Sistemi a Eventi Discreti

Colognese, Rossini

Febbraio 2018

Concetti di Base

Composizione parallela: $H||G$

- Si crea uno stato iniziale composto dagli stati iniziali dei due grafi;
- si effettua una transizione di uno dei grafi, modificando la nuova coppia di stati, ad esempio:
  - nel grafo G, il nodo A va in B con la transizione a_1;
  - nel grafo H, se con a_1 il nodo 1 va in 2, allora il nuovo stato è $(B, 2)$;
  - nel grafo H, se con a_1 il nodo 1 non ha mosse disponibili, allora il nuovo stato è $(B, 1)$;
- effettuare tutte le transizioni di entrambi i grafi generando tutte le combinazioni possibili.

  Per alcuni eventi, bisogna tenere conto della specifica dell'esercizio ($H_{spec} \mid\mid G$).

Prodotto: $H \times G$

- Si crea uno stato iniziale composto dagli stati iniziali dei due grafi;
- si effettua una transizione, modificando la nuova coppia di stati, verificando che entrambi possano eseguirla. Ad esempio, se nel grafo G, il nodo A va in B con la transizione a_1:
  - se nel grafo H il nodo 1 va in 2 con a_1, allora il nuovo stato è $(B, 2)$;
  - se il grafo H non ha mosse disponibili con a_1 dal nodo 1, la transizione viene saltata;
- effettuare tutte le transizioni di entrambi i grafi generando tutte le combinazioni possibili.

Coaccessibilità: $CoAcc(G)$

Elimina tutti gli stati che andranno solamente in stati di *deadlock* oppure *livelock*.

Accessibilità: $Acc(G) = Trim(G)$

Elimina tutti gli stati che non possono essere raggiunti da S_0 attraverso una o più transizioni.

Complemento: $Compl(G)$

Si convertono tutti gli stati non-terminali in stati terminali e viceversa. Si aggiungono inoltre tutte le transizioni mancanti per completare l'alfabeto del linguaggio; se necessario si aggiunge uno stato accettante (terminale) per le transizioni mancanti.

Linguaggi e Automi

Osservatore

Considerando un automa (possibilmente non deterministico) $G_{nd} = (X, E \cup \{\epsilon\}, f_{nd}, \gamma, x_0, X_m)$, l'insieme degli eventi $E = E_o \cup E_{uo}$ è così suddiviso:

- E_o : insieme degli eventi osservabili;
- E_{uo} : insieme degli eventi non osservabili (comprende le ϵ-transizioni).

Per costruire l'osservatore per il grafo G, si seguono i seguenti passi:

- sostituire con ϵ tutte le transizioni corrispondenti a eventi non osservabili;
- convertire l'automa NFA in DFA:
  - si prende lo stato iniziale e si crea uno stato includendo anche gli stati raggiungibili con ϵ-transizioni;
  - per ogni altra transizione t si creano gruppi di stati raggiungibili con t e con ϵ;
  - lo stato che contiene S_f di G sarà lo stato finale dell'osservatore.

Osservabilità

Per verificare l'osservabilità di un sistema:

- costruire il grafo corrispondente al linguaggio marcato;
- riscriverlo disabilitando gli archi con transizione controllabile uscente dallo stato iniziale (lasciare gli incontrollabili);
- costruire osservatore sostituendo le transizioni non osservabili con ϵ:
  - se uno stato non terminale (composto da più stati) ha una transizione controllabile che è stata disabilitata per uno dei suoi componenti, allora il sistema *non è osservabile*;
  - se viene disabilitata per un componente dello stato finale allora il sistema *è osservabile*.

