Diagnosi di un sistema ad eventi discreti mediante automi e reti di Petri

Maria Paola Cabasino

Dip. di Ing. Elettrica ed Elettronica, Università di Cagliari, Piazza d'Armi, 09123 Cagliari, Italy

email:mariapaolacabasino@tiscali.it

5 settembre 2005

Indice

Capitolo 1

Introduzione

In questa tesi verrà presentato un lavoro di ricerca sulla diagnosi dei sistemi ad eventi discreti. La proprietà di diagnosticabilità verrà introdotta nel contesto del problema della diagnosi dei guasti.
La scoperta e la localizzazione del guasto è un'attività importante nel controllo automatico di un sistema altamente complesso. La crescente richiesta di performance e affidabilità dei sistemi complessi ha reso necessario lo sviluppo di metodi sistematici e sofisticati per un'accurata diagnosi dei guasti del sistema.
Il problema della diagnosi dei guasti ha ricevuto larga attenzione nella letteratura e un'ampia gamma di schemi sono stati sviluppati. L'approccio con i sistemi ad eventi discreti (SED) è applicabile sia ai sistemi che ricadono naturalmente in questa classe, sia ai sistemi dinamici a variabili continue che possono essere ricondotti ai SED con un più alto livello di astrazione.
Gli stati del SED riflettono lo stato normale e quello di guasto dei componenti del sistema, mentre gli eventi di guasto fanno parte dell'insieme degli eventi. Il problema è di scoprire il verificarsi di tali eventi. Il maggior vantaggio di questo approccio è che non richiede dettagli approfonditi per costruire il modello del sistema che deve essere diagnosticato e quindi è particolarmente adatto per la diagnosi di sistemi che sono difficili da modellare. Tipici esempi includono estesi e/o complessi sistemi di riscaldamento, ventilazione e aria condizionata, impianti di potenza,ecc.

Il lavoro è organizzato come segue: il capitolo 2 contiene una breve descrizione sulla diagnosi in generale e sulla diagnosi dei sistemi ad eventi discreti in particolare. La diagnosi è il processo che individua le anormalità nel comportamento del sistema e isola la causa o la sorgente di queste anormalità. I guasti in un sistema industriale potrebbero nascere per diversi motivi come un errore nel pro-

getto, un'apparecchiatura mal funzionante, uno sbaglio dell'operatore, e così via. Ci sono tre principali fattori che spingono allo studio della diagnosi dei guasti: i guasti sono inevitabili; la diagnosi dei guasti è importante (se non cruciale); la diagnosi dei guasti è difficile. Dal punto di vista dell'implementazione, un sistema diagnostico può essere classificato come on-line o off-line. I metodi off-line assumono che il sistema sia in un banco di prova e debba essere testato per i possibili guasti a priori, mentre nella diagnosi on-line il sistema si assume che sia operativo e il sottosistema diagnostico è progettato per monitorare continuamente il comportamento del sistema, identificare ed isolare i guasti.
La seconda parte del capitolo contiene un breve descrizione della diagnosi dei sistemi ad eventi discreti. I sistemi ad eventi discreti sono dei sistemi dinamici il cui comportamento è governato dal manifestarsi di eventi fisici che causano cambiamenti repentini nello stato del sistema. Questo approccio è applicabile sia ai sistemi che ricadono naturalmente nella classe dei sistemi ad eventi discreti, sia ai i sistemi dinamici a variabili continue, che possono essere ricondotti ai SED con un più alto livello di astrazione. Inoltre, questo approccio alla diagnosi è appropriato per guasti notevoli o improvvisi che portano significativi cambiamenti allo stato dei componenti del sistema, ma non necessariamente portano il sistema alla rottura.

Il capitolo 3 è dedicato alla descrizione dei modelli ad eventi discreti che verranno trattati nella tesi. E' diviso in due parti: la prima parte presenta gli automi, mentre la seconda parte introduce le reti di Petri. Un sistema ad eventi discreti è un sistema il cui comportamento dinamico è caratterizzato dall'accadimento asincrono di eventi che individuano lo svolgimento di attività di durata non necessariamente nota. Un SED è caratterizzato da un insieme di eventi accadibili, da uno spazio di stato e da un evoluzione dello stato regolata dagli eventi. I sistemi ad eventi discreti si dividono in logici e temporizzati. Questo capitolo è dedicato alla trattazione dei SED logici, cioè quei SED il cui comportamento può essere rappresentato mediante le sequenze di eventi che essi generano, senza la necessità di conoscere la temporizzazione associata agli eventi. In particolare la prima parte del capitolo è dedicata agli automi finiti deterministici e non deterministici, mentre la seconda parte del capitolo è dedicata alle reti di Petri logiche o reti posto/transizione e alle regole che ne governano l'evoluzione. Una rete posto/transizione consiste in un modello logico, che non consente di rappresentare la temporizzazione degli eventi, ma solo l'ordine con cui essi si verificano. In particolare parleremo di abilitazione e scatto di una transizione, di marcatura di una rete e insieme di raggiungibilità della stessa e infine esporremo alcune proprietà che risulteranno utili ai fine della trattazione del quinto capitolo.

Il capitolo 4 presenta il primo approccio di diagnosi di un sistema ad eventi discreti. Tale approccio, dello studioso Stephane Lafortune, utilizza gli automi a stati finiti. In questo capitolo vengono introdotte due nozioni, tra loro legate, di diagnosticabilità. Queste sono la diagnosticabilità e la I-diagnosticabilità. Inoltre verrà presentata una procedura sistematica per la scoperta e la localizzazione dei guasti usando i diagnosticatori e verranno fornite le condizioni necessarie e sufficienti per la diagnosticabilità. Alla fine del capitolo verrà introdotto brevemente anche il software UMDES, particolarmente utile per modellare gli automi a stati finiti e svolgere molte operazioni su questi.

Il capitolo 5 presenta il secondo approccio di diagnosi di un sistema ad eventi discreti mediante reti di Petri. In questo capitolo si presenta un efficiente approccio per la diagnosi dei guasti che utilizza le reti di Petri. Tale approccio è stato proposto da Alessandro Giua e Carla Seatzu in [2]. Altri approcci per la diagnosi dei sistemi ad eventi discreti basati sulle reti di Petri sono stati presentati anche in [4], [5], [6]. In questo capitolo, per prima cosa daremo una caratterizzazione delle sequenze di scatto, corrispondenti a una data osservazione, basata sulla nozione di marcature di base e giustificazioni. Proporremo un algoritmo per il calcolo dell'insieme delle marcature di base, che useremo per determinare un automa deterministico, che chiameremo *albero di raggiungibilità delle marcature di base* o più brevemente albero base di raggiungibilità, che potrà essere usato come diagnosticatore.

Il capitolo 6 presenta un toolbox MATLAB comprendente due programmi che implementano due algoritmi sviluppati nel secondo approccio e presentati nel quinto capitolo. Il primo algoritmo data in ingresso una rete di Petri, una marcatura ed una transizione trova le spiegazioni minime; mentre il secondo algoritmo data una rete di Petri fornisce i risultati necessari per la costruzione dell'albero base di raggiungibilità.

Il capitolo 7 è dedicato al confronto tra i due approcci. Verranno introdotte delle analogie tra i due approcci e verranno messe in risalto le diversità. Inoltre verrà presentato un teorema, proposto per la prima volta in questa tesi, che dimostrerà come la definizione di aciclicità comune ai due approcci, ma derivante da esigenze diverse, sia sostanzialmente equivalente. Inoltre in ultimo presenteremo due esempi sviluppati con i due approcci. Il secondo esempio mira a dimostrare come l'approccio mediante reti di Petri sia molto più efficiente se consideriamo sistemi concorrenziali.

Il capitolo 8 è il capitolo conclusivo.

I contributi che questa tesi ha portato alla ricerca sono di natura teorica ed implementativa (sviluppo di software).

- Un primo contributo di questa tesi consiste nella dimostrazione che, per classi interessanti di sistemi, i due approcci possono entrambi venire applicati. Per tali classi di sistemi sono stati confrontati i due metodi e sono stati presentati degli esempi che mettono in risalto pregi e difetti di entrambe le tecniche.
- In questa tesi inoltre viene presentato per la prima volta un teorema che dimostra che per una sottoclasse di reti di Petri (reti limitate) l'ipotesi di aciclicità di Lafortune e quella di Giua-Seatzu sono sostanzialmente equivalenti, anche se derivanti da esigenze diverse.
- Un ulteriore contributo originale della tesi consiste nello sviluppo di un toolbox MATLAB per la diagnosi mediante reti di Petri.

Capitolo 2

La diagnosi

2.1 La diagnosi

La scoperta e l'isolamento dei guasti nei sistemi industriali è un tema che ha ricevuto moltissima attenzione negli ultimi decenni. Un guasto è definito come uno scostamento del sistema dal suo comportamento normale; la diagnosi è il processo che individua le anormalità nel comportamento del sistema e isola la causa o la sorgente di questa anormalità. I guasti in un sistema industriale potrebbero nascere per diversi motivi come un errore nel progetto, un'apparecchiatura mal funzionante, uno sbaglio dell'operatore, e così via. Ci sono tre principali fattori che spingono lo studio del problema della diagnosi dei guasti:

1. i guasti sono inevitabili;
2. la diagnosi dei guasti è importante (se non cruciale);
3. la diagnosi dei guasti è difficile.

Nei paragrafi che seguono esamineremo brevemente ognuno dei tre fattori sopra elencati.

I guasti sono inevitabili nei moderni e complessi ambienti industriali. Così come la tecnologia evolve e come noi continuiamo a costruire sistemi sempre più complessi e funzionali, pretendendo da questi sempre maggiori prestazioni, così la complessità di questi sistemi aumenta. Conseguentemente (e sfortunatamente)

noi aumentiamo la possibilità che il sistema si guasti, e se noi non ci preoccupiamo di rendere sicuri i nostri progetti, di migliorare la qualità delle tecniche di controllo e di migliorare la formazione degli operatori, allora i guasti nei sistemi saranno inevitabili. Infatti, data la complessa interazione fra i componenti, i sottosistemi e i processi, un guasto del sistema potrebbe essere considerato come un evento normale, o come una caratteristica intrinseca della maggior parte dei sistemi industriali.

Constatato il fatto che i guasti sono inevitabili, il bisogno di un efficace strumento per la loro scoperta è abbastanza evidente se consideriamo le loro conseguenze e i loro impatti non solo sui sistemi coinvolti, ma sulla stessa società. Molti dei maggiori disastri industriali e incidenti nel passato, come l'incidente ″Three Mile Island″, l'esplosione all' impianto petrolchimico nel Texas, il più grande black-out di New York e l'incidente dell'Apollo 13 hanno avuto la loro origine da una valvola che perdeva, o da un collegamento difettoso, o da un interruttore fuso. La scoperta precisa e tempestiva di questi guasti avrebbe potuto prevenire l'effetto a cascata che questi ″semplici″ guasti hanno causato. Concludendo perciò i guasti in un sistema complesso conducono alla fine a questi incidenti più gravi, a cui purtroppo abbiamo assistito. La sicurezza e l'affidabilità sono quindi due primi fattori che ci spingono allo studio di questo problema, specialmente nel caso di sistemi che sono potenzialmente ad alto rischio, come quelli citati sopra. La ″disponibilità″ del sistema è un fattore in più che ci motiva quando consideriamo sistemi quali impianti di potenza e industrie petrolchimiche, dove una interruzione dovuta a un guasto potrebbe condurre a maggiori problemi nello svolgimento dei servizi, oltre ad un serio impatto economico. Infine, c'è da notare che dei metodi efficaci nella diagnosi dei guasti non solo aiutano ad evitare effetti indesiderati, ma possono anche aumentare la qualità e il prestigio delle industrie. La migliore qualità delle prestazioni, l'integrità e l'affidabilità di un prodotto, la riduzione dei costi per la manuntenzione delle apparecchiature e dei servizi sono alcuni dei maggiori benefici che possono portare degli schemi efficaci di diagnosi, specialmente per quei prodotti e servizi orientati alle industrie come quelle di controllo di ambienti industriali e domestici, di servizi d'automazione, industrie manifatturiere di automobili e semiconduttori. Quindi, possiamo notare come dei metodi efficaci e tempestivi di diagnosi dei guasti possano aumentare la sicurezza, l'affidabilità, la qualità e l'economia dei processi industriali.

La natura complessa dei sistemi industriali non solo aumenta la potenziale avvenuta di un guasto, ma rende inoltre i problemi diagnostici molto difficili e impegnativi. Molti sistemi sono dotati di indicatori quali allarmi, luci di avviso, e così via, per indicare lo stato del sistema agli operatori che stanno monitorando il suo comportamento ed aiutarli nel loro ragionamento sulla diagnosi. Tuttavia, il costo

e la fattibilità tecnologica limitano il numero di sensori e quindi il numero delle variabili di sistema che possono essere direttamente monitorate. In questo modo l'informazione sullo stato del sistema che gli operatori (o tecnici) hanno è indiretta e quindi devono basare la diagnosi su modelli mentali del processo. Le complesse e spesso non evidenti interazioni ed accoppiamenti fra i componenti del sistema rendono le deduzioni un'attività impegnativa, specialmente se le decisioni sono da prendere in fretta. Infine, anche se uno ha a disposizione tutti i sensori di cui ha bisogno, il problema rimane ugualmente difficile. Infatti confrontare i dati provenienti da un largo numero di sensori, spesso apparentemente contraddittori, specialmente in situazioni di guasto, e correlare questi dati con i possibili guasti non è un compito semplice. Questo qualche volta porta gli operatori a ignorare gli allarmi. In più, ci sono stati nella storia degli incidenti industriali, causati da valori sbagliati o letti erroneamente, e/o interpretazioni sbagliate degli operatori sulle informazioni dei sensori.

In vista dei fattori sopra elencati, non è difficile capire quanto i meccanismi per una tempestiva e accurata diagnosi dei guasti siano davvero essenziali. Infatti, questo bisogno è stato ben capito e apprezzato sia nell'industria che nella ricerca. Una grande quantità di ricerche e sforzi sono stati spesi e tuttora vengono dedicati alla creazione e allo sviluppo di un sistema diagnostico automatico e sono stati proposti una varietà di schemi, che differiscono nella struttura teorica e nella filosofia di progetto e implementazione.

Poichè dal punto di vista concettuale esistono moltissimi metodi per la diagnosi dei guasti, possiamo classificarli come (i) metodi basati sull'albero dei guasti; (ii)metodi basati su modelli analitici; (iii) metodi basati sulla costruzione di modelli a partire dal ragionamento; e (iv) metodi basati sui sistemi ad eventi discreti.

Dal punto di vista dell'implementazione questi sistemi diagnostici possono essere classificati come on-line e off-line.

Nella diagnosi off-line, il sistema che deve essere diagnosticato non è operativo e può essere pensato come un sistema in un banco di prova. La procedura diagnostica prevede l'esecuzione di una serie di comandi di prova, poi si osservano i risultati di questi e si traggono le conclusioni sull'insieme dei possibili stati in cui il sistema potrebbe trovarsi. Per questo il problema della diagnosi off-line potrebbe essere considerato equivalente a un problema di verifica.

Nella diagnosi on-line il sistema si assume in condizioni di normale funzionamento. Lo scopo qui, come nel caso della diagnosi off-line, è di emettere una sequenza di comandi e identificare unicamente lo stato del sistema. Però, al contrario del

caso di diagnosi off-line, ora durante il processo di diagnosi bisogna rendere conto del possibile verificarsi di eventi non osservabili.

Riassumendo, i metodi off-line assumono che il sistema sia in un banco di prova e debba essere testato per i possibili guasti a priori, mentre nella diagnosi on-line, il sistema si assume che sia operativo e il sottosistema diagnostico è progettato per monitorare continuamente il comportamento del sistema, identificare ed isolare i guasti.

La diagnosi che esporremo in questa tesi è quella basata sui sistemi ad eventi discreti.

