1. Supervisione dei sistemi ad eventi discreti

1.1 Introduzione

Come visto nei capitoli precedenti lo stato di un sistema ad eventi discreti cambia al verificarsi di un evento. Per imporre quindi un determinato comportamento al sistema in esame è necessario poter influenzare in qualche modo la sequenza con cui gli eventi si susseguono. Ovviamente per cambiare in maniera automatica il comportamento di un sistema a eventi discreti, cosi come nel caso dei sistemi a stato vettore, è conveniente ricorrere ad uno schema di controllo in retroazione.

In seguito si assumerà come riferimento lo schema di controllo illustrato in Fig. 1, in cui il sistema da controllare è rappresentato da un sistema ad eventi discreti *G*, mentre il controllore è costituito da un sistema *S*, il quale osserva gli eventi che l'automa *G* esegue. Assumendo che *S* sia in grado di disabilitare alcuni degli eventi in modo che essi non occorrano, il meccanismo di controllo consiste nel disabilitare gli eventi che portano a transizioni indesiderate tra gli stati di *G*. Come vedremo il controllore *S* (che nel caso del controllo d sistemi ad eventi discreti prende anche il nome d supervisore) è a sua volta un sistema ad eventi discreti rappresentabile attraverso un automa.

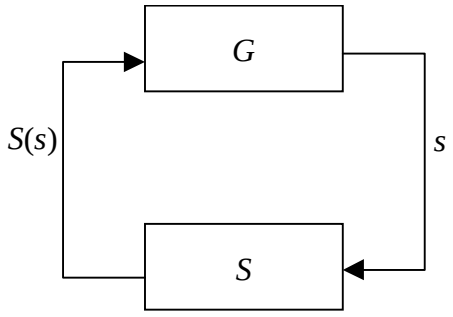

Fig. 1. Il ciclo di retroazione del controllo di supervisione, *G* rappresenta il sistema controllato, ed *S* il supervisore.

1.2 Un esempio di supervisione

Prima di iniziare formalmente a studiare il problema di supervisione dei sistemi ad eventi discreti, vediamo come in un caso semplice sia possibile arrivare alla definizione di un supervisore, direttamente dalle specifiche di comportamento che si vogliono assegnare al sistema da controllare.

A tal scopo si consideri un elaboratore in grado di eseguire un solo task per volta, e si assuma che gli utenti attraverso un terminale d'ingresso possano caricare il programma ed i dati che costituiscono tale task. Per evitare agli utenti che devono sottomettere un nuovo task l'attesa della fine di quello in corso, è possibile far si che il sistema operativo dell'elaboratore sia in grado di gestire una coda dei task via via sottoposti. Per questo motivo, il sistema operativo contiene un programma, detto schedulatore, il quale prima provvede a inserire i task in una coda, e poi al loro turno li sottopone all'unità centrale di elaborazione (questa modalità di elaborazione viene chiamata a lotti).

Il sistema descritto può essere modellato come una coda in cui il processo d'arrivo coincide con l'immissione da parte di un utente di un nuovo task, ed il processo di partenza con il completamento da parte dell'unità centrale di un task. Se si assume che tale coda abbia una lunghezza massima pari a 3, un automa che la descrive è mostrato in Fig. 2. Gli eventi A e P denotano rispettivamente l'arrivo e la partenza di un task.

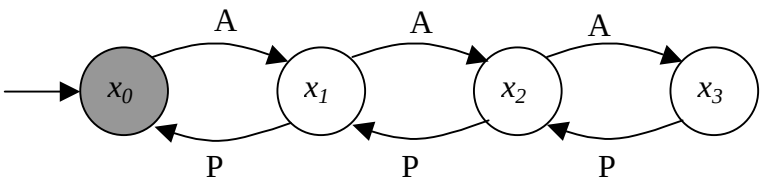

Fig. 2. Automa che descrive la coda gestita dallo schedulatore dell'elaboratore.

Si supponga ora di aggiungere altri due eventi al sistema, in particolare un evento S (stop) che segnala la volontà di porre termine all'elaborazione, ed un evento R (resume) che segnala la possibilità di riprendere l'elaborazione. Questi due eventi possono essere ad esempio generati da un amministratore del sistema attraverso un terminale di controllo. L'automa di Fig. 2 può allora essere modificato nell'automa di Fig. 3, dove sono stati aggiunti a ciascuno stato degli auto-anelli per tenere conto della possibile occorrenza dei nuovi eventi.

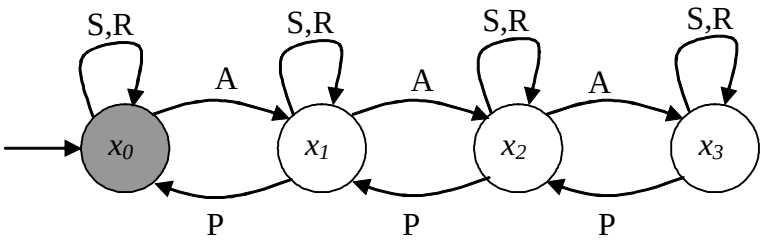

Fig. 3. Automa modificato in modo da includere gli eventi S ed R.

Osservando l'automa di Fig. 3 si vede che, allo stato attuale, gli eventi S ed R sono ignorati. Si vuole allora progettare un supervisore *S* che modifichi il comportamento di tale sistema, in modo che esso risponda in modo appropriato a questi due eventi.

