Appunti di
Modelli della Concorrenza

A cura di:
Francesco Refolli
Matricola 865955

Anno Accademico 2023-2024

Part I

Teoria

Chapter 1

CCS e LTS

1.1 Programmazione Sequenziale

- Semantica operazionale: data una Macchina Astratta si descrivono i passi di computazione come funzione $M : I \to O$.
- Semantica denotazionale: dato un Programma, assegno una funzione $f : I \to O$ (= lambda calcolo).
- Semantica assiomatica: specificare le proprieta' di un programma come funzione $P : I \to O$, dove i dati in input vengono specificati come i valori che gli input possono assumere in termini di condizioni, e vice versa gli output.

Caratteristiche della programmazione sequenziale:

- Terminazione
- Composizionalita': composizione funzionale f,g: $h : D(f) \to C(g) = f * g$, con $C(f) \equiv D(g)$

Notazione 'I P O', ex: $\{x > 0\}$ P $\{x < 0\}$

1.1.1 Con programmi concorrenti

- P1: $x = 2$, allora $\{x = V\}$ P1 $\{x = 2\}$.
- P2: $x = 3$, allora $\{x = V\}$ P2 $\{x = 3\}$.
- P3: $x+ = 1$, allora $\{x = V\}$ P3 $\{x = V\}$.

Come si nota, in presenza di concorrenza non c'è determinismo e quindi composizionalita' semplice (come nei programmi sequenziali).

- Esecuzione di $P1 \to P2$. $\{x = V\}$ P1 P2 $\{x = 3\}$.
- Se si mettono in parallelo: $\{x = V\}$ P1 && P2 $\{x = 2 \vee x = 3\}$.
- Con esecuzione in parallelo di P1,P2,P3: $\{x = V\}$ P2 && P3 $\{x = 2 \vee x = 3 \vee x = 4\}$.

1.2 Labelled Transition System (LTS)

Sistema di Transizioni Etichettate scritto (S, AcT, T) dove:

- S è insieme di stati
- Act è insieme di nomi di azioni
- T è la funzione di transizione denotata $T : S \times AcT \to S$, dove:
  - $(s, a, s') \in T$
  - $s \xrightarrow{a} s'$
  - $(s) \xrightarrow{a} (s')$
- formalmente, $s \xrightarrow{w} s'$ sse:
  - o) $w = \epsilon \wedge s = s'$
  - o) $w = a * x \wedge s \xrightarrow{a} s'' \xrightarrow{x} s'$

1.3 Calculus of Communicating Systems (CCS)

- Lambda calcolo di processi concorrenti
- Communicating Sequential Processes.

definizione Un set di programmi è un insieme di processi che vengono composti con l'operatore di esecuzione parallela e tramite la sincronizzazione l'interazione con l'ambiente.

CCS è un sistema di calcolo interpretato come LTS che viene denotato con la seguente espressione BNF:

$P ::= 0 \mid a.P_1 \mid A \mid P_1 + P_2 \mid P_1|P_2 \mid P_1[b/a] \mid P_1\backslash a$

Che significa rispettivamente:

- il processo inattivo generico 0
- P esegue a e diventa P_1
- definendo $A \stackrel{def}{\equiv} P_1$ si usa A come identificatore
- P diventa P_1 **o** P_2
- P_1 e P_2 coesistono
- a P_1 si rinomina l'azione $a \to b$
- a P_1 si rimuove l'azione a

Due processi P1 e P2 sono detti equivalenti non in termini funzionali, ma "all'osservazione", cioè se un processo esterno non è in grado di distinguere tra i processi P1 e P2 interagendo con essi. (Liskov dei poveri). La Bisimulazione è tecnica di verifica per identificare se due processi sono dissimili.

Si denota un sistema CCS come tupla (K, Act, T), dove

- K, insieme di nomi di Processi
- A, insieme di nomi di azioni
- $\overline{A}$, insieme di conomi di azioni, ($= \overline{a} \in \overline{A}$ è il reciproco di $a \in A$, es: SYN e ACK)
- $Act = A \cup B \cup \{t\}$, con $t \notin A$
- $(A \cup B)$ sono le azioni osservabili
- t è azione di handshake tra due processi (azione che non aspetta nulla indrè, ndr)

Sia P un processo $P \in K$, ricordando la notazione LTS:

- es: $(P = a.\ P)$:= P esegue a e poi si comporta come P (ricorsione)
- $(P1) \xrightarrow{a} (P2)$ denota sincronizzazione e mutamento
- $(P1) \xrightarrow{a} (P1)$ denota ricorsione di processo

1.3.1 B^1

- ${B^1}_0 = in\ .\ {B^1}_1$
- ${B^1}_1 = \overline{out}\ .\ {B^1}_0$

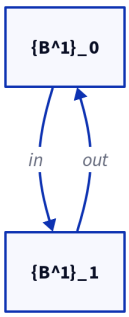


Figure 1.1: diagram

1.3.2 B^2

- ${B^2}_0 = in\ .\ {B^2}_1$
- ${B^2}_1 = \overline{out}\ .\ {B^2}_0\ +\ in\ .\ {B^2}_2$
- ${B^2}_2 = \overline{out}\ .\ {B^2}_1$

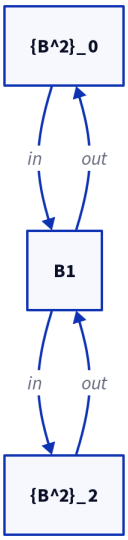


Figure 1.2: diagram

1.3.3 L'esecuzione parallela $B^1{}_0|B^1{}_0$ è bisimile a $B^2{}_0$

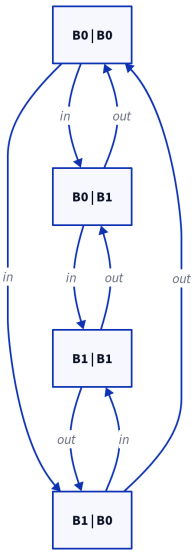


Figure 1.3: diagram

Chapter 2

Bisimulazione

- $S = (P_1|B_1|C_1) \setminus \{dep, est\}$
- $P_1 = prod\ .\ P_2$
- $P_2 = \overline{dep}\ .\ P_1$
- $C_1 = estr\ .\ C_2$
- $C_2 = cons\ .\ C_1$
- $B_1 = dep\ .\ B_2$
- $B_2 = \overline{estr}\ .\ B_1$

$(P_1|B_1|C_1) \xrightarrow{prod} (P_2|B_1|C_1) \xrightarrow{\overline{dep}/dep} (P_1|B_2|C_1) \xrightarrow{prod} (P_2|B_2|C_1) \xrightarrow{\overline{estr}/estr} (P_1|B_1|C_2) \xrightarrow{prod} (P_2|B_1|C_2) \xrightarrow{cons} (P_2|B_1|C_1)$

2.1 Nozione di equivalenza

Nel caso di programmi sequenziali è sufficiente una congruenza funzionale $H : L(I, O) \to L(I, O)$. Invece nei programmi CCS non sequenziali, si deve tener conto dell'interazione dei moduli con l'environment. Serve una relazione di equivalenza $R : C \to C$ che permetta di valutare l'interscambiabilita' di moduli.