Si considerino i linguaggi K e $M = \overline{M}$ definiti sull'alfabeto di eventi E, con $E_c \subseteq E$, $E_o \subseteq E$ e P la proiezione naturale da $E* \Rightarrow E_0*$.
K è osservabile rispetto a M, E_o, E_c se per tutte le stringhe $s \in \overline{K}$ e per tutti gli eventi $\sigma \in E_c$ si ha:

$$(s\sigma \notin \overline{K}) \wedge (s\sigma \in M) \Rightarrow P^{-1}[P(s)]\sigma \cap \overline{K} = \emptyset$$

L'insieme di stringhe denotato dal termine $P^{-1}[P(s)]\sigma \cap \overline{K}$ contiene tutte le stringhe che hanno la medesima proiezione di s e possono essere prolungate in K con il simbolo σ. Se tale insieme non è vuoto, allora K contiene due stringhe s e s' tali che $P(s) = P(s')$ per cui s$\sigma \notin \overline{K}$ e $s'\sigma \in \overline{K}$. Un supervisore non saprebbe distinguere tra s e s' per l'osservabilità. Non potrebbe quindi esistere un supervisore che ottiene esattamente il linguaggio $\overline{K}$.

La proiezione va fatta sul linguaggio (prefissi).

$P^{-1}[P(\epsilon)] = P^{-1}[\epsilon] = \{e^*\}$, con $e \in E_{uo}$.
$P(aaa) = \{aaa\}$, ad esempio con $E = \{a, b\}$.
$P^{-1}[P(aa)] = \{aa, bb, ab, ba\}$, se a e b sono eventi indistinguibili con $E = \{a, b\}$.

Controllabilità

Un linguaggio può essere controllabile (esiste una strategia di controllo) anche se non è realizzabile da un automa a stati finiti (es. $K = \{a^n b^m \text{ con } n \geq m \geq 0\}$).

Per verificare la controllabilità di un sistema:

- l'evento incontrollabile non può essere disabilitato, quindi disabilitare quello controllabile;
- dopo che l'impianto ha prodotto l'evento incontrollabile si riabilita l'evento controllabile.

Se la stringa prodotta corrisponde a quella voluta K allora *è controllabile.*
Cioè è controllabile solo se l'evento incontrollabile è il primo evento dalla stringa K.

Si prende una stringa $s \in \overline{K}$, si prende poi una stringa $\sigma \in E_{uc}$. Se la concatenazione $s\sigma \in M \in \overline{K}$ allora il sistema è controllabile. Se invece $s\sigma \in M \notin \overline{K}$ allora non è controllabile ($\overline{K} = L(H)$ e $H = H_{spec}||G$).

Controllore Massimo Supremo

Controllore massimo supremo: $K^{\uparrow C} = L(H)^{\uparrow C}$.
Per $K = \overline{K}$ abbiamo: $K^{\uparrow C} = K \backslash [M \backslash K / E^*_{uc}] E^*$.
Dati G_1 e G_2:

- costruire $G = G_1||G_2$, $M = L(G)$;
- costruire H_{spec} seguendo le specifiche e individuare $L(H_{spec})$;
- costruire $H = H_{spec}||G$ tenendo conto della specifica ($K = L(H)$);
- verificare la controllabilità con $s\sigma \in M \in \overline{K}$;
- costruire $H_0 = H \times G$ assumendo $\mathcal{L}_m(H_0) = K$ e $\mathcal{L}(H_0) = \overline{K}$;
- calcolare $K^{\uparrow C}$ con il seguente algoritmo:
  - rimuovere per incontrollabilità da H_0 i nodi composti che non hanno l'arco uscente con evento non controllabile perché uno dei due nodi che lo compongono non ha l'uscita corrispondente;
  - rimuovere per potatura gli stati non terminali che non vanno in uno stato marcato;
  - se la precedente eliminazione causa la perdita di archi con evento non controllabile, ripetere l'algoritmo.

L'automa ottenuto (anche vuoto) sarà il Controllore Massimo Supremo.