Per fare questo è necessario specificare in dettaglio che si intende per risposta appropriata da parte del sistema; in questo caso assumeremo che all'atto dell'occorrenza di un segnale di stop (evento S) il sistema debba continuare l'elaborazione fino al termine di tutti i task in coda, non accettando più alcun task dall'esterno fino al successivo segnale di resume (evento R). Da una analisi di questa specifica si evince che per imporre un corretto funzionamento al sistema è necessario tenere memoria dell'ultimo evento S o R occorso. Questo può essere ottenuto scomponendo ciascun stato dell'automa di Fig. 3 in due stati, uno attivo se è occorso un evento S, l'altro attivo se è occorso un evento R. In questo modo si ottiene l'automa di Fig. 5.

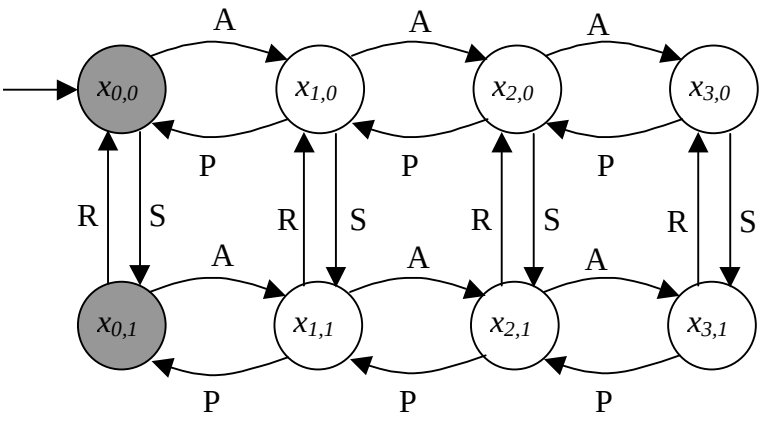

Fig. 4. Automa ottenuto per scomposizione degli stati.

Ragionando sull'automa di Fig. 4 è ora semplice rendersi conto che il corretto funzionamento dello schedulatore può essere ottenuto semplicemente disabilitando l'evento A, quando il sistema si trova in uno degli stati $x_{k,1}$, con k=1, 2, 3. Questo sarà quindi il compito del supervisore. Si noti che il comportamento del sistema a ciclo chiuso può essere rappresentato attraverso l'automa di Fig. 5, se ne deduce che il sistema a ciclo chiuso è a sua volta un sistema ad eventi discreti.

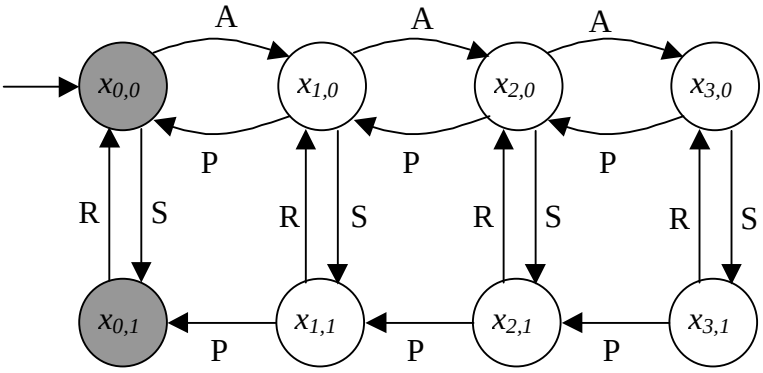

Fig. 5. Automa che descrive il comportamento a ciclo chiuso dello schedulatore.

1.3 Sistemi ad eventi discreti controllati

Si consideri un sistema ad eventi discreti rappresentato da un automa $G=(E, X, f, \Gamma, x_0, X_m)$, e si assuma di connettere a G un supervisore S secondo lo schema in retroazione illustrato in Fig. 1. Per caratterizzare il modo in cui il supervisore S funziona, si partizioni l'insieme degli eventi E in due sottoinsiemi disgiunti

$$E=E_c \cup E_{nc},$$

dove

- E_c è l'insieme degli eventi controllabili; gli eventi cioè di cui il supervisore è in grado di prevenire l'occorrenza, o in altre parole di disabilitare;
- E_{nc} è l'insieme degli eventi non controllabili; questi eventi non possono essere disabilitati dal supervisore. Casi tipici di eventi non controllabili sono i seguenti:

- eventi inerentemente imprevedibili, ad esempio guasti;
- eventi che modellano la lettura di sensori che misurano grandezze fisiche non controllate;
- eventi legati al naturale scorrere del tempo (ad esempio un evento potrebbe essere generato periodicamente da un clock);
- eventi ad alta priorità che si sceglie deliberatamente di non disabilitare.

Assumendo che S sia in grado di osservare tutti gli eventi dell'insieme E, esso è in grado di controllare la funzione di transizione dell'automa G abilitando o disabilitando dinamicamente gli eventi appartenenti a E_c. Quindi il supervisore S può essere visto come una funzione che ha per dominio il linguaggio eseguito da G e come codominio l'insieme potenza[1] di E:

$$S : s \in \mathsf{L}(G) \rightarrow S(s) \in 2^E .$$

Se l'automa G (sotto la supervisione di S) ha eseguito in passato una stringa $s \in \mathsf{L}(G)$, l'insieme degli eventi abilitati che G può eseguire in futuro dal suo stato corrente $f(x_0,s)$ è dato da

$$S(s) \cap \Gamma(f(x_0, s)) .$$

In altre parole G, sotto la supervisione di S, non può eseguire eventi che, oltre ad appartenere al suo insieme degli eventi attivi $\Gamma(f(x_0,s))$, non appartengano anche ad $S(s)$.

