

Confronto tra sistemi

In varie occasioni, dato l'insieme dei comportamenti di due sistemi, possiamo essere interessati a confrontarli tra loro (ad esempio per comprendere quale dei due soddisfa maggiormente ad alcune specifiche). Nel caso di DEDS come quelli che stiamo trattando, ciò equivale a confrontare tra loro due linguaggi L ed L' , costruiti su un alfabeto comune E . Se si dispone di due automi $G = (X, E, f, \Gamma, x_0, X_m)$ e $G' = (X', E, f', \Gamma', x'_0, X'_m)$ che generano L ed L' , il confronto può essere fatto utilizzando questi ultimi.

Spesso, ciò che interessa paragonare nel confronto sono insiemi di stringhe aventi il medesimo prefisso. In particolare, data una stringa t in $L \cap L'$, ad esempio, interessa sapere quali delle stringhe te (con $e \in E$) appartenenti ad L appartengono anche ad L' e viceversa.

Per poter fare questo è conveniente identificare lo stato $x = f(x_0, t) \in G$ con lo stato $x' = f'(x'_0, t) \in G'$, allo scopo di confrontare agevolmente $\Gamma(x)$ con $\Gamma'(x')$.

Da un punto di vista generale, occorre sviluppare una procedura che consenta di mappare l'insieme degli stati di G sopra l'insieme degli stati di G' .

Confronto tra sistemi

Prendiamo in considerazione il caso in cui $L \subseteq L'$. In generale, non è detto che i grafi soggiacenti a G e G' abbiano una struttura simile o compatibile. Può accadere che in G si abbia $f(x_0, t) = f(x_0, s) = x$, mentre in G' $f'(x'_0, t) = x' \neq f'(x'_0, s) = x''$, e quindi non sia possibile mappare x né su x' né su x'' .

Per ovviare a questo tipo di situazioni si può procedere a modificare G , senza alterare il linguaggio L da esso generato, in modo da rendere la struttura del grafo soggiacente compatibile con quella del grafo soggiacente a G' .

Questa operazione prende il nome di **raffinemento dello stato** di G e può essere attuata ricorrendo all'operazione di prodotto.

Raffinamento tramite prodotto

Consideriamo

$$L = \Lambda(G) \subseteq L' = \Lambda(G')$$

e proponiamoci di effettuare un confronto tra i linguaggi mappando G , o un grafo equivalente ottenuto per raffinamento, su G' . A tale scopo costruiamo

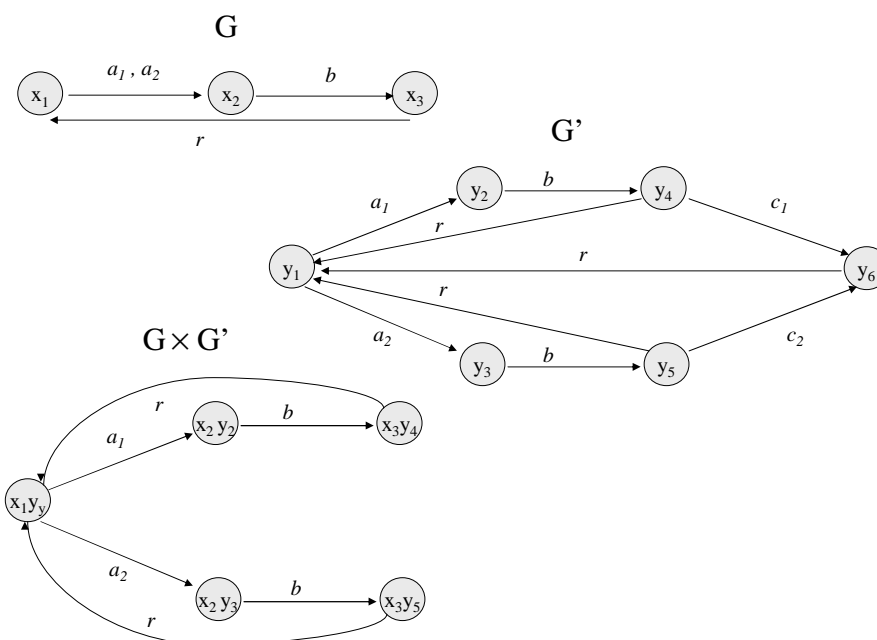
$$G_{\text{new}} = G \times G'$$

Gli stati di G_{new} sono coppie (x, x') , $x \in X$ ed $x' \in X'$.

G_{new} è equivalente a G , cioè genera lo stesso linguaggio L .

Il desiderato mapping del grafo soggiacente a G_{new} su quello soggiacente a G' consiste nell'associare a (x, x') la seconda componente x' .

Esempio



Linguaggi regolari

Quanto precede mostra l'utilità di rappresentare i linguaggi tramite automi, purchè gli automi stessi siano facilmente maneggiabili. E' questo il caso degli automi aventi uno spazio degli stati finito, o automi a stati finiti, che possono, in particolare, essere trattati mediante calcolatori a memoria finita.