Sovralinguaggio Controllabile Infimo

Sovralinguaggio controllabile infimo chiuso rispetto al prefisso: $K^{\downarrow C} = \overline{K} E^*_{uc} \cap M$.
Dati G_1 e G_2:

- costruire $G = G_1||G_2$, $M = L(G)$;
- costruire H_{spec} seguendo le specifiche e individuare $L(H_{spec})$;
- costruire $H = H_{spec}||G$ tenendo conto della specifica ($K = L(H)$);
- calcolare $K^{\downarrow C}$ con il seguente algoritmo:
  - aggiungere un nodo N avente un auto-anello con gli eventi non controllabili e collegare tutti gli altri nodi a N con ϵ transizioni;
  - $\forall$ nodo sostituire ϵ con l'evento/i non controllabile a esso mancante, ottenendo così H_{aug};
  - costruire l'automa $H_{aug} \times G$ ottenendo $K^{\downarrow C}$.

Segnali discreti

Se $T \subseteq R$ e' l'insieme dei tempi in cui e è presente, cioè $T = \{t \in R : e(t) \neq assente\}$, il segnale e è discreto se esiste una funzione iniettiva $f : T \to N$ che preserva l'ordine, cioè $\forall t_1, t_2 \in T$ se $t_1 \leq t_2$ allora $f(t_1) \leq f(t_2)$. L'esistenza di tale funzione iniettiva garantisce che possiamo contare gli eventi secondo un ordine temporale.

Un segnale è discreto se:

- riesco a metterlo in corrispondenza dei numeri naturali;
- è definito sugli interi non negativi: basta prendere come f la funzione identità che è iniettiva e preserva banalmente l'ordine;
- non è definito sui razionali: i tempi in cui è presente non possono essere contati in ordine (essi non sono una successione di eventi istantanei nel tempo, bensì un insieme di eventi istantanei nel tempo);
- se $t = 1 - \frac{1}{n}$ per ogni intero positivo n: poiché in questo caso l'insieme dei tempi in cui il segnale y è presente è $T = \{1 - \frac{1}{1}, 1 - \frac{1}{2}, 1 - \frac{1}{3}, ..., 1 - \frac{1}{n}, ...\} = \{0, \frac{1}{2}, \frac{2}{3}, ...\}$ possiamo definire f come $\forall t \in T : f(t) = n$, dove $t = 1 - \frac{1}{n}$ che è iniettiva e preserva l'ordine;
- non è combinazione di due segnali di cui almeno uno non è discreto.

Se due segnali sono discreti, la loro combinazione può non esserlo: $x(n)$ e $y(1 - \frac{1}{n})$ con $n \in \mathbb{Z}$.

Reti di Petri

Albero di Copertura (o Raggiungibilità)

Dato una marcatura iniziale si effettuano tutte le possibili transizioni ottenendo nuove marcature. L'albero si conclude quando non è possibile effettuare altre transizioni oppure quando si ottengono marcature già trovate (*dup*).
Se un nodo y è già presente nell'albero e il nuovo nodo x copre quello vecchio (e c'è un percorso da y a x), allora si mette $x(p) = \omega$ per ogni p avente $x'(p) > y(p)$.

Dall'albero si può ricavare il linguaggio della Rete di Petri (ogni transizione corrisponde ad un carattere).

Grafo di Copertura (o Raggiungibilità)

Dato un albero di copertura si sostituiscono i *dup* con archi all'indietro.

Grafo Marcato: rete in cui ogni posto ha esattamente un arco in ingresso e uno in uscita.
Reti a Scelta Libera: rete in cui ogni arco da p a t, o è l'unico uscente da p o è l'unico entrante in t.