2.1.1 $S_1 \sim S_2$, S_1 isomorfo a S_2

Detto $LTS_1 = (Q_1, E_1, T_1, q_{0_1})$ e $LTS_2 = (Q_2, E_2, T_2, q_{0_2})$, allora

- $\alpha : Q_1 \to Q_2$
- $\beta : E_1 \to E_2$
- $\alpha(q_{0_1}) = q_{0_2}$

$(q_1, a, q_1) \in T_1 \to (\alpha(q_1), \beta(a), \alpha(q_1)) \in T_2$

- si dice traccia una sequenza $s \in Act^*$ di azioni percorribili dalla radice (ovvero dal processo considerato).

- l'insieme delle tracce di un processo è composto da tutte le possibili sequenze di azioni e di sincronizzazioni a partire da esso.

Due processi P_1 e P_2 sse $Tracce(P_1) \equiv Tracce(P_1)$

2.2 CCS di un Dispensatore di Bevande

$Uni = (M|LP) \setminus \{coin,\ caffe\}$

- $M_1 = \overline{coin}\ .\ (caffe\ .\ M_1 + \overline{coin}\ .\ tea\ .\ M_1)$
- $M_2 = \overline{coin}\ .\ caffe\ .\ M_2 + \overline{coin}\ .\ \overline{coin}\ .\ tea\ .\ M_2$

2.2.1 Schema di M_1

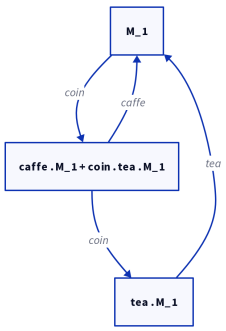


Figure 2.1: diagram

2.2.2 Schema di M_2

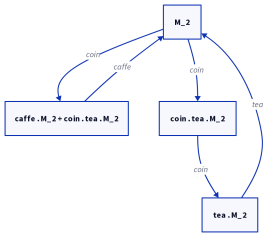


Figure 2.2: diagram

2.2.3 Comparazione

Il sistema di transizioni di M_1 e M_2 non sono isomorfi, ovviamente.
tracce di M_1:

- coin caffe
- coin coin tea

tracce di M_2

- coin caffe
- coin coin tea

Le tracce sono uguali, ma il comportamento di queste CCS è potenzialmente differente perchè non c'è isomorfismo. Si dicono equivalenti perchè eseguono le stesse sequenze, ma non c'è garanzia dell'effettiva esecuzione di M_2 perchè è non deterministica (i.e. è concorrenti sull'azione $\overline{coin}$).

2.3 Equivalenza ad Osservazione

M_1 e M_2 sono intercambiabili sse un osservatore esterno non è in grado di distinguere i due attori distinti.

Transizioni:

- $p \stackrel{a}{\Rightarrow} p' \equiv p \stackrel{t^*}{\rightarrow} q \stackrel{a}{\rightarrow} q \stackrel{t^*}{\rightarrow} p'$: transizione debole, si puo' usare delle **t-mosse** per percorrere l'azione a
- $p \stackrel{a}{\rightarrow} p'$: transizione forte, non si possono usare delle **t-mosse** per percorrere l'azione a
- $p \stackrel{t}{\rightarrow} p'$: **t-mossa**, l'avversario puo' fare una **t-mossa** o stare fermo

L'attaccante puo' sempre decidrere se usare **P** o **Q** come mossa.

2.3.1 Bisimulazione debole

La relazione $R : P \times P$, è una relazione di equivalenza (di bisimulazione forte) sse
$\forall p, q \in P \mid p, q \in P$

- $\forall \alpha \in Act = A \cup \overline{A} \cup \{t\}, \;\; se\ p \stackrel{a}{\rightarrow} p' \mid \exists q' \in Q \mid q \stackrel{a}{\Rightarrow} q'$
- $\forall \alpha \in Act = A \cup \overline{A} \cup \{t\}, \;\; se\ q \stackrel{a}{\rightarrow} q' \mid \exists p' \in P \mid p \stackrel{a}{\Rightarrow} p'$

$p \stackrel{bis}{\longleftrightarrow} q$ sse $\exists R$ relazione di Bisimulazione t.c. $p\ R\ q$.

2.3.2 Bisimulazione forte

La relazione $R : P \times P$, è una relazione di equivalenza (di bisimulazione forte) sse
$\forall p, q \in P \mid p, q \in P$

- $\forall \alpha \in Act = A \cup \overline{A} \cup \{t\}, \;\; se\ p \stackrel{a}{\rightarrow} p' \mid \exists q' \in Q \mid q \stackrel{a}{\rightarrow} q'$
- $\forall \alpha \in Act = A \cup \overline{A} \cup \{t\}, \;\; se\ q \stackrel{a}{\rightarrow} q' \mid \exists p' \in P \mid p \stackrel{a}{\rightarrow} p'$

$p \stackrel{bis}{\longleftrightarrow} q$ sse $\exists R$ relazione di Bisimulazione t.c. $p\ R\ q$.

2.4 Bisimulazione debole come simulazione avversaria

- Attaccante: cerca di dimostrare $p \not\equiv_{BIS} q$
- Difensore: cerca di dimostrare $p \equiv_{BIS} q$

2.4.1 Una simulazione G(p, q)

è composta di varie **run**. Ogni **run** è una sequenza finita o infinita di configurazioni (p0,q0), (p1,q1), ... (pi,qi) In ogni **step** di **run**, si passa da una configurazione all'altra (parte l'attaccante).

- si sceglie uno dei due processi ed esegue una azione **a**, $p \xrightarrow{a} p'$, si trasforma l'altro processo con l'azione **a** verificando che il processo q' è bisimile a p'.
- è possibile che uno dei due attori non possa muovere, in quel caso vince l'altro
- se la **run** è finita, allora vince il difensore
- importante: il difensore puo' usare la **t** come un ponte jolly.

per ogni simulazione, solo uno dei due attori ha una strategia vincente che gli da la possibilita' di vincere ogni partita

- P_1 = t . a . NIL + b . NIL
- P_2 = t (a . NIL + b . NIL)

attaccante con t . b su P_2

Chapter 3

Reti

3.1 Rete

$N = (B, E, F)$ è una rete sse:

- B è l'insieme di condizioni (stati locali)
- E è l'insieme di eventi
- tali che $B \cap E = \emptyset$ e $B \cup E \neq \emptyset$
- $F \subseteq (B \times E) \cup (E \times B)$ relazione di flusso rappresenta da $\rightarrow$

La relazione di flusso definisce pre-elementi ($\bullet x$) e post-elementi ($x\bullet$).