2.2 La diagnosi dei sistemi ad eventi discreti

In questa tesi viene proposto l'approccio con i sistemi ad eventi discreti. I sistemi ad eventi discreti sono dei sistemi dinamici il cui comportamento è governato dal manifestarsi di eventi fisici che causano cambiamenti repentini nello stato del sistema. Questo approccio è applicabile sia ai sistemi che ricadono naturalmente nella classe dei sistemi ad eventi discreti, sia ai sistemi dinamici a variabili continue, che possono essere ricondotti ai SED con un più alto livello di astrazione. Gli stati del SED riflettono lo stato normale e quello di guasto dei componenti del sistema, mentre gli eventi di guasto fanno parte dell'insieme degli eventi. Il problema è di scoprire il verificarsi di tali eventi. Il maggior vantaggio di questo approccio è che non richiede dettagli approfonditi per costruire il modello del sistema che deve essere diagnosticato e quindi è particolarmente adatto per la diagnosi di sistemi che sono difficili da modellare. Tipici esempi includono estesi e/o complessi sistemi di riscaldamento, ventilazione e aria condizionata, impianti di potenza e processi manifatturieri di semiconduttori e di automobili. Inoltre, questo approccio alla diagnosi è appropriato per guasti notevoli o improvvisi che portano significativi cambiamenti allo stato dei componenti del sistema, ma non necessariamente portano il sistema alla rottura.

Capitolo 3

Automi e Reti di Petri

3.1 Introduzione

Un sistema ad eventi discreti (SED) è un sistema il cui comportamento dinamico è caratterizzato dall'accadimento asincrono di eventi che individuano lo svolgimento di attività di durata non necessariamente nota. Formalmente, un sistema ad eventi discreti è caratterizzato da:

- un insieme E degli eventi accadibili;
- uno spazio di stato costituito da un insieme discreto X;
- un' evoluzione dello stato event–driven, cioè regolata dagli eventi: lo stato evolve nel tempo solo in dipendenza dell'accadimento di eventi asincroni, appartenenti all'insieme E.

L'equazione che descrive l'evoluzione dello stato a partire dallo stato iniziale x_0 è

$$x_{k+1} = \delta(x_k, e_k)$$

dove

- x_{k+1} è lo stato del sistema dopo l'accadimento dell'evento k-esimo;
- e_k è il k-esimo evento accaduto dall'istante iniziale considerato, che fa transire lo stato da x_k a $x_k + 1$;

- $\delta : X \times E \longrightarrow X$ è la funzione di transizione di stato.

I sistemi ad eventi discreti si dividono in due grandi famiglie: i modelli logici e i modelli temporizzati, a seconda del tipo di traccia (o traiettoria) degli eventi adottata.

Nei modelli logici la traccia degli eventi è costituita semplicemente da una sequenza di eventi $\{e_1, e_2, ...\}$, in ordine di occorrenza, senza alcuna informazione circa i tempi di occorrenza degli eventi; dato uno stato iniziale x_0, la traiettoria dello stato verrà costruita nel tempo come la sequenza di stati $\{x_0, x_1, x_2, ...\}$ risultanti dall'accadimento della sequenza di eventi, ma non è possibile specificare gli istanti di tempo in cui avvengono le transizioni di stato.

Nei modelli temporizzati, invece, la traccia degli eventi è costituita da una sequenza di coppie $\{(e_1, \tau_1), (e_2, \tau_2), ...\}$ dove ogni elemento e_i è accoppiato al suo tempo di accadimento τ_i, eventualmente stocastico; dato uno stato iniziale x_0, la traiettoria dello stato sarà evidentemente ancora la sequenza di stati $\{x_0, x_1, x_2, ...\}$ risultanti dall'accadimento della sequenza di eventi; in questo caso però si sa che le transizioni di stato avvengono negli stati di occorrenza degli eventi.

I modelli logici rendono agevole lo studio delle proprietà qualitative del sistema e consentono quindi di effettuare l'analisi strutturale di un SED, mentre i modelli temporizzati permettono di studiare i diversi comportamenti nel tempo del sistema, pertanto sono indispensabili qualora si voglia effettuare l'analisi prestazionale di un SED. In ogni caso, si può concludere che un modello ad eventi discreti costituisce sempre una descrizione matematica finita dell'insieme infinito di tracce che rappresentano il comportamento di un sistema ad eventi discreti ad un certo livello di astrazione.

In questa tesi verranno trattati i sistemi ad eventi logici ed in particolare nella prima parte di questo capitolo verranno introdotti gli automi finiti deterministici e non deterministici, e nelle seconda parte del capitolo le reti di Petri.

3.1.1 Automi finiti deterministici (AFD)

Definizione 3.1.1. *Un* automa finito deterministico *è una quintupla che si denota* $G = (X, E, \delta, x_0, X_m)$ *dove:*

- *X è un insieme finito di stati;*

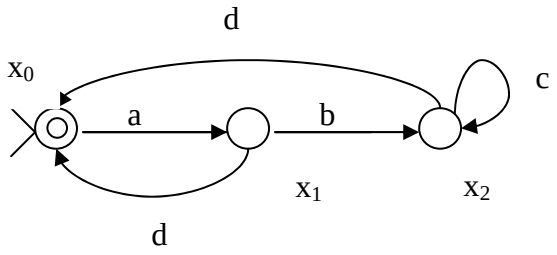


Figura 3.1: Un automa finito deterministico

- *E è un insieme finito di eventi (cioè un alfabeto di simboli);*
- $\delta : X \times E \longrightarrow X$ *è la funzione di transizione;* $\delta(x, e)$ *indica lo stato raggiunto dall'automa quando si verifica l'evento* e *a partire dallo stato x;*
- x_0 *è lo stato iniziale;*
- $X_m \subseteq X$ *è l'insieme di stati finali (o stati marcati).*

Un automa può essere descritto da un grafo in cui ogni stato è rappresentato con un nodo, lo stato iniziale da un nodo con una freccia, e lo stato finale da un nodo cerchiato. Se $\overline{x} = \delta(x, e)$ vi sarà un arco orientato dal nodo x al nodo $\overline{x}$ etichettato con il simbolo e per rappresentare la transizione da x a $\overline{x}$; tale arco viene anche indicato con il nome di e-transizione.

Esempio 3.1.1. *La figura 3.1 rappresenta un automa con* $X = \{x_0.x_1, x_2\}$*, alfabeto* $E = \{a, b, c, d\}$*, stato iniziale* x_0 *e insieme di stati finali* $X_m = \{x_0\}$*. La funzione di transizione è data dalla seguente tabella:*

δ	a	b	c	d
x_0	x_1	-	-	-
x_1	-	x_2	-	x_0
x_2	-	-	x_2	x_0

Ad esempio il simbolo x_1 *all'inserzione tra la riga* x_0 *e la colonna a indica che vale* $\delta(x_0, a) = x_1$*. Una casella vuota come quella all'inserzione fra la riga* x_0 *e la colonna b, indica che la corrispondente transizione non è definita. La c-transizione che parte dal nodo* x_2 *e arriva allo stesso nodo è detta cappio.*
L'automa in figura 3.1, ad esempio, può descrivere in modo semplificato il comportamento di una macchina spenta (stato x_0*) che su comando di un operatore*

(evento a) viene accesa (stato x_1). Una volta accesa viene attrezzata (evento b) raggiungendo lo stato di avviamento (stato x_2) potendo iniziare una lavorazione su un pezzo (evento c). Una volta avviata, la macchina può continuare a lavorare un pezzo dopo l'altro un numero indefinito di volte (l'evento c può sempre verificarsi una volta raggiunto lo stato x_2). Da un qualunque stato di funzionamento l'operatore può spegnere la macchina (evento d). Si considera come finale lo stato x_0, per indicare che si desidera riportare la macchina nello stato in cui è spenta in un ciclo di lavorazione.

Occorre notare che poichè in un AFD δ è una funzione parziale, per ogni $x \in X$ e per ogni $e \in E$ il valore della transizione $\delta(x, e)$ può non essere definito o essere univocamente definito. Se non fosse definito, non ci sarà una e-transizione uscente dal nodo x. Se fosse definito, vi sarà una e-transizione uscente dal nodo x. Non è possibile, però, che dallo stesso nodo escano due o più archi etichettati con lo stesso simbolo.
Ancora si osservi che, poichè ad ogni transizione corrisponde un evento, le etichette delle transizioni uscenti da uno stato x indicano quali eventi possono verificarsi in tale stato. E' possibile dunque dare la seguente definizione:

Definizione 3.1.2. *Dato un AFD $G = (X, E, \delta, x_0, X_m)$, definiamo l'*insieme degli eventi abilitati (o attivi) *nello stato $x \in X$ come*

$$A(x) = \{e \in E \mid \ \delta(x, e) \textit{è definito}\}.$$

Per indicare che $e \in A$ può anche usarsi la notazione $\delta(x, e)!$ che indica appunto che la funzione δ è definita per la coppia (x,e).

Il comportamento di un automa è dato dalle possibili evoluzioni, cioè sequenze di transizioni, che esso può generare. Ad ogni sequenza di transizioni corrisponde una produzione.

Definizione 3.1.3. *Dato un $AFD\ G = (X, E, \delta, x_0, X_m)$ si definisce* produzione *una sequenza*

$$x_{j_0} \xrightarrow{e_1} x_{j_1} \xrightarrow{e_2} ...x_{j_{k-1}} \xrightarrow{e_k} x_{j_k}$$

dove per ogni $i = 0, ...k$ vale $x_{j_i} \in X$ e inoltre da ogni stato $x_{j_{i-1}}$ il verificarsi dell'evento e_i porta allo stato x_{j_i}, cioè per ogni $i = 1, ...k$ vale $x_{j_i} = \delta(x_{j_{i-1}}, e_i)$. Si dice anche che tale produzione genera la parola $\omega = e_1e_2...e_k$ partendo dallo stato x_{j_0} e raggiungendo lo stato x_{j_k}.

Ad esempio una possibile produzione dell'automa in figura 3.1 è la seguente:

$$x_0 \xrightarrow{a} x_1 \xrightarrow{b} x_2 \xrightarrow{c} x_2 \xrightarrow{c} x_2$$

Questa produzione raggiunge lo stato x_2 a partire dallo stato x_0 e descrive un'evoluzione in cui la macchina viene accesa, attrezzata e lavora due parti. Tale produzione genera la parola $\omega = abcc$.
Si noti che una produzione può essere definita a partire da uno stato qualunque e non necessariamente dallo stato iniziale. Poichè δ è una funzione, è facile capire che non possono esistere due produzioni diverse che partono dallo stesso stato e generano la stessa parola.

Poichè a ogni evoluzione è associata una parola formata da simboli dell'alfabeto degli eventi E, all'insieme delle possibili evoluzioni che partono dallo stato iniziale è possibile associare un linguaggio $L \subseteq E^*$.

Definizione 3.1.4. *Dato un AFD $G = (X, E, \delta, x_0, X_m)$ si dice che una parola $\omega \in E^*$ è*

- generata *da G se $\delta^*(x_0, \omega)!$, cioè se esiste una produzione che genera ω a partire dallo stato iniziale;*
- accettata (o marcata) *da G se $\delta^*(x_0, \omega) = x \in X_m$, cioè se esiste una produzione che genera ω a partire dallo stato iniziale e raggiunge uno stato finale.*

Ad esempio, la parola $abcc$ è generata dall'automa in figura 3.1 perchè $\delta^*(x_0, abcc) = x_2$; tuttavia non accettata perchè x_2 non è uno stato finale. Viceversa la parola ad è accettata (e dunque anche generata) perchè $\delta^*(x_0, ad) = x_0$ e x_0 è finale. Infine la parola ac non è generata (e dunque nemmeno accettata) perchè $\delta^*(x_0, ac)$ non è definita. Da notare che la parola vuota è sempre generata e viene accettata solo se lo stato iniziale è anche finale.

Definizione 3.1.5. *Dato un AFD $G = (X, E, \delta, x_0, X_m)$ è possibile associare ad esso due linguaggi:*

- *Il* linguaggio generato

$$L(G) = \{\omega \in E^* \mid \delta^*(x_0, \omega)!\} \subseteq E^*,$$

cioè il linguaggio che costituisce l'insieme di tutte le parole generate.

- *Il* linguaggio accettato (o marcato)

$$L_m(G) = \{\omega \in E^* \mid \delta^*(x_0, \omega) \in X_m\} \subseteq L(G),$$

cioè il linguaggio che costituisce l'insieme di tutte le parole accettate.

E' importante osservare che il linguaggio generato da un AFD è necessariamente chiuso per prefisso, cioè vale $L(G) = \overline{L(G)}$: questo perchè se una parola può essere generata allora devono poter essere generati tutti i suoi prefissi.
Viceversa, il linguaggio accettato da un AFD non è necessariamente chiuso per prefisso, perchè non sempre tutti i prefissi di una parola accettata sono accettati. Ad esempio nell'automa in figura 3.1 la parola ad è accettata, ma il suo prefisso a non lo è. In generale dunque vale $L_m(G) \subseteq \overline{L_m(G)}$; come caso particolare in questa relazione vale l'eguaglianza se e solo se tutti gli stati sono finali.
Inoltre se una parola può essere accettata, allora tale parola e tutti i suoi prefissi possono anche a fortiori essere generati: vale dunque $\overline{L_m(G)} \subseteq L(G)$.

3.1.2 Automi finiti non deterministici (AFN)

Definizione 3.1.6. *Un* automa finito non deterministico *è una quintupla che si denota* $G = (X, E, \Delta, x_0, X_m)$ *dove:*

- *X è un insieme finito di stati;*
- *E è un insieme finito di eventi (cioè un alfabeto di simboli);*
- $\Delta : X \times E_\varepsilon \times X$ *è la relazione di transizione, dove* $E_\varepsilon = E \cup \{\epsilon\}$. *Se* $(x, e, \overline{x}) \in \Delta$*, allora a partire dallo stato* x *ed eseguendo una* e*-transizione (qui* e *può essere un simbolo dell'alfabeto oppure la parola vuota) si può raggiungere lo stato* $\overline{x}$*.*
- x_0 *è lo stato iniziale;*
- $X_m \subseteq X$ *è l'insieme di stati finali (o stati marcati).*

Anche di un AFN può essere data una rappresentazione del tutto analoga a quella di un AFD.

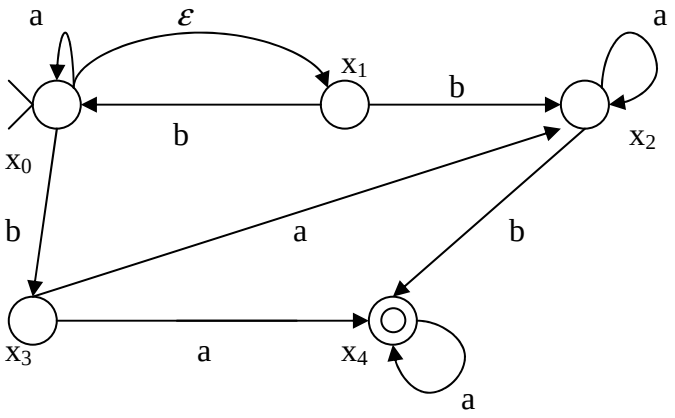


Figura 3.2: Un automa finito non deterministico

Esempio 3.1.2. *La figura 3.2 rappresenta un AFN avente insieme di stati* $X = \{x_0, x_1, x_2, x_3, x_4\}$*, alfabeto* $E = \{a, b\}$*, stato iniziale* x_0 *e insieme di stati finali* $X_m = \{x_4\}$*. La relazione di transizione è data da:*

$$\Delta = \{(x_0, \varepsilon, x_1), (x_0, a, x_0), (x_0, b, x_3), (x_1, b, x_0), (x_1, b, x_2),$$

$$(x_2, a, x_2), (x_2, b, x_4), (x_3, a, x_2), (x_3, a, x_4), (x_4, a, x_4)\}.$$

Gli AFN possono essere visti come una generalizzazione degli AFD. Infatti, la relazione Δ è una generalizzazione della funzione δ e consente di avere:

1. Transizioni etichettate con la parola vuota ε (dette anche ε-transizioni). Tali transizioni possono essere viste come corrispondenti ad eventi $''$silenziosi$''$ che fanno passare da uno stato all'altro senza che essi possano essere osservati;

2. Più transizioni uscenti dallo stesso stato e aventi la stessa etichetta. Tali transizioni possono essere viste come corrispondenti a eventi $''$parzialmente osservabili$''$: cioè si osserva il verificarsi di un evento, ma non si è in grado di determinare esattamente quale, fra due o più eventi etichettati, si è verificato.

Anche per gli AFN è possibile definire una produzione.