Macchine a stati

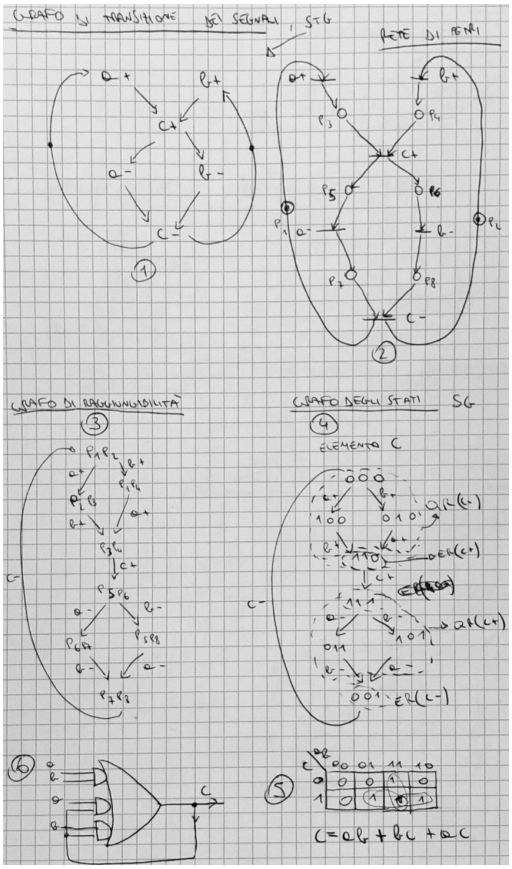
Macchina a Stati: è una rete di Petri in cui ogni transizione ha un arco in ingresso e uno in uscita.
Le reti di Petri sono più espressive degli automi a stati finiti. Esse corrispondono ad automi a stati finiti quando hanno un numero finito di marcature raggiungibili (rete di Petri limitata).

Date le tracce degli ingressi/uscite costruire:

- *Grafo di Transizione dei Segnali (STG)*: grafo con input e output seguiti da +/- e collegati tra loro. È un modello per diagrammi temporali. È una rete di Petri interpretata dove le transizioni sono rappresentate dalle loro etichette e si omettono i posti. I gettoni indicano lo stato iniziale del sistema;
- *Reti di Petri*: questi diventano transizioni e tra una transizione e l'altra si inseriscono i posti, inserire anche i token nei posti precedenti alle prime transizioni;
- *Grafo di Raggiungibilità*: fare tutte le possibili esecuzioni (abilitando una transazione alla volta) partendo dai posti iniziali e unendo i posti;
- *Grafo degli Stati*: sostituire i posti con i valori binari, effettuare incrementi/decrementi, poi per ogni uscita u individuare le regioni di eccitazione $ER(u+)$ e $ER(u-)$ e di crescenza $QR(u+)$ e $QR(u-)$:
  - $ER(u+)$: stati in cui u passa da 0 a 1 (u sta per cambiare);
  - $ER(u-)$: stati in cui u passa da 1 a 0 (u sta per cambiare);
  - $QR(u+)$: stati in cui u è 1 e resta 1 (u è appena stato cambiato o resta uguale);
  - $QR(u-)$: stati in cui u è 0 e resta 0 (u è appena stato cambiato o resta uguale).

Codifica Consistente: per ogni transizione tra due stati del grafo degli stati, i due vettori binari degli stati differiscono solo per il cambio di valore del segnale che etichetta la transizione.
Codifica Unica: a ogni stato del grafo degli stati e' assegnato un codice binario unico.
Codifica Completa: se due stati del grafo degli stati hanno lo stesso codice binario, essi hanno gli stessi segnali d'uscita abilitati.
Limitatezza: il grafo degli stati ha un numero finito di stati, cioè la rete di Petri originale ha un insieme di raggiungibilità finito.

- *Tabella di Karnaugh*: creare una tabella per ogni uscita inserendo 1 in corrispondenza delle combinazioni $\in \{QR+, ER+\}$ di quell'uscita, 0 altrimenti e ricavare l'equazione dell'uscita;
- *Realizzazione*: rappresentare con porte logiche le equazioni trovate ($+$ diventa OR cioè quella con la conca, $*$ diventa AND cioè quella fatta a D).



