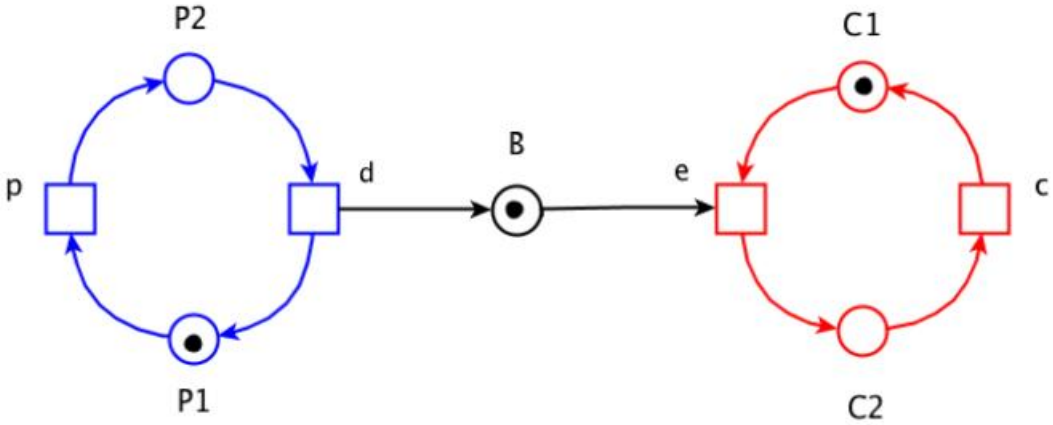
$$c_1[U > c_2$$

sse:

$$(c_1[U) \wedge \left(c_2 = (c_1 -^\bullet U) \cup U^\bullet\right)$$

ovvero sse U è un passo abilitato in c_1 e lo scatto degli eventi in U porta alla configurazione c_2 che si ottiene togliendo da c_1 l'insieme delle precondizioni degli eventi in U e aggiungendo quindi l'insieme delle postcondizioni degli eventi in U

Esempio 3. *Riprendiamo l'esempio del produttore e del consumatore. Sia dato il sistema Σ che modella produttore e consumatore*



Si hanno:

- *$\{p, e\}, \{p, c\}, \{d, c\}$ esempi di insiemi di eventi indipendenti*
- *$\{p, e\}$ che è un passo abilitato in $\{P_1, B, C_1\}$*
- *$\{P_1, B, C_1\}[\{p, w\} > \{P_2, C_2\}$ ovvero lo scatto del passo $\{p, e\}$ ci porta in $\{P_2, C_2\}$*

Diamo ora una definizione formale di **sistema elementare**.

Definizione 7. *Un **sistema elementare** $\Sigma = (B, E, F; c_{in})$ è definito come una rete $N = (B, E, F)$ e a cui è associato un caso iniziale, una configurazione iniziale, ovvero un sottoinsieme di condizioni che rappresentano lo stato iniziale da cui inizia la computazione e l'evoluzione del sistema. Formalmente il caso iniziale si indica con $C_{in} \in B$*

Definizione 8. *Dato un sistema elementare $\Sigma = (B, E, F; c_{in})$ si indica con C_Σ l'insieme dei **casi raggiungibili** da tale sistema a partire dal caso iniziale c_{in}.*
Formalmente l'insieme dei casi raggiungibili è il di piccolo sottoinsieme dell'insieme delle parti di B, ovvero 2^B, tale che:

- *$c_{in} \in C_\Sigma$, ovvero sicuramente il caso iniziale appartiene all'insieme dei casi raggiungibili*
- *se $c \in C_\Sigma$, $U \subseteq E$ e $c' \subseteq B$ sono tali che $c[U > c'$ allora $c' \in C_\Sigma$, ovvero se ho un generico caso c che appartiene ai casi raggiungibili, se ho un insieme di eventi U tale che questo insieme di eventi (che abbiamo visto essere indipendenti, per la definizione di passo abilitato) è abilitato in c in un unico passo e la sua occorrenza mi porta in c', allora anche c' appartiene a C_Σ.*