Un **caso** è un'insieme di condizioni $c \subseteq B$ che rappresenta lo stato globale del sistema.

regola di scatto L'evento e è abilitato in c ($c[e >$) sse: $\bullet e \subseteq c$ and $e\bullet \cap c = \emptyset$. Quando e scatta (sse è abilitato), si genera un caso $c' = (c - \bullet e) \cup e\bullet$.

Una rete si dice **semplice** se $\bullet x = \bullet y \wedge x\bullet = y\bullet \Rightarrow x = y$.

Una rete si dice **pura** se $\forall e \in E : \bullet e \cap e\bullet = \emptyset$.
Sia $U \subseteq E$

- due eventi $e_1, e_2 \in U$ sono indipendenti se $(\bullet e_1 \cup e_1\bullet) \cap (\bullet e_2 \cup e_2\bullet) = \emptyset$.
- U è un insieme di eventi indipendenti se tutti gli eventi sono a due a due indipendenti
- U è un passo abilitato sse U è un insieme di eventi indipendenti e $\forall e \in U$, $c[e >$.
- U è un passo da c_1 a c_2 ($c_1[U > c_2$) sse: $c_2 = (c_1 - \bullet U) \cup U\bullet$

3.2 Sistema Elementare

E' detto sistema elementare $\Sigma = (B, E, F, c_{in})$ sse:

- $N = (B, E, F)$ è una rete e il caso iniziale è $c_{in} \subseteq B$.

L'insieme dei casi raggiungibili del sistema elementare è definito dal piu' piccolo sottoinsieme $C_\Sigma \subseteq 2^B$ tale che:

- $c_{in} \in C_\Sigma$
- detti $c \in C_\Sigma, U \subseteq E, c' \subseteq B$, se $c[U > c'$ allora $c' \in C_\Sigma$

Del resto, è detto insieme dei passi il corrispondente insieme $U_\Sigma \subseteq 2^E$ tale che $\forall U \subseteq E \mid \exists c, c' \ : \ c[U > c'$.

Di conseguenza il grafo dei casi sequenziale è il sistema a transizioni etichettate LTS $CG_\Sigma = (C_\Sigma, U_\Sigma, A, c_{in})$ dove:

- C_Σ è l'insieme dei nodi del grafo
- U_Σ è l'alfabeto
- A è l'insieme degli archi etichettati con le parole $p \in U$
- $A = \{(c, U, c') | c, c' \in C_\Sigma, U \in U_\Sigma, c[U > c'\}$

3.3 Diamond Property

Anche detto "Teorema dei Due Passi" In quel tempo il figlio dell'uomo si stava dilettando con i sistemi elementari, al che ne prese uno, lo spezzo', ne diede un po' ai suoi discepoli e disse:

- Sia un sistema elementare $\Sigma = (B, E, F, c_{in})$
- Sia $U \subseteq E$ un sottoinsieme siano U_1, U_2 una sua partizione ($U_1, U_2 \neq \emptyset$, $U_1 \cup U_2 = U \wedge U_1 \cap U_2 = \emptyset$)
- Siano $C, D \subseteq B$ due casi in Σ

Se $C[U > D$ allora esiste un caso intermedio $E \subseteq B$ tale che siano anche $C[U_1 > E$ e $E[U_2 > D$.
Vice versa se $C[U_1 > E$ e $E[U_2 > D$, allora $C[U > D$.

3.4 E le sue consequenze

Per la **Diamond Property** il **Grafo dei Casi Sequenziale** e il relativo **Sistema Elementare** sono tra solo semanticamente equivalenti e percio' si possono ricavare l'uno dall'altro.

Dato questo principio base, è immediato che l'equivalenza di due Sistemi Elementari puo' essere data dall'isomorfismo tra i due relativi Grafi dei Casi Sequenziali.

Tale isomorfismo tra S_0 e S_1 , $I = (a, b)$, è composto da due mappature biunivoche tali che:

- $a : S_0 \to S_1$
- $b : E_0 \to E_1$
- $a(s_0^0) = s_0^1$
- $\forall s, s' \in S; \forall e \in E; (s, e, s') \in T_0 \Leftrightarrow (a(s), b(e), a(s')) \in T_1$

3.5 Sono state un disastro per la razza umana

Dato un sistema etichettato $A = (S, E, T, s_0)$, esiste un sistema elementare $\Sigma = (B, E, F, c_0)$ tale che il grafo dei casi CG_Σ sia isomorfo ad A (e in tal costruire Σ)?

Bella domanda, comunque è stato risolto dalla **Teoria delle Regioni**.

3.6 Contatto

Sia un sistema elementare $\Sigma = (B, E, F, c_0)$, si definisce contatto un caso $c \subseteq B$ tale che: $\bullet e \subseteq c \wedge c \cap e\bullet \neq \emptyset$.

Cioè un evento potrebbe anche scattare se non fosse che alcune post-condizioni non sono inattive.

Un sistema **senza contatti** è un sistema in cui $\forall e \in E, c \in C_\Sigma \mid \bullet e \subseteq c \rightarrow c \cap e\bullet = \emptyset$.

E' possibile trasformare un sistema elementare **con contatti** in uno **senza contatti** aggiungendo il complementare di ogni condizione facendo rimanere isomorfo il grafo dei casi sequenziale. Ovvero: $\forall b \in B\ \exists b' \in B' \supseteq B\ \forall e \in b\bullet \rightarrow b' \in \bullet e$.

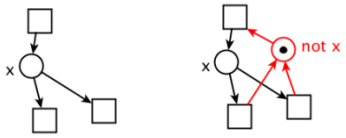


Figure 3.1: example

3.7 Dipendenza

Due eventi e_1, e_2 si dicono sequenziali sse $c[e_1 > \wedge \neg c[e_2 > \wedge c[e_1 e_2 >$, ovvero se $c[e_1 > c[e_2 >$, ovvero se c'è dipendenza causale fra i due.

Vice versa due eventi si dicono concorrenti se $c[e_1 > \wedge c[e_2 > \wedge (\bullet e_1 \cup e_1\bullet) \cap (\bullet e_2 \cup e_2\bullet) = \emptyset$. Ovvero se sono entrambi abilitati ed indipendenti.

3.8 Conflitto

Due eventi si dicono in **conflitto** sse $c[e_1 > \wedge c[e_2 > \wedge \bullet e_1 \cap e_2\bullet \neq \emptyset$.

Importante notare che in molti casi non è possibile stabilire se si è risolto un conflitto o no. C'è molta confusione visto che il sistema è visto dall'alto senza vedere i passi intermedi che compongono il passaggio da un caso all'altro.

3.9 Genesi

Data una rete $N = (B, E, F)$ è possibile generare delle sottoreti.

Sia $N' = (B', E', F')$ una rete.

Essa si dice genericamente sottorete di N sse:

- $B' \subseteq B$
- $E' \subseteq E$

- $F' = F \cap (B'xE') \cup (E'xB')$

Quindi N' si dice sottorete generata da B sse:

- $B' \subseteq B$
- $E' = \bullet B \cup B\bullet$
- $F' = F \cap (B'xE') \cup (E'xB')$

Similarmente N' si dice sottorete generata da E sse:

- $B' = \bullet E \cup E\bullet$
- $E' \subseteq E$
- $F' = F \cap (B'xE') \cup (E'xB')$

3.10 Composizione

Non ho la minima idea del se l'abbiano fatto o meno, comunque giusto per completezza ...