Definizione 3.1.7. *Dato un AFN* $G = (X, E, \Delta, x_0, X_m)$ *si definisce* produzione *una sequenza*

$$x_{j_0} \xrightarrow{e'_1} x_{j_1} \xrightarrow{e'_2} ...x_{j_{k-1}} \xrightarrow{e'_k} x_{j_k}$$

dove per ogni $i = 0, ...k$ vale $x_{j_i} \in X$ e inoltre da ogni stato $x_{j_{i-1}}$ eseguendo la transizione etichettata e'_i (si noti che e'_i potrebbe essere un evento in E o anche la parola vuota ε) si raggiunge lo stato x_{j_i}, cioè $e'_i \in E_\varepsilon$ e $(x_{j_{i-1}}, e'_i, x_{j_i}) \in \Delta$ per ogni i. Si dice anche che tale produzione genera la parola $\omega = e'_1 e'_2 ... e'_k$ partendo dallo stato x_{j_0} e raggiungendo lo stato x_{j_k}.

Ad esempio una possibile produzione dell'automa in figura 3.2 è la seguente:

$$x_0 \xrightarrow{a} x_0 \xrightarrow{\varepsilon} x_1 \xrightarrow{b} x_0 \xrightarrow{a} x_0 \xrightarrow{a} x_0$$

Questa produzione raggiunge lo stato x_0 a partire dallo stato x_0 e genera la parola $\omega = abaa$. Inoltre, in questo caso la lunghezza della parola generata è inferiore a quella della produzione: infatti $|\omega| = 4$, mentre la produzione contiene cinque transizioni.
Si osservi che poichè Δ è una relazione (e non una funzione), possono esistere due produzioni diverse che partono dallo stesso stato e generano la stessa parola. Ad esempio anche

$$x_0 \xrightarrow{a} x_0 \xrightarrow{b} x_3 \xrightarrow{a} x_4 \xrightarrow{a} x_4$$

è una produzione che parte dallo stato x_0 e genera ancora la parola $\omega = abaa$, raggiungendo però lo stato x_4.
E' proprio questa caratteristica, cioè il fatto che a una parola di eventi generata a partire da uno stato dato non corrisponde univocamente una produzione, che fa definire tali automi non deterministici.

La nozione di parola accettata da un AFN va considerata con particolare attenzione, perchè, a causa del non determinismo, alla stessa parola possono essere associate diverse produzioni.

Definizione 3.1.8. *Dato un AFN $G = (X, E, \Delta, x_0, X_m)$ si dice che una parola $\omega \in E^*$ è*

- generata *da G se esiste $x \in X$ tale che $(x_0, \omega, x) \in \Delta^*$, cioè se esiste una produzione che genera ω a partire dallo stato iniziale;*
- accettata (o marcata) *da G se esiste $x \in X_m$ tale che $(x_0, \omega, x) \in \Delta^*$, cioè se esiste una produzione che genera ω a partire dallo stato iniziale raggiungendo uno stato finale.*

Si noti che a causa del non determinismo possono esistere più stati raggiunti generando ω a partire dallo stato iniziale. La parola ω è accettata se almeno uno

fra questi è finale. Ad esempio nel caso dell'automa in figura 3.2 si è già visto che la parola $\omega = abaa$ può essere generata (fra le altre) dalle due produzioni

$$x_0 \xrightarrow{a} x_0 \xrightarrow{\varepsilon} x_1 \xrightarrow{b} x_0 \xrightarrow{a} x_0 \xrightarrow{a} x_0$$

$$x_0 \xrightarrow{a} x_0 \xrightarrow{b} x_3 \xrightarrow{a} x_4 \xrightarrow{a} x_4$$

La prima non porta a uno stato finale. Tuttavia poichè la seconda porta allo stato x_4 che è finale, si può concludere che la parola $abaa$ è accettata.

Definizione 3.1.9. *Dato un AFN $G = (X, E, \Delta, x_0, X_m)$ è possibile associare ad esso due linguaggi:*

- *Il* linguaggio generato

$$L(G) = \{\omega \in E^* \mid \exists x \in X : (x_0, \omega, x) \in \Delta^*\} \subseteq E^*,$$

  cioè il linguaggio che costituisce l'insieme di tutte le parole generate.

- *Il* linguaggio accettato (o marcato)

$$L_m(G) = \{\omega \in E^* \mid \exists x \in X_m : (x_0, \omega, x) \in \Delta^*\} \subseteq L(G),$$

  cioè il linguaggio che costituisce l'insieme di tutte le parole accettate.

Anche per gli AFN come già visto per il caso degli AFD, valgono le seguenti inclusioni: $L_m(G) \subseteq \overline{L_m(G)} \subseteq L(G) = \overline{L(G)}$.

3.1.3 Alcune proprietà degli automi

In questo paragrafo vedremo due delle proprietà degli automi (raggiungibilità e blocco) che ci serviranno nel seguito della tesi, quando parleremo dell'approccio di Lafortune.

Definizione 3.1.10. *Sia $G = (X, E, \delta, x_0, X_m)$ un AFD. Uno stato $x \in X$ è detto:*

- raggiungibile *dallo stato $\overline{x} \in X$ se esiste una parola $\omega \in E^*$ per cui $\delta^*(\overline{x}, \omega) = x$. Se $\overline{x} = x_0$, lo stato x è più brevemente detto raggiungibile;*

- coraggiungibile *verso lo stato $\overline{x} \in X$ se esiste una parola $\omega \in E^*$ per cui $\delta^*(x, \omega) = \overline{x}$. Se lo stato x è coraggiungibile verso uno stato $\overline{x} \in X_m$, x è più brevemente detto coraggiungibile;*

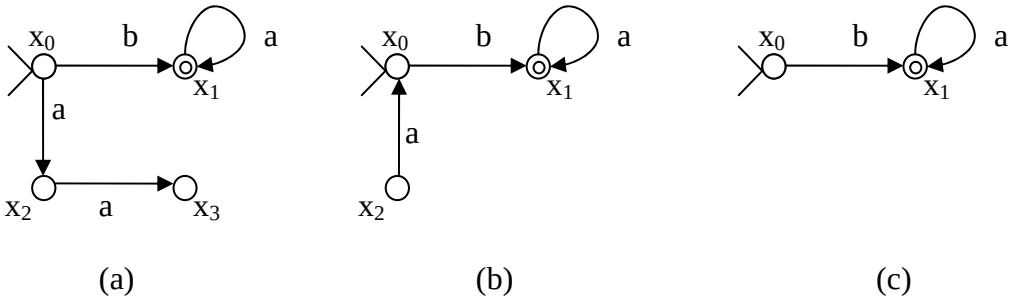


Figura 3.3: (a) Un automa raggiungibile e bloccante; (b) un automa non raggiungibile e non bloccante; (c) un automa rifinito

- bloccante *se è raggiungibile ma non coraggiungibile;*
- morto*, se qualunque sia* $e \in E$*, la transizione* $\delta(x, e)$ *non è definita.*

Si noti che morto e bloccante sono due proprietà diverse. Uno stato morto può essere non bloccante purchè finale, mentre uno stato bloccante può non essere morto. Dall'esame del grafo di un automa si può concludere che uno stato x è: raggiungibile dallo stato $\overline{x}$ se esiste un cammino orientato che parte da $\overline{x}$ e arriva a x; coraggiungibile verso lo stato $\overline{x}$ se esiste un cammino orientato che parte da x e arriva a $\overline{x}$; morto se non esistono archi che escono dal nodo.
Per l'automa invece valgono le seguenti proprietà:

Definizione 3.1.11. *Un AFD G è detto:*

- raggiungibile *se tutti i suoi stati sono raggiungibili;*
- coraggiungibile *se tutti i suoi stati sono coraggiungibili;*
- non bloccante *se tutti i suoi stati sono non bloccanti;*
- rifinito *se è raggiungibile e coraggiungibile;*
- reversibile *se ogni stato raggiungibile dallo stato iniziale è anche coraggiungibile verso lo stato iniziale.*

Facciamo un esempio per chiarire meglio i concetti appena definiti.

Esempio 3.1.3. *Nell'automa in figura 3.3(a) tutti gli stati sono raggiungibili, gli stati* x_3 *e* x_4 *non sono coraggiungibili, lo stato* x_3 *è anche morto; dunque tale automa è raggiungibile, non coraggiungibile e bloccante. Nell'automa in*

figura 3.3(b) tutti gli stati sono coraggiungibili, ma lo stato x_2 non è raggiungibile: dunque tale automa è non raggiungibile, coraggiungibile e non bloccante. Nell'automa in figura 3.3(c), tutti gli stati sono raggiungibili e coraggiungibili: dunque tale automa è rifinito. Nessuno dei tre automi in figura è reversibile.
Un esempio di AFD reversibile è l'automa in figura 3.1.

La raggiungibiltà permette dunque di studiare quali sono i possibili stati in cui un sistema può trovarsi a partire da uno stato. Il blocco indica uno stato anomalo, a partire dal quale il sistema non ha la possibilità di evolvere verso uno stato terminale. Lo stato morto indica una condizione in cui nessun evento può più verificarsi. La reversibiltà permette di verificare che il sistema possa essere reinizializzato. Si osservi inoltre che le precedenti definizioni possono anche applicarsi a un AFN, rifinendole dove necessario in termini della funzione Δ^*.

3.2 Reti di Petri

Le reti di Petri sono un modello di sistemi ad eventi discreti che trae origine dal lavoro di Carl Adam Petri, un ricercatore tedesco, che nel 1962 discusse la sua tesi di dottorato dal titolo 'Kommunication mit Automaten', in cui presentava questo nuovo modello logico che in seguito avrebbe appunto preso il suo nome. Esse rappresentano uno degli strumenti più efficaci nell'analisi di tutto quel ramo dell'automatica che si occupa di studiare i sistemi ad eventi discreti (SED). Le reti di Petri si dividono in logiche e temporizzate. Queste ultime si dividono a loro volta in deterministiche e stocastiche. Noi tratteremo le reti di Petri logiche o rete posto/transizione (o rete P/T), che consistono in un modello logico, che non consente di rappresentare la temporizzazione degli eventi, ma solo l'ordine con cui essi si verificano.

Fra i vari modelli ad eventi discreti, le reti di Petri hanno un'importanza predominante a causa di vari fattori:

- forniscono un formalismo grafico e matematico per la modellazione dei SED;
- permettono di dare una rappresentazione compatta in termini di spazio di stato (rappresentano SED con un numero infinito di stati, mediante un grafo con un numero finito di nodi);

- permettono di rappresentare esplicitamente il concetto di concorrenza, cioè di attività che possono venire svolte parallelamente;
- consentono una rappresentazione modulare; cioè se un sistema è composto da più sottosistemi che interagiscono tra loro, è generalmente possibile rappresentare ciascun sottosistema come una semplice sottorete e poi, mediante operatori di rete, unire le varie sotto-reti per ottenere il modello del sistema complessivo.

3.2.1 Struttura delle reti posto/transizione

Una rete di Petri P/T è un grafo bipartito, orientato e pesato. I due tipi di vertici sono detti posti (rappresentati da cerchi) e transizioni (rappresentati da barre). Gli archi, che devono essere orientati, connettono i posti alle transizioni e viceversa.

Definizione 3.2.1. *Una* rete P/T *è una struttura* $N = (P, T, Pre, Post)$ *dove:*

- $P = \{p_1, p_2, ..., p_m\}$ *è l'insieme degli* m posti*;*
- $T = \{t_1, t_2, ..., t_n\}$ *è l'insieme delle* n transizioni*;*
- $Pre : P \times T \longrightarrow N$*: è la* funzione di pre-incidenza *che specifica gli archi diretti dai posti alle transizioni (detti archi "pre") e viene rappresentata mediante una matrice* $m \times n$*;*
- $Post : P \times T \longrightarrow N$*: è* funzione di post-incidenza *che specifica gli archi diretti dalle transizioni ai posti (detti archi "post") e viene rappresentata mediante una matrice* $m \times n$*.*

Si suppone che $P \cap T = \emptyset$, cioè posti e transizioni sono insiemi disgiunti e che $P \cup T \neq \emptyset$, cioè la rete è costituita da almeno un posto o da una transizione.

Le matrici Pre e $Post$ sono delle matrici di interi non negativi. Si denota con $Pre(\cdot, t)$ la colonna della matrice Pre relativa alla transizione t, e con $Pre(p, \cdot)$ la riga della matrice Pre relativa al posto p. La stessa notazione vale per la matrice $Post$. L'informazione sulla struttura di rete contenuta nelle matrici Pre e $Post$ può essere compattata in un'unica matrice, detta di incidenza.

Definizione 3.2.2. *Data una rete* $N = (P, T, Pre, Post)$*, con* m *posti ed* n *transizioni, la* matrice di incidenza $C : P \times T \longrightarrow Z$ *è la matrice* $m \times n$ *definita come:*

$$C = Pre - Post$$

cioè il generico elemento di C vale $C(p,t) = Post(p,t) - Pre(p,t)$.

Data C non posso ricostruire il grafo, mentre date le matrici Pre e $Post$ posso ricostruire perfettamente il grafo. Un esempio chiarirà questi concetti.

Esempio 3.2.1. *In figura 3.4 è rappresentata la rete $N = (P, T, Pre, Post)$ con insieme dei posti $P = \{p_1, p_2, p_3, p_4\}$ e insieme delle transizioni $T = \{t_1, t_2, t_3, t_4, t_5\}$ le matrici Pre e Post valgono:*

$$Pre = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad Post = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

La matrice di incidenza vale:

$$C = \begin{pmatrix} 0 & -1 & 0 & 0 & 1 \\ 0 & 2 & -1 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$

Si noti che $Post(p_2, t_2) = 2$ e dunque vi sono due archi che vanno dalla transizione t_2 al posto p_2. Nella figura, invece di rappresentare i due archi è usata una notazione semplificata che consiste nel rappresentare un solo arco avente per etichetta un numero (2 in questo caso) che indica la sua molteplicità. L'arco bi-orientato viene usato per una migliore chiarezza grafica. Esso rappresenta in verità due archi: uno in ingresso alla transizione e l'altro in uscita.

Infine, data una transizione si definiscono i seguenti sistemi di posti:

${}^\bullet t = \{p \in P \mid Pre(p,t) > 0\}$: è l'insieme dei posti in ingresso a t.
$t^\bullet = \{p \in P \mid Post(p,t) > 0\}$: è l'insieme dei posti in uscita da t.
${}^\bullet p = \{t \in T \mid Post(p,t) > 0\}$: è l'insieme delle transizioni in ingresso a p.
$p^\bullet = \{t \in T \mid Pre(p,t) > 0\}$: è l'insieme delle transizioni in uscita da p.

Ad esempio nella rete in figura 3.4 vale ${}^\bullet t_2 = \{p_1\}$, $t_2^\bullet = \{p_2\}$, ${}^\bullet p_2 = \{t_2, t_5\}$, $p_2^\bullet = \{t_3, t_4\}$.

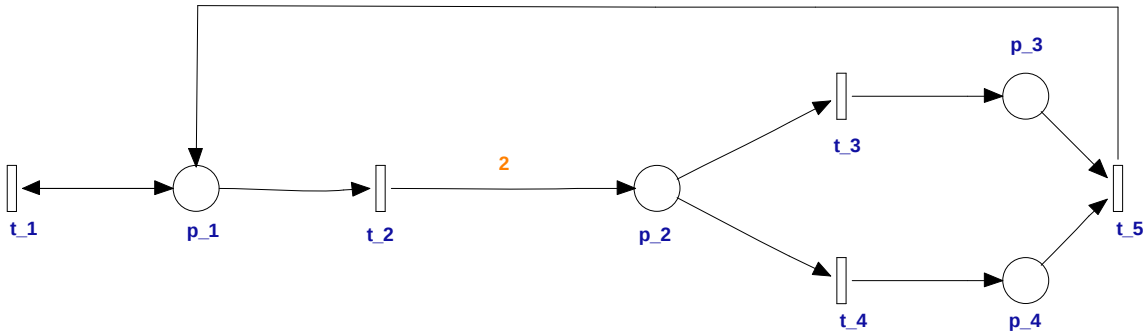


Figura 3.4: Una rete di Petri posto-transizione

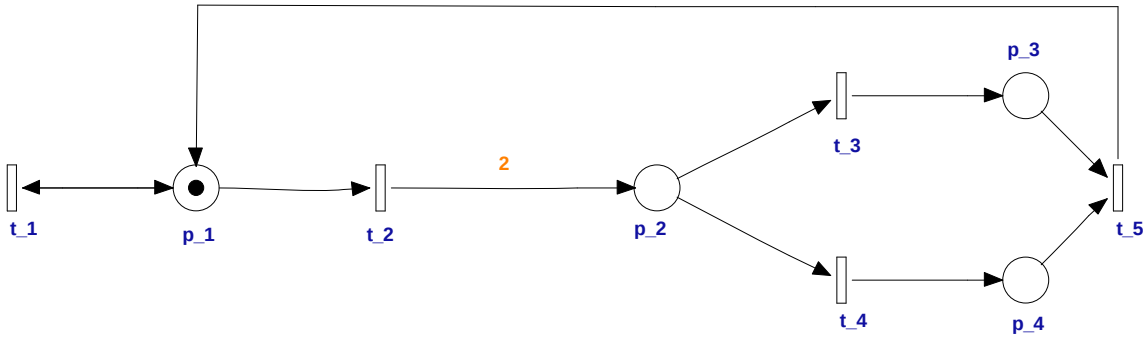


Figura 3.5: Evoluzione di una rete marcata. Marcatura iniziale

3.2.2 Marcatura e sistema di rete

Mediante la marcatura è possibile definire lo stato di una rete P/T.