D'altra parte il supervisore non può disabilitare eventi non controllabili e quindi deve risultare

$$E_{nc} \cap \Gamma(f(x_0, s)) \subseteq S(s).$$

Un supervisore che rispetta tale proprietà viene detto ammissibile.

È importante notare che l'azione del supervisore S non è di tipo statico, ma di tipo dinamico, infatti gli eventi abilitati da S non dipendono esclusivamente dallo stato in cui si trova G, ma anche dalla traiettoria che G ha seguito per portarsi in tale stato. Formalmente questo è messo in evidenza dal fatto che il dominio di S è $\mathsf{L}(G)$ e non X. Ne consegue che l'azione di controllo può cambiare in visite successive allo stesso stato di G.

Il sistema complessivo ottenuto dalla connessione in retroazione di S e G, sarà indicato in seguito con il simbolo S/G; si dimostra che tale sistema è a sua volta un sistema ad eventi discreti, e che risulta[2]

$$\mathsf{L}(S/G) \subseteq \mathsf{L}(G) .$$

Il linguaggio marcato di S/G è inoltre definito attraverso la relazione

$$\mathsf{L}_m(S/G) = \mathsf{L}(S/G) \cap \mathsf{L}_m(G) ,$$

[1] L'insieme potenza di un insieme A, indicato con il simbolo 2^A, è l'insieme di tutti i sottoinsiemi di A.

[2] È possibile dimostrare che il linguaggio generato da S/G è ottenibile a partire dalla seguente regola ricorsiva:

1) $\varepsilon \in \mathsf{L}(S/G)$; 2) $[s\sigma \in \mathsf{L}(S/G)] \Leftrightarrow [(s \in \mathsf{L}(S/G)$ and $(s\sigma \in \mathsf{L}(G))$ and $(\sigma \in S(s))]$.

da cui si ricava

$$L_m(S/G) \subseteq L_m(G) \subseteq L(G).$$

In conclusione il sistema a ciclo chiuso *S*/*G* è in grado di interpretare un sottolinguaggio del sistema a ciclo aperto. Proprio in questo si esplica l'azione del supervisore, che tra tutti i possibili sottolinguaggi deve selezionare quello che consente di soddisfare le specifiche.

1.4 Progetto e implementazione del supervisore

Si consideri ancora lo schema di Fig. 1. Il problema del progetto del supervisore si può formulare nel seguente modo. Dato il sistema ad eventi discreti *G* ed una specifica di comportamento desiderato SP, progettare un supervisore *S* tale che il sistema a ciclo chiuso *S*/*G* segua il comportamento desiderato SP.

Le specifiche riguardanti il comportamento desiderato SP saranno usualmente assegnate attraverso proposizioni del linguaggio naturale; ad esempio nel problema del Paragrafo 1.2 la specifica assegnata era SP = "*all'occorrenza di un evento S il sistema non deve più accettare nuovi task, fino all'occorrenza di un evento R*".

Al fine di arrivare alla realizzazione del supervisore, si supponga di poter costruire un automa $R=(Y, E, g, \Gamma_R, y_0, Y)$ completamente raggiungibile che rappresenti il comportamento desiderato per il sistema a ciclo chiuso (si noti che si è imposto che tutti gli stati di *R* siano marcati, questo non costituisce una limitazione come sarà chiarito in seguito). Il supervisore deve essere progettato in modo che valga la relazione

$$L(R) = L(S/G).$$

Ovviamente per quanto detto al Paragrafo 1.3 ciò sarà realizzabile solo se risulta

$$L(R) \subseteq L(G). \qquad (1)$$

Si costruisca ora l'automa $R\times G=(Y\times X, E, (g,f), \Gamma_R\cap\Gamma, Y\times X_m)=(Y\times X, E, g\times f, \Gamma_{R\times G}, Y\times X_m)$, e si noti che

$$\begin{aligned} L(R\times G) &= L(R)\cap L(G) \\ &= L(R) \\ &= L(S/G) \\ L_m(R\times G) &= L_m(R)\cap L_m(G) \\ &= L(R)\cap L_m(G) \\ &= L(S/G)\cap L_m(G) \\ &= L_m(S/G). \end{aligned}$$

Da tali relazioni segue che il comportamento desiderato a ciclo chiuso è implicitamente codificato all'interno della struttura dell'automa $R\times G$. Se tale automa si trova nello stato (y,x) un evento può occorrere senza violare le specifiche SP se solo se esso è nell'insieme degli eventi attivi $\Gamma_{R\times G}(y,x)$. Quindi il supervisore *S* può essere scritto come

$$S(s) = \Gamma_{R\times G}(g \times f((y_0, x_0), s)) = \Gamma_R(g(y_0, s)). \qquad (2)$$

Tale relazione consente di realizzare il supervisore *S* in una maniera estremamente semplice; infatti basterà implementare l'automa *R* e far si che esso esegua come un osservatore passivo gli eventi eseguiti da *G*, gli eventi da abilitare saranno tutti quelli dati dal corrente insieme degli eventi attivi di *R* (si veda la Fig. 6).