Due Reti sono componibili in tre modi:

- sincrono
- asincrono
- misto

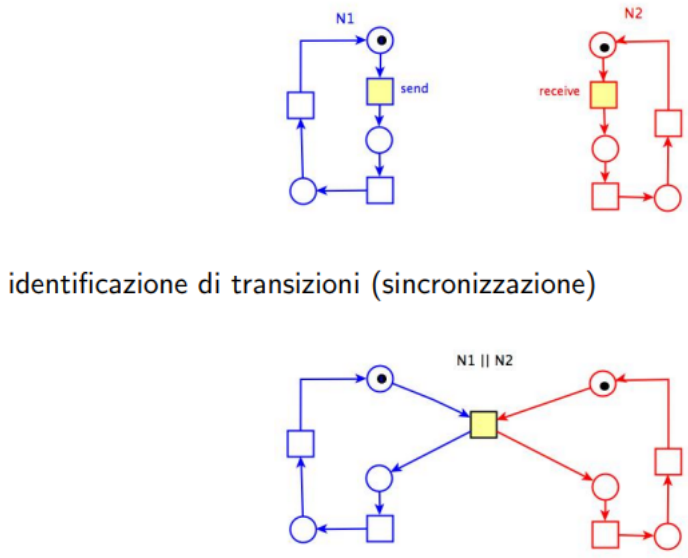
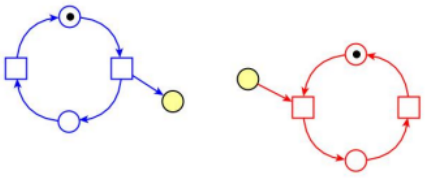


Figure 3.2: sincrono



identificazione di posti (canali di comunicazione)

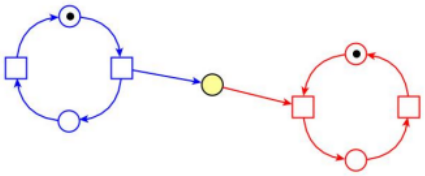


Figure 3.3: asincrono

Chapter 4

Processi

4.1 Rete Causale

$N = (B, E, F)$ è una rete causale sse:

- $\forall b \in B, |\bullet b| \leq 1 \wedge |b \bullet| \leq 1$ (quindi non ci sono conflitti)
- $\forall x, y \in B \cup E \ : \ (x, y) \in F^+ \Rightarrow (y, x) \notin F^+$ (quindi non ci sono cicli)
- $\forall e \in E$, $\{x \in B \cup E \mid (x, e) \in F\}$ e un insieme finito (la rete puo' essere infinita)

Ad una rete causale corrisponde un ordine parziale $(X, \leq) = (B \cup E, F^*)$, quindi:

- $x, y \in X$
- $x \leq y \Leftrightarrow$ x causa y
- $x \ li \ y$, $x \leq y \vee y \leq x$, causalmente dipendenti
- $x \ co \ y$, $\neg(x < y) \wedge \neg(y < x)$, causalmente indipendenti
- le relazioni co, li, sono simmetriche, riflessive ma **non** transitive

Definite queste due relazioni si definiscono i sequenti sottoinsiemi:

- $C \subseteq X$ è detto **co-set** sse $\forall x, y \in C$, $x \ co \ y$
- C è detto **taglio** sse co-set massimale $(\forall x \in (X \ C), \exists c \in C \ : \ c \ li \ x)$
- C è detto **B-taglio** se $C \subseteq B$
- $L \subseteq X$ è detto **li-set** sse $\forall x, y \in L$, $x \ li \ y$
- L è detto **linea** sse li-set massimale $(\forall x \in (X \ L), \exists l \in L \ : \ l \ co \ x)$

4.1.1 K-Density

Una Rete Causale N è detta **K-densa** sse $\forall l \in Linee(N), \forall c \in Tagli(N) \mid |l \cap c| = 1$.
NB: Se N è finita allora è anche densa.

4.2 Intermezzo

Questo esempio di Sistema Elementare Σ verra' usato per illustrare i Processi non Sequenziali e Ramificati.

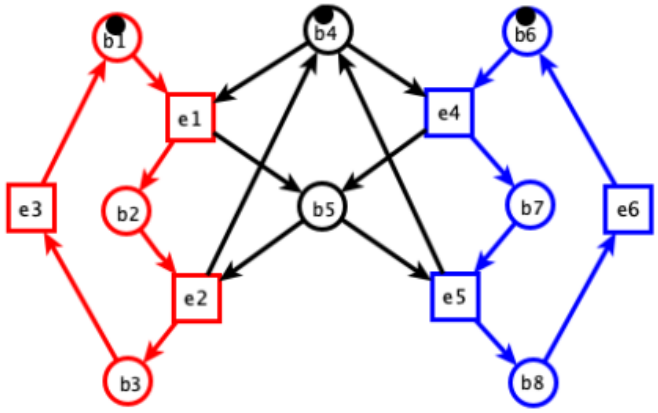


Figure 4.1: example

4.3 Processo Non Sequenziale

Dato un sistema elementare senza contatti e finito $\Sigma = (S, T, F, c_{in})$, sia $N = (B, E, F)$ una **rete causale**.

$P =< N, \phi : B \cup E \to S \cup T >$ è detto **Processo non Sequenziale** sse:

- $\phi(B) \subseteq S, \phi(E) \subseteq T$
- $\forall x, y \in B \cup E \ : \ \phi(x) = \phi(y) \Rightarrow x \ li \ y$
- $\forall e \ inE \ : \ \phi(\bullet e) = \bullet\phi(e) \wedge \phi(e\bullet) = \phi(e)\bullet$
- $Min(N) = \{x \in B \cup E\} \mid \nexists y \ : \ (y, x) \in F\}$
- $Min(N)$ de facto è il set dei minimali della rete causale
- ecco, $\phi(Min(N)) = c_{in}$, (con un po' di immaginazione il concetto è quello)

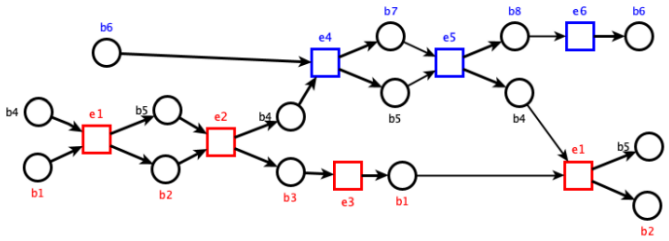


Figure 4.2: example

4.4 K-Density 2.0

Sia $< N, \phi >$ un processo non sequenziale di un sistema elementare Σ finito e senza contatti;

Teorema $N = (B, E, F)$ è K-Densa, e $\forall K \subseteq B$, K è B-Taglio di N tale che K è finito e $\exists c \in C_\Sigma \ : \ \phi(K) = c$.