Il problema che ci si pone, dunque, è quello di determinare quali linguaggi (eventualmente composti da infinite parole) si possano rappresentare tramite automi a stati finiti.

Definizione

Dato un alfabeto E , diremo linguaggio regolare ogni linguaggio L costruito su E ($L \subseteq E^*$) che possa essere rappresentato da un automa a stati finiti.

Nota

L'insieme \mathfrak{R} dei linguaggi regolari su E è un sottoinsieme proprio di 2^{E^*} .

Linguaggi regolari

Se L ed L' sono linguaggi regolari, allora sono linguaggi regolari anche

- \bar{L}
- $\overline{L^*}$
- $C(L) = E^* \setminus L$
- LL'
- $L \cup L'$
- $L \cap L'$

Linguaggi regolari

Esercizio

Si dimostri quanto asserito nella precedente trasparenza.

(Suggerimenti. Si rifletta su cosa occorre dimostrare e, successivamente, detti G e G' due automi che generino L ed L' ,

- Si consideri $\text{Trim}(G)$ e si marchino tutti i suoi stati
- Si aggiunga un nuovo stato iniziale a G , lo si marchi e lo si connetta al vecchio stato iniziale con ϵ . Poi si aggiunga una transizione ϵ da ciascuno stato marcato di G al vecchio stato iniziale
- Si consideri l'operazione di complemento
- Si crei un nuovo stato iniziale e lo si connetta con due ϵ agli stati iniziali di G e G' ,
- Si connettano gli stati di G allo stato iniziale di G' mediante ϵ e si tolga la marcatura a tutti gli stati di G
- Si consideri il prodotto $G \times G'$

Linguaggi regolari

I linguaggi regolari possono essere descritti tramite espressioni dette anch'esse regolari.

Nelle espressioni regolari si usa

- il segno $+$ anziché il segno \cup ad indicare l'aggregazione di due o più insiemi di stringhe a formarne un altro: $A+B = A \cup B$
- il segno $*$ ad indicare la chiusura di Kleene di un insieme A (es. $A = \{a\} \Rightarrow A^* = \{\epsilon, a, aa, aaa, \dots\}$)
- la giustapposizione ad indicare l'insieme di stringhe ottenute giustappoendo le stringhe di un insieme a quelle di un altro: $AB = \{ab, a \in A, b \in B\}$.

Linguaggi regolari

Stabiliamo che:

- le espressioni Φ , $\{\epsilon\}$, $\{e\}$ per ogni $e \in E$ sono regolari
- se A e B consistono di espressioni regolari, anche A^* , B^* , AB , $A+B$ consistono di espressioni regolari
- non ci sono altre espressioni regolari oltre quelle ottenibili a partire dalle costruzioni implicite nelle due regole precedenti.

Esempio

$E=\{a,b,g\}$

$(a+b)g^*$ indica il linguaggio $L=\{a,b,ag, bg, agg, bgg, aggg, bggg, \dots\}$

$(ab)^*+g$ indica il linguaggio $L=\{\epsilon, g, ab, abab, ababab, \dots\}$

Teorema

Ogni linguaggio che può essere scritto con espressioni regolari è un linguaggio regolare e viceversa

Riconoscitore canonico

Osserviamo che dato un linguaggio L possono esistere più automi diversi che lo marcano (come si fa a costruire un esempio?). Possiamo classificare tali automi in base alla cardinalità del loro spazio degli stati e fissare l'interesse su quelli per i quali si ha cardinalità minima.

Definizione

Un automa che marchi il linguaggio L e abbia insieme degli stadi di cardinalità minima tra tutti gli automi che marcano L è detto riconoscitore canonico di L.

Proposizione

Se $L=E^*$, allora il riconoscitore canonico di L è unico.

(Esercizio: si dimostri quanto detto sopra nel caso $E = \{a,b\}$)

Riconoscitore canonico

L'interesse nel considerare il riconoscitore canonico è dovuto al fatto che esso rappresenta il modo più economico per descrivere un dato linguaggio e, quindi per studiarne le proprietà.

Dato un automa a stati finiti $G = (X, E, f, \Gamma, x_0, X_m)$, una prima misura della memoria occorrente per rappresentarlo in un calcolatore è data da $\text{card}(X) \times \text{card}(E)$.