*Questa è una definizione data per **induzione strutturale**, nel primo punto si ha la base, nel secondo l'ipotesi e la conseguenza*

Definizione 9. *Dato un sistema elementare $\Sigma = (B, E, F; c_{in})$ si indica con U_Σ l'**insieme dei passi** di Σ, ovvero di tutti i possibili insiemi di eventi indipendenti che possono occorrere in qualche caso. Formalmente:*

$$U_\Sigma = \{U \subseteq E \mid \exists\, c, c' \in C_\Sigma :\ c[U > c'\}$$

Ovvero l'insieme dei sottoinsiemi di eventi tali per cui esistano due casi raggiungibili in C_Σ e U è abilitato in c e il suo scatto mi porta in c'.

Definiamo ora il comportamento dei sistemi elementari.

Definizione 10. *Sia $\Sigma = (B, E, F; c_{in})$ un sistema elementare e siano $c_i \in C_\Sigma$ ed $e_i \in E$.*
Definiamo;

- *un **comportamento sequenziale** come una sequenza di eventi che possono occorrere dal caso iniziale. Facendo scattare in maniera sequenziale gli eventi uno alla volta in c_n:*

$$c_{in}[e_1 > c_1[e_2 > \ldots [e_n > c_n$$

Scrittura che può essere alleggerita in:

$$c_{in}[e_1 e_2 \dots e_n > c_n$$

Possiamo dire di avere a che fare con una ***simulazione sequenziale non deterministica, detta anche semantica a interleaving****, infatti ho più eventi abilitati da prendere uno alla volta*

- *un* ***comportamento non sequenziale****, in quanto possiamo anche considerare insiemi di eventi, ovvero passi. Considero quindi sequenze di passi, avendo a che fare con la* ***step semantics****. Non ho quindi una simulazione sequenziale non deterministica in quanto dal caso iniziale faccio scattare un insieme di eventi, in maniera concorrente (e quindi senza ordine specificato), per poi far scattare un altro insieme di eventi fino ad arrivare a* c_n*:*

  $$c_{in}[U_1 > c_1[U_2 > \dots [U_n > c_n$$

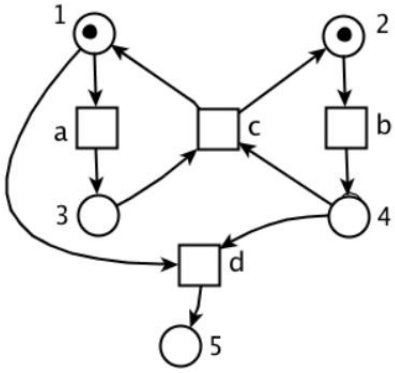
  Scrittura che può essere alleggerita in:

  $$c_{in}[U_1 U_2 \dots U_n > c_n$$

  Gli insiemi U_i *non sono insiemi massimali abilitati ma sottoinsiemi indipendenti e abilitati in* c_{in}*.*
  Posso avere anche un altro tipo di ***comportamento non sequenziale****, definito da Petri stesso, in una* ***semantica ad ordini parziali*** *in cui si definiscono processi non sequenziali. Il comportamento di tale sistema viene registrato in una rete di Petri*

In ogni caso si considerano sia sequenze finite che infinite (con cicli) di eventi o passi.

Esempio 4. *Dato il sistema elementare* Σ*:*



si ha, per esempio, la seguente sequenza di occorrenza di eventi:

$$\{1,2\}[a>\{3,2\}[b>\{3,4\}[c>\{1,2\}[b>\{1,4\}[d>\{5\}$$

arrivati in "5" abbiamo un caso finale, ovvero una situazione di ***deadlock****, in quanto il sistema non può evolvere ulteriormente.*
Vediamo anche la seguente possibile sequenza di passi. In "1" e "2" sia "a" che "b" sono indipendenti e sono entrambi abilitati (scattano in maniera concorrente in un unico passo... ovviamente posso avere passi con lo scatto di un solo evento):

$$\{1,2\}[\{a,b\}>\{3,4\}[\{c\}>\{1,2\}[\{b\}>\{1,4\}$$

Come ricordato posso finire in una sequenza infinita.