4.5 Rete di Occorrenze

$N = (B, E, F)$ è detta **Rete di Occorrenze** sse:

- $\forall b \in B, \ | \bullet b| \leq 1$ (due eventi non hanno una stessa post-condizione, conflitti solo in avanti)
- $\forall x, y \in B \cup E \ : \ (x, y) \in F^+ \Rightarrow (y, x) \notin F^+$ (quindi non ci sono cicli)
- $\forall e \in E$, $\{x \in B \cup E \mid (x, e) \in F\}$ e un insieme finito (la rete puo' essere infinita)

L'ordine parziale $(X, \leq) = (B \cup E, F^*)$ persiste.

4.6 Conflitti

Sia definita la relazione $\# : B \cup E \rightarrow B \cup E$ simmentrica ma non riflessiva nè transitiva:

- $x \# y$, sse $\exists e_1, e_2 \in E \ : \ \bullet e_1 \wedge \bullet e_2 \neq \emptyset \wedge e_1 \leq x \wedge e_2 \leq y$.
- Ovvero, se esistono due eventi con precondizioni in comune tali che x e y dipendono da essi

4.7 Processo Ramificato

Dato un sistema elementare senza contatti e finito $\Sigma = (S, T, F, c_{in})$, sia $N = (B, E, F)$ una **rete di occorrenze**.

$P =< N, \phi : B \cup E \rightarrow S \cup T >$ è detto **Processo Ramificato** sse:

- $\phi(B) \subseteq S, \phi(E) \subseteq T$
- $\forall x, y \in E \ : \ \phi(x) = \phi(y) \wedge \bullet x = \bullet y \Rightarrow x \ = \ y$
- ϕ è una biiezione su $\phi(\bullet e) \Leftrightarrow \bullet \phi(e)$
- ϕ è una biiezione su $\phi(e \bullet) \Leftrightarrow \phi(e) \bullet$
- ϕ è una biiezione su $\phi(Min(N)) \Leftrightarrow c_{in}$

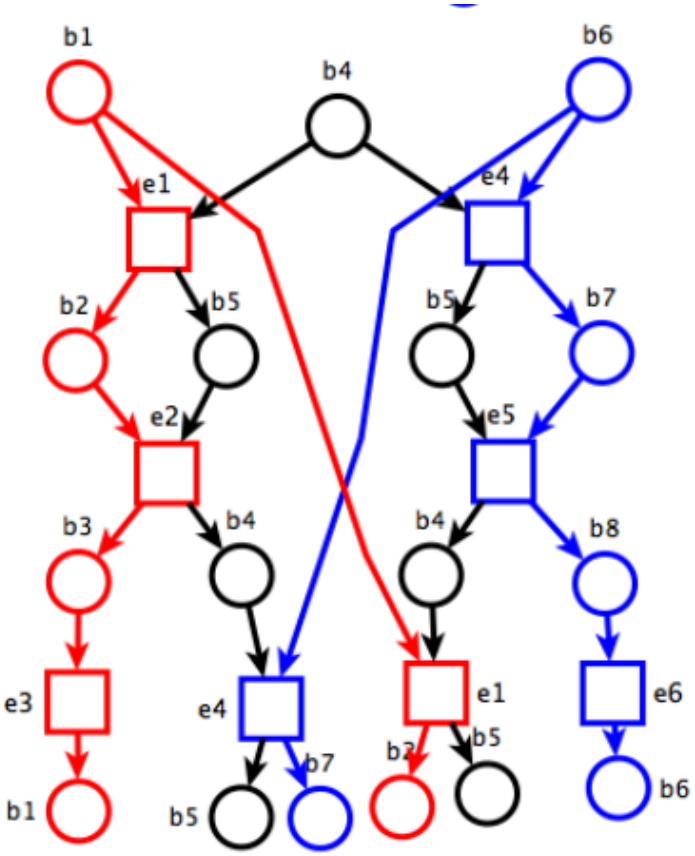


Figure 4.3: example

4.8 Prefissi

Dato un sistema elementare Σ e due suoi processi ramificati $P_1 = (N_1, \phi_1), P_2 = (N_2, \phi_2)$, P_1 si dice prefisso di P_2 sse:

N_1 è sottorete di N_2 e $\phi_{2|N_1} = \phi_1$

4.9 Unfolding

Un sistema elementare Σ ammette un solo massimale rispetto alla relazione di prefisso tra processi ramificati. Questo processo è chiamato **unfolding** ($= Unf(\Sigma)$). Ogni processo non sequenziale di Σ è un prefisso di $Unf(\Sigma)$.

Da prendere con le pinze, ma a quanto pare un processo sequenziale è un processo ramificato tale che la rete sia una rete causale (senza conflitti). Boh si chiama anche corsa (run).

Chapter 5

Reti ad Alto Livello

Le reti elementari sono belle ma nel momento in cui io dovessi arricchire il protocollo che rappresentano come potrei aggiungere dettagli senza modificare la struttura di una rete? Semplice: reti piu' complesse!

5.1 Reti di Posti e Transizioni

$\Sigma = (S, T, F, K, W, M_0)$ si dice sistema di Posti e Transizioni (sistema P/T) sse:

- (S, T, F) è una rete
- $K : S \to N^+ \cup \{\infty\}$ è la funzione capacita' dei posti
- $W : F \to N$ è la funzione peso degli archi
- $M_0 : S \to N : \forall s \in S \;\; M_0(s) \leq K(s)$ è la marcatura iniziale

regola di scatto

- $M[t$ sse $\forall s \in S \;\; M(s) \geq W(s,t) \wedge M(s) + W(t,s) \leq K(s)$
- $M[t > M'$ sse $M'(s) = M(s) - W(s,t) + W(t,s)$

L'insieme delle marcature raggiungibili di Σ è $[M_0$:

- $M_0 \in [M_0 >$
- se $M \in [M_0 > \wedge \exists t \in T : M[t > M' alloraM' \in [M_0 >$

Valgono tutte le considerazioni delle Reti di petri su conflitto, concorrenza e indipendenza.

Grafo delle marcature raggiungibili Sia $\Sigma = (S, T, F, K, W, M_0)$ un sistema P/T. $RG(\Sigma) = ([M_0 >, U_\Sigma, A, M_0)$ è il suo grafo dove:

$A = \{(M, U, M') \;:\; M, M' \in [M_0 > \wedge U \in U_\Sigma \wedge M[U > M'\}$.

Se U è una singola transizione (insieme singoletto), allora si ha il grafo di raggiungibilita' sequenziale $SRG(\Sigma)$.

Tensio': La **Diamond Property** non è piu' valida in generale perchè qui si ammettono i self loop.

Σ è detto limitato sse $\exists n \in N$, tale che $\forall s \in S, \forall M \in [M_0 >, M(s) \leq N$. Σ è limitato $\Leftrightarrow [M_0 >$ è un insieme finito (il grafo delle marcature è finito).