(Esercizio: si valuti l'accuratezza della misura descritta sopra)

Costruzione del riconoscitore canonico

Per costruire il riconoscitore canonico di un dato linguaggio L si opera in genere come segue:

- 1) si determina un automa G che marchi L ;
 - 2) si modifica G , se necessario, riducendo il numero degli stati fino ad ottenere la cardinalità minima, senza modificare il linguaggio marcato.
- Si noti che al passo 1) non ci si preoccupa di limitare il numero degli stati. Il passo 2) consiste essenzialmente in una operazione, detta di aggregazione, che raggruppa più stati in un unico stato. L'aggregazione si basa sulla seguente definizione:

Definizione

Due stati x e x' di un automa G sono detti equivalenti se $\Lambda_m(G(x)) = \Lambda_m(G(x'))$, cioè se l'insieme dei percorsi marcati ottenibili a partire da x e quello dei percorsi marcati ottenibili a partire da x' sono uguali.

L'operazione di aggregazione consiste nell'individuare tutti gli stati equivalenti tra loro e sostituirli con un unico stato, detto stato aggregato.

(Esercizio: si descriva come compiere il passo 1))

Costruzione del riconoscitore canonico

Dati due stati x e y di G , osserviamo che

- se $x \in X_m$ e $y \notin X_m$, allora x non è equivalente a y ;
- se entrambi appartengono o entrambi non appartengono a X_m e
 - $f(x, e) = f(y, e)$ per ogni $e \in E \cup \{\epsilon\}$,

allora x ed y sono equivalenti;

- se entrambi appartengono o entrambi non appartengono a X_m e
 - $f(x, e) = f(y, e)$ per ogni $e \in E' \subseteq E \cup \{\epsilon\}$
 - $f(x, e) = y$ e $f(y, e) = x$ per ogni $e \in E \cup \{\epsilon\} \setminus E'$

allora x ed y sono equivalenti.

Più in generale, se X' è un insieme di stati tali che $X' \subseteq X_m$ oppure $X' \cap X_m = \emptyset$, si ha che R è fatto di stati equivalenti se

- $f(x, e) = z \notin R$ per $x \in R$ implica $f(y, e) = z$ per ogni $y \in R$.

(Esercizio: si dimostri quanto affermato sopra)

Esempio

➤ Consideriamo una **macchina** che **legge cifre** dall'insieme $\{1, 2, 3\}$ e **marca** qualunque **stringa** che finisca con la **sottostringa 123**.

➤ $E = \{1, 2, 3\}$; evento: "la macchina legge n ", $n = 1, 2, 3$.

➤ L'**automa** dovrà generare il linguaggio E^* e marcare

$$L_m = \{st \in E^* : s \in E^* \text{ e } t = 123\}$$

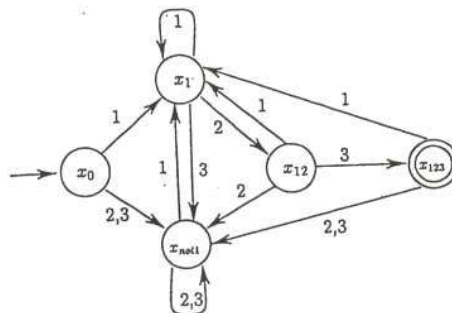
➤ Per determinare X e costruire f , cominciamo fissando x_0 , che indica la condizione di partenza, prima della quale non è stata letta alcuna cifra.

➤ Poiché siamo interessati alla stringa 123, **definiamo**

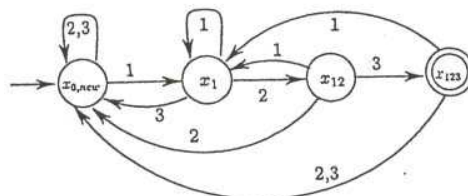
- X_1 la prima cifra letta è "1"
- X_{non1} la prima cifra letta è 2 o 3

- Occorre che l'automa memorizzi che il suffisso della stringa letta è 1, 12 oppure 123, oppure nessuno di questi
- Allora x_1 è esteso a rappresentare che l'evento più recente è 1 (suffisso 1)
 - X_{12} suffisso 12
 - X_{123} suffisso 123, da marcare.
- Qualunque suffisso diverso non deve essere memorizzato
- Allora eventi che risultino in suffissi che non sono 1, 12, 123 riconducono allo stato x_{non1} , che indica che 1, 12, 123 non sono suffissi della stringa letta finora.

Al passo 1) si ha : $X = \{x_0, x_1, x_{non1}, x_{12}, x_{123}\}$.



Se operiamo aggregando gli stati di $R = \{x_0, x_{non1}\}$, si ha:



Procedura per ridurre il numero degli stati

1) flag (x, y) per tutti gli $x \in X_m, y \notin X_m$

1+i) per ogni coppia (x, y) non flagged al passo i), se $(f(x, e), f(y, e))$ è flagged per qualche $e \in E$, flag (x, y) e itera su i finchè si possono aggiungere flags

Le coppie senza flag rimanenti al termine della procedura sono costituite da stati equivalenti.

Si procede a raggruppare tali coppie, mettendo in un unico insieme coppie che hanno un elemento in comune. Si ottiene in tal modo una suddivisione in insiemi disgiunti di stati equivalenti.