Definizione 3.2.3. *Una* marcatura *è una funzione* $M : P \longrightarrow N$ *che assegna ad ogni posto un numero intero non negativo di marche (o gettoni) rappresentate graficamente con dei pallini neri dentro i posti.*

Considerando l'esempio in figura 3.4, una marcatura possibile M è $M(p_1) = 1, M(p_2) = M(p_3) = M(p_4) = 0$ come mostrato in figura 3.5. Un'altra marcatura possibile è quella mostrata in figura 3.6, dove $M(p_1) = 0, M(p_2) = 2, M(p_3) = M(p_4) = 0$, ottenuta dalla precedente con lo scatto di t_2.

Definizione 3.2.4. *Una rete* N *con una marcatura iniziale* M_0 *è detta* rete marcata o sistema di rete*, e viene indicata come* $\langle N, M_0 \rangle$.

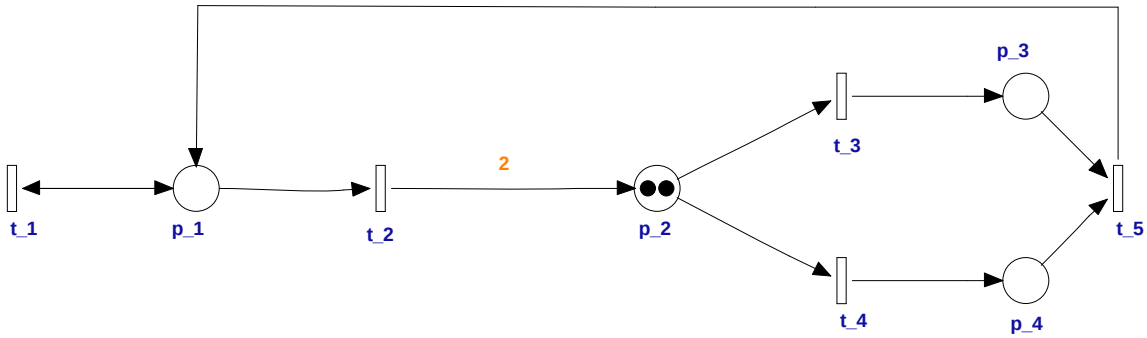


Figura 3.6: Evoluzione di una rete marcata. Marcatura raggiunta dopo lo scatto di t_2

Una rete marcata è, in effetti, un sistema ad eventi discreti a cui è associato un comportamento dinamico.

3.2.3 Abilitazione e scatto

Definizione 3.2.5. *Una* transizione t *è* abilitata *dalla marcatura* M *se*

$$M \geqslant Pre(\cdot, t)$$

cioè se ogni posto $p \in P$ *della rete contiene un numero di marche pari o superiore a* $Pre(p, t)$. *Per indicare che* t *è abilitata da* M *si scrive* $M[t\rangle$. *Per indicare che* t' *non è stata abilitata da* M *si scrive* $\neg M[t'\rangle$.

Definizione 3.2.6. *Una transizione* t *abilitata da una marcatura* M *può scattare. Lo scatto di* t *rimuove* $Pre(p, t)$ *marche da ogni posto* p *appartenente a* P *e aggiunge* $Post(p, t)$ *in ogni posto* p *appartenente a* P, *determinando una nuova marcatura* M'. *Cioè vale:*

$$M' = M - Pre(\cdot, t) + Post(\cdot, t) = M + C(\cdot, t)$$

Per indicare che lo scatto di t *da* M *determina la marcatura* M' *si scrive* $M[t\rangle M'$.

Definizione 3.2.7. *Una sequenza di transizioni* $\sigma = t_{j_1} t_{j_2} ... t_{j_r} \in T^*$ *è abilitata da una marcatura* M *se: la transizione* t_{j_1} *è abilitata da* M *e il suo scatto porta a* $M_1 = M + C(\cdot, t_{j_1})$ *; la transizione* t_{j_2} *è abilitata da* M_1 *e il suo scatto porta a*

$M_2 = M_1 + C(\cdot, t_{j_2})$ *ecc. Una sequenza abilitata* σ *viene anche detta sequenza di scatto e ad essa corrisponde la traiettoria:*

$$M[t_{j_1}\rangle M_1[t_{j_2}\rangle M_2 ... [t_{j_r}\rangle M_r.$$

Per indicare che la sequenza σ è abilitata da M si scrive $M[\sigma\rangle$. Per indicare che lo scatto di σ da M determina la marcatura M' si scrive $M[\sigma\rangle M'$.
Ad esempio nella rete in figura 3.6 una possibile sequenza di transizioni abilitata dalla marcatura data è $\sigma = t_3 t_4 t_5 t_1$ il cui scatto porta alla marcatura iniziale $M_0 = [\ 1 \quad 0 \quad 0 \quad 0\ ]^T$.

Definizione 3.2.8. *Il comportamento (o linguaggio) di una rete marcata* $\langle N, M_0\rangle$ *è l'insieme delle sequenze di scatto abilitate dalla marcatura iniziale, cioè l'insieme:*

$$L(N, M_0) = \{\sigma \in T^* : M_0[\sigma\rangle\}.$$

T^* è l'insieme di tutte le possibili combinazioni ottenibili dagli elementi di T.

Definizione 3.2.9. *Una marcatura* M *è detta raggiungibile in* $\langle N, M_0\rangle$ *se esiste una sequenza di scatto tale che* $M_0[\sigma\rangle M$. *L'*insieme di raggiungibilità *di una rete marcata* $\langle N, M_0\rangle$ *è l'insieme delle marcature che possono venir raggiunte a partire dalla marcatura iniziale, cioè l'insieme:*

$$R(N, M_0) = \{M \in N^m \mid \exists \sigma \in L(N, M_0) : M_0[\sigma\rangle M\}.$$

Definiamo insieme potenzialmente raggiungibile $PR(N, M_0)$ *, l'insieme:*

$$PR(N, M_0) = \{M \in N^m \mid \exists y \in N^n : M = M_0 + C \cdot y\}.$$

I due insiemi sono legati dalla relazione $R(N, M_0) \subseteq PR(N, M_0)$.

3.2.4 Equazione di stato

Definizione 3.2.10. *Sia* $\langle N, M_0\rangle$ *una rete marcata e* C *sia la sua matrice di incidenza. Se* M *è raggiungibile da* M_0 *scattando la sequenza di transizioni* σ *vale:*

$$M = M_0 + C \cdot \sigma.$$

σ *è detto vettore di scatto e ha tante componenti quante sono le transizioni.*

3.2.5 Proprietà di una rete di Petri

In questo paragrafo daremo alcune proprietà che ci saranno molto utili nel capitolo quinto dove si parla dell'approccio con le reti di Petri.

Una rete di Petri che non ha cicli orientati è detta **aciclica**. Per questa sottoclasse di reti valgono i seguenti risultati:

Teorema 3.2.1. *Consideriamo una* rete *di Petri* aciclica N.

(i) Se il vettore $y \in \mathbb{N}^n$ soddisfa l'equazione $M_0 + C \cdot y \geq 0$, allora esiste una sequenza di scatto σ "scattabile" dalla marcatura M_0 e tale che il vettore di scatto associato a σ è uguale a y.

(ii) Una marcatura M è raggiungibile da una marcatura M_0 se e solo se esiste una soluzione intera non negativa y che soddisfa l'equazione di stato $M = M_0 + C \cdot y$.

Un posto p è *limitato* se:

$$bound(p) = \max_{M \in R(N,M_0)} M(p) = k < +\infty \tag{3.1}$$

Un sistema di rete $\langle N, M_0 \rangle$ è *limitato* se esiste una costante k positiva tale che, per $M \in R(N, M_0)$, $M(p) \leq k$. Una rete è detta *strutturalmente limitata* se è limitata per qualunque marcatura iniziale.

La *funzione di etichettatura* $L : T \longrightarrow E \cup \{\varepsilon\}$ assegna a ogni transizione $t \in T$ o un simbolo appartenente a un dato alfabeto E o la stringa vuota ε. Indichiamo con T_u l'insieme delle transizioni che hanno etichetta ε:

$$T_u = \{t \in T \mid L(t) = \varepsilon\}.$$

Le transizioni appartenenti all'insieme T_u sono chiamate non osservabili o silenziose. Nel capitolo quinto dedicato all'approccio con le reti di Petri assumeremo che la stessa etichetta $e \in E$ non possa essere associata a più di una transizione. Le transizioni appartenenti all'insieme T_o sono dette osservabili.

Nel seguito indicheremo con C_u (C_o) la restrizione della matrice di incidenza a T_u (T_o). Indichiamo con w la stringa di eventi associata alla sequenza σ, tale che $w = L(\sigma)$. Notiamo che la lunghezza di una sequenza σ (indicata con $|\sigma|$) è sempre maggiore o uguale della lunghezza della corrispondente parola w (indicata con $|w|$). Infatti se σ contiene k' transizioni etichettate con ε allora $|\sigma| = k' + |w|$.

Inoltre indichiamo con σ_0 la sequenza di lunghezza zero e con ε la parola vuota. Usiamo la notazione $w_i \preccurlyeq w$ per indicare il generico prefisso di w lunghezza $i \leq k$, dove k è la lunghezza di w.

Definition 3.2.1. *Data una rete $N = (P, T, Pre, Post)$ e un sottoinsieme $T' \subseteq T$ delle sue transizioni, definiamo la sottorete $T'-$indotta di N la nuova rete $N' = (P, T', Pre', Post')$, dove $Pre', Post'$ sono le restrizioni di Pre e $Post$ a T'. La nuova rete N' può essere pensata come ottenuta da N rimuovendo tutte le transizioni $T \setminus T'$. Scriveremo anche $N' \prec_{T'} N$.*

Capitolo 4

Diagnosi mediante automi

4.1 Introduzione

In questo capitolo viene trattata la diagnosi di un sistema ad eventi discreti mediante l'approccio con gli automi. Questo approccio è stato studiato ed esposto da Stephane Lafortune, docente presso l'università del Michigan, in collaborazione con diversi suoi studenti di dottorato, tra cui Meera Sampath, Raja Sengupta e altri. Per questo capitolo ci riferiremo all'articolo [1]. La diagnosi di un guasto in un sistema ad eventi discreti si sviluppa in due principali fasi: la costruzione del modello del sistema ad eventi discreti che deve essere diagnosticato, seguito dalla costruzione del ″protocollo diagnostico″, in altre parole l'insieme di regole impiegate per la scoperta e la localizzazione del guasto.

Il comportamento del sistema è modellato con un linguaggio regolare ed è rappresentato da un automa a stati finiti. In breve, un linguaggio è detto diagnosticabile se è possibile scoprire, con un ritardo finito, il verificarsi di certi eventi non osservabili, detti eventi difettosi o guasti. Presenteremo nel seguito una procedura sistematica per la scoperta e l'isolamento dei guasti usando i diagnosticatori. Il diagnosticatore è una macchina a stati finiti costruita a partire dal modello di macchina a stati finiti del sistema. Questa macchina esegue la diagnosi osservando il comportamento on-line del sistema. Gli stati del diagnosticatore portano informazioni sul guasto e il verificarsi del guasto può essere scoperto (con un ritardo finito) esaminando questi stati. Le condizioni necessarie e sufficienti affinché un linguaggio sia diagnosticabile sono basate sul diagnosticatore. Quindi, il diagnosticatore ha due scopi:

- la verifica off-line delle proprietà di diagnosticabilità del sistema,
- la scoperta e la localizzazione on-line dei guasti.

Nella seconda sezione di questo capitolo introdurremo il modello del sistema e le notazioni necessarie per le sezioni che seguono. Poi definiremo il diagnosticatore e in ultimo le condizioni necessarie e sufficienti per la diagnosticabilità e la I-diagnosticabilità.

4.2 La nozione di diagnosticabilità

4.2.1 Il modello del sistema

Il sistema da diagnosticare è modellato come un automa a stati finiti:

$$G = (X, \Sigma, \delta, x_0) \tag{4.1}$$

dove X è lo spazio di stato, Σ è l'insieme di eventi, $\delta : X \times E \longrightarrow X$ è la funzione di transizione parziale, come visto nella sezione 3.1.1, e x_0 è lo stato iniziale del sistema.

Una delle ragioni per cui si usano gli automi a stati finiti per rappresentare un sistema ad eventi discreti è la loro facilità nell'analizzare il comportamento del sistema. Usando delle efficienti strutture dati, come i puntatori, noi possiamo facilmente percorrere il loro diagramma stato-transizione, avanti e indietro, e così avere risposta sulle proprietà dei linguaggi generato e marcato dell'automa.
Il modello G rende conto sia del comportamento normale, che di quello di guasto del sistema. Il comportamento del sistema è descritto dal linguaggio a prefisso-chiuso $L = L(G)$ generato da G. L è un sottoinsieme di Σ^*, dove Σ^* rappresenta la chiusura di Kleene dell'insieme Σ.

Un automa non deterministico è caratterizzato da eventi non osservabili. Le ε-transizioni di un automa non deterministico sono eventi che si verificano nell'automa, ma che non sono visibili, o osservabili, dall'esterno. Questa perdita di osservabilità è dovuta all'assenza di un sensore che rileva l'occorrenza dell'evento o al fatto che l'evento accade in una locazione remota, ma non è comunicato al sito centrale; questa è una tipica situazione nei sistemi di informazione distribuiti. Gli eventi di guasto che non causano un immediato cambiamento nella lettura del sensore sono un esempio di eventi non osservabili. Invece che etichettare tutte le

transizioni dovute ad eventi non osservabili con ε e ottenere un automa non deterministico come modello del sistema, definiamo gli eventi di queste transizioni come ″autentici″, ma caratterizziamo questi eventi come non osservabili.

In altre parole, il nostro modello del sistema sarà un automa deterministico il cui insieme di eventi è diviso in due insieme disgiunti:

$$\Sigma = \Sigma_o \cup \Sigma_u, \tag{4.2}$$

dove Σ_o rappresenta l'insieme degli eventi osservabili e Σ_u rappresenta l'insieme degli eventi non osservabili. Gli eventi osservabili nel sistema possono essere o comandi generati dal controllore o letture del sensore successive all'esecuzione del comando del controllore o un cambiamento del valore letto dal sensore. Gli eventi non osservabili possono essere guasti o altri eventi che causano cambiamenti nello stato del sistema, non rilevati dai sensori.

Consideriamo $\Sigma_f \subseteq \Sigma$ l'insieme degli eventi di guasto che devono essere diagnosticati. Assumiamo, senza perdita di generalità, che $\Sigma_f \subseteq \Sigma_u$, dal momento che un guasto osservabile può essere facilmente diagnosticato. Il nostro obiettivo è di identificare il manifestarsi dei guasti, se ne accadono, date le tracce di eventi osservati dal sistema, in cui ci sono solo gli eventi osservabili appartenenti a Σ_o. Dividiamo l'insieme dei guasti in insiemi disgiunti non vuoti corrispondenti ai diversi tipi di guasto:

$$\Sigma_f = \Sigma_{f1} \cup \Sigma_{f2} \cup ... \cup \Sigma_{fm}.$$

Indichiamo con $\Pi_f = \{1, 2, ...m\}$ questa partizione. Essa è motivata dalle seguenti considerazioni:

- Una strumentazione inadeguata può rendere impossibile la diagnosi del singolo guasto.
- Non possiamo pretendere di identificare singolarmente il verificarsi di ogni singolo evento di guasto. Possiamo solamente essere interessati a conoscere se ne è accaduto uno, appartenente a un insieme in cui l'effetto dell'insieme dei guasti sul sistema è lo stesso. In seguito, quando scriveremo che ″*un guasto di tipo F_i si è verificato*″ significherà che si è verificato qualche evento appartenente all'insieme Σ_{fi}.

