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

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 → 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: input[M_1]=input[M_2]; output[M_1]=output[M_2].
- X Non-det: comportamento[M_1]=comportamento[M_2].
- Cioè se M_1 **raffina** m2 e viceversa.
- C'è equivalenza se c'è **bisimulazione**.

1.5 Raffinamento

M_1 **raffina** M_2 sse input[M_1]=input[M_2]; output[M_1]=output[M_2] e comportamento[M_1] $\subseteq$ 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 minimize(M_1) e 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 → 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 → 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 $\in$ 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, $\forall$ p1 $\in$ 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.

1.12 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.

1.13 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.

1.14 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 $E_c \subseteq E$, $E_c \subseteq E$ 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 medesiman 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 $\overline{K}$.