

UNIVERSITÀ DEGLI STUDI DI VERONA

Sistemi ad eventi discreti

RIASSUNTO DEI PRINCIPALI ARGOMENTI

Giorgia Gulino, Davide Bianchi

February 6, 2018

Contents

1	Macchine a stati	2
1.1	Output-Deterministico	2
1.2	Non-Deterministico	2
1.3	Non-Deterministico, Progressiva	2
1.4	Equivalenza	2
1.5	Raffinamento	2
1.6	Bisimulazione	2
1.7	Isomorfe	2
1.8	Rel. RAFFINAMENTO/SIMULAZIONE AFSND \rightarrow AFSD	2
1.9	Rel. RAFFINAMENTO/SIMULAZIONE AFSND \rightarrow AFSpseudo-nondet	2
1.10	Rel. RAFFINAMENTO/SIMULAZIONE AFSND \rightarrow AFSND	3
1.11	Simulazione per Det $\rightarrow M_1$ da M_2	3
1.12	Simulazione per Output-Det	3
1.13	Simulazione	3
1.14	Bisimulazione per Det	3
2	LINGUAGGI	3
2.1	Il linguaggio K è <i>controllabile</i> ?	3
2.2	Osservabilità	4
2.3	Proprietà di Controllabilità	4
2.4	Riguardo il sottolinguaggio supremo	4
2.5	Riguardo il sovralinguaggio infimo	5

1 Macchine a stati

Definizione 1.0.1 (Macchina a stati deterministica) *Una macchina a stati deterministica esiste un solo uno stato iniziale. Inoltre per ogni stato e per ogni input esiste solo un stato successivo. Se M_2 è deterministica allora M_1 è **simulata** da M_2 sse è **equivalente** a M_2 .*

1.1 Output-Deterministico

Solo uno stato iniziale e per ogni stato e ogni coppia di I/O c'è 1 solo stato successivo. Se M_2 è **Output-Det** allora M_2 **simula** M_1 sse M_1 **raffina** M_2 .

Deterministico implica **Output-Det**, ma **non** viceversa.

1.2 Non-Deterministico

Può esserci più di uno stato iniziale e per ogni stato e ogni coppia di I/O può esserci + di uno stato successivo. Se M_2 è NON-DET, M_1 è **simulata** da M_2 allora M_1 **raffina** M_2 , ma non viceversa.

1.3 Non-Deterministico, Progressiva

Progressiva significa che l'evoluzione è definita per ogni ingresso, cioè la funzione è definita come: Stati x ingressi \rightarrow P(Stati x uscite)/insieme vuoto, dove P rappresenta l'insieme potenza e l'insieme vuoto impone che sia progressiva.

1.4 Equivalenza

- X Det: $\text{input}[M_1] = \text{input}[M_2]; \text{output}[M_1] = \text{output}[M_2]$.
- X Non-det: $\text{comportamento}[M_1] = \text{comportamento}[M_2]$.
- Cioè se M_1 **raffina** M_2 e viceversa.
- C'è equivalenza se c'è **bisimulazione**.

1.5 Raffinamento

M_1 **raffina** M_2 sse $\text{input}[M_1] = \text{input}[M_2]; \text{output}[M_1] = \text{output}[M_2]$ e $\text{comportamento}[M_1] \subseteq \text{comportamento}[M_2]$.

1.6 Bisimulazione

Bisimulazione tra M_1 e M_2 sse l'unione delle **simulazioni** è simmetrica e c'è **isomorfismo** tra $\text{minimize}(M_1)$ e $\text{minimize}(M_2)$.

1.7 Isomorfe

Si dicono isomorfe se hanno lo stesso numero di stati con nome uguale.

1.8 Rel. RAFFINAMENTO/SIMULAZIONE AFSND \rightarrow AFSD

Se M_1 è det, M_1 è **simulata** da M_2 sse M_1 è equivalente a M_2 , cioè se M_1 raffina M_2 e viceversa.