Determinizzazione di un NFA (Subset Construction)

Dato un automa *NFA*, si può ottenere un automa *output-deterministico* con il seguente algoritmo:

- lo stato iniziale del DFA corrisponde a tutti gli stati iniziali dell'NFA;
- per ciascun stato componente prendo una alla volta le coppie (input/output) e genero i nuovi macrostati composti dai diversi stati target delle transizioni;
- ripetere il passo precedente fino a raggiungere un macrostato finale (contiene uno stato finale).

	stati presenti				
	(s_{1A}, s_1)	(s_{2B}, s_1)	(s_{2B}, s_2)	(s_{3B}, s_1)	(s_{3B}, s_3)
X/Z	*stati futuri*				
0/0	$(s_{2B}, s_2), (s_{3B}, s_3)$	(s_{2B}, s_2)	–	$(s_{2B}, s_2), (s_{3B}, s_3)$	–
0/1	–	–	(s_{1A}, s_1)	–	(s_{1A}, s_1)
1/0	–	–	–	–	–
1/1	$(s_{2B}, s_1), (s_{3B}, s_1)$	(s_{2B}, s_1)	(s_{1A}, s_1)	$(s_{2B}, s_1), (s_{3B}, s_1)$	(s_{1A}, s_1)

Gli stati (s_{2B}, s_2) e (s_{3B}, s_3) sono equivalenti (le loro colonne di stati futuri coincidono). Perciò ci riduciamo a una tavola con una colonna in meno, dove lo stato (s_{3B}, s_3) è rimpiazzato dallo stato (s_{2B}, s_2).

	stati presenti			
	(s_{1A}, s_1)	(s_{2B}, s_1)	(s_{2B}, s_2)	(s_{3B}, s_1)
X/Z	*stati futuri*			
0/0	(s_{2B}, s_2)	(s_{2B}, s_2)	–	(s_{2B}, s_2)
0/1	–	–	(s_{1A}, s_1)	–
1/0	–	–	–	–
1/1	$(s_{2B}, s_1), (s_{3B}, s_1)$	(s_{2B}, s_1)	(s_{1A}, s_1)	$(s_{2B}, s_1), (s_{3B}, s_1)$

Gli stati (s_{1A}, s_1), (s_{2B}, s_1) e (s_{3B}, s_1) sono equivalenti (ci riduciamo a una tavola con due colonne in meno).

	stati presenti	
	(s_{1A}, s_1)	(s_{2B}, s_2)
X/Z	*stati futuri*	
0/0	(s_{2B}, s_2)	–
0/1	–	(s_{1A}, s_1)
1/0	–	–
1/1	(s_{1A}, s_1)	(s_{1A}, s_1)

Minimizzazione

Input: macchina a stati M.
Output: *minimize(M)*, cioè la macchina a stati con il minor numero di stati che sia bisimile a M (il risultato è unico salvo rinominazione degli stati).
L'algoritmo si basa su due passi:

- *Output split*: in un set, se tutte le transizioni con un certo input hanno stesso output ok, se stesso input ma diverso output allora divido gli stati mettendoli in set diversi;
- *Next-state split*: preso un set ottenuto, se ho degli stati che con uno stesso input raggiungono set diversi, allora faccio uno split.

Ogni insieme di stati ottenuto rappresenta un singolo stato della macchina a stati bisimile a M.

Simulazione

Date due macchine a stati M_1 e M_2, possiamo dire che M_2 simula M_1 se:

- per ogni mossa di M_1, M_2 deve poter rispondere raggiungendo la stessa coppia (input/output);
- si aggiunge all'insieme della relazione di simulazione la coppia di stati raggiunti dalle due macchine.

Ogni stato iniziale di M_1 è simulato da uno stato iniziale di M_2, e se uno stato p di M_1 è simulato da uno stato q di M_2, per ogni ingresso e per ogni stato futuro p' di p c'è uno stato futuro q' di q tali che p' e q' producono la medesima uscita e p' è simulato da q'.