Vediamo ora come modellare e registrare il comportamento del sistema. Un modo è usando il **grafo dei casi raggiungibili**.

Definizione 11. *Il* ***grafo dei casi raggiungibili*** *di un sistema elementare* $\Sigma = (B, E, F; c_{in})$ *è il sistema di transizioni etichettato:*

$$CG_\Sigma = (C_\Sigma, U_\Sigma, A, c_{in})$$

dove:

- C_Σ *è l'insieme dei nodi del grafo, ovvero gli stati globali sono i casi raggiungibili dal sistema* Σ
- $U\Sigma$*, è l'alfabeto, ovvero i passi del sistema rappresentano l'alfabeto*

- *A è l'insieme di archi etichettati, formalmente definito come:*

$$A = \{(c, U', c') |\, c, c' \in C_\Sigma, U \in U_\Sigma, c[U > c']\}$$

ovvero sono archi che connettono uno caso c con un caso c′ e sono etichettati con un passo U sse U è abilitato in c e porta in c′. Ovviamente c e c′ sono devono essere raggiungibili e U deve appartenere all'insieme dei passi di Σ

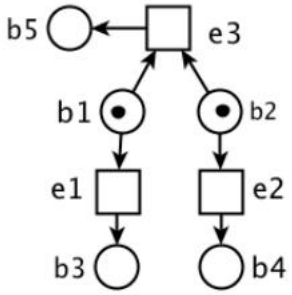


Figura 2.8: il sistema Σ

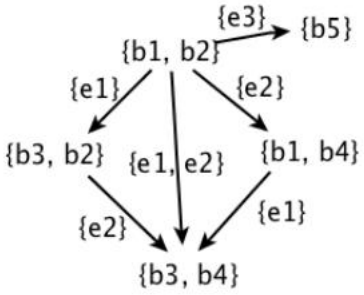


Figura 2.9: Grafo dei casi del sistema Σ

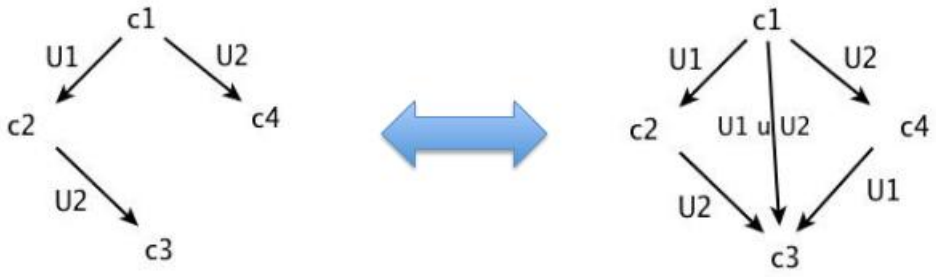
2.1.1 Diamond Property

Dato un sistema elementare $\Sigma = (B, E, F, c_{in})$ e il suo grafo dei casi $CG_\Sigma = (C_\Sigma, U_\Sigma, A, c_{in})$ si ha che il grafo soddisfa una particolare proprietà, detta **diamond property**, tipica solo dei sistemi elementari.

Definizione 12. *La **diamond property** stabilisce una proprietà della struttura del grafo della rete elementare, ovvero, dati $U_1, U_2 \in U_\Sigma$ tali che:*

- $U_1 \cap U_2 = \emptyset$
- $U_1 \neq \emptyset$
- $U_2 \neq \emptyset$

e dati $c_i \in C_\Sigma$ allora vale, per esempio:



ovvero se posso rilevare come sottografo una struttura come quella a sinistra nell'immagine allora sicuramente tale sottografo contiene anche l'arco gli archi per ottenere l'immagine di destra. Il discorso vale anche all'opposto.