Infine, si sostituisce un singolo stato ad ogni insieme di stati equivalenti.

AUTOMI INGRESSO-USCITA

La definizione di automa è:

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

con $\Gamma : X \rightarrow 2^E$ funzione eventi attivi

Esistono due varianti a tale definizione

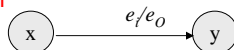
➤ **Automa di MOORE:** automa con funzione di uscita dallo stato

esiste una funzione di uscita che assegna una uscita ad ogni stato. L'uscita è emessa dall'automata quando essa entra nello stato a seguito di un evento interpretabile come ingresso

➤ **Automa di MEALY:** automa ingresso/uscita

esiste una funzione di uscita che assegna una uscita ad ogni arco. L'uscita è emessa dall'automata quando esegue la transizione di stato indicata. Le transizioni sono marcate da: **evento IN / evento OUT**

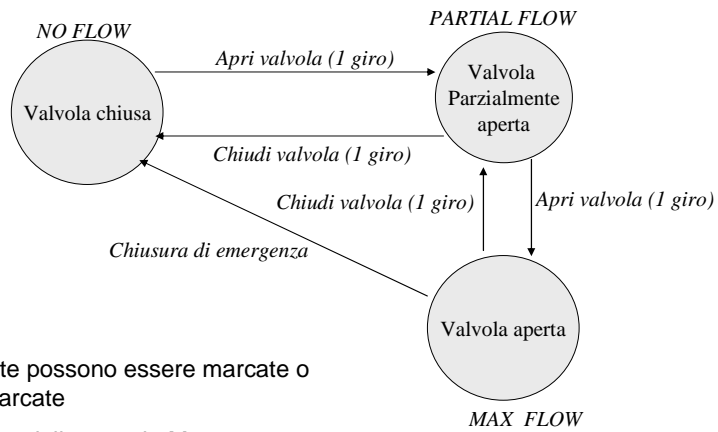
Esempio:



quando l'automata è in x e riceve l'evento di ingresso e_i , opera una transizione ad y e in questo processo emette l'uscita e_o

Esempio: Automa di MOORE

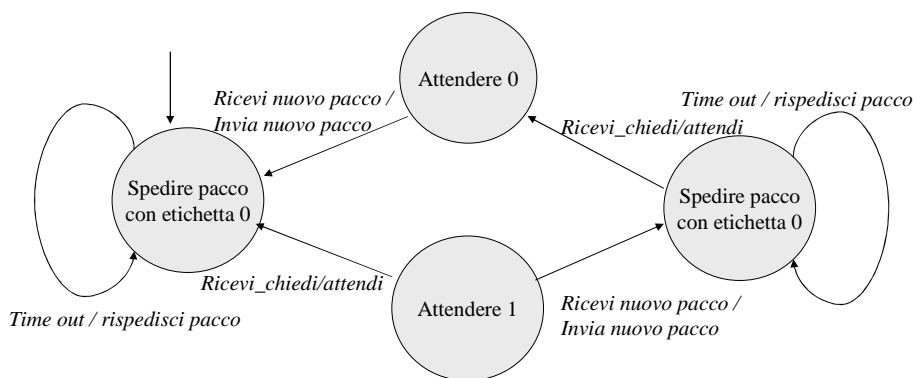
➤ Automa di MOORE



- le uscite possono essere marcate o non marcate
- le uscite dello stato in Moore possono essere viste come una generalizzazione della marcatura

Esempio: Automa di MEALY

➤ Automa di MEALY

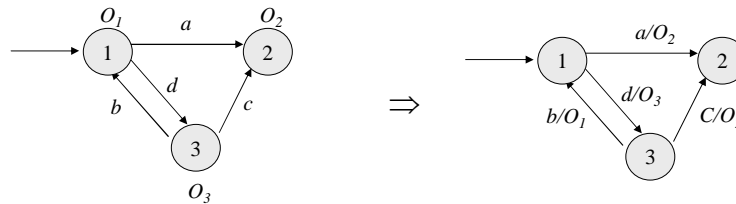


- Le nozioni di **uscita associata allo stato** o **alla transizione** possono essere utili nel costruire modelli di DES.
- I sistemi composti di componenti elettromeccanici integrati (p.e. assemblaggio, motori, sistemi di controllo di processi, ...) sono in genere equipaggiati con un insieme di sensori che misurano lo stato fisico del sistema.
- **Moore** è adatto a fornire rappresentazioni quando **l'uscita associata allo stato corrisponde alle letture dei sensori**.
- **Mealy** risulta adatto a fornire rappresentazioni di protocolli di comunicazione o di sistemi SW in generale.