Bisimulazione

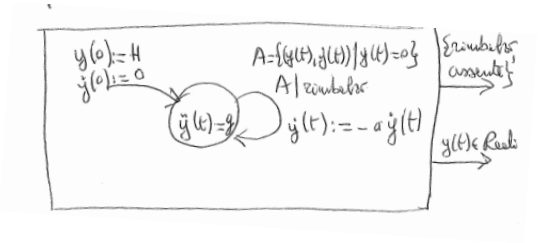
Vale la bisimulazione se vale la simulazione in entrambi i versi (condizione necessaria ma non sufficiente). Se le relazioni di simulazione sono simmetriche, vale sicuramente la bisimulazione. Si testano le simulazioni in entrambi i versi scegliendo arbitrariamente quale macchina muoverà per prima ad ogni turno.

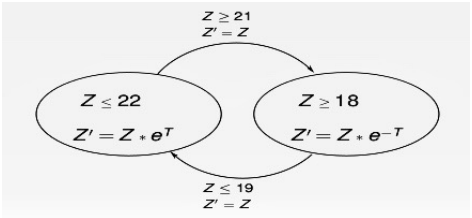
La *bisimulazione* tra M_1 e M_2 indica che gli stati iniziali delle due macchine sono in relazione e, per ogni stato p di M_1, in relazione con q di M_2, per ogni valore in input p e q producono lo stesso valore in output e gli stati successivi sono ancora in relazione.

Automi ibridi

Si consideri una palla che rimbalza. Al tempo $t = 0$, la palla è fatta cadere da un'altezza di $y(0) = H$ metri con $\dot{y}(0) = 0\ metri/sec$. Dopo cade liberamente seguendo la legge $\ddot{y}(t) = -g$, con $g = 10\ m/sec^2$ costante di gravità. A un tempo successivo t_1, la palla colpisce il suolo con una velocità $\dot{y}(t_1) < 0$ metri al secondo e si produce un evento discreto rimbalzo. La collisione è inelastica e la palla rimbalza con velocità $\dot{y}(t) = -a\dot{y}(t_1)$, con $0 < a < 1$ costante. Poi la palla risale sino a una certa altezza e ricade di nuovo al suolo e così ripetutamente.

- locazioni: l_1, dove l_1 è la locazione iniziale con condizioni iniziali $y(0) := H$, $\dot{y}(0) := 0$;
- dinamica della locazione l_1 : $\ddot{y}(t) = -g$;
- transizione da l_1 a l_1 : $A/rimbalzo$, $\dot{y}(t) := -a\dot{y}(t)$, dove $A = \{(y(t), \dot{y}(t))|y(t) = 0\}$ (la sintassi delle annotazioni di una transizione è *guardia/uscita, azione*);
- ingresso assente perché il sistema è autonomo;
- uscita $e_u(t) \in \{rimbalzo, assente\}$,
  uscita $y(t) \in \mathbb{R}$.





Termostato

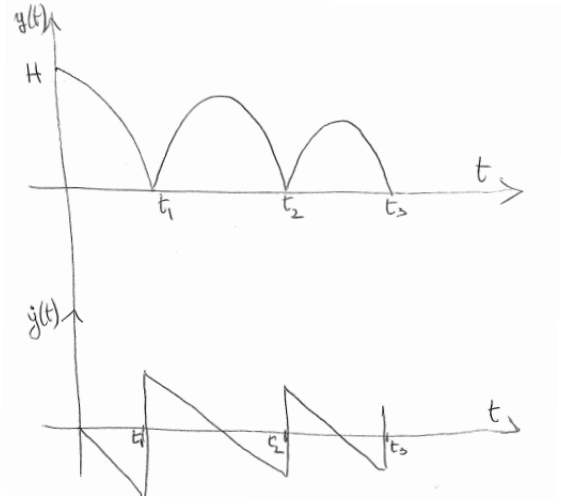
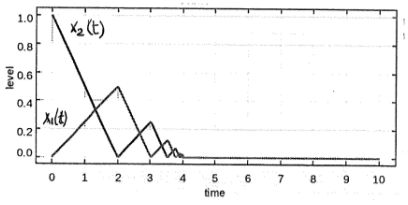


Grafico qualitativo

Comportamento Zenoniano: numero infinito di transizioni discrete in un tempo finito.