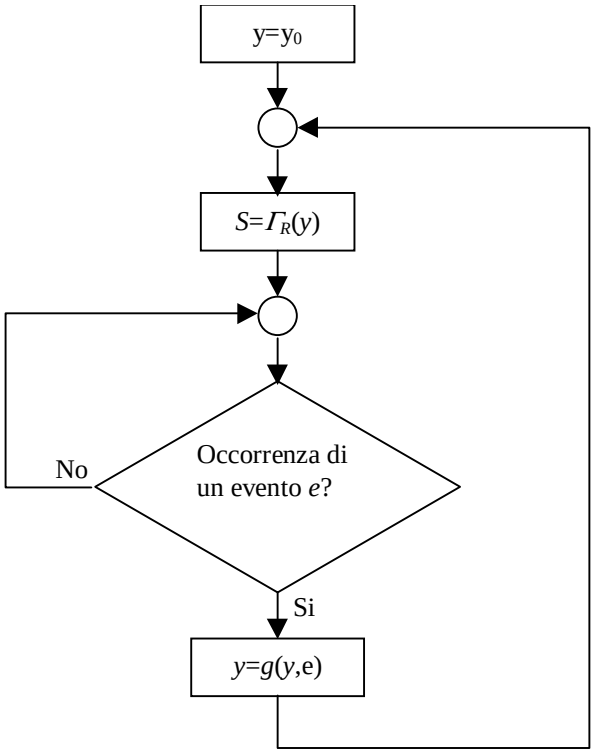

Fig. 6. Algoritmo di controllo per l'implementazione del supervisore.

A questo punto è necessario porsi la questione dell'ammissibilità del supervisore descritto dall'equazione (2). Una condizione necessaria è, come già detto, che risulti verificata la relazione (1) , ma questo non è in generale sufficiente quando esistono eventi non controllabili. Il seguente teorema invece fornisce una condizione necessaria e sufficiente per l'esistenza di un supervisore ammissibile.

Teorema. *Si consideri l'automa* $G=(E, X, f, \Gamma, x_0, X_m)$*, e sia* $E_{nc}\subseteq E$ *l'insieme degli eventi non controllabili; sia* $R=(Y, E, g, \Gamma_R, y_0, Y)$ *un automa completamente raggiungibile, tale che* $L(R)\subseteq L(G)$*. Esiste un supervisore S tale che* $L(S/G)=L(R)$*, se e solo se*

$$\forall s \in L(R), \sigma \in E_{nc} \quad s\sigma \in L(G) \Rightarrow s\sigma \in L(R)$$

■

Quest'ultimo teorema, noto come *Teorema della Controllabilità*, afferma che esiste un supervisore ammissibile che risolve il problema in esame, se, data una stringa eseguibile dal sistema desiderato a ciclo chiuso, allora le stringhe ottenute facendo seguire tale stringa da un evento non controllabile devono essere allora volta eseguibili a ciclo chiuso, se lo sono a ciclo aperto.

Si noti che la scelta di marcare tutti gli stati di *R* ha il solo scopo di far si che il linguaggio marcato da $R\times G$ coincida con il linguaggio marcato da *S*/*G*. L'automa *R* va quindi costruito imponendo le specifiche desiderate e poi semplicemente marcando tutti i suoi stati.

In conclusione il progetto e l'implementazione di un supervisore può avvenire secondo i seguenti passi:

1) Traduzione delle specifiche assegnate nel linguaggio naturale in un automa *R*, che rappresenterà il comportamento desiderato a ciclo chiuso; tutti gli stati di tale automa devono essere marcati.
2) Minimizzazione delle dimensioni di *R* eliminando gli stati equivalenti e quelli non raggiungibili;
3) Implementazione del supervisore secondo l'algoritmo mostrato in Fig. 6.

In realtà tra i passi 2 e 3 è necessario includere un passo intermedio in cui si vada a verificare l'esistenza di un supervisore ammissibile. Questo problema è affrontato nel Paragrafo successivo.

1.5 Verifica della proprietà di controllabilità

Nel paragrafo precedente si è visto che un supervisore può essere costruito sulla base delle specifiche, una volta che queste siano state tradotte in automa rappresentante il comportamento desiderato a ciclo chiuso. Affinché però il supervisore ottenuto risulti ammissibile è necessario verificare che le ipotesi del Teorema della controllabilità siano soddisfatte. Questa verifica consiste nel controllare che ogni stringa eseguibile a ciclo aperto, che sia formata da un prefisso eseguibile a ciclo chiuso e da un evento non controllabile, risulti ancora eseguibile a ciclo chiuso. Tale verifica per sistemi semplici può essere fatta attraverso una ispezione del grafo di transizione dell'automa *R*.

Esempio. Si consideri un sistema ad eventi discreti da controllare rappresentato dall'automa *G* in Fig. 7. Si assuma che a ciclo chiuso si voglia imporre la seguente specifica: "*Dopo una occorrenza dell'evento b_1, b_1 non deve più occorrere fino a quando non occorre per almeno una volta l'evento b_2* ". Tale specifica può essere tradotta nell'automa *R*, anch'esso illustrato in Fig. 7.