- Si possono sempre trasformare automi di Mealy e di Moore in automi standard:
 - ✓ **Mealy** → standard
 $E \equiv \{\text{etichette I/O}\}$
 quindi il linguaggio generato dall'automa è l'insieme delle stringhe i/o che possono essere generate dall'automa di Mealy
 - ✓ **Moore** → standard
 L'uscita di stato può essere vista come l'evento di uscita associato a tutti gli eventi che portano in quello stato

Esempio

Trasformazione di un automa di Mealy in automa standard



Analisi e sintesi DEDS

La rappresentazione di un DEDS mediante un automa a stati finiti ci consente di

- analizzare le proprietà di funzionamento del DEDS (**analisi**),
- definire una politica di controllo che ci consenta di far funzionare il DEDS rispettando determinate specifiche (**sintesi**).

In generale, lo svolgimento delle procedure di analisi da parte di un calcolatore che abbia memorizzato un modello del DEDS sotto forma di automa a stati finiti, nell'ipotesi frequente che $\text{card}E$ sia molto minore di $\text{card}X$, ha una complessità computazionale proporzionale alla $\text{card}X$ o a $(\text{card}X)^2$.

In genere, nei problemi di analisi, si suppone che tutti gli eventi siano osservabili per chi compie l'analisi. In alcune situazioni, tuttavia, può accadere che il verificarsi di particolari eventi non possa essere rilevato, si parla in tal caso di eventi non osservabili e, nella problematica di analisi, rientra anche quella di dedurre informazione su quanto non è osservabile per mezzo di quanto lo è.

Analisi e sintesi DEDS

Problemi di analisi:

- Proprietà e problemi di sicurezza
- Proprietà e problemi di blocco
- Problemi di diagnostica e/o di stima dello stato

Proprietà e problemi di sicurezza

I cosiddetti problemi di sicurezza relativi ad un dato DEDS riguardano la possibilità del verificarsi di singoli eventi o di comportamenti indesiderati.

Trasferendo questo al livello di automa che rappresenta il DEDS il problema diviene quello di verificare la presenza di stringhe o sottostringhe indesiderate nel linguaggio generato.

In alternativa, si può considerare il problema dell'inclusione nel linguaggio generato di un linguaggio (detto legale o ammissibile), che non contenga stringhe o sottostringhe indesiderate.

Estendendo tali problematiche, si può considerare il problema di pervenire ad uno stato indesiderato.

Proprietà e problemi di sicurezza

I problemi di sicurezza vengono affrontati in genere mediante procedure di calcolo dirette.

- La presenza di stringhe indesiderate può venire studiata analizzando l'insieme delle stringhe che si possono comporre a partire da ciascuno stato accessibile.
- La relazione $L \subseteq \Lambda(G)$ può essere studiata sfruttando la sua equivalenza con la relazione $L \cap \Lambda(C(G)) = \Phi$ (si ricordi la costruzione di un automa che genera $C(G)$ e quella del prodotto ...).
- La raggiungibilità di un dato stato x a partire da x' si studia costruendo la parte accessibile dopo aver fissato x come stato iniziale.

Proprietà e problemi di blocco

Problemi e proprietà di blocco sono legati al verificarsi della condizione $\Lambda_m(G) \neq \Lambda(G)$: se si ha un blocco, allora $\Lambda_m(G) \neq \Lambda(G)$.

Se, in un automa bloccante, occorre determinare tutti gli stati di blocco o deadlocks e tutti i livelocks, si può partire col determinare la parte CoAc e poi esaminare l'insieme degli eventi possibili a partire da ciascuno degli stati non CoAc. In questo modo, se ce ne sono, si trovano i deadlocks.

Per quanto riguarda i livelock, essi vanno ricercati, utilizzando opportune procedure di calcolo, tra i gruppi di stati connessi che sono fuori da CoAc.

(Esercizio: Si dimostri quanto affermato).

Diagnostica e stima dello stato

I problemi di diagnostica e stima dello stato si pongono quando non tutti gli eventi si verificano in modo osservabile per chi compie l'analisi.

Non tratteremo qui in dettaglio tali problemi, limitandoci a segnalare che essi possono venire affrontati limitandosi a considerare un sottoautoma che non presenti eventi inosservabili. I comportamenti descritti mediante il sottoautoma permettono di avere una stima dei comportamenti del DEDS rappresentato dall'automa di partenza.

Analisi e sintesi DEDS

Problemi di sintesi:

- Definire una politica di controllo (insieme di regole) che applicata ad un dato DEDS ne modifichi in maniera desiderata il comportamento
- Definire una procedura di implementazione della politica di controllo (controllore)

Problemi di sintesi

Si consideri un DEDS Σ rappresentato da un automa $G = (X, E, f, \Gamma, x_0, X_m)$ e si supponga che l'insieme dei comportamenti di Σ contenga degli elementi indesiderati, o, in altri termini, che $\Lambda(G)$ contenga delle stringhe indesiderate (può trattarsi, ad esempio di comportamenti o stringhe che violano certe condizioni di funzionamento, come "esegui l'operazione B solo dopo aver eseguito l'operazione A", o di comportamenti che portano a situazioni, cioè a stati, di blocco o a condizioni di livelock).