Esempio: cisterne

Definizioni

Macchina di Moore:
L'uscita non dipende dall'ingresso, ma solo dallo stato corrente.

Macchina di Mealy:
L'uscita è determinata dallo stato corrente e dall'ingresso corrente.

Supervisore:
Se il sistema è sia controllabile che osservabile allora esiste il *supervisore*.

Composizione Prodotto:
$f((x_1, x_2), \sigma) := (f_1(x_1, \sigma), f_2(x_2, \sigma))$ se $\sigma \in \Gamma_1(x_1) \cap \Gamma_2(x_2)$
Proprietà:

- $\Gamma_{1\times 2}(x_1, x_2) = \Gamma_1(x_1) \cap \Gamma_2(x_2)$
- $\mathcal{L}(G_\times G_2) = \mathcal{L}(G_1) \cap \mathcal{L}(G_2)$
- $\mathcal{L}_m(G_\times G_2) = \mathcal{L}_m(G_1) \cap \mathcal{L}_m(G_2)$

Composizione Parallela:

$$f((x_1, x_2), \sigma) := \begin{cases} (f_1(x_1, \sigma), f_2(x_2, \sigma)) & \text{se } \sigma \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ (f_1(x_1, \sigma), x_2) & \text{se } \sigma \in \Gamma_1(x_1) \backslash E_2 \\ (x_1, f_2(x_2, \sigma)) & \text{se } \sigma \in \Gamma_2(x_2) \backslash E_1 \end{cases}$$

Proprietà:

- $\mathcal{L}(G_1 || G_2) = P_1^{-1}[\mathcal{L}(G_1)] \cap P_2^{-1}[\mathcal{L}(G_2)]$
- $\mathcal{L}_m(G_1 || G_2) = P_1^{-1}[\mathcal{L}_m(G_1)] \cap P_2^{-1}[\mathcal{L}_m(G_2)]$

Controllabilità:
Siano K e $M = \overline{M}$ linguaggi dell'alfabeto di eventi E, con $E_{uc} \subseteq E$. Si dice che K è controllabile rispetto a M e E_{uc} se per tutte le stringhe $s \in \overline{K}$ e per tutti gli eventi $\sigma \in E_{uc}$ si ha: $s\sigma \in M \Rightarrow s\sigma \in \overline{K}$.
Per la definizione di controllabilità si ha che K è controllabile sse $\overline{K}$ è controllabile.

Macchina a stati deterministica:
Una macchina a stati è deterministica se esiste un solo stato iniziale. Inoltre per ogni stato e *per ogni input* esiste solo uno stato successivo.

Non-Deterministico:
Per ogni segnale in input ci sono uno o più segnali in output, dunque questi sistemi hanno più *behaviors*. Può esserci più di uno stato iniziale e per ogni stato e ogni coppia di I/O può esserci più di uno stato successivo.

Output-Deterministico:
C'è solo uno stato iniziale e per ogni stato e per ogni coppia di I/O c'è solo uno stato successivo. Per ogni *behavior* (x, y) c'è una sola esecuzione. Per ogni macchina a stati non deterministica si può trovare una macchina a stati output-deterministica equivalente (attraverso la *subset construction*).
Deterministico implica Output-Deterministico, ma non viceversa.

Non-Deterministico, Progressivo:
Significa che l'evoluzione è definita per ogni ingresso, cioè la funzione di transizione è definita come: $Stati \times Ingressi \rightarrow P(Stati \times uscite)\backslash\emptyset$, dove P è l'insieme potenza e $\emptyset$ impone che sia progressiva.