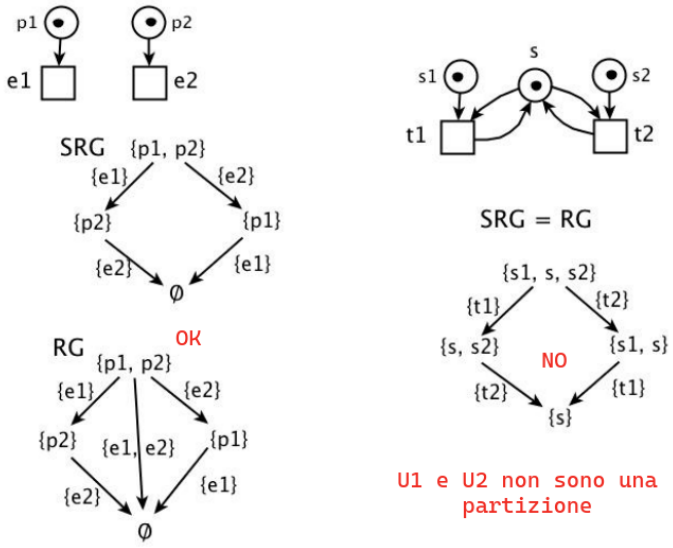


Figure 5.1: example

5.2 Contatti

Un sistema $P/T\ \Sigma = (S, T, F, K, W, M_0)$ è detto **senza contatti** sse:

- $\forall M \in [M_0 >, \forall t \in T, \forall s \in S$
- $M(s) \leq W(s,t) \Rightarrow M(s) + W(t,s) \leq K(s)$

Per rendere un sistema senza contatti si puo' complementare i posti come nel caso delle reti di petri. La conseguenza di questa condizione è che l'abilitazione di una transizione $t \in T$ in $M \in [M_0 >$ non dipende piu' dalla capacita' dei posti, ma solo dalla marcatura del posto e dal peso degli archi.

5.3 Reti Marcate

Un sistema $P/T\ \Sigma = (S, T, F, K, W, M_0)$ è detto **rete marcata** sse:

- $\forall s \in S,\ M_0(s) \in N \wedge K(s) = \infty$
- $\forall t \in T,\ W(s,t) \leq 1 \wedge W(t,s) \leq 1$

Ovvero nel computo dell'abilitazione di una transizione, il peso degli archi e la capacita' dei posti sono ridondanti ovvero ininfluenti.

Sicurezza Una Rete Marcata è detta sicura ss: $\forall M \in [M_0 >, \forall s \in S,\ M(s) \leq 1$.

Sorpresa delle sorprese, una rete marcata **sicura** e **pura** è anche una rete di petri **pura**.

5.4 Matrice di Incidenza

Sia $\Sigma = (S, T, F, K, W, M_0)$ un sistema P/T tale che $\forall s \in S$, $K(s) = \infty$ e $N = (S, T, F)$ sia una pura.

Il sistema puo' essere rappresentato con la **matrice di incidenza** $\underline{N} : S \times T \to N$

Chapter 6

Modelli

6.1 Modelli di Kripke

Sia $AP = \{p, q, z...\}$ un insieme di proposizioni atomiche. Si dice Modello di Kripke la tupla $M = (Q, T, I)$ dove (Q, T) è un sistema di transizioni e $I : Q \to 2^{AP}$ è la semantica che definisce l'insieme di proposizioni vere nello stato q.

Si denota cammino $\pi = q_0q_1q_2q_3q_4....$, dove $\forall i, (q_i, q_{i+1}) \in T$. Il suo suffisso i-esimo è denotato $\pi^{(i)} = q_iq_{i+1}q_{i+2}....$

Quello che è importante identificare nei modelli di Kripke sono tutti i cammini massimali, tra cui i cammini π infiniti, distinti del sistema. In buona sostanza è simile alle tracce.

6.2 LTL

La **Logica Temporale Lineare** è l'insieme delle formule ben formattate FBF_{LTL}:

- le proposizioni atomiche in AP
- True e False
- le composizioni della logica proposizionale $\alpha \to \beta$, $\alpha \wedge \beta$ ...
- $X\alpha$, nel prossimo stato
- $F\alpha$, prima o poi
- $G\alpha$, sempre
- $U(\alpha, \beta)$, vale α fino a che non vale β (implica $F\beta$)

Data una semantica per definire se una fbf è vera in un certo cammino massimale π, si dice che la formula sia vera in q sse è vera in tutti i cammini massimali che partono da q.

6.2.1 In altre parole

Si definisce l'operatore di induzione $\models$, tale che $\pi \models \alpha$ sse α è vera nel cammino π. Due formule si dicono *equivalenti* sse $\pi \models \alpha \Leftrightarrow \pi \models \beta$. Quindi posso riscrivere la semantica degli operatori della LTL come:

- $\pi \models X\alpha$ sse $\pi^1 \models \alpha$
- $\pi \models F\alpha$ sse $\exists i \in N \ : \ \pi^i \models \alpha$
- $\pi \models G\alpha$ sse $\forall i \in N \ : \ \pi^i \models \alpha$
- $\pi \models U(\alpha, \beta)$ sse $\exists i \in N \ : \ \pi^i \models \beta \ \wedge \ \forall \in [0, i) \ : \ \pi^j \models \alpha$

Allo stesso modo due modelli M_1, M_2 con stati iniziali p, q, si dicono equivalenti rispetto ad una logica L sse $\forall \alpha \in FBF_L \ : \ M_0, p \models \alpha \Leftrightarrow M_1, q \models \alpha$.

6.2.2 Operatori Derivati

- While (o Weak Until), $W(\alpha, \beta) \equiv G\alpha \vee U(\alpha, \beta)$
- Release, $R(\alpha, \beta) \equiv W(\beta, \beta \wedge \alpha)$, ovvero α rilascia β

6.2.3 Insieme Minimale di Operatori

L'insieme $\{X, U\}$ forma un insieme minimale perchè è possibile derivare tutti gli altri. (TODO: provaci)

6.2.4 Limiti espressivi

Uno dei più importanti riguarda la negazione delle formule riguardo all'implicazione che una sia vera nello stato sse è vera in tutti i cammini. In fact non è possibile esprimere appunto questi quantificatori in modo accurato, e in generale non è possibile esprimere la nozione di esistenza di cammino speciale. Un esempio:

"Non è vero che $F\alpha$" non è equivalente a dire $\neg F\alpha$ perchè la negazione è immersa nel quantificatore implicito di "tutti i cammini" che costituisce la semantica della LTL. Al contrario $\neg F\alpha \equiv G\neg\alpha$, ovvero in ogni cammino non è vero che prima o poi α diventa vera.