Faremo le seguenti assunzioni sul sistema da analizzare:

1. Il linguaggio L generato da G è vivo, ossia esiste una transizione per ogni stato x appartenete a X; in altre parole il sistema non può raggiungere uno stato morto nel quale nessun evento è possibile.

2. Non esiste in G alcun ciclo di eventi non osservabili.

La prima assunzione, che riguarda la vivezza del linguaggio, è fatta per motivi di semplicità; mentre la seconda assunzione assicura che le osservazioni avvengano con una certa regolarità. Dal momento che la scoperta dei guasti è basata sulle transizioni osservabili del sistema, richiediamo che il sistema non generi sequenze arbitrariamente lunghe di eventi non osservabili.

Supponiamo che il sistema da diagnosticare consista di molti componenti fisici distinti e di un insieme di sensori. Prima costruiamo gli automi a stati finiti, che modellano il comportamento dei singoli componenti. Questi modelli tengono conto sia del comportamento normale, che di quello difettoso del sistema. Consideriamo, ad esempio, un semplice sistema d'aria condizionata (HVAC) composto da una pompa, una valvola ed un controllore. Partendo dai modelli dei componenti e dalla mappa dei sensori, generiamo un modello composito che raccoglie le interazioni fra i componenti e incorpora in esso la mappa dei sensori. Questo modello composto è il sistema G su cui svolgeremo la diagnosi.

Introduciamo ora alcune nozioni e la costruzione del generatore G' che sarà usato nel seguito.

4.2.2 Notazione

La traccia vuota è indicata con ε. Indichiamo con $\overline{s}$ l'insieme dei prefissi di ogni traccia $s \in \Sigma^*$. Indichiamo con L/s il post-linguaggio di L dopo s, ad esempio:

$$L/s = \{t \in \Sigma^* | st \in L\}.$$

Definiamo la proiezione $P : \Sigma^* \longrightarrow \Sigma_o^*$ dove:

$$\begin{cases} P(\varepsilon) = \varepsilon & \\ P(\sigma) = \sigma, & \text{se } \sigma \in \Sigma_o \text{ ;} \\ P(\sigma) = \varepsilon, & \text{se } \sigma \in \Sigma_u \text{ ;} \\ P(s\sigma) = P(s)P(\sigma), & s \in \Sigma^*, \sigma \in \Sigma \text{ .} \end{cases}$$

P cancella semplicemente gli eventi non osservabili in una traccia.
L'operatore di proiezione inversa P_L^{-1} è definito come:

$$P_L^{-1}(y) = \{s \in L | P(s) = y\}.$$

Indichiamo con s_f l'evento finale della traccia s. Definiamo:

$$\Psi(\Sigma_{fi}) = \{s\sigma_f \in L | \sigma_f \in \Sigma_{fi}\}.$$

$\Psi(\Sigma_{fi})$ indica l'insieme di tutte le tracce di L che terminano con un evento di guasto appartenente alla classe Σ_{fi}. Consideriamo $\sigma \in \Sigma$ e $s \in \Sigma^*$. Usiamo la notazione $\sigma \in s$ per indicare che σ è un evento appartenente alla traccia s. Con un leggero abuso di notazione, scriviamo $\Sigma_{fi} \in s$ per indicare il fatto che $\sigma_f \in s$, per qualche $\sigma_f \in \Sigma_{fi}$, o in modo più formale $\overline{s} \cap \Psi(\Sigma_{fi}) \neq \emptyset$.

Indichiamo:

$$X_o = \{x_0\} \cup \{x \in X | (\exists x' \in X), (\exists \sigma \in \Sigma_o) : \delta(x', \sigma) = x\}. \tag{4.3}$$

X_o è quindi l'insieme degli stati in X che sono raggiunti da almeno una transizione osservabile, insieme a x_0. Indichiamo con $L(G, x)$ l'insieme di tutte le tracce che hanno origine dallo stato x di G. Definiamo:

$$L_o(G, x) = \{s \in L(G, x) \quad | \quad s = u\sigma, u \in \Sigma_u^*, \sigma \in \Sigma_o\} \tag{4.4}$$

e

$$L_\sigma(G, x) = \{s \in L_o(G, x) \quad | \quad s_f = \sigma\}. \tag{4.5}$$

$L_o(G, x)$ rappresenta l'insieme di tutte le tracce che hanno origine nello stato x e terminano al primo evento osservabile, mentre $L_\sigma(G, x)$ indica quelle tracce in $L_o(G, x)$ che terminano con un particolare evento osservabile σ.

Nella sezione seguente avremo bisogno di usare un generatore del linguaggio.

Definizione 4.2.1. ***Generatore del linguaggio osservato***

Definiamo un generatore G' del linguaggio:

$$P(L) = \{t : t = P(s) \quad con \quad s \in L\}.$$

G' sarà in generale non deterministico, ed è così costruito:

$$G' = (X_o, \Sigma_o, \delta_{G'}, x_0) \tag{4.6}$$

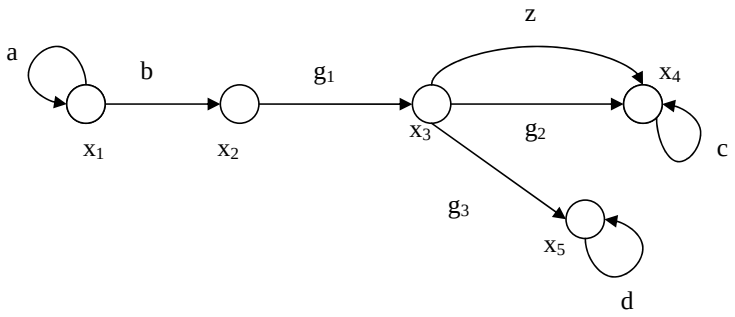


Figura 4.1: Esempio di un sistema con guasti multipli

dove X_o, Σ_o e x_0 sono definite rispettivamente in eq. (4.3)*, eq.* (4.2) *e in eq.* (4.1)*, e la relazione di transizione è data da:*

$$\delta_{G'} \subseteq (X_o \times \Sigma \times X_o)$$

ed è definita come segue:

$$(x, \sigma, x') \in \delta_{G'} \quad \textit{se} \quad \delta(x, s) = x' \quad \textit{per qualche } s \in L_\sigma(G, x).$$

E' facile verificare che $L(G') = P(L)$.

Facciamo ora un semplice esempio per chiarire le notazioni finora introdotte.

Esempio 4.2.1. *Consideriamo la rete in figura 4.1. Se consideriamo le tracce $s_1 = bg_1$ ed $s_2 = bg_1g_3d$ gli eventi finali rispettivamente della prima e della seconda traccia sono uguali a $s_{f1} = g_1$ ed $s_{f2} = d$.*
Se considero le classi $\Sigma_{f1} = \{g_1, g_2\}$, $\Sigma_{f2} = \{g_3\}$, l'insieme $\Psi(\Sigma_{f1})$ di tutte le tracce di L che terminano con un evento di guasto appartenente alla classe Σ_{f_1} è $\Psi(\Sigma_{f1}) = \{a^n bg_1 \mid n \geq 0\} \cup \{a^n bg_1g_2 \mid n \geq 0\}$. Possiamo definire anche l'insieme $\Psi(\Sigma_{f2})$ che è pari a $\Psi(\Sigma_{f2}) = \{a^n bg_1g_3 \mid n \geq 0\}$.
L'insieme degli stati in X che sono raggiunti da almeno una transizione osservabile per la rete in figura 4.1 è pari a $X_o = \{x_1, x_2, x_4, x_5\}$.
L'insieme di tutte le tracce che hanno origine nello stato x_2 e terminano al primo evento osservabile è pari a $L_o(G, x_2) = \{g_1g_3d, g_1g_2c, g_1zc\}$. Invece l'insieme delle tracce appartenenti a $L_o(G, x_2)$ che terminano con l'evento osservabile c è pari a $L_c(G, x_2) = \{g_1g_2c, g_1zc\}$.

4.2.3 Approcci per definire la diagnosticabilità

Siamo ora pronti a definire la nozione di diagnosticabilità. In poche parole, un linguaggio L è diagnosticabile se è possibile scoprire, con un ritardo finito, che è accaduto un guasto di un certo tipo, usando la traccia degli eventi osservati. Presentiamo ora due nozioni di diagnosticabilità; la prima è più rigorosa della seconda. La prima sarà la definizione di diagnosticabilità, la seconda, meno stringente, sarà quella di I-diagnosticabilità.

1. **Diagnosticabilità**:

   Definizione 4.2.2. *Un linguaggio L, chiuso per prefisso e vivo, è detto diagnosticabile, con riferimento alla proiezione P e alla partizione Π_f su Σ_f, se vale la seguente scrittura:*

   $$(\forall i \in \Pi_f)(\exists n_i \in \mathbb{N})[\forall s \in \Psi(\Sigma_{fi})](\forall t \in L/s)$$
   $$[\|t\| \geq n_i \Longrightarrow D(st, f_i)]$$

   dove la condizione di diagnosticabilità $D(s', f_i)$ di un guasto f_i data la stringa s' è:

   $$\omega \in P_L^{-1}[P(s')] \Longrightarrow \Sigma_{fi} \in \omega.$$

   La definizione appena data di diagnosticabilità significa che: data una traccia s generata dal sistema, che termina in un evento di guasto appartenente all'insieme Σ_{fi}, consideriamo una continuazione t di s sufficientemente lunga. La condizione D implica che ogni traccia appartenente al linguaggio che genera lo stesso insieme di eventi osservabili della traccia $''$st$''$, debba contenere in essa un evento di guasto dell'insieme Σ_{fi}. Questo implica che lungo una qualunque continuazione t di s, uno può scoprire il verificarsi di un guasto di tipo F_i con un ritardo finito, in particolare in al massimo n_i transizioni del sistema dopo s. In altre parole, la diagnosticabilità richiede che ogni evento di guasto conduca a delle osservazioni distinte, sufficienti per permettere unicamente l'identificazione del guasto con un ritardo finito.

   Il caso di guasti multipli dello stesso tipo, ossia appartenenti allo stesso insieme della partizione, richiede particolare attenzione. Quando più di un guasto dello stesso tipo, detto F_i, si manifesta lungo una traccia s di L, la definizione sopra data di diagnosticabilità, non richiede che ognuna di queste occorrenze sia scoperta. E' sufficiente essere capaci di concludere,

dopo un numero limitato di eventi dopo il verificarsi del primo guasto, che lungo s, un guasto dell'insieme Σ_{fi} è avvenuto.

Facciamo un esempio sulla nozione di diagnosticabilità data sopra.

Esempio 4.2.2. *Consideriamo il sistema rappresentato in figura 4.1. Gli eventi a, b, c e d sono osservabili, z è un evento non osservabile, mentre g_1, g_2 e g_3 rappresentano i guasti.*

$$\Sigma_o = \{a, b, c, d\}, \Sigma_u = \{z, g_1, g_2, g_3\}, \Sigma_f = \{g_1, g_2, g_3\}$$

Consideriamo che lo stato iniziale del sistema sia x_0. Se scegliamo le partizioni:

$$\Sigma_{f1} = \{g_1, g_2\}, \Sigma_{f2} = \{g_3\}$$

non è richiesto che si debba distinguere tra i guasti g_1 e g_2. Quindi il sistema illustrato è diagnosticabile con $n_1 = 2$ e $n_2 = 1$. Infatti considerate le parole $\omega_1 = bg_1g_2c$, $\omega_2 = bg_1zc$, $\omega_3 = bg_1g_3d$, in tutte possiamo identificare come traccia che termina in un evento di guasto, la traccia $s = bg_1$, quindi poichè n_1 è il numero massimo di transizioni del sistema in cui si può scoprire il verificarsi di un guasto dopo s, risulta $n_1 = 2$. Allo stesso modo per l'insieme Σ_{f2} possiamo considerare la parola $\omega_1 = bg_1g_3d$. La traccia s in questo caso è pari a $s = bg_1g_3$, conseguentemente il numero massimo di transizioni del sistema in cui si può scoprire il verificarsi di un guasto dopo s risulta $n_2 = 1$. D'altra parte se la partizione fosse stata:

$$\Sigma_{f1} = \{g_1\}, \Sigma_{f2} = \{g_2\} \text{ e } \Sigma_{f3} = \{g_3\},$$

allora il sistema sarebbe stato non diagnosticabile dal momento che non è possibile determinare il manifestarsi dell'evento g_2. Infatti se osservo la stringa bc non sono in grado di dire se il guasto g_2 si è verificato o meno, perchè i due eventi potrebbero appartenere indifferentemente alle due parole $\omega_1 = bg_1g_2c$ e $\omega_2 = bg_1zc$. Quindi g_2 non è diagnosticabile perchè non conduce a delle osservazioni distinte, sufficienti per permettere unicamente l'identificazione del guasto con un ritardo finito.

2. **I-Diagnosticabilità**: La precedente condizione di diagnosticabilità richiede che la condizione D sia applicata a tutte le tracce che contengono un guasto. Proponiamo ora una definizione più rilassata di diagnosticabilità, detta I-diagnosticabilità, che prevede che la condizione di diagnosticabilità D non sia applicata a tutte le tracce contenenti un guasto, ma solo a quelle in cui il guasto è seguito da un certo evento indicatore osservabile associato a ogni tipo di guasto.

Questa modifica è motivata dalle seguenti considerazioni fisiche. Consideriamo una pompa che fornisce acqua ad un serbatoio mediante una valvola. La portata d'acqua della pompa è controllata dalla posizione della valvola. Assumiamo ora che la valvola sia bloccata aperta. Fintanto che il controllore rimane nella modalità di ″valvola aperta″, esso non agisce sulla valvola. In questo caso il sistema è considerato non diagnosticabile secondo la precedente definizione. Nel caso della definizione modificata, associamo come indicatore l'evento ″valvola chiusa″ all'evento di guasto ″valvola bloccata aperta″, e chiediamo che il sistema esegua l'evento ″valvola chiusa″ prima di decidere sulla sua diagnosticabilità.
Il sistema è considerato diagnosticabile se, dopo l'esecuzione del corrispondente evento indicatore, è possibile scoprire il guasto della valvola; mentre è considerato non diagnosticabile se, anche dopo che l'evento indicatore è avvenuto, non si riesce ad individuare il guasto della valvola.

Riassumendo, la I-diagnosticabilità richiede la scoperta dei guasti solo dopo che si è verificato l'evento indicatore associato al guasto.

Associamo per prima cosa a ogni evento di guasto in Σ_f, uno o più eventi indicatori osservabili. Con $\Sigma_I \subseteq \Sigma_o$ indichiamo l'insieme degli eventi indicatori e con $I_f : \Sigma_f \longrightarrow 2^{\Sigma_I}$ indichiamo la mappa degli indicatori. Scegliamo infine una partizione Π_f su Σ_f tale che:

$$\bigcup\nolimits_{i\in\Pi_f} \Sigma_{fi} = \Sigma_f$$

come prima, con il vincolo aggiuntivo che per ogni $i = 1, ..., m$

$$\sigma_{f1}, \sigma_{f2} \in \Sigma_{fi} \Longrightarrow I_f(\sigma_{f1}) = I_f(\sigma_{f2})$$

e definiamo

$$I(\Sigma_{fi}) = I_f(\sigma_f) \quad \text{con} \quad \sigma_f \in \Sigma_{fi}.$$

Abbiamo ora un insieme di eventi indicatori osservabili $I(\Sigma_{fi})$ associati a ogni tipo di guasto F_i. Definiamo ora la I-diagnosticabiltà.

Definizione 4.2.3. *Un linguaggio L, chiuso per prefisso e vivo, è detto* I-diagnosticabile *con riferimento alla proiezione* P *, alla partizione* Π_f *su* Σ_f *e alla mappa degli indicatori* I *se vale la seguente scrittura:*

$$(\forall i \in \Pi_f)(\exists n_i \in \mathbb{N})[\forall s \in \Psi(\Sigma_{fi})](\forall t_1 t_2 \in L/s : st_1 \in \Psi[I(\Sigma_{fi})])$$

$$[\|t_2\| \geq n_i \Longrightarrow D(st_1t_2, f_i)]$$

dove la condizione di diagnosticabilità $D(s', f_i)$ di un guasto f_i data la stringa s' è:

$$\omega \in P_L^{-1}[P(s')] \Longrightarrow \Sigma_{fi} \in \omega.$$

come già visto in def. 4.2.2.

