Università degli Studi di Verona

Sistemi ad eventi discreti

Riassunto dei principali argomenti

Giorgia Gulino, Davide Bianchi

February 15, 2018

Contents

1 Macchine a stati

Definizione 1.0.1 (Macchina a stati deterministica) *Una macchina a stati è deterministica se valgono:*

- *esiste un solo uno stato iniziale;*
- *per ogni stato e per ogni input esiste solo un stato successivo;*

Inoltre se M_2 è deterministica allora M_1 è **simulata** da M_2 sse è **equivalente** a M_2.

1.1 Output-Determinismo

Definizione 1.1.1 *Una macchina a stati è **output-deterministica** se esiste un solo stato iniziale e per ogni stato e ogni coppia di I/O c'è un solo stato successivo. Se M_2 è **output-deterministica** allora M_2 **simula** M_1.*

determinismo $\Rightarrow$ **output-determinismo**, non vale il viceversa.

1.2 Non-Determinismo

In una macchina a stati non deterministica può esistere più di uno stato iniziale e per ogni stato e ogni coppia di I/O può esistere più di uno stato successivo. Se M_2 è non deterministica, M_1 è **simulata** da M_2 allora M_1 **raffina** M_2, ma non viceversa.

Una macchina a stati è progressiva quando l'evoluzione è definita per ogni ingresso, cioè la funzione è definita come

$$States \times Inputs \Rightarrow \mathcal{P}(States \times Outputs) \setminus \emptyset$$

dove $\mathcal{P}$ rappresenta l'insieme potenza e l'insieme vuoto impone che sia progressiva.

1.3 Equivalenze

Data una macchina a stati X:

- se X è deterministica: $input[M_1] = input[M_2]; output[M_1] = output[M_2]$.
- se X è non deterministica: $Behaviour[M_1] = Behaviour[M_2]$.
- se due macchine a stati M_1 e M_2 sono bisimili, allora sono equivalenti.

Definizione 1.3.1 (Raffinamento) M_1 ***raffina*** $M_2 \Leftrightarrow \Big(Inputs[M_1] = Inputs[M_2] \wedge Outputs[M_1] = Outputs[M_2] \wedge Behaviour[M_1] \subseteq Behaviour[M_2]\Big)$.

1.4 Bisimulazione

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

1.5 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.

1.6 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.

1.7 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.

1.8 Simulazione per Det $\rightarrow$ M_1 da M_2

- $\forall$ p $\in$ PossibiliInitialState[M_1], $\exists$ q $\in$ PossibiliInitialState[M_2], (p,q) $\in$ S.
- $\forall$ p $\in$ Stati[M_1], $\forall$ q $\in$ Stati[M_2].
  - if $(p, q) \in S \Rightarrow \forall x \in Input, \forall y \in Output, \forall p_1 \in Stati[M_1]$;
  - if (p1,y) $\in$ PossibiliUpdates[M_1](p,x) $\Rightarrow$ $\exists$ q1 $\in$ Stati[M_1], (q1,y) $\in$ PossibiliUpdates[M_2](q,x) e (p1,q1) $\in$ S. (S contiene coppie di stati iniziali e coppie consultanti l'algoritmo).
  - $\forall$ p $\in$ Stati[M_1] $\exists$ q $\in$ Stati[M_2] per cui $\forall$ I/O possibili c'è corrispondenza tra I/O uguali di p e (p,q) $\in$ S.

1.9 Simulazione per Output-Det

Data M ASFND trova la macchina output-det det(M) equivalente a M.
SUBSET CONSTRUCTION

- InitialState[det(M)] = PossibileInitialState[M]
- States[det(M)]=InitialState[det(M)]
- Ripeti finché nuove transizioni possono essere aggiunte a det(M). Scegli
  - P $\in$ States[det(M)] e (x,y) $\in$ Input x Output
  - Q = q $\in$ States[M] — $\exists$ p $\in$ P, (q,y) $\in$ PossibleUpdates[M](p,x) Se Q $\neq$ 0 allora
    States[det(M)]= States[det(M)] $\cup$ Q
    Update[det(M)](p,x)=(q,y)
    Raggruppa tutti gli stati iniziali, $\forall$ coppia I/O raggruppa tutti gli stati per cui quest'ultima è Possibleupdate.

1.10 Simulazione

- Se p $\in$ PossibleInitialState[M_1] e PossibleInitialState[m2] = q $\Rightarrow$ (p,q) $\in$ S.
- Se (p,q) $\in$ S e (p1,y) $\in$ PossibleUpdates[M_1](p,x) e PossibleUpdates[M_2](q,x) = q.

1.11 Bisimulazione per Det

Una relazione binaria B è una *bisimulazione* sse:

- InitialState[M_1], InitialState[M_2] $\in$ B
- $\forall$ p $\in$ Stati[M_1], $\forall$ q $\in$ Stati[M_2]:
  - if (p,q) $\in$ B $\Rightarrow$ $\forall$ x $\in$ Input[M_1], Output[M_1](p,x) = Output[M_2](q,x)
    (nextState[M_1](p,x),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 : $\mathbf{s}\sigma \in \mathbf{M} \Rightarrow \mathbf{s}\sigma \in \overline{K}$. (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_0*.
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 *il sottolinguaggio supremo di K*
- $K^{\downarrow C}$ il *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 $\mathbf{K_1}$ e $\mathbf{K_2}$ 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:

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

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

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:

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

  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_{out}(K) è un(*poset*) che è chiuso sotto intersezioni arbitrarie (e unioni).
- CC_{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_{out}(K).
- Chiamiamo K$^{\downarrow\mathrm{C}}$ il sovralinguaggio infimo a prefisso-chiuso e controllabile di K.
  - Nel caso peggiore, K$^{\downarrow\mathrm{C}}$ = M, dal momento che M $\in CC_{out}$(K).
  - Se K è controllabile, allora K$^{\downarrow\mathrm{C}}$ = $\overline{K}$.