6.3 CTL

La **Computation Tree Logic** è l'insieme delle formule ben formattate FBF_{CTL}:

- le proposizioni atomiche in AP
- True e False
- le composizioni della logica proposizionale $\alpha \rightarrow \beta$, $\alpha \wedge \beta$ ...
- $AX\alpha$, $EX\alpha$, nel prossimo stato
- $AF\alpha$, $EF\alpha$, prima o poi
- $AG\alpha$, $EG\alpha$, sempre
- $AU(\alpha, \beta)$ $EU(\alpha, \beta)$, vale α fino a che non vale β (implica $F\beta$)

Dove A e E sono i quantificatori sui cammini: $A \equiv$ per ogni cammino, $E \equiv$ esiste un cammino tale che.

6.3.1 Differenze Espressive

Come già detto prima, in LTL non è possibile dare la nozione di possibilità di esistenza di un cammino. Ebbene anche CTL non può esprimere alcuni concetti. Prendiamo un esempio.

"In tutti i cammini prima o poi si raggiunge uno stato a partire dal quale p è sempre vera".

- In logica LTL si può dire FGp (ricordando che la nozione di tutti i cammini è assunta per costruzione di semantica LTL)
- In logica CTL si è tentati di usare lo stetto costrutto, ma dovendo inserire i quantificatori in precedenza agli operatori temporali si è costretti a dire $AFAGp$

Se queste due formule fossero equivalenti allora $M, q \models AFAGp \Leftrightarrow M, q \models FGp$ per ogni $q \in Q$. Tuttavia non è così:

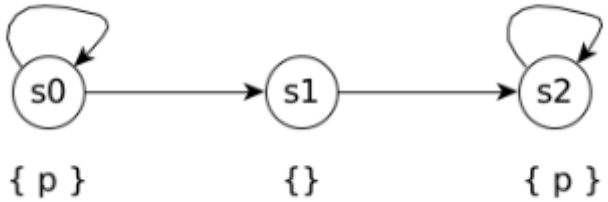


Figure 6.1: Esempio

$M, s_0 \not\models AFAGp$ ma $M, s_0 \models FGp$. Per comprendere meglio questa diversità è utile spezzettare la formula CTL:

- $k \equiv AGp$.
- $AFAGp \equiv AFk$.
- $M, s_0 \not\models k$, $M, s_1 \not\models k$, $M, s_2 \models k$.
- ora, $M, s_1 \models AFk$ e $M, s_2 \models AFk$ ma $M, s_0 \not\models AFk$ perchè esiste il cammino massimale $s_0, s_0, s_0...$ dove non è vero che prima o poi k.

6.4 CTL^*

Questa logica estende LTL e CTL rimuovendo il vincolo di avere un quantificatore accanto ad un operatore.

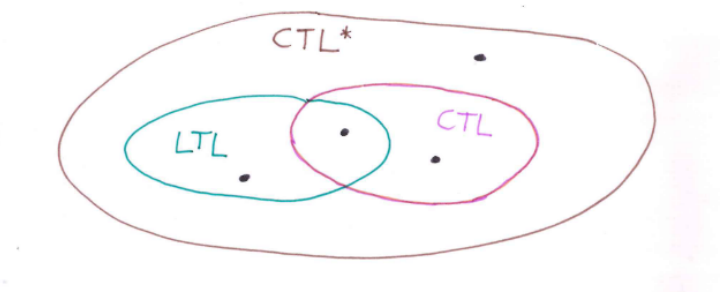


Figure 6.2: Esempio

In questo modo FGp è una formula FBF.

6.4.1 Perchè no?

La ragione ricalca un pò alcuni problemi di LTL: quest'ultimo oltre ad essere poco espressivo possiede un tempo di model checking esponenziale. Visto che CTL può essere verificato bottom up (pezzo per pezzo, in forma di enunciati semplici) la sua verifica ha tempo lineare. Poi il problema di tutto questo è che nonostante sia lineare rispetto al numero di stati de facto se è estratta da una rete di petri, il numero di stati tende ad essere esponenziale.

6.5 $\mu - calcolo$

È una logica più complessa di CTL che permette di definire formule ricorsive.

Si supponga per esempio di voler esprimere $EF\alpha$ con il solo operatore X: $EF\alpha = \alpha \vee EX\alpha \vee EXEX\alpha \vee EXEXEX\alpha$.... Ovvero $EF\alpha \equiv \alpha \vee EX(EF\alpha)$, con il μ-calcolo: $\mu Y.(a \vee EXY)$.

Allo stesso modo si supponga per esempio di voler esprimere $AG\alpha$ con il solo operatore X: $AG\alpha = \alpha \wedge AX\alpha \wedge AXAX\alpha \wedge AXAXAX\alpha$.... Ovvero $AG\alpha \equiv \alpha \wedge AX(AG\alpha)$, con il μ-calcolo: $\mu Y.(a \wedge AXY)$.

A quanto pare la semantica del calcolo μ è definita sui modelli di Kripke attraverso operatori di punto fisso.

Questa logica ha la massima potenza espressiva ($\mu - calcolo \supset CTL^*$) al costo di un'alta complessità di calcolo nella verifica e nella leggibilità delle formule stesse (formule "oscure").

A proposito di questo riporto le stime di complessità (indicative) di un modello di Kripke M e una formula f:

- $CTL : O(|M| \times |f|)$
- $LTL : O(|M| \times 2^{|f|})$

Chapter 7

Model Checking

7.1 Reticoli

Un **reticolo** è un insieme ordinato parzialmente ordinato $R = (L, \leq)$ tale che $\forall x, y \in L$ esistono $x \wedge y$ e $x \vee y$. Un reticolo si dice completo se $\vee B$ e $\wedge B$ esistono per ogni $B \subseteq L$.

7.2 Monotonia

Siano X, Y due insiemi parzialmente ordinati, data una funzione $f : X \to Y$, essa si dice monotona sse $\forall a, b \in X$, $a \geq b \Rightarrow f(a) \geq f(b)$ e $a \leq b \Rightarrow f(a) \leq f(b)$.

7.3 Punti Fissi

Data una funzione monadica $f : X \to X$, $x \in X$ è un punto fisso sse $f(x) = x$.

7.4 Teorema di Knaster Tarski

Sia $R = (L, \leq)$ un reticolo completo, e $f : L \to L$ una funzione monotona. Allora f ha un minimo e un massimo punto fisso.

7.5 Teorema di Kleene

Sia una funzione $f : 2^A \to 2^A$, essa si dice continua se $f(\cup X_i) = \cup F(X_i)$.
Allora posso trovare:

- il massimo punto fisso calcolando $f(A), f(f(A)), f(f(f(A)))$...
- il minimo punto fisso calcolando $f(\emptyset), f(f(\emptyset)), f(f(f(\emptyset)))$...