Da notare che $\Psi[I(\Sigma_{fi})]$ indica l'insieme di tutte le tracce di L che terminano in un evento osservabile dell'insieme $I(\Sigma_{fi})$. Perciò, nel caso della I-diagnosticabilità, si richiede che il verificarsi di un evento di guasto di tipo F_i debba essere scoperto al massimo in n_i transizioni del sistema, dopo che si è verificato l'evento indicatore appartenente all'insieme $\Psi[I(\Sigma_{fi})]$.

Esempio 4.2.3. *Consideriamo il sistema rappresentato in figura 4.1. Supponiamo che gli eventi indicatori siano scelti come segue: $I(\Sigma_{f1}) = \{c\}$, $I(\Sigma_{f2}) = \{d\}$, e $I(\Sigma_{f3}) = \{d\}$. Come partizioni scegliamo $\Sigma_{f1} = \{g_1\}$, $\Sigma_{f2} = \{g_2\}$ e $\Sigma_{f3} = \{g_3\}$. Questo sistema è I-diagnosticabile con $n_1 = 0$ e $n_3 = 0$, in quanto in questo caso n_1 ed n_3 sono il numero massimo di transizioni del sistema in cui viene scoperto rispettivamente il guasto f_1 ed il guasto f_3 dopo che si sono verificati i rispettivi eventi indicatori c e d. E' da notare che sebbene non sia possibile scoprire il verificarsi dell'evento g_2, l'evento indicatore d corrispondente a g_2 non segue il guasto, quindi la condizione di diagnosticabilità non è violata. Infatti è facile constatare guardando la figura 4.1 che dopo che si verifica il guasto f_2 si arriva allo stato morto c, e quindi l'evento indicatore d del guasto f_2 non potrà mai manifestarsi.*

4.3 Il Diagnosticatore

Introduciamo ora il diagnosticatore che è un nuovo automa a stati finiti costruito a partire dal modello del sistema G dato in eq. 4.1. Questa macchina è usata per diagnosticare on-line il comportamento di G. Il diagnosticatore è anche usato per verificare le condizioni necessarie e sufficienti per la diagnosticabilità. Mentre il diagnosticatore ″base″ presentato in questa sezione è adeguato per gli scopi di diagnosi, le modifiche aggiuntive, che sono presentate nella prossima sezione, sono necessarie per testare la diagnosticabilità. Vediamo ora la costruzione del diagnosticatore. Definiamo l'insieme delle etichette di guasto $\Delta_f = \{F_1, F_2, ...F_m\}$ dove $|\Pi_f| = m$ e l'insieme completo delle possibili etichette è:

$$\Delta = \{N\} \cup 2^{\{\Delta_f \cup \{A\}\}}.$$

Qui N è da interpretare con il significato di normale, A di ambiguo, e F_i con $i = 1, ...m$ sta a significare che un evento di guasto di tipo F_i si è verificato. Definiamo:

$$Q_o = 2^{X_o \times \Delta}.$$

Il diagnosticatore per G è l'automa a stati finiti:

$$G_d = (Q_d, \Sigma_o, \delta_d, q_0)$$

dove Q_d, Σ_o, δ_d e q_0 hanno la solita interpretazione.

Assumiamo che il sistema sia ″normale″ alla partenza, cioè che il sistema parta da uno stato in cui non è possibile che si sia ancora verificato alcun guasto. Quindi lo stato iniziale del diagnosticatore q_0 è definito come $\{(x_0, \{N\})\}$.

La funzione di transizione è definita come spiegato sotto. Lo spazio di stato Q_d è il sottoinsieme risultante di Q_o composto dagli stati del diagnosticatore, che sono raggiungibili da q_0 con δ_d. Dal momento che lo spazio di stato Q_d del diagnosticatore è un sottoinsieme di Q_o, uno stato q_d di G_d è della forma:

$$q_d = \{(x_1, l_1), ..., (x_n, l_n)\}$$

dove x_i appartiene a X_o e l_i appartiene a Δ; l_i è della forma $l_i = \{N\}$, $l_i = \{A\}$, $l_i = \{F_{i_1}, F_{i_2}, ...F_{i_k}\}$, o $l_i = \{A, F_{i_1}, F_{i_2}, ...F_{i_k}\}$ dove negli ultimi due casi $\{i_1, i_2, ...i_k\} \subseteq \{1, 2, ...m\}$. Un osservatore di G dà la stima dello stato corrente del sistema dopo il verificarsi di ogni evento osservabile. Il diagnosticatore G_d potrebbe essere pensato come un osservatore esteso, dove viene appeso a ogni stato un'etichetta della forma menzionata sopra. Le etichette attaccate agli stati portano informazioni sui guasti e i guasti sono diagnosticati controllando queste etichette.

Prima di definire la funzione di transizione del diagnosticatore, definiamo le seguenti tre funzioni: la funzione di propagazione delle etichette LP, la funzione di range R e la funzione di correzione delle etichette LC.

Definizione 4.3.1. *La* funzione di propagazione delle etichette *è* $LP : X_o \times \Delta \times \Sigma^* \longrightarrow \Delta$. *Dati* $x_0 \in X_o$, $l \in \Delta$ *e* $s \in L_o(G, x)$, LP *propaga l'etichetta* l *lungo* s, *partendo da* x *e seguendo le dinamiche di* G, *secondo* $L(G, x)$. *La funzione* LP *è definita come segue:*

$$LP(x, l, s) = \begin{cases} \{N\}, & \text{se } l = \{N\} \wedge \forall i[\Sigma_{fi} \notin s] \text{ ;} \\ \{A\}, & \text{se } l = \{A\} \wedge \forall i[\Sigma_{fi} \notin s] \text{ ;} \\ \{F_i : F_i \in l \vee \Sigma_{fi} \in s\}, & \text{altrimenti .} \end{cases}$$

Definizione 4.3.2. *La* funzione di range $R : Q_o \times \Sigma_o \longrightarrow Q_o$ *è definita come segue:*

$$R(q, \sigma) = \cup_{(x,l)\in q} \cup_{s\in L_\sigma(G,x)} \{(\delta(x, s), LP(x, l, s))\}.$$

Definizione 4.3.3. *La* funzione di correzione delle etichette $LC : Q_o \longrightarrow Q_o$ *è definita come segue:*

$$\begin{aligned} LC(q) = \quad & \{(x, l) \in q \mid x \textit{ compare solo una volta in tutte le coppie di } q \\ & \cup\{(x, \bar{l}) \mid (x, l_{i_1}), ..., (x, l_{i_k}) \in q, k > 1, \bar{l} = \{A\} \cup (l_{i_1} \cap ... \cap l_{i_k})\}. \end{aligned}$$

L'uso della funzione di correzione delle etichette LC e dell'etichetta A è spiegata nel seguito. L'etichetta acquisita da qualunque stato x lungo una traccia s indica il verificarsi o meno di un guasto, quando il sistema si muove lungo una traccia s e transiziona nello stato x. Supponiamo che esistano due coppie (x, l) e (x, l') in $R(q, \sigma)$ per uno stato q del diagnosticatore. Allora questo implica che lo stato x potrebbe essere raggiunto con un evento di guasto di un particolare tipo, detto F_i, o meno. In tale condizione, attacchiamo allo stato x l'etichetta A per sottolineare il fatto che c'è un'ambiguità. In altre parole, l'etichetta A è da interpretare col significato $''F_i$ e non F_i'' per $i \in \{1, ..., m\}$. E' da notare che non distinguiamo fra i casi $''F_i$ o F_j'', $''F_j$ o F_k'', $''N$ o F_i'', e così via. In tutte queste situazioni noi usiamo semplicemente l'etichetta A. Mentre questo fatto potrebbe condurre a una perdita di informazione, necessaria per determinare la diagnosticabilità del linguaggio, è adeguato per lo scopo della diagnosi trattare allo stesso modo tutte le classi prima menzionate. Definiamo ora la funzione transizione del diagnosticatore $\delta_d : Q_o \times \Sigma_o \longrightarrow Q_o$ come:

$$q_2 = \delta_d(q_1, \sigma) \Longleftrightarrow q_2 = LC[R(q_1, \sigma)]$$

con $\sigma \in e_d(q_1)$ dove :

$$e_d(q_1) = \cup_{(x,l)\in q_1}\{P(s) : s \in L_o(G, x)\}.$$

A parole, $e_d(q_1)$ è l'insieme degli eventi attivi (abilitati) di G_d nello stato q_1.

Riassumendo, il diagnosticatore G_d è costruito come segue. Consideriamo lo stato corrente del diagnosticatore (composto dagli stati stimati di G con le loro rispettive etichette) essere q_1, e consideriamo che il successivo evento osservato sia σ. Il nuovo stato q_2 del diagnosticatore è calcolato seguendo un algoritmo.

Algoritmo 4.3.1. *L'algoritmo è strutturato in tre passi:*

1. *Per ogni stato stimato x in q_1, calcola lo stato raggiunto in seguito all'evento σ, dato da $S(x,\sigma) = \{\delta(x, s\sigma)$ dove $s \in \Sigma_u^*\}$.*

2. *Consideriamo $x' \in S(x,\sigma)$ con $\delta(x, s\sigma) = x'$. Propaghiamo l'etichetta l associata a x con l'etichetta l' associata a x' in accordo con le seguenti regole:*

(a) *se $l = \{N\}$ ed s non contiene eventi di guasto, allora l'etichetta l' è ancora $\{N\}$.*

(b) *se $l = \{A\}$ ed s non contiene eventi di guasto, allora l'etichetta l' è ancora $\{A\}$.*

(c) *Se $l = \{A, F_i\}$ ed s non contiene eventi di guasto, allora l'etichetta l' è $\{F_i\}$.*

(d) *Se $l = \{N\}$ o $\{A\}$ ed s contiene un evento di guasto appartenente agli insiemi Σ_{fi}, Σ_{fj}, allora $l' = \{F_i, F_j\}$.*

(e) *Se $l = \{F_i, F_j\}$ o $\{A, F_i, F_j\}$ ed s contiene un evento di guasto appartenente all'insieme Σ_{fk}, allora $l = \{F_i, F_j, F_k\}$.*

3 *Indichiamo con q_2 tutte le coppie (x', l') calcolate seguendo il punto 1 e 2, per ogni (x, l) contenuto in q_1. Sostituiamo con $\{x', A, F_i, F_j\}$ tutte le coppie $(x', l'), (x', l'') \in q_2$ tali che F_i ed F_j sono componenti sia di l' che di l''. Cioè, se lo stesso stato stimato x' compare più di una volta in q_2 con diverse etichette, associamo a x' tutte le componenti comuni di queste etichette, e in aggiunta attacchiamo a x' l'etichetta A di stato ambiguo.*

Notiamo che nei casi $(c), (d)$ ed (e) visti sopra, non propaghiamo l'etichetta A da uno stato all'altro. Ciò conduce a una riduzione dello spazio di stato del diagnosticatore, ma non conduce a una perdita di informazione necessaria alla determinazione della proprietà di diagnosticabilità di un linguaggio o all'implementazione della diagnostica. La ragione di ciò diventerà chiara nelle sezioni seguenti.

Esempio 4.3.1. *Consideriamo il sistema G rappresentato in figura 4.2. Gli eventi $\alpha, \beta, \gamma, \delta$ e σ sono osservabili, mentre $\sigma_u, \sigma_{f1}, \sigma_{f2}$ e $\sigma_{f2'}$ sono eventi non osservabili. Consideriamo le partizioni $\Sigma_{f1} = \{\sigma_{f1}\}$ e $\Sigma_{f2} = \{\sigma_{f2}, \sigma_{f2'}\}$. In basso nella stessa figura è rappresentato il diagnosticatore, dove per chiarezza abbiamo rappresentato le coppie (x, l), con xl. Lo stato iniziale x_0 di G è lo stato 1.*

Sottolineiamo il fatto che nella procedura di costruzione del diagnosticatore abbiamo assunto noto lo stato iniziale del sistema, dal momento che abbiamo assunto che il diagnosticatore partisse in parallelo con il sistema dall'inizio del funzionamento. Ma la procedura rimane valida anche nel caso in cui lo stato iniziale sia sconosciuto.

4.4 Criteri per la diagnosticabilità

In questa sezione presentiamo le condizioni necessarie e sufficienti affinché un linguaggio L sia diagnosticabile e $I-$diagnosticabile. Queste condizioni sono enunciate sul diagnosticatore o su variazioni di questo. Per provare queste condizioni usiamo, in aggiunta al diagnosticatore, il generatore del linguaggio osservato G' introdotto nella def. 4.2.1.

4.4.1 Condizioni per la diagnosticabilità

Consideriamo solo il caso di guasti singoli dello stesso tipo, trascurando il caso di guasti multipli dello stesso tipo. Enunciamo ora un paio di proprietà del diagnosticatore che discendono dalla sua costruzione. Queste proprietà e le definizioni che seguono saranno usate successivamente per enunciare e dimostrare le condizioni per la diagnosticabilità.

Abbiamo definito l'automa a stati finiti G nell'eq. 4.1. Abbiamo poi introdotto il generatore del linguaggio osservato in def. 4.2.1 e il diagnosticatore nella def. 4.2.2. Per quest'ultimo valgono le seguenti proprietà.

Proprietà e definizioni di G_d

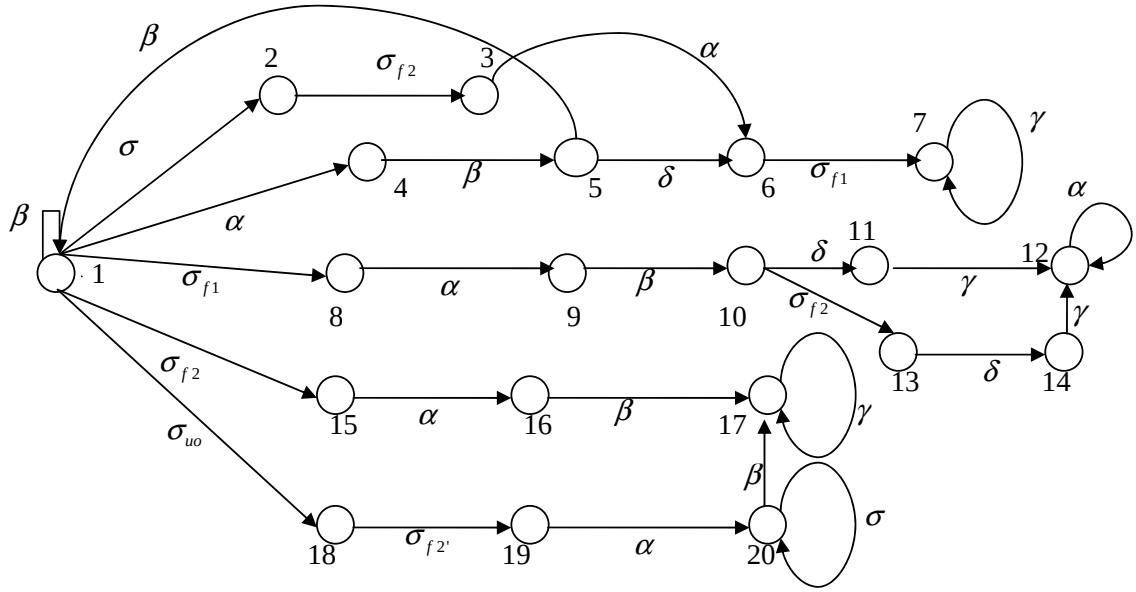
P1) Per costruzione, qualunque x_i appartenente a X_o compare in al massimo una coppia (x_i, l_i) in ogni stato di Q_d.