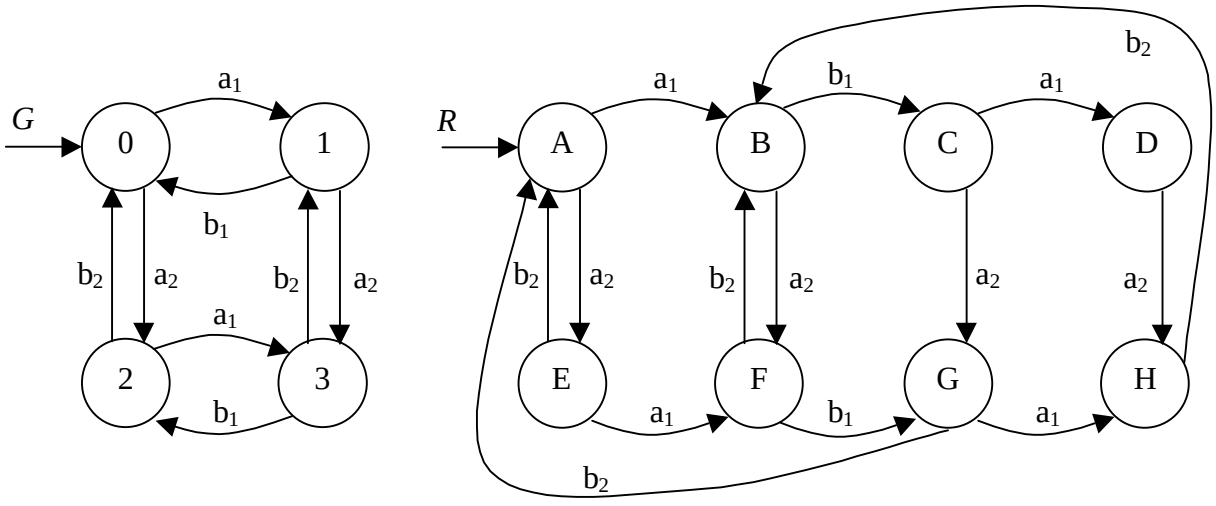

Fig. 7. Gli automi *G* ed *R* dell'esempio

Si consideri ora l'automa $R \times G$, che limitandosi a rappresentare gli stati raggiungibili è mostrato in Fig. 8. Se si assume che l'insieme degli eventi non controllabili sia $E_{nc}=\{b_1\}$, si può facilmente verificare che non esiste un supervisore ammissibile che riesca ad imporre il comportamento desiderato. Infatti nel diagramma di Fig. 8 si vede che nello stato (H,3) l'evento b_1 non è abilitato, mentre esso lo è nello stato 3 dell'automa G, quindi un supervisore dovrebbe, quando G si trova in tale stato, disabilitare b_1.

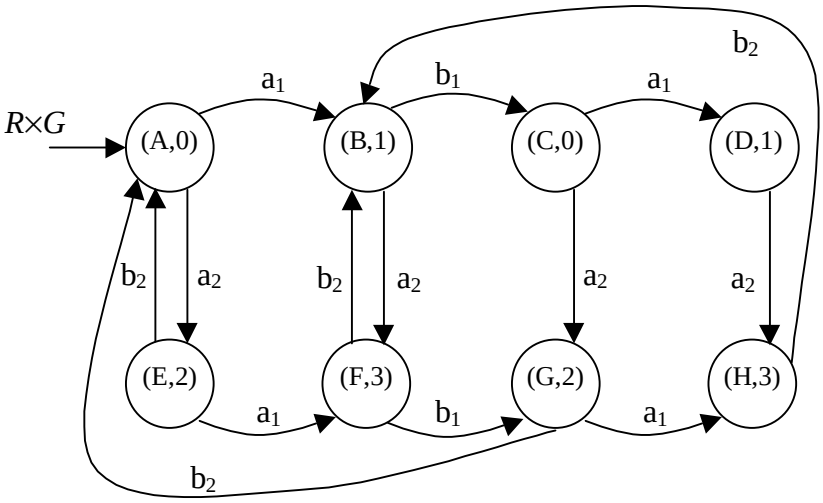

Fig. 8. L'automa $R \times G$ dell'esempio.

Questo tipo di verifica visiva è possibile solo quando si è di fronte a sistemi con un ridotto numero di stati. In casi più complessi si può ricorrere ad algoritmi che consentono di verificare la controllabilità in maniera automatica. Quest'argomento per la sua complessità non sarà però sviluppato in questi appunti; cosi come non si discuterà il problema di come affrontare la non controllabilità di un sistema ad eventi.

Tale problema può essere posto nei seguenti termini: assumendo che non esista un supervisore ammissibile per il problema in esame, come trovare un supervisore ammissibile che soddisfi almeno in parte alle specifiche, ovvero come modificare le specifiche, restringendo il comportamento del sistema a ciclo chiuso, in modo da ottenere un supervisore ammissibile.

In questo caso, ad esempio, se si mantiene sempre disabilitato l'evento a_1 si ottiene un sistema a ciclo chiuso che rispetta la specifica iniziale, ma che limita molto il comportamento del sistema a ciclo chiuso; infatti in questo caso l'evento b_1 non potrà mai occorrere.

1.6 Traduzione delle specifiche di controllo

In questo paragrafo saranno sviluppati una serie di esempi che mostrano come passare dalle specifiche e dal modello a ciclo aperto, al modello desiderato a ciclo chiuso. Tali esempi sono stati selezionati in modo da fornire un campione rappresentativo delle situazioni che più spesso si possono trovare nelle applicazioni del controllo di supervisione.