Raffinamento:
Il sistema S_1 raffina il sistema S_2 sse con $Time[S_1] = Time[S_2]$ hanno stessi input e output ed inoltre $behaviors[S_1] \subseteq behaviors[S_2]$. S_1 è più dettagliato mentre S_2 è un'astrazione delle proprietà di S_1.

Equivalenza:
Due macchine a stati M_1 e M_2 sono equivalenti sse implementano lo stesso sistema (funzione di I/O), cioè $\forall x \in input, M_1(x) = M_2(x) \in output$.
Sono equivalenti se $behaviors[M_1] = behaviors[M_2]$.
Sono equivalenti sse esiste una bisimulazione tra $M_1 e M_2$.
Sono equivalenti sse M_1 raffina M_2 e M_2 raffina M_1.

Simulazione - Raffinamento:
Se M_2 è deterministica, M_1 è simulata da M_2 sse M_1 raffina M_2 e M_2 raffina M_1.
Se M_2 è pseudo-nondeterministica, M_1 è simulata da M_2 sse M_1 raffina M_2.
Se M_2 è nondeterministica, M_1 è simulata da M_2 se M_1 raffina M_2 (ma M_1 raffina M_2 non implica che M_1 è simulata da M_2).

Linguaggi Normali: siano dati $M = \overline{M} \subseteq E^*$ e la proiezione P. Un linguaggio $K \subseteq M$ si dice *normale* rispetto a M e P se: $\overline{K} = P^{-1}[P(\overline{K})] \cap M$

Linguaggi Regolari - Reti di Petri:
I Linguaggi Regolari sono un sottoinsieme di quelli accettati dalle Reti di Petri.

Osservabilità e Intersezione:
L'osservabilità non è preservata dall'intersezione, dati K_1 e K_2 osservabili, $K_3 = K_1 \cap K_2$ non è osservabile. Dato che l'intersezione non preserva l'osservabilità, non è garantita l'esistenza del sovralinguaggio osservabile infimo.

Osservabilità e Unione:
L'osservabilità non è preservata dall'unione, dati K_1 e K_2 osservabili, $K_3 = K_1 \cup K_2$ non è osservabile.

Rete di Petri Limitata (può non essere conservativa, ad esempio se ha un posto e una transizione):
È limitata se non ci sono ω nell'albero di copertura, quindi k è il massimo numero di gettoni che possono accumularsi in un posto, e non ∞.
Un posto è limitato se $\exists k \geq 0$ tale che, per ogni marcatura raggiungibile, il numero di gettoni del posto $\leq k$. Una rete è limitata se ogni posto è limitato.
Rete di Petri Viva:
Una rete di Petri con marcatura iniziale x_0 è viva se da ogni transizione t e da ogni marcatura x_M raggiungibile da x_0 esiste una marcatura x_t raggiungibile da x_M dove t è abilitata a scattare. Cioè se da ogni nodo dell'albero si può far scattare qualsiasi transizione (da ogni marcatura si può ritornare alla marcatura iniziale e da lì far scattare qualsiasi transizione).
Rete di Petri Conservativa (dunque anche limitata):
Il numero di token nella rete è costante, cioè $\sum_{i=1}^{b} \gamma_i x(p_i) = C$ dove b sono i posti limitati (senza mai ω).
È *strettamente conservativa* se per ogni marcatura raggiungibile, è invariata la somma dei gettoni.
Una rete è conservativa se esiste un vettore di interi tale che, per ogni marcatura raggiungibile, è invariata la somma pesata dei gettoni nei posti.
Rete di Petri Reversibile (se il suo grafo di raggiungibilità è fortemente connesso):
È reversibile se da ogni marcatura raggiungibile è possibile ritornare alla marcatura iniziale.

Rete di Petri Ordinaria: è ordinaria se tutti gli archi anno peso unitario.
Rete di Petri Pura: è pura se non contiene cappi.
Rete di Petri Ristretta: è ristretta se è ordinaria e pura.