P2) Consideriamo q appartenente a Q_d. Allora:

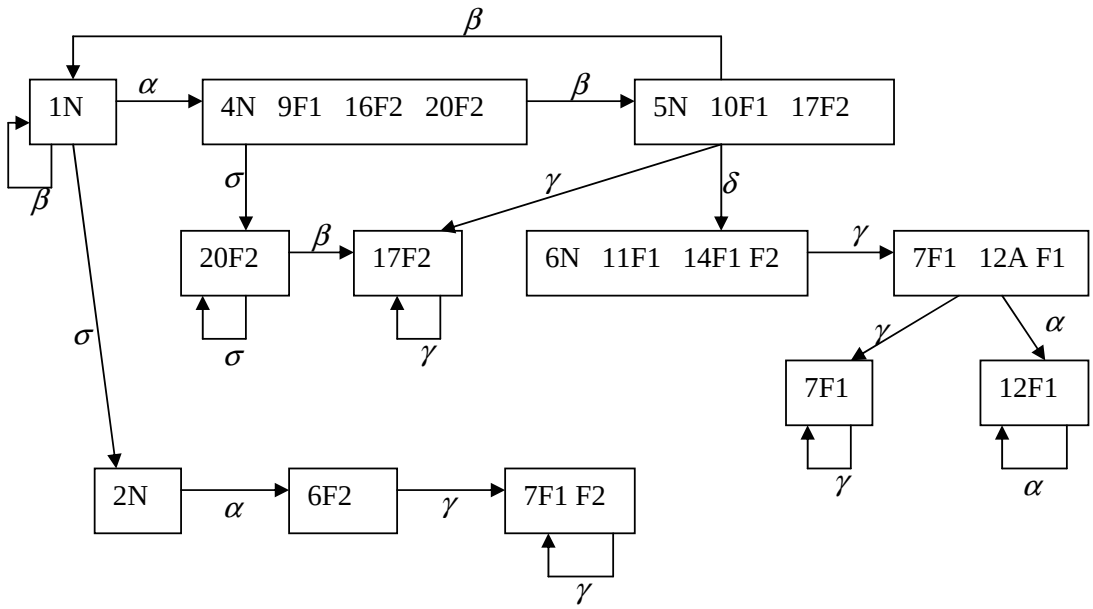
$$(x_1, l_1), (x_2, l_2) \in q \iff \exists s_1, s_2 \in L$$

tale che

$$s_1 = s_1' \sigma_{1_f},\ s_2 = s_2' \sigma_{2_f},\ \sigma_{1_f}, \sigma_{2_f} \in \Sigma_o,\ \delta(x_0, s_1) = x_1,\ \delta(x_0, s_2) = x_2$$



Il sistema G



Il diagnosticatore G_d

Figura 4.2: Esempio che illustra la costruzione del diagnosticatore G_d

e

$$P(s_1) = P(s_2).$$

P3) Consideriamo $q_1, q_2 \in Q_d$ e $s \in \Sigma^*$ tale che $(x_1, l_1) \in q_1, (x_2, l_2) \in q_2, \delta(x_1, s) = x_2$ e $\delta_d[q_1, P(s)] = q_2$. Allora:

$$(F_i \notin l_2) \wedge (A \notin l_2) \Longrightarrow F_i \notin l_1.$$

La proprietà P3) semplicemente enuncia che le etichette di guasto F_i sono propagate da stato a stato, salvo che siano rimpiazzate dall'etichetta A come conseguenza della funzione di correzione delle etichette LC.
Inoltre se lungo una traccia s appartenente a L, uno stato x porta l'etichetta N, allora anche i suoi predecessori porteranno la stessa etichetta, a meno che non venga rimpiazzata dall'etichetta A.

Definizione 4.4.1. *Uno stato q appartenente a Q_d è detto*

1. F_i-certo *se* $\forall (x, l) \in q, F_i \in l$.
2. F_i-incerto *se* $\exists (x, l), (y, l') \in q$, *tale che* $F_i \in l$ *e* $F_i \notin l'$.
3. ambiguo *se* $\exists (x, l) \in q$, *tale che* $A \in l$.

Notiamo che nella definizione di sopra di stato F_i-incerto, $x \neq y$ per la proprietà P1). Notiamo inoltre che se lo stato q non è F_i-incerto, ciò non implica necessariamente che q sia F_i-certo, dal momento che uno stato $q \in Q_d$ tale che $\forall (x, l) \in q, F_i \notin l$ non è nè F_i-certo, né F_i-incerto. Il seguente risultato è una diretta conseguenza della costruzione del diagnosticatore.

Lemma 4.4.1. *Valgono i seguenti risultati:*

1. *Dato $\delta_d(q_0, u) = q$, se q è F_i-certo allora* $[\forall \omega \in P_L^{-1}(u)]\ \Sigma_{fi} \in \omega$.
2. *Se uno stato $q \in Q_d$ è F_i-incerto, allora $\exists s_1, s_2 \in L$ tale che $\Sigma_{fi} \in s_1$, $\Sigma_{fi} \notin s_2$, $P(s_1) = P(s_2)$, $\delta_d(q_0, P(s_1)) = q$ e $\delta(x_0, s_1) \neq \delta(x_0, s_2)$.*
3. *Se uno stato $q \in Q_d$ è ambiguo, allora $\exists s_1, s_2 \in L$ e $\exists i \in \Pi_f$ tale che $\Sigma_{fi} \in s_1$, $\Sigma_{fi} \notin s_2$, $P(s_1) = P(s_2)$, $\delta_d(q_0, P(s_1)) = q$ e $\delta(x_0, s_1) = \delta(x_0, s_2)$.*

Dalla definizione di uno stato F_i-certo e dal lemma 4.4.1, è ovvio che se lo stato corrente del diagnosticatore è F_i-certo, allora possiamo concludere che si è verificato un guasto di tipo F_i, senza tenere conto di quale sia lo stato attuale di G. Questa è precisamente il tipo di diagnosi che viene trattata in questa tesi. D'altra parte la presenza di uno stato F_i-incerto in G_d, corrisponde alla situazione dove ci sono due tracce s_1 e s_2 in L tali che s_1 contiene un guasto di tipo F_i, mentre s_2 no e in più, le tracce s_1 e s_2 producono lo stesso insieme di eventi osservabili. Ogni volta che il diagnosticatore raggiunge uno stato F_i-incerto, concludiamo che un guasto di tipo F_i è accaduto, ma non è possibile stabilire in quale punto della sequenza di eventi osservati il guasto sia davvero accaduto.
Infine la presenza di uno stato ambiguo in G_d corrisponde alla situazione in cui ci sono due tracce s_1 e s_2 in L tali che l'insieme di tutti i possibili prolungamenti di s_1 in L sono gli stessi di s_2; s_1 contiene un evento di guasto di tipo F_i, mentre s_2 no, e in più s_1 e s_2 producono lo stesso insieme di eventi osservabili. D'ora in poi chiameremo tali sequenze, tracce F_i-*ambigue*.

Definizione 4.4.2. *Un insieme di stati $x_1, x_2, ..., x_n \in X$ forma un ciclo in G se $\exists s \in L(G, x_1)$ tale che $s = \sigma_1\sigma_2...\sigma_n$ e $\delta(x_l, \sigma_l) = x_{(l+1)modn}$, $l = 1, 2, ...n$.*

La seguente definizione di un ciclo F_i-indeterminato è basata sull'esame dei cicli in G_d e in G'.

Definizione 4.4.3. *Un insieme di stati F_i-incerti $q_1, q_2, ..., q_n \in Q_d$ forma un ciclo F_i-indeterminato se:*

1. *$q_1, q_2, ..., q_n$ formano un ciclo in G_d con $\delta_d(q_l, \sigma_l) = q_{l+1}$, $l = 1, 2, ..., n-1$, $\delta_d(q_n, \sigma_n) = q_1$, dove $\sigma_l \in \Sigma_o$, $l = 1, 2, ..., n$, e*

2. *$\exists (x_l^k, l_l^k), (y_l^r, \widetilde{l_l^r}) \in q_l$, $l = 1, ..., n$, $k = 1, ..., m$ e $r = 1, ..., m'$ tale che:*

   (a) *$F_i \in l_l^k$, $F_i \notin \widetilde{l_l^r}$ per tutti gli l, k, r;*

   (b) *Le sequenze di stati $\{x_l^k\}$, con $l = 1, ..., n$, $k = 1, ...m$ e $\{y_l^r\}$, con $l = 1, ..., n$, $r = 1, ...m'$ formano cicli in G' con:*

$$(x_l^k, \sigma_l, x_{(l+1)}^k) \in \delta_{G'}, \quad l = 1, ..., n-1, \quad k = 1, ..., m$$

$$(x_n^k, \sigma_n, x_1^{(k+1)}) \in \delta_{G'}, \quad k = 1, ..., m-1$$

e

$$(x_n^m, \sigma_n, x_l^1) \in \delta_{G'}$$

e

$$(y_l^r, \sigma_l, y_{(l+1)}^r) \in \delta_{G'}, \quad l = 1, ..., n-1, \quad r = 1, ..., m'$$

$$(y_n^r, \sigma_n, y_1^{(r+1)}) \in \delta_{G'}, \quad r = 1, ..., m'-1$$

e

$$(y_n^{m'}, \sigma_n, y_1^1) \in \delta_{G'}$$

In altre parole, un ciclo F_i-indeterminato in G_d è un ciclo composto esclusivamente di stati F_i-incerti per i quali esiste:

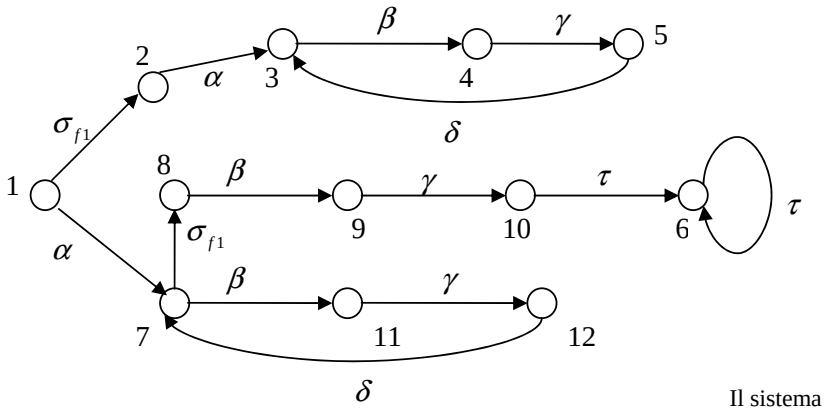
1. un corrispondente ciclo (di eventi osservabili) in G' che coinvolge solo stati che contengono F_i nelle loro etichette nel ciclo in G_d (questa è la sequenza $\{x_l^k\}$) e
2. un corrispondente ciclo (di eventi osservabili) in G' che coinvolge solo stati che non contengono F_i nelle loro etichette nel ciclo in G_d (questa è la sequenza $\{y_l^r\}$).

Osserviamo che nella definizione data prima, m ed m' indicano il numero di volte che il ciclo $q_1, q_2, ..., q_n$ in G_d è completato prima che il ciclo in G' sia completato, ad esempio nm ed nm' sono le lunghezze dei cicli in G' per $\{x_l^k\}$ e $\{y_l^r\}$, rispettivamente. Un ciclo F_i-indeterminato in G_d indica la presenza in L di due tracce s_1 ed s_2 di arbitraria lunghezza, tali da avere la stessa proiezione osservabile, ed s_1 contiene un evento di guasto dell'insieme Σ_{fi}, mentre s_2 no.
La nozione di un ciclo F_i-indeterminato è l'elemento più cruciale nello sviluppo delle condizioni necessarie e sufficienti per la diagnosticabilità.

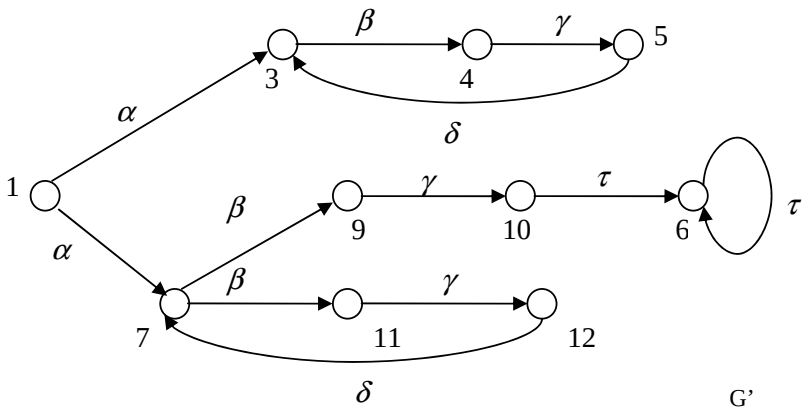
Chiariamo ora il concetto di cicli indeterminati con un esempio:

Esempio 4.4.1. *La figura 4.3 mostra un sistema e il suo diagnosticatore. Consideriamo $\Sigma_f = \Sigma_{f1} = \sigma_{f1}$. Il diagnosticatore ha un ciclo di stati F_1-incerti, corrispondente alla sequenza di eventi $\beta\gamma\delta$. Corrispondente a questo ciclo nel diagnosticatore, troviamo due cicli nell'automa G': il primo coinvolge gli stati $3, 4,$ e 5 che compaiono con l'etichetta F_1 nel ciclo del diagnosticatore, mentre il secondo coinvolge gli stati $7, 11,$ e 12 che portano l'etichetta N nel ciclo del diagnosticatore. Quindi il ciclo in G_d è un ciclo F_1-indeterminato con $m = m' = 1$, $x_1^1 = 3$, $x_2^1 = 4$, $x_3^1 = 5$, e $y_1^1 = 7$, $y_2^1 = 11$, $y_3^1 = 12$.*

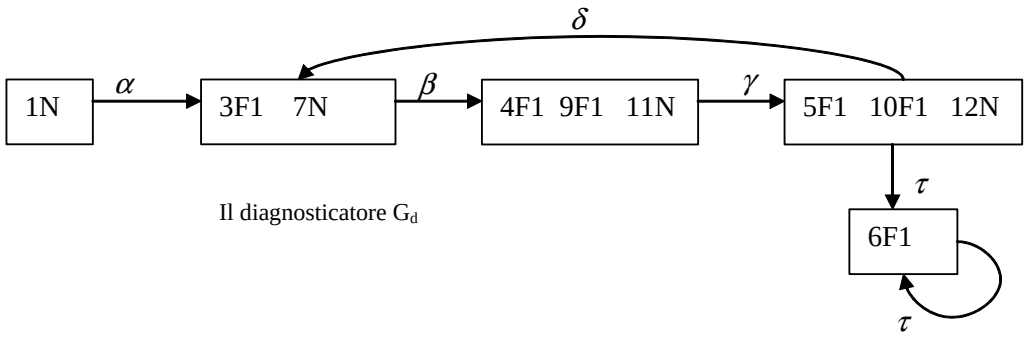
Esempio 4.4.2. *La figura 4.4 mostra un sistema e il suo diagnosticatore. Consideriamo $\Sigma_f = \Sigma_{f1} = \sigma_{f1}$. Il diagnosticatore ha un ciclo di stati F_1-incerti. Infatti si nota che i due diagnosticatori in figura 4.3 e 4.4 sono identici. In questo*



Il sistema G

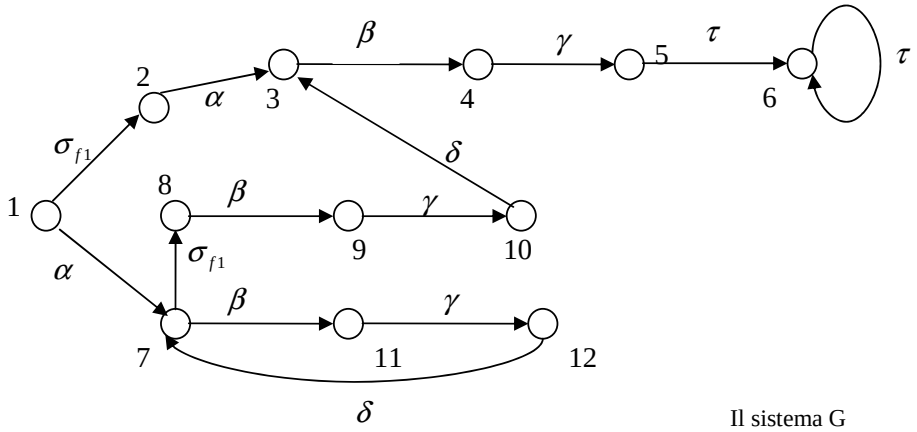


G'