1.6.1 Esclusione di stati illegali

Spesso si verifica che il comportamento desiderato a ciclo chiuso porta a vietare la visita di alcuni stati di *G*; tali stati vengono detti illegali. In questo caso l'automa *R*, che specifica il comportamento a ciclo chiuso, può essere ottenuto andando ad eliminare da *G* gli stati illegali. L'eliminazione degli stati illegali può portare alcuni stati a diventare non raggiungibili, è allora opportuno eseguire su *R* l'operazione *Ac*, in modo da ottenere un automa *R* completamente raggiungibile.

Esempio. Si consideri l'ingresso di una banca (si veda la Fig. 9); per problemi legati alla sicurezza tali ingressi sono costituiti da due porte, su ciascuna delle quali è presente un pulsante attraverso cui un utente può richiedere l'apertura; la porta si richiude automaticamente dopo il passaggio dell'utente. Indicato con a_i l'evento "si è aperta la porta *i*", e con c_i l'evento "si è chiusa la porta *i*", con *i*=1,2, il comportamento non controllato del sistema costituito dalle due porte è mostrato in Fig. 10. Gli stati dell'automa *G* sono stati codificati su due cifre: la prima indica la condizione (1=aperta, 0=chiusa) della porta 1, La seconda indica la condizione della porta 2.

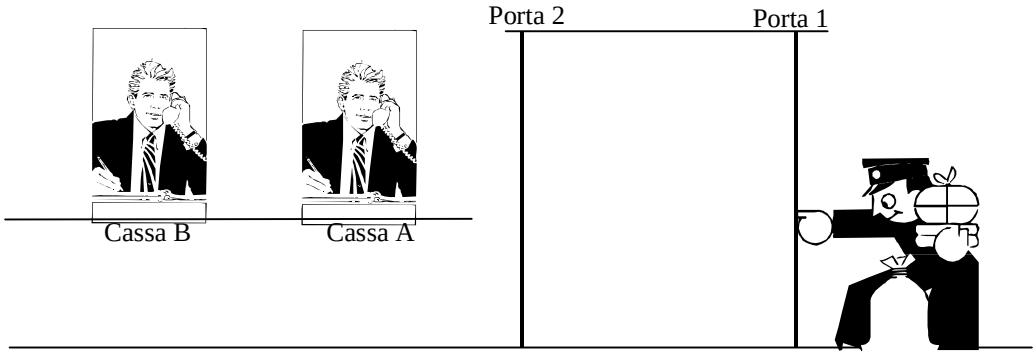

Fig. 9. L'ingresso di una banca.

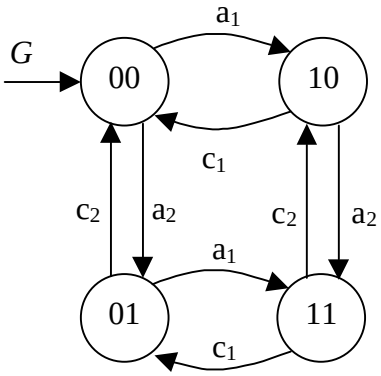

Fig. 10. L'automa *G* che descrive il comportamento non controllato dell'ingresso della banca.

La specifica a ciclo chiuso che si vuole imporre è la seguente: "*non devono mai essere aperte contemporaneamente entrambe le porte*". Tale specifica porta immediatamente ad identificare lo stato "11" come uno stato illegale. L'automa *R*, ottenuto eliminando tale stato, è mostrato in Fig. 11.

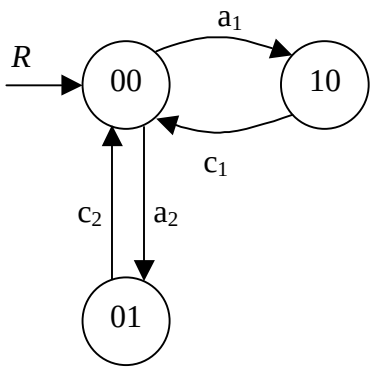

Fig. 11. L'automa *R* che descrive il comportamento desiderato a ciclo chiuso.

L'implementazione del supervisore porterà ad un sistema di controllo che provvederà a disabilitare l'apertura della porta 2 quando ci si trova nello stato 10, e a disabilitare l'apertura della porta 1 quando ci si trova nello stato 01.

1.6.2 Decomposizione di stati

Se, per poter determinare il comportamento futuro, le specifiche richiedono di tenere memoria di come un particolare stato è stato raggiunto, allora è necessario eseguire una decomposizione di tale stato in più stati. L'insieme degli eventi attivi di ciascuno dei nuovi stati va opportunamente determinato in base a come si vuole che l'evoluzione futura del sistema continui.

Esempio. Si consideri un sistema costituito da un nastro (*nastro 0*) trasportatore da cui si diramano altri due nastri (*nastri 1 e 2*). Tale sistema fa parte di un processo manifatturiero, su esso arrivano due tipi di pezzi (indicati semplicemente con le dizioni "*pezzo di tipo 1*", e "*pezzo di tipo 2*"), i pezzi di tipo 1 devono essere smistati sul nastro 1, mentre i pezzi di tipo 2 devono essere smistati sul nastro 2 (si veda la Fig. 12).