1.9 Rel. RAFFINAMENTO/SIMULAZIONE AFSND \rightarrow AFSpseudo-nondet

Se M_2 è psuedo-non det, M_1 è **simulata** da M_2 sse M_1 è equivalente a M_2 , cioè se M_1 **raffina** M_2 .

1.10 Rel. RAFFINAMENTO/SIMULAZIONE AFSND \rightarrow AFSND

Se M_2 non è deterministica, M_1 è **simulata** da M_2 implica M_1 **raffina** M_2 , ma M_1 raffina M_2 non implica M_1 **simula** M_2 .

1.11 Simulazione per Det $\rightarrow M_1$ da M_2

- $\forall p \in \text{PossibiliInitialState}[M_1], \exists q \in \text{PossibiliInitialState}[M_2], (p,q) \in S$.
- $\forall p \in \text{Stati}[M_1], \forall q \in \text{Stati}[M_2]$.
 - if $(p,q) \in S \Rightarrow \forall x \in \text{Input}, \forall y \in \text{Output}, \forall p1 \in \text{Stati}[M_1]$;
 - if $(p1,y) \in \text{PossibiliUpdates}[M_1](p,x) \Rightarrow \exists q1 \in \text{Stati}[M_1], (q1,y) \in \text{PossibiliUpdates}[M_2](q,x)$ e $(p1,q1) \in S$. (S contiene coppie di stati iniziali e coppie consultanti l'algoritmo).
 - $\forall p \in \text{Stati}[M_1] \exists q \in \text{Stati}[M_2]$ per cui $\forall I/O$ possibili c'è corrispondenza tra I/O uguali di p e $(p,q) \in S$.

1.12 Simulazione per Output-Det

Data M ASFND trova la macchina output-det $\text{det}(M)$ equivalente a M.

SUBSET CONSTRUCTION

- $\text{InitialState}[\text{det}(M)] = \text{PossibileInitialState}[M]$
- $\text{States}[\text{det}(M)] = \text{InitialState}[\text{det}(M)]$
- Ripeti finché nuove transizioni possono essere aggiunte a $\text{det}(M)$. Scegli
 - $P \in \text{States}[\text{det}(M)]$ e $(x,y) \in \text{Input} \times \text{Output}$
 - $Q = \{q \in \text{States}[M] \mid \exists p \in P, (q,y) \in \text{PossibileUpdates}[M](p,x)\}$ Se $Q \neq \emptyset$ allora $\text{States}[\text{det}(M)] = \text{States}[\text{det}(M)] \cup Q$
 $\text{Update}[\text{det}(M)](p,x) = (q,y)$
Raggruppa tutti gli stati iniziali, \forall coppia I/O raggruppa tutti gli stati per cui quest'ultima è Possibileupdate.

1.13 Simulazione

- Se $p \in \text{PossibileInitialState}[M_1]$ e $\text{PossibileInitialState}[M_2] = q \Rightarrow (p,q) \in S$.
- Se $(p,q) \in S$ e $(p1,y) \in \text{PossibileUpdates}[M_1](p,x)$ e $\text{PossibileUpdates}[M_2](q,x) = q$.

1.14 Bisimulazione per Det

Una relazione binaria B è una *bisimulazione* sse:

- $\text{InitialState}[M_1], \text{InitialState}[M_2] \in B$
- $\forall p \in \text{Stati}[M_1], \forall q \in \text{Stati}[M_2]$:
 - if $(p,q) \in B \Rightarrow \forall x \in \text{Input}[M_1], \text{Output}[M_1](p,x) = \text{Output}[M_2](q,x)$
 $(\text{nextState}[M_1](p,x), \text{nextState}[M_2](q,x)) \in B$. Stati iniziali di M_1 e M_2 sono in relazione e ogni coppia (p,q) relazionati, \forall input producono lo stesso output e nextState Relazionati.

2 LINGUAGGI

2.1 Il linguaggio K è *controllabile*?

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 : $s\sigma \in M \Rightarrow s\sigma \in \overline{K}$. (Equivalente a $\overline{K}E_{uc} \cap M \subseteq \overline{K}$.)