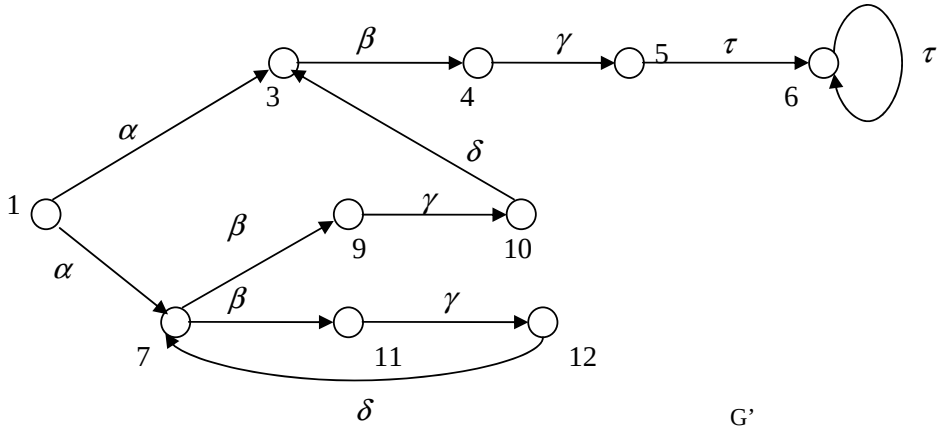


Il diagnosticatore G_d

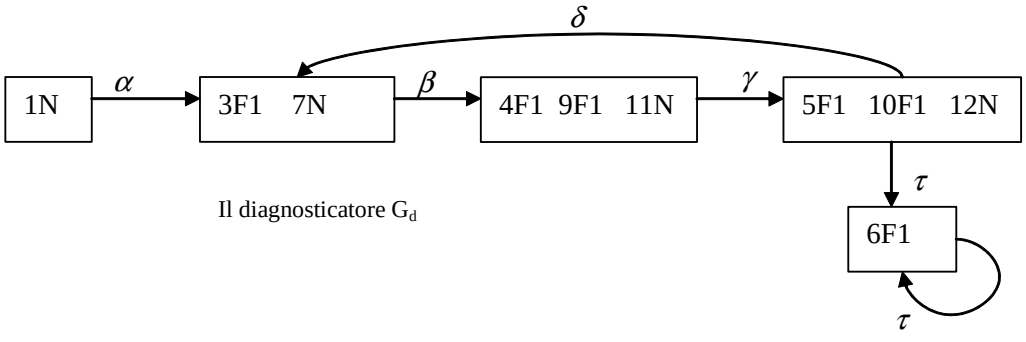
Figura 4.3: Esempio di un sistema con un ciclo F_1-indeterminato nel diagnosticatore G_d



Il sistema G



G'



Il diagnosticatore G_d

Figura 4.4: Esempio di un sistema con un ciclo di stati F_1-incerti (ma non F_1-indeterminati) nel diagnosticatore G_d

caso però il ciclo di stati F_1-incerti non è F_1-indeterminato, in quanto agli stati che portano l'etichetta F_1 nel ciclo in G_d, non corrisponde alcun ciclo in G' che li coinvolge; gli stati a cui ci stiamo riferendo sono: $3, 4, 5, 9$ *e* 10.

Siamo ora pronti per enunciare le condizioni necessarie e sufficienti.

Condizioni necessarie e sufficienti per la diagnosticabilità:

Teorema 4.4.1. *Un linguaggio L, senza guasti multipli dello stesso tipo, è diagnosticabile se e solo se il suo diagnosticatore G_d soddisfa le seguenti due condizioni:*

C1) Non ci sono cicli indeterminati in G_d, per tutti i tipi di guasto F_i.

C2) Nessuno stato $q \in Q_d$ è ambiguo.

Le condizioni C1) e C2), insieme con l'assunzione che L sia vivo, implicano che se L è diagnosticabile, allora ogni stato F_i-incerto conduce a uno stato F_i-certo in un numero finito di transizioni del diagnosticatore.

4.4.2 Condizioni per la $I-$diagnosticabilità

Studiamo ora le condizioni necessarie e sufficienti affinchè un linguaggio sia $I-$diagnosticabile. Ricordiamo che nel caso di I-diagnosticabilità siamo interessati a scoprire i guasti solo dopo che si sono verificati i corrispondenti eventi indicatori, ossia richiediamo che la condizione di diagnosticabilità D sia verificata solo per quelle tracce nelle quali un evento indicatore segue un guasto. Sulla base di quanto detto introduciamo le seguenti modifiche al diagnosticatore di base G_d per ottenere G_d^I.

Il diagnosticatore G_d^I

Definiamo, come prima, l'insieme delle etichette di guasto $\Delta_f = \{F_1, F_2, ..., F_m\}$ dove $|\Pi_f| = m$. In aggiunta, definiamo un insieme di etichette di indicatori $\Delta_i = \{I_1, I_2, ..., I_m\}$. Interpretiamo la scrittura $\{I_{i1}, ..., I_{ik}\}$ col significato che gli eventi indicatori del tipo I_{i1}, fino al tipo I_{ik} sono accaduti. L'insieme completo delle possibili etichette è definito come:

$$\Delta^I = \{N\} \cup 2^{\Delta_f \cup \Delta_i}$$

con il vincolo che

$$(\forall l \in \Delta^I) \quad I_i \in l \Longrightarrow F_i \in l$$

(verrà spiegato nei prossimi paragrafi).

Il diagnosticatore modificato G_d^I è l'automa a stati finiti:

$$G_d^I = (Q_d^I, \Sigma_o, \delta_d^I, q_0)$$

con lo stato iniziale $q_0 = \{(x_0, \{N\})\}$, come definito prima. La funzione di propagazione delle etichette LP^I, la funzione di range R, la funzione di correzione delle etichette LC^I, la funzione di transizione δ_d^I, e lo spazio di stato Q_d^I di G_d^I sono definiti come segue.

Definizione 4.4.4. *La funzione di propagazione delle etichette $LP^I : X_o \times \Delta^I \times \Sigma^* \longrightarrow \Delta^I$. Dati $x_0 \in X_o$, $l \in \Delta^I$ e $s \in L_o(G, x)$, LP^I propaga l'etichetta l lungo s, partendo da x e seguendo le dinamiche di G, ossia in accordo con $L(G, x)$. La funzione LP è definita come segue:*

$$LP^I(x, l, s) =$$

$$\begin{cases} \{N\}, & \text{se } l = \{N\} \wedge \forall i[\Sigma_{fi} \notin s] \text{ ;} \\ \{F_i : F_i \in l \vee \Sigma_{fi} \in s\} \cup & \\ \{I_i : I_i \in l \vee [I(\Sigma_{fi}) \in s \wedge (F_i \in l \vee \Sigma_{fi} \in s)]\}, & \text{altrimenti .} \end{cases}$$

La figura 4.5 mostra la propagazione delle etichette in accordo con la funzione di propagazione delle etichette LP^I definita sopra. In questa figura abbiamo indicato con σ_{f1} un guasto di tipo F_1 e con σ_{I1} un evento indicatore di tipo I_1.

Definizione 4.4.5. *La funzione di range $R : Q_o \times \Sigma_o \longrightarrow Q_o$ è definita come segue:*

$$R(q, \sigma) = \cup_{(x,l) \in q} \cup_{s \in L_\sigma(G,x)} \{(\delta(x, s), LP^I(x, l, s))\}.$$

Definizione 4.4.6. *La funzione di correzione delle etichette $LC^I : Q_o \longrightarrow Q_o$ è definita come segue:*

$$LC^I(q) = q - \{(x, l) \in q : (x, l') \in q \wedge \forall i[F_i \in l \Longleftrightarrow F_i \in l'] \wedge [l \subset l']\}.$$

L'uso della funzione di correzione delle etichette LC^I è così spiegata. Supponiamo che esistano due coppie (x, l) e (x, l') come descritto sopra, appartenenti a $R(q, \sigma)$ per un dato stato q di G_d^I. Questo implica la presenza in L di due tracce s_1 ed s_2 tali che esse hanno le stesse proiezioni e conducono allo stesso stato x, ma

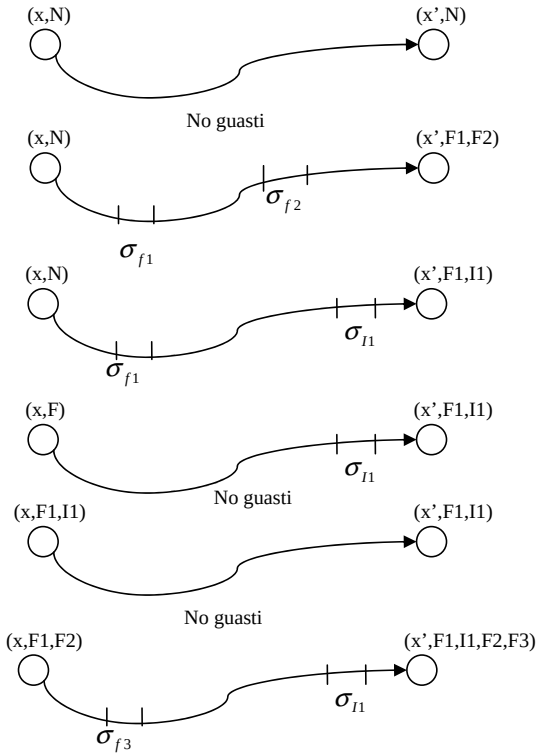


Figura 4.5: Propagazione delle etichette lungo una traccia di L

s_1 contiene un evento indicatore di tipo I_i seguito da un guasto di tipo F_i, mentre s_2 no.
Dal momento che per la I-diagnosticabilità siamo interessati solo a quelle tracce nelle quali l'evento indicatore segue il guasto, possiamo lasciar perdere le coppie (x, l) che non contengono l'etichetta I_i, senza perdita di generalità.

La funzione transizione $\delta_d^I : Q_o \times \Sigma_o \longrightarrow Q_o$ è definita come:

$$q_2 = \delta_d^I(q_1, \sigma) \Longleftrightarrow q_2 = LC^I[R(q_1, \sigma)]$$

con $\sigma \in e_d(q_1)$ dove :

$$e_d(q_1) = \cup_{(x,l)\in q_1}\{P(s) : s \in L_o(G, x)\}.$$

Lo spazio di stato Q_d^I è il risultante sottoinsieme di Q_o composto dagli stati del diagnosticatore che sono raggiungibili da q_0 sottostando alla funzione di transizione δ_d^I. Uno stato q_d di G_d^I assume ora la forma :

$$q_d = \{(x_1, l_1), ..., (x_n, l_n)\}$$

dove $x_i \in X_o$ e $l_i \in \Delta^I$, ossia l_i assume la forma $l_i = \{N\}$ o $l_i = \{F_{i1}, F_{i2}, ..., F_{ik}$, $I_{j1}, I_{j2}, ...I_{jl}\}$ dove $\{i_1, i_2, ..., i_k\} \subseteq \{1, 2, ..., m\}$ e $\{j_1, j_2, ..., j_l\} \subseteq \{i_1, i_2, ..., i_k\}$.

Facciamo le seguenti osservazioni sul diagnosticatore modificato G_d^I:

1. Oltre all' informazione sul guasto, le etichette ora contengono l'informazione sull'accadimento degli eventi indicatori che seguono i guasti.

2. Attacchiamo l'etichetta I_i a ogni l, solo se un evento indicatore appartenente all'insieme $I(\Sigma_{fi})$ segue un guasto appartenente all'insieme Σ_{fi}. L'insieme delle etichette I_i è sempre un sottoinsieme dell'insieme delle etichette F_i per qualunque coppia $(x, l) \in q \in Q_d$.

3. Le etichette I_i si propagano di stato in stato esattamente come le etichette F_i.

4. In questo caso non usiamo l'etichetta A. Come detto prima, siamo concentrati solo su quelle tracce in cui il guasto è seguito dal corrispondente evento indicatore. Perciò, potrebbero essere presenti in L due tracce F_i-ambigue per un certo $i \in \Pi_f$ e tuttavia L potrebbe essere diagnosticabile se nessuna traccia, nel prolungamento di queste tracce, contiene un evento indicatore dell'insieme $I(\Sigma_{fi})$. Perciò, per verificare la I-diagnosticabilità, abbiamo bisogno di ricordare quali tipi di guasti causano l'ambiguità, sempre nel caso di guasti singoli dello stesso tipo.

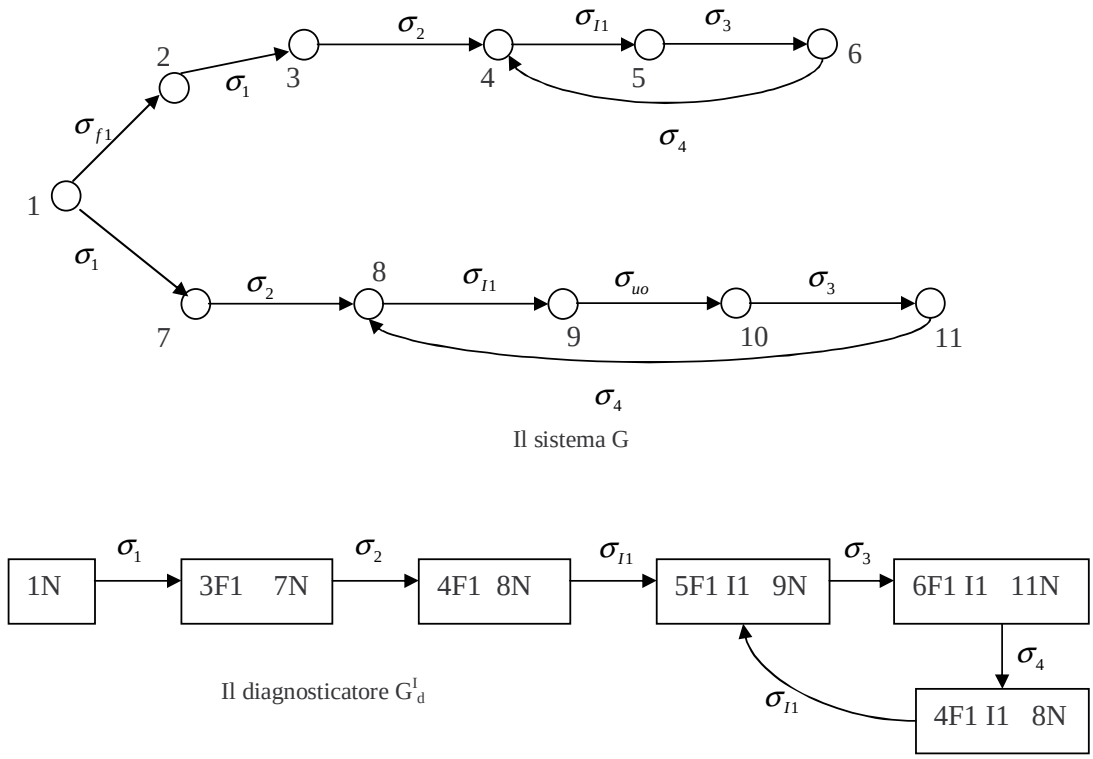


Figura 4.6: Costruzione del diagnosticatore G_d^I

Esempio 4.4.3. *La figura 4.6 illustra la costruzione del diagnostictore G_d^I. In questo caso σ_i, con $i \in \{1, ..., 4\}$, e σ_{I1} sono eventi osservabili, mentre σ_u è non osservabile. L'evento indicatore corrispondente al guasto σ_{f1} è $I(\sigma_{f1}) = \{\sigma_{I1}\}$ e la partizione è $\Sigma_{f1} = \{\sigma_{f1}\}$.*

4.4.3 Proprietà e definizioni di G_d^I

In questo caso non esiste l'etichetta A di stato ambiguo. Le proprietà P1) e P3) cambiano:

P1) Potrebbe esistere uno stato $q \in Q_d$ tale che $(x, l), (x, l') \in q$ con $l \neq l'$.

P3) Consideriamo $q_1, q_2 \in Q_d$ ed $s \in \Sigma^*$ tale che $(x_1, l_1) \in q_1, \quad (x_2, l_2) \in q_2$, $\delta(x_1, s) = x_2, \quad e \quad \delta_d[q_1, P(s)] = q_2$. Allora

$$F_i \notin l_2 \Longrightarrow F_i \notin l_1.$$

La proprietà P3) dice semplicemente che le etichette di guasto F_i si propagano da uno stato all'altro e se lungo una traccia L di s uno stato x riceve un'etichetta F_i, allora ogni successore x' di x avrà la stessa etichetta F_i.

La definizione di stati F_i-certi rimane la stessa che abbiamo definito per la diagnosticabilità, mentre la definizione di stato ambiguo non ci interessa. Anche le definizioni di stati F_i-incerti e cicli F_i-indeterminati non variano, mentre il lemma diventa:

Lemma 4.4.2. *1. Consideriamo $\delta_d