7.6 Algoritmo per CTL

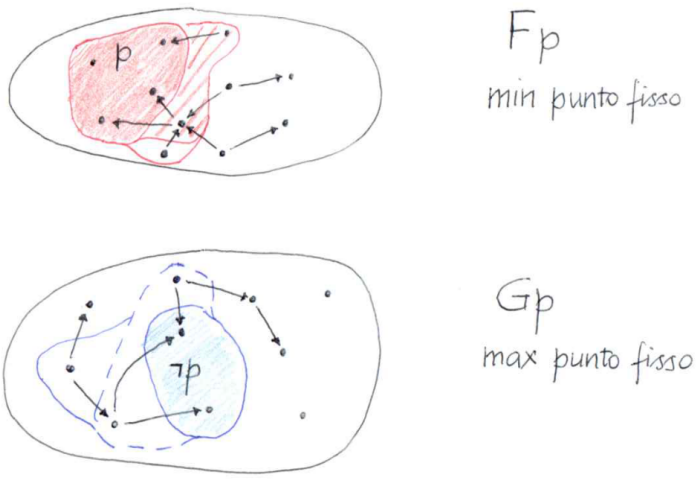


Figure 7.1: Esempio

Siano $M = (Q, T, I)$ un modello di Kripke e α una formula. Si definisce $[[\alpha]]$ l'estensione di *alpha*, dove $[[\alpha]] = \{q \in Q \mid M, q \models \alpha\}$.

Ora sia $\alpha \equiv AF\beta$. A questa formula si associa una funzione $f_\alpha : 2^Q \to 2^Q$ tale che, per ogni $H \subseteq Q$:

- $f_\alpha(H) = [[\beta]] \cup \{p \in Q \mid \forall (p, q) \in T \; q \in H\}$
- ovvero ad ogni passo seleziono solo gli stati tali per cui posso andare solo in H e "li aggiungo"
- $f_a(\emptyset) = \beta$ trivialmente
- Il minimo punto fisso di f_a è $[[\alpha]]$

Vice versa sia ora $\alpha \equiv EG\beta$. A questa formula si associa una funzione $g_\alpha : 2^Q \to 2^Q$ tale che, per ogni $H \subseteq Q$:

- $g_\alpha(H) = [[\beta]] \cap \{p \in Q \mid \exists (p, q) \in T \; q \in H\}$
- ovvero ad ogni passo seleziono solo gli stati tali per cui posso andare in H e "rimuovo" gli altri
- $g_a(Q) = \beta$ trivialmente
- Il massimo punto fisso di g_a è $[[\alpha]]$

7.7 Automi di Buchi

Gli automi di Buchi sono automi che riconoscono parole infinite su un alfabeto finito Σ. Invece che avere stati finali abbiamo stati "accettanti", e il criterio di accettazione per questo genere di automi e' di conseguenza: "l'automa accetta se durante la computazione passa infinite volte per almeno uno degli stati accettanti".

7.8 Algoritmo per LTL

Per calcolare la validità di una formula LTL su un Modello di Kripke dobbiamo usare il seguente metodo.

- Si costruisce l'automa di Buchi $B_{\neg\alpha}$
- Si trasforma M in automa etichettato da insiemi di proposizioni atomiche (ogni arco entrante ha come label le proposizioni atomiche di uno stato)
- Si calcola il prodotto sincrono dei due automi PS
- Si verifica se $L(PS) = \emptyset$ (decidibile), allora $M, q_0 \models \alpha$

Questa cosa "funziona" solo perchè con la composizione completamente sincrona si fa in modo che i due automi eseguano le stesse istruzioni somehow. Non l'ho capita allora ma non l'ha nemmeno spiegato in realtà, l'ha dato per scontato.

7.9 Fairness

Un ultimo concetto da presentare è la nozione di Fairness su sistemi elementari. Un'esecuzione si dice **unfair** se un evento è abilitato ma non scatta mai. Questa eventualità esiste (e per certi versi rappresenta i limiti espressivi di un modello sequenziale, ma sono parole in libertà), per questa ragione si vuole limitare la valutazione di una formula alle esecuzioni **fair**.

Esistono due tipi di fairness:

- Fairness debole: $\forall i : (FGen_i) \Rightarrow (F(ex_i))$
- Fairness forte: $\forall i : (GFen_i) \Rightarrow (GF(ex_i))$

Part II

Esami

Chapter 8

Esami

8.1 Indicazioni sullo Scritto

- fare attenzione alle conseguenze logice nella dimostrazione di correttezza
- fare attenzione ad indicare le derivazioni corrette e a partire dalla post-condizione per ricavare la precondizione (mai vice versa, perchè sarebbe cannato)
- scrivere l'esercizio sulla bisimulazione solo dopo aver identificato la strategia vincente per attaccante o difensore
- se usi la notazione sbagliata per qualcosa "fa niente" ma te lo fa notare lo stesso
- la dimostrazione di correttezza e la bisimulazione sono molto importanti a livello di punteggio, e personalmente da molto valore a queste
- assicurarsi di aver considerato tutte le possibili mosse per il difensore prima di fare qualcosa
- al 90% i sistemi CCS non sono bisimili, ma può capitare che lo siano. Ad ogni modo a volte nello scritto ti dice direttamente che non lo sono o vice versa.

8.2 Indicazioni sull'Orale

- mi ha chiesto rispettivamente
-
  - Cos'è una dimostrazione nella logica di Hoare, di fare un esempio di derivazione e cose cosí
  - Di spiegare come funziona il gioco della bisimulazione (proprio in parole molto semplici e astratte, cioè l'attaccante fa una mossa e il difensore risponde, le regole di base e chi vince e quando vince)
  - Quando due eventi si dicono indipendenti e che tipo di informazioni ne ricaviamo (passo abilitato, diamond property per il grafo dei casi)
  - Cosè CTL^* e perchè non si usa

In generale ho visto che è abbastanza buono con l'orale, non tanto per i voti quanto per il fatto che prova ad ogni costo a farti ragionare e cerca di aiutarti a capire dove hai sbagliato spiegandoti per filo e per segno.

8.3 Esercizi da portare all'orale

8.3.1 Dimostrazione del Teorema di Knaster Tarski per il massimo punto fisso

- Costruiamo l'insieme $Z = \{T \subseteq A \mid T \subseteq f(T)\}$.
- Poniamo $M = \cup Z$.
- $\forall T \in Z \mid T \subseteq M$, Quindi $T \subseteq f(T) \subseteq f(M)$.
- $\forall T \in Z \mid T \subseteq f(M)$, Quindi $\cup Z = M \subseteq f(M)$.
- Quindi $M \in Z$.
- Visto che f é monotona: $M \subseteq f(M) \ \rightarrow \ f(M) \subseteq f(f(M))$.
- Il che significa che $f(M) \in Z$, ovvero che $f(M) \subseteq M$.
- Quindi M è il massimo punto fisso del Reticolo su f.

8.3.2 Formula f di espansione della formula $c \equiv AU(\alpha, \beta)$

- $c \equiv A\ \alpha\ U\ \beta$
- $f(H) = [[b]] \cup (\{q \in Q \mid \forall (q,p) \in T \mid p \in H\} \cap [[a]])$
- $f(\emptyset) = [[b]]$
- $min\ f \equiv H \subseteq ([[a]] \cup [[b]])$