Il problema che ci poniamo è quello di modificare l'insieme dei comportamenti di Σ allo scopo di evitare quelli indesiderati.

Il paradigma di controllo, che descrive la nostra possibilità di intervenire su Σ , prevede che

- si conosca lo stato attuale x di Σ
- sia possibile inibire tutti o parte degli eventi appartenenti a $\Gamma(x)$

Problemi di sintesi

La politica di controllo consisterà nell'inibire specifici eventi in modo tale che non si verifichino comportamenti indesiderati.

L'implementazione di tale politica avviene

- valutando la condizione attuale del sistema, cioè lo stato x in cui esso si trova,

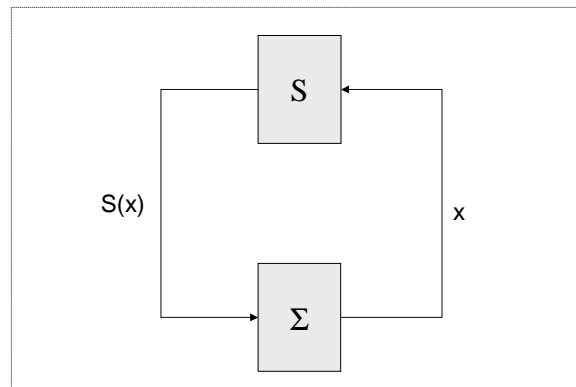
questo ci permetterà di individuare l'insieme $\Gamma(x)$, al quale appartengono gli eventi da inibire

- inibendo (oppure: abilitando solo) specifici eventi di $\Gamma(x)$.

Poiché la scelta di quali eventi inibire si basa sulla conoscenza dello stato attuale, parleremo di implementazione della politica di controllo in **retroazione dallo stato**. La modalità di implementazione descrive la struttura del controllore, che indicheremo con S . $S(x)$ indicherà l'azione di controllo (quali eventi inibire) nella situazione (stato) x .

Problemi di sintesi

Schema del controllo in controeazione dallo stato



Problemi di sintesi

In termini più generali (e formali) un problema di sintesi può essere formulato prendendo in considerazione, per un sistema Σ rappresentato dall'automa $G = (X, E, f, \Gamma, x_0, X_m)$, opportuni sottolinguaggi di $\Lambda(G)$, che descrivono il comportamento **ammissibile** o **legale** del sistema controllato.

In particolare, ci si può ricondurre a situazioni del tipo

$$\Lambda_r \subset \Lambda_a \subseteq \Lambda(G)$$

dove, essendo $\Lambda(G)$ una rappresentazione del comportamento del sistema senza controllo,

Λ_r rappresenta il comportamento (minimo) desiderato, da ottenersi mediante l'azione di controllo,

Λ_a rappresenta il comportamento ammissibile (massimo) che si è disposti ad accettare

Definizione della politica di controllo

Dato il sistema Σ rappresentato dall'automa $G = (X, E, f, \Gamma, x_0, X_m)$, eventuali eventi di E che non siano disabilitabili mediante politiche di controllo verranno detti eventi non controllabili e si porrà $E = E_C \cup E_{NC}$ (la presenza di eventi non disabilitabili può essere dovuta all'esistenza di eventi intrinsecamente non prevedibili, come i guasti, o a limiti della capacità di intervento sul sistema, come limiti dovuti agli attuatori, o a scelte di modellazione).

Definizione

Una politica di controllo è una funzione $S: X \rightarrow 2^E$ tale che $S(x) \subseteq \Gamma(x)$ (in pratica, $S(x)$ rappresenta l'insieme degli eventi abilitati quando il sistema è nello stato x).

Una politica di controllo S è detta ammissibile se $E_{NC} \cap \Gamma(x) \subseteq S(x)$ per ogni $x \in X$.

Effetti della politica di controllo

L'attuazione della politica di controllo S su G dà luogo ad un sistema indicato con S/G , che viene detto controllato in controeazione dallo stato o in catena chiusa. Il comportamento di S/G viene indicato con $\Lambda(S/G)$, cioè come linguaggio generato da S/G .

In pratica, $\Lambda(S/G)$ è costituito dall'insieme delle stringhe di $\Lambda(G)$ (cioè $\Lambda(S/G) \subseteq \Lambda(G)$) che si possono formare mediante gli eventi abilitati da S . Analogamente, $\Lambda_m(S/G) = \Lambda(S/G) \cap \Lambda_m(G)$.