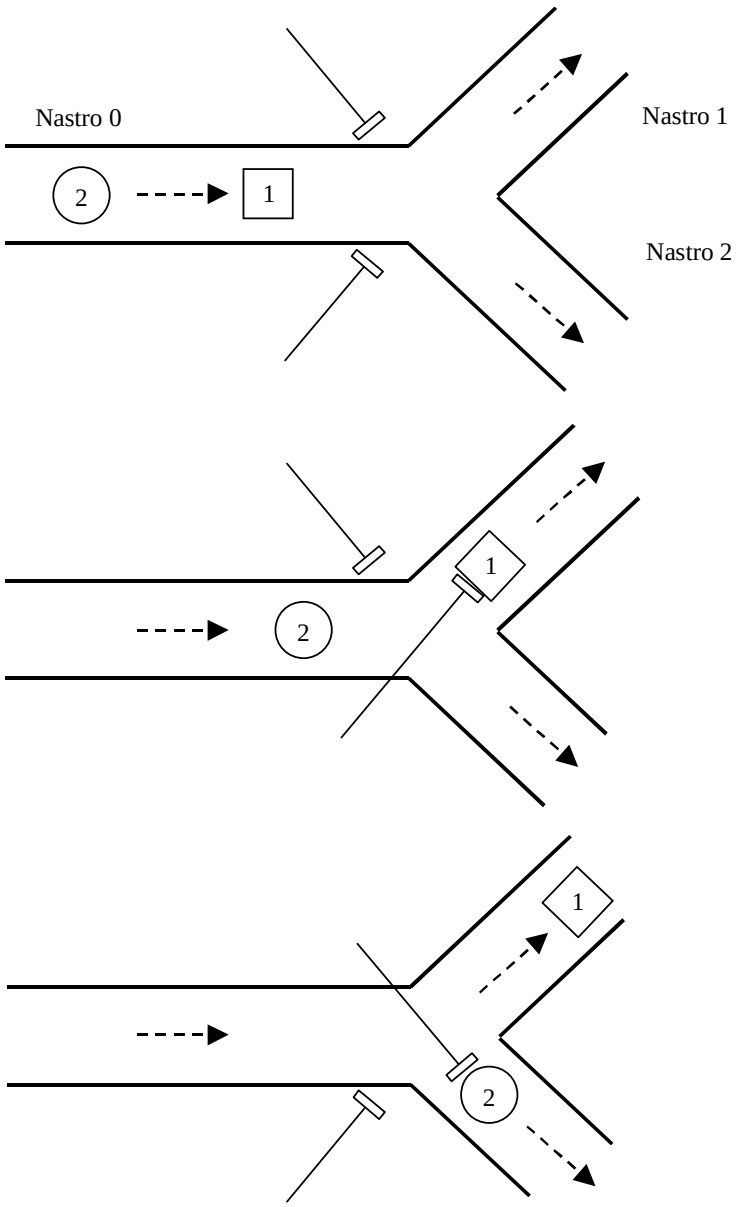

Fig. 12. Esempio del nastro trasportatore

Sul nastro si alternano sequenze di due pezzi, uno per ciascun tipo, e quindi un ciclo di funzionamento del sistema consiste nello smistamento di un pezzo di tipo 1 e di un pezzo di tipo 2.

Un sensore all'inizio del nastro 0 è in grado di determinare il tipo di pezzo che sta passando, mentre alla fine del nastro un altro sensore rileva la presenza di un pezzo (senza essere in grado di determinarne il tipo). Due sistemi indipendenti provvedono ad attivare i pistoni che trasferiscono il pezzo al termine del nastro 0 sui nastri 2 e 3.

Denotando con a_1 e a_2 gli eventi relativi alla presenza all'inizio del nastro 0 di un pezzo di tipo 1 o di tipo 2, e con p_1 e p_2 gli eventi relativi all'azionamento dei pistoni 1 e 2. Il comportamento non controllato del sistema è descritto dall'automa di Fig. 13.

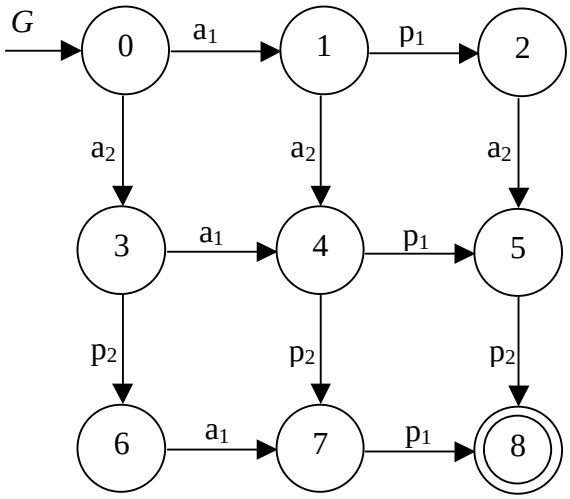

Fig. 13. Esempio dei nastri.

Tale comportamento non rispetta le specifiche; in quanto, a partire dallo stato 4, si verifica prima l'evento p_1 o l'evento p_2 a seconda della velocità dei due sistemi che controllano i pistoni, senza tener conto del tipo di pezzo che effettivamente si trova al termine del nastro 1. Questo comportamento è scorretto anche perché è possibile che i due pistoni vengano azionati contemporaneamente, portando ad una loro possibile collisione[3]

Un automa che rispetti le specifiche può essere ottenuto scomponendo lo stato 4 in due stati differenti, in modo da tenere memoria del pezzo che è arrivato per primo sul nastro. Tale soluzione è mostrata nell'automa di Fig. 14.