Si possono fare delle prove:

1. **prima prova:**
   Dimostriamo che possiamo passare all'immagine di destra da quella di sinistra aggiungendo i due archi mancanti.
   Per semplicità diciamo che U_i è un singolo evento e_i, con $i = 1, 2$. Siano inoltre $c_1, c_2 \in C_\Sigma$, ovvero sono casi raggiungibili, ed $e_1, e_2 \in E$ tali che $c_1[e_1 > c_2[e_2 >$ e $c_1[e_2 >$, i due eventi quindi sono abilitati in sequenza e da c_1 è anche abilitato c_2 . Si vuole dimostrare che:

   $$({}^\bullet e_1 \cup e_1^\bullet) \cap ({}^\bullet e_2 \cup e_2^\bullet) = \emptyset$$

   ovvero che i due eventi sono indipendenti, che sono entrambi abilitati e che sono eseguibili in qualsiasi ordine.

Da $c_1[e_1 >$ e $c_1[e_2 >$ segue che:

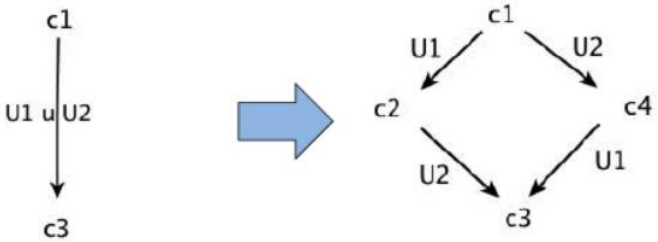
- ${}^\bullet e_1 \cap e_2^\bullet = \emptyset$
- ${}^\bullet e_2 \cap e_1^\bullet = \emptyset$

infatti se e_1 e e_2 sono entrambi abilitati in c_1, le loro pre-condizioni sono vere e le post-condizioni false, e quindi non è possibile che una condizione sia contemporaneamente precondizione di e_1 (vera) e anche postcondizione di e_2 (falsa), e viceversa. Quindi le precondizioni di un evento sono disgiunte dalle postcondizioni dell'altro. Inoltre dal fatto che ho $c_1[e_1 > c_2[e_2$, ovvero che da c_1 è abilitato e_1 e che dopo lo scatto di e_1 è ancora abilitato e_2 possiamo dire che:

- $e_1^\bullet \cap e_2^\bullet = \emptyset$
- ${}^\bullet e_1 \cap {}^\bullet e_2 = \emptyset$

in c_2, infatti, le pre-condizioni di e_1 sono false mentre le precondizioni di e_2 sono vere e quindi e_1 e e_2 non possono avere precondizioni in comune; inoltre sempre in c_2 le postcondizioni di e_1 sono vere, mentre quelle di e_2 sono false, e quindi e_1 e e_2 non possono avere post-condizioni in comune. Quindi le precondizioni dei due eventi sono disgiunte, come del resto anche le postcondizioni, in quanto i due eventi sono sequenziali.
Si è quindi dimostrato che i due eventi hanno precondizioni e post-condizioni completamente disgiunte e quindi la tesi è verificata

2. **seconda prova:**
   Analizzando la situazione:



Si supponga che $U_1 \cup U_2 \in U_\Sigma$ e che si abbiano:

- $U_1 \cap U_2 = \emptyset$, ovvero sono disgiunti
- $U_1 \neq \emptyset$
- $U_2 \neq \emptyset$

allora se $c_1[(U_1 \cup U_2) > c_3$, quindi è abilitato il passo $U_1 \cup U_2$ in c_1, sicuramente si ha che sono abilitati anche i singoli passi:

- $c_1[U_1 >$
- $c_1[U_2 >$

resta da dimostrare che dopo lo scatto di U_1 è ancora abilitato U_2 in c_2. Ma se $U_1 \cup U_2$ è un passo abilitato significa che posso eseguirli in qualsiasi ordine, quindi anche prima U_1 e poi U_2, e questo comporta sicuramente che U_2 è abilitato e che porta a c_3. Analogamente invertendo U_1 e U_2, formalmente:

- $c_1[U_1 > c_2[U_2 > c_3$
- $c_1[U_2 > c_4[U_1 > c_3$

Si dimostra così che l'immagine di sinistra comporta quella di destra.

Grazie alla diamond property possiamo non considerare il grafo dei casi raggiungibili ma solo il **grafo dei casi sequenziale**:

Definizione 13. *Un **grafo dei casi sequenziale** del sistema elementare $\Sigma = (B, E, F, c_{in})$ è una quadrupla:*

$$SCG_\Sigma = (C_\Sigma, E, A, c_{in})$$

dove le etichette sono i singoli eventi (mentre il resto rimane definito come nel grafo dei casi raggiungibili). Formalmente si ha quindi che:

$$A = \{(c, e, c') | , , c'_\Sigma, e \in E : c[e > c'\}$$

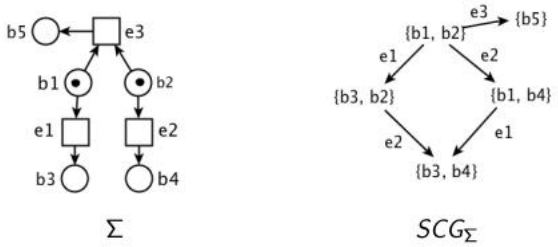


Figura 2.10: Esempio di grafo dei casi sequenziale

Si registra quindi l'occorrenza di un evento alla volta. Il grafo dei casi sequenziale è quindi il sistema di transizione con gli archi etichettati dai singoli eventi.

Riprendendo l'immagine precedente si ha che per la diamond property possono aggiungere l'arco "centrale" che trasformerebbe nuovamente il grafo dei casi sequenziale in quello dei casi raggiungibili quindi: *per la diamond property, nei sistemi elementari il grafo dei casi e il grafo dei casi sequenziale sono* ***sintatticamente equivalenti****, ovvero possono essere ricavati a vicenda.*
Questo implica il fatto che due sistemi elementari hanno grafi dei casi ***isomorfi*** *sse hanno grafi dei casi sequenziali isomorfi.*

2.1.2 Isomorfismo tra Sistemi di Transizione Etichettati

Si ricorda che:

Si parla di isomorfismo quando due strutture complesse si possono applicare l'una sull'altra, cioè far corrispondere l'una all'altra, in modo tale che per ogni parte di una delle strutture ci sia una parte corrispondente nell'altra struttura; in questo contesto diciamo che due parti sono corrispondenti se hanno un ruolo simile nelle rispettive strutture.

Diamo ora una definizione formale di isomorfismo tra sistemi di transizione etichettati, che possono quindi essere grafi dei casi o grafi dei casi sequenziali.

Definizione 14. *Siano dati due sistemi di transizione etichettati:*
$A_1 = (S_1, E_1, T_1, s_{01})$ *e* $A_2 = (S_2, E_2, T_2, s_{02})$.
e siano date due ***mappe biunivoche****:*

1. $\alpha : S_1 \to S_2$
2. $\beta : E_1 \to E_2$

allora:

$$\langle \alpha, \beta \rangle : A_1 = (S_1, E_1, T_1, s_{01}) \to A_2 = (S_2, E_2, T_2, s_{02})$$

è un ***isomorfismo*** *sse:*

- $\alpha(s_{01}) = s_{02}$
- $\forall s, s' \in S_1, \forall e \in E_1 : (s, e, s') \in T_1 \Leftrightarrow (\alpha(s), \beta(e), \alpha(s')) \in T_2$