Da quanto abbiamo premesso, segue che, dato un sistema Σ rappresentato dall'automa $G = (X, E, f, \Gamma, x_0, X_m)$ e indicato con Λ_r ($\subseteq \Lambda_a \subseteq \Lambda(G)$) un comportamento desiderato, il problema di sintesi che ci si pone è innanzitutto quello di definire una politica di controllo S tale che $\Lambda_r \subseteq \Lambda(S/G) \subseteq \Lambda_a \subseteq \Lambda(G)$, o anche $\Lambda_r \subseteq \Lambda_m(S/G) \subseteq \Lambda_a \subseteq \Lambda(G)$. Inoltre, occorrerà determinare il modo per implementare S .

Note

Possiamo definire $\Lambda(S/G)$ ricorsivamente nel modo seguente:

1) $\varepsilon \in \Lambda(S/G)$

2) $[(s \in \Lambda(S/G)) \text{ e } (s\sigma \in \Lambda(G)) \text{ e } (\sigma \in S(f(s, x_0)))] \Leftrightarrow [s\sigma \in \Lambda(S/G)]$

Osserviamo che:

▪ $\Lambda(S/G) \subseteq \Lambda(G)$ è per definizione chiuso rispetto al prefisso

▪ $\Lambda_m(S/G) \subseteq \overline{\Lambda_m(S/G)} \subseteq \Lambda(S/G) \subseteq \Lambda(G)$

▪ ε (la stringa vuota) è contenuta in $\Lambda(S/G)$ poichè è contenuta in $\Lambda(G)$

Note

Possiamo avere un comportamento $\Lambda(S/G)$ che prevede situazioni di blocco. Ciò si verifica se

$$\Lambda(S/G) \neq \overline{\Lambda_m(S/G)}$$

In tal caso diremo che la politica di controllo o di supervisione S è di blocco. Ricordiamo che, poiché le stringhe marcate rappresentano compiti ultimati o la registrazione del completamento di una qualche operazione (a seconda di come si è deciso di modellare), una S di blocco dà luogo ad un sistema controllato che può non terminare l'esecuzione del compito.

Esempio

➤ $\Lambda(G) = \overline{\{abc\}}$, $\Lambda_m(G) = \{ab\}$

- **G** è **blocking** perché la stringa $abc \in \Lambda(G)$ non è prefisso di ab
- Se aggiungiamo a G

$$S : S\{\varepsilon\} = \{a\}, S(a) = \{b\}, S(ab) = \emptyset$$

allora S/G è **non blocking** perché

$$\Lambda(S/G) = \overline{\Lambda_m(S/G)} = \overline{\{ab\}}$$

➤ Se però l'evento **c** fosse **incontrollabile non** potremmo **eliminare** le stringhe bloccate **tramite il controllo**

➤ **P.E.**

- Se prendiamo $S\{\varepsilon\} = \{a\}$, $S(a) = \{b\}$,
 - ✓ allora per l'ammissibilità $S(a, b) = \{c\}$ e quindi S/G coincide con G
- Se si prendesse $S(a) = \emptyset$,
 - ✓ S/G si bloccherebbe dopo la stringa a
- Se $S\{\varepsilon\} = \emptyset$,
 - ✓ S/G si bloccherebbe dopo la stringa ε

Teorema di controllabilità

Teorema

Dato un sistema Σ rappresentato da un automa $G = (X, E, f, \Gamma, x_0, X_m)$, un linguaggio $K \subseteq \Lambda(G)$ ed un insieme di eventi incontrollabili E_{uc} , se K verifica la seguente condizione

per ogni $s \in K$ ed ogni $e \in E$, se $e \in K$ implica $s'e \in K$ per ogni $s' \in K$ con $f(x_0, s) = f(x_0, s')$ (condizione di completezza)

esiste una politica di controllo S tale che $\Lambda(S/G) = \overline{K}$, se e solo se $\overline{K} E_{uc} \cap \Lambda(G) \subseteq \overline{K}$.

La condizione $\overline{K} E_{uc} \cap \Lambda(G) \subseteq \overline{K}$ è detta condizione di controllabilità di K rispetto a $\Lambda(G)$. Essa è equivalente ad affermare che, per ogni $s \in K$ e per ogni $e \in E_{uc}$, se $e \in \Lambda(G)$ implica $se \in K$.

Test di controllabilità

Per verificare se K è controllabile rispetto a $\Lambda(G)$, si può operare nel seguente modo.

- 1) Si costruisce un automa H tale che $\Lambda(H) = \overline{K}$.
- 2) Si costruisce il prodotto $H \times G$.
- 3) Per ogni stato $(h, g) \in H \times G$ si confronta $E_{uc} \cap \Gamma(h, g)$ con $E_{uc} \cap \Gamma(g)$.
- 4) Se esiste un evento e tale che $e \in E_{uc} \cap \Gamma(g)$ ed $e \notin E_{uc} \cap \Gamma(h, g)$, allora K non è controllabile.

