Università degli Studi di Verona

Sistemi ad eventi discreti

Riassunto dei principali argomenti

Giorgia Gulino, Davide Bianchi

February 17, 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 → 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 → 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 → M_1 da M_2

- ∀ p ∈ PossibiliInitialState[M_1], ∃ q ∈ PossibiliInitialState[M_2], (p,q) ∈ S.
- ∀ p ∈ Stati[M_1], ∀ q ∈ 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) ∈ PossibiliUpdates[M_1](p,x) ⇒ ∃ q1 ∈ Stati[M_1], (q1,y) ∈ PossibiliUpdates[M_2](q,x) e (p1,q1) ∈ S. (S contiene coppie di stati iniziali e coppie consultanti l'algoritmo).
  - ∀ p ∈ Stati[M_1] ∃ q ∈ Stati[M_2] per cui ∀ I/O possibili c'è corrispondenza tra I/O uguali di p e (p,q) ∈ 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 ∈ States[det(M)] e (x,y) ∈ Input x Output
  - Q = q ∈ States[M] — ∃ p ∈ P, (q,y) ∈ PossibleUpdates[M](p,x) Se Q ≠ 0 allora
    States[det(M)]= States[det(M)] ∪ Q
    Update[det(M)](p,x)=(q,y)
    Raggruppa tutti gli stati iniziali, ∀ coppia I/O raggruppa tutti gli stati per cui quest'ultima è Possibleupdate.

1.10 Simulazione

- Se p ∈ PossibleInitialState[M_1] e PossibleInitialState[m2] = q ⇒ (p,q) ∈ S.
- Se (p,q) ∈ S e (p1,y) ∈ 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] ∈ B
- ∀ p ∈ Stati[M_1], ∀ q ∈ 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 Introduzione

Definizione 2.1.1 (Linguaggio) *Dato un insieme di simboli E, un traccia (ovvero una sequenza finita di simboli di E), definiamo linguaggio un qualsiasi sottoinsieme $L \subseteq E^*$.*

Definiamo inoltre l'insieme dei prefissi di un linguaggio come l'insieme

$$\overline{L} := \{s \in E^* : (\exists t \in E^*) st \in L\}$$

2.2 Automi

In parole povere, un automa è una struttura matematica che genera un determinato linguaggio.

Definizione 2.2.1 (Automa) *Un automa è un tupla di sei elementi*

$$G = (X, E, f, \Gamma, x_0, X_m)$$

dove:

- *X è l'insieme degli stati;*
- *E è l'insieme di eventi associati alle transizioni in G;*
- *$f : X \times E \to X$ è la **funzione di transizione**; dicendo che $f(x, e) = y$ si sta dicendo che esiste una transizione etichettata con e che porta dallo stato x allo stato y;*
- *$\Gamma : X \to 2^E$ è la **funzione degli eventi attivi**; $\Gamma(x)$ indica l'insieme degli eventi per i quali $f(x, e)$ è definita;*
- *x_0 è lo stato iniziale;*
- *$X_m \subseteq X$ è l'insieme degli **stati marcati**.*

Definizione 2.2.2 (Linguaggio generato) *Il linguaggio generato da un automa è definito come:*

$$L(G) = \{s \in E^* : f(x_0, s) \text{ è definita}\}$$

Definizione 2.2.3 (Linguaggio marcato) *Il linguaggio marcato da un automa è definito come:*

$$L_m(G) = \{s \in L(G) : f(x_0, s) \in X_m\}$$

Definizione 2.2.4 (Equivalenza tra automi) *Due automi G_1 e G_2 si dicono equivalenti se vale che*

$$L(G_1) = L(G_2) \wedge L_m(G_1) = L_m(G_2)$$

2.2.1 Composizione di automi

Proiezioni naturali. Le proiezioni naturali sono funzioni del tipo

$$P_i : (E_1 \cup E_2)^* \to E_i^*$$

definite come

$$P_i(\epsilon) = \epsilon$$
$$P_i(\sigma) = \begin{cases} \sigma & \text{se } \sigma \in E_i \\ \epsilon & \text{se } \sigma \notin E_i \end{cases}$$
$$P_i(s\sigma) = P_i(s)P_i(\sigma) \text{ per } s \in (E_1 \cup E_2)^*, \sigma \in E_1 \cup E_2$$

Delle proiezioni naturali esistono anche le inverse: come le proiezioni sono estese ai linguaggi come

$$P_i(L) = \{t \in E_i^* : \exists s \in L(P_i(s) = t)\} \text{ con } L \subseteq (E_1 \cup E_2)^*$$

anche le proiezioni inverse sono estese come

$$P_i^{-1}(L_i) = \{s \in (E_1 \cup E_2)^* : \exists t \in L_i(P_i(s) = t)\}$$

per $L_i \subseteq E_i^*$.

Prodotto di automi. Il prodotto di due automi $G_1 = (X_1, E_1, f_1, \Gamma_1, x_{01}, X_{m1})$ e $G_2 = (X_2, E_2, f_2, \Gamma_2, x_{02}, X_{m2})$ è l'automa risultato

$$G_1 \times G_2 = (X_1 \times X_2, E_1 \cap E_2, f, \Gamma_{1\times 2}, (x_{01}, x_{02}), X_{m1} \times X_{m2})$$

dove

$$f((x_1, x_2), \sigma) = \begin{cases} (f(x_1, \sigma), f_2(x_2, \sigma)) & \text{se } \sigma \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ \text{indefinita} & \text{altrimenti} \end{cases}$$
$$\Gamma_{1\times 2}(x_1, x_2) = \Gamma_1(x_1) \cap \Gamma_2(x_2)$$

Le proprietà di questa composizione sono le seguenti:

1. $L(G_1 \times G_2) = L(G_1) \cap L(G_2)$
2. $L_m(G_1 \times G_2) = L_m(G_1) \cap L_m(G_2)$

Composizione parallela di automi. La composizione parallela di due automi $G_1 = (X_1, E_1, f_1, \Gamma_1, x_{01}, X_{m1})$ e $G_2 = (X_2, E_2, f_2, \Gamma_2, x_{02}, X_{m2})$ è l'automa risultato

$$G_1 || G_2 = (X_1 \times X_2, E_1 \cup E_2, f, \Gamma_{1||2}, (x_{01}, x_{02}), X_{m1} \times X_{m2})$$

dove

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

La proiezione parallela gode delle seguenti proprietà:

1. $L(G_1 || G_2) = P_1^{-1}[L(G_1)] \cap P_2^{-1}[L(G_2)]$
2. $L_m(G_1 || G_2) = P_1^{-1}[L_m(G_1)] \cap P_2^{-1}[L_m(G_2)]$
3. $G_1 || G_2 = G_2 || G_1$

2.3 Controllabilità e osservabilità di un linguaggio

Definizione 2.3.1 (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* $\sigma \in E_{uc}$ *vale*

$$s\sigma \in M \Rightarrow s\sigma \in \overline{K}$$[1]

2.4 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:

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

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 $\overline{K}$.

2.5 Proprietà di Controllabilità

Esistono due tipi di linguaggi derivati da K:

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

Abbiamo i seguenti rapporti:

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

- 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:

[1] Equivalente a $\overline{K} E_{uc} \cap M \subseteq \overline{K}$.

2.6 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).

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

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