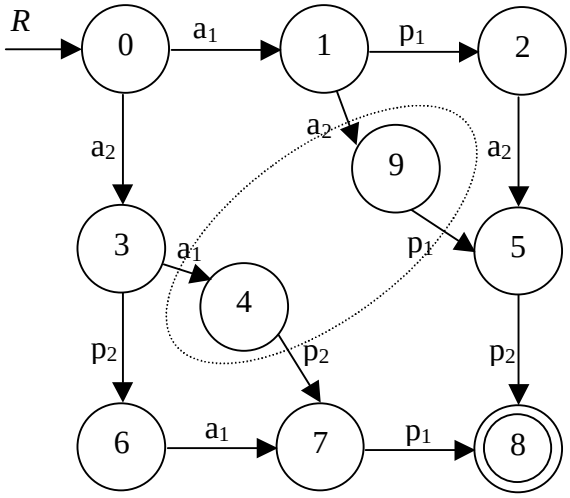

Fig. 14. Esempio dei nastri: comportamento desiderato a ciclo chiuso.

[3] In questo caso è come se gli eventi p_1 e p_2 occorressero contemporaneamente. Si tenga però presente che la teoria dei sistemi ad eventi discreti, cosi come sviluppata in questi appunti, vieta questa situazione, e due eventi non possono mai occorrere contemporaneamente.

1.6.3 Alternanza di eventi

Se le specifiche prevedono l'alternarsi di due eventi, detti *a* e *b*, questa alternanza può essere catturata dall'automa *H* di Fig. 15.

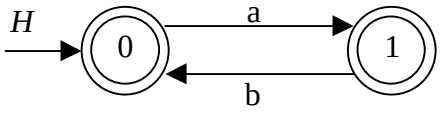

Fig. 15. Automa a due stati che forza l'alternanza degli eventi *a* e *b*.

A questo punto l'automa *R* può essere ottenuto attraverso l'operazione *H*||*G*.

Esempio. Torniamo all'esempio dell'ingresso di una banca, esempio già trattato nel Paragrafo 1.6.1, aggiungendo una nuova specifica. In particolare si vuole imporre un comportamento per cui la stessa porta non possa aprirsi due volte di seguito senza che nel frattempo si sia aperta anche l'altra. Per semplicità supponiamo di poter assumere che il primo evento che debba verificarsi sia l'apertura della porta 1. L'automa *H* che cattura questa specifica è indicato in Fig. 16. Il comportamento desiderato a ciclo chiuso può essere ottenuto eseguendo la combinazione parallela tra l'automa *R* di Fig. 11 (automa che gia contiene una parte delle specifiche del sistema di controllo) e l'automa *H*. Questa operazione va eseguita tenendo conto che gli eventi c_1 e c_2 non appartengono all'insieme degli eventi dell'automa *H*.

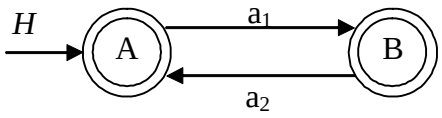

Fig. 16. Automa a due stati che forza l'alternanza nell'apertura delle due porte.

L'automa *R*||*G* è mostrato in Fig. 17, si noti che, come spesso capita nelle operazioni di composizione, in questo automa compaiono degli stati (01B e 10A) che non sono raggiungibili, e che quindi possono essere eliminati.

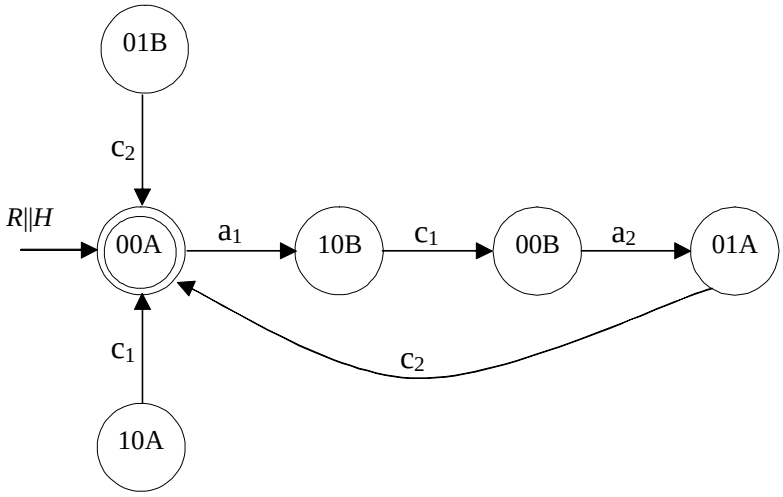

Fig. 17. Comportamento desiderato per il sistema di apertura delle porte di una banca.

1.6.4 Prevenzione di blocchi critici

Una specifica molto importante è quella che il sistema a ciclo chiuso sia non bloccante, in questo caso per ottemperare a questa specifica basterà eliminare dall'automa a ciclo aperto gli stati non controllabili ad uno stato marcato, o in altre parole di andare ad eseguire sull'automa *G* una operazione di *coAc*.

Esempio. *da sviluppare*