Implementazione di S

Dato un sistema Σ rappresentato dall'automa $G = (X, E, f, \Gamma, x_0, X_m)$, un linguaggio $K \subseteq \Lambda(G)$ che verifichi la condizione di completezza ed un insieme di eventi incontrollabili E_{uc} , assumiamo che K sia controllabile (escludiamo i casi non interessanti $K = \Lambda(G)$ e $K = \Phi$).

La politica di controllo S definita da

$$S(x) = [E_{uc} \cap \Gamma(x)] \cup \{\sigma \in E_c \text{ tali che, per ogni } s \in K \text{ con } f(x_0, s) = x, \text{ si ha } s\sigma \in K\}$$

è tale che $\Lambda(S/G) = \overline{K}$ (lo si verifichi).

Per implementare S si considera un automa trim $R = (Y, E, g, \Gamma_R, y_0, Y_m)$ che marca il linguaggio \overline{K} , tale cioè che $\Lambda_m(R) = \Lambda(R) = \overline{K}$.

Diremo che R è una realizzazione di S .

L'automa prodotto $R \times G$ descrive il comportamento di Σ controllato in retroazione mediante S .

Implementazione di S

Quanto affermato segue dal fatto che

$$\Lambda(R \times G) = \Lambda(R) \cap \Lambda(G) = \overline{K} \cap \Lambda(G)$$

$$\Lambda_m(R \times G) = \Lambda_m(R) \cap \Lambda_m(G) = K \cap \Lambda_m(G) = \Lambda(S/G) \cap \Lambda_m(G) = \Lambda_m(S/G)$$

Osservando che, per ogni $s \in K$, è possibile identificare $g(y_0, s)$ con $f(x_0, s)$, possiamo dire che

$$S(x) = S(f(x_0, s)) = [E_{uc} \cap \Gamma(f(x_0, s))] \cup \{\sigma \in E_c : s\sigma \in K\} = \Gamma_R(g(y_0, s)) = \Gamma_{R \times G}(g \times f((y_0, x_0), s))$$

dove la prima eguaglianza segue dalla controllabilità di K e la seconda eguaglianza segue dall'essere $K \subseteq \Lambda(G)$.

L'automa R , in pratica, ci consente di memorizzare in maniera appropriata la politica di controllo S .

Implementazione di S

L'implementazione effettiva di S attraverso il prodotto $R \times G$ avviene come segue:

1) se G è nello stato $f(x_0, s)$ ed R nello stato corrispondente $f(y_0, s)$, l'evento $e \in \Gamma(f(x_0, s))$ è abilitato solo se esso è presente anche in $\Gamma(f(y_0, s))$, cioè solo se $e \in \Gamma(f(y_0, s))$.

2) l'accadimenti dell'evento e, in tal caso, porta negli stati $f(x_0, se)$ ed $f(y_0, se)$ e il processo si ripete.

Un tale modo di operare prende, a volte, il nome di table lookup

Note

In genere, la modifica del comportamento di un sistema tramite un controllore o supervisore è motivata dall'esigenza di rispettare certe specifiche.

Di solito, tali specifiche sono espresse in linguaggio naturale, come, per esempio, le seguenti:

- evitare alcuni stati di G;
- eseguire un insieme di eventi in un dato ordine di priorità,
- non permettere che un evento accada più di n volte fra due accadimenti di un altro evento,
- Permettere l'accadimento di un evento solo a seguito di un determinato altro;
- eccetera.

Note

In pratica, il percorso che occorre compiere per giungere ad ottenere un comportamento desiderato è in sintesi il seguente:

dati il sistema Σ rappresentato da $G = (X, E, f, \Gamma, x_0, X_m)$, un insieme di eventi incontrollabili E_{uc} e un insieme di specifiche s_1, \dots, s_n espresse in termini linguistici,

1) si costruisce, se possibile, un sottolinguaggio $K \subseteq \Lambda(G)$ le cui parole rispettano le specifiche (in pratica, si traducono le specifiche espresse in termini linguistici in un modello di linguaggio) e che verifica la condizione di completezza, ;

2) si verifica la controllabilità di K ;

3) si costruisce la politica di controllo S tale che $L(S/G) = K$;

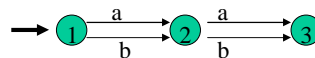
4) si implementa S mediante una sua realizzazione R .

La costruzione di K è in genere molto complessa e richiede esperienza. Può anche accadere che non esista un sottolinguaggio $K \subseteq \Lambda(G)$ le cui parole rispettano le specifiche. In tal caso, occorre modificare in maniera opportuna i dati Σ e G .

Note

Una delle difficoltà più frequenti nella costruzione di K riguarda la necessità di soddisfare specifiche che richiedono di ricordare come si è giunti ad un determinato stato.

Per esempio si consideri



con la specifica aa e ba ammissibili, ab non ammissibile.

Per poter trovare K , in questo caso, è necessario modificare G suddividendo lo stato 2 in due stati distinti, così da aggiungere memoria al sistema