Per la def di *controllabilità* si ha che K è controllabile sse \overline{K} è controllabile.

2.2 Osservabilità

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 $E^* \Rightarrow E_o^*$.

Si dice che K è osservabile rispetto a M , E_o, E_c se per tutte le stringhe $s \in \overline{K}$ e per tutti gli eventi $\sigma \in E_c$ abbiamo:

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

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 $s'\sigma \in \overline{K}$. Tali due stringhe richiederebbero un'azione di controllo diversa rispetto a σ (disabilitare σ nel caso di s , abilitare σ nel caso di s'), ma un supervisore non saprebbe distinguere tra s e s' per l'osservabilità ristretta. Non potrebbe quindi esistere un supervisore che ottiene esattamente il linguaggio \overline{K} .

2.3 Proprietà di Controllabilità

Esistono due tipi di linguaggi derivati da K :

- $K^{\uparrow C}$ il *sottolinguaggio supremo* di K
- $K^{\downarrow C}$ il *sovralinguaggio controllabile infimo* di K

Abbiamo i seguenti rapporti:

$$\emptyset \subseteq K^{\uparrow C} \subseteq K \subseteq \overline{K} \subseteq K^{\downarrow C} \subseteq M$$

- Se K_1 e K_2 sono controllabili, allora $K_1 \cup K_2$ è controllabile.
- Se K_1 e K_2 sono controllabili, allora $K_1 \cap K_2$ non ha bisogno di essere controllabile.
- Se K_1 e K_2 sono non in conflitto ed entrambi controllabili, allora $K_1 \cap K_2$ è controllabile. **Si ricorda che K_1 e K_2 si dicono non in conflitto qualora $\overline{K_1} \cap \overline{K_2} = \overline{K_1 \cap K_2}$**
- Se K_1 e K_2 sono prefisso-chiuso e controllabili, allora $K_1 \cap K_2$ è prefisso-chiuso e controllabile.

Definiamo due classi di linguaggi:

$$C_{in}(K) := L \subseteq K : \overline{L}E_{uc} \cap M \subseteq \overline{L}$$

$$CC_{out}(K) := L \subseteq E^* : (K \subseteq L \subseteq M) \wedge (\overline{L} = L) \wedge (\overline{L}E_{uc} \cap M \subseteq \overline{L})$$

2.4 Riguardo il sottolinguaggio supremo

- $C_{in}(K)$ è un insieme parzialmente ordinato (*o poset*) che è chiuso sotto unioni arbitrarie.
- $C_{in}(K)$ possiede un unico elemento *supremo*. Definito come:

$$K^{\uparrow C} := \bigcup_{L \in C_{in}(K)} L$$

che è un elemento ben-definito di $C_{in}(K)$.

- $K^{\uparrow C}$ è chiamato *sottolinguaggio supremo controllabile* di K .
 - Nel caso peggiore, $K^{\uparrow C} = \emptyset$, dal momento che $\emptyset \in C_{in}(K)$
 - Se K è controllabile, allora $K^{\uparrow C} = K$
 - Osserviamo che $K^{\uparrow C}$ non necessita di essere prefisso-chiuso in generale

2.5 Riguardo il sovralinguaggio infimo

- $CC_{\text{out}}(K)$ è un(*poset*) che è chiuso sotto intersezioni arbitrarie (e unioni).
- $CC_{\text{out}}(K)$ possiede un unico elemento *infimo*. Definito come:

$$K^{\downarrow C} := \bigcap_{L \in CC_{\text{out}}(K)} L$$

che è un elemento ben-definito di $CC_{\text{out}}(K)$.

- Chiamiamo $K^{\downarrow C}$ il sovralinguaggio infimo a prefisso-chiuso e controllabile di K .
 - Nel caso peggiore, $K^{\downarrow C} = M$, dal momento che $M \in CC_{\text{out}}(K)$.
 - Se K è controllabile, allora $K^{\downarrow C} = \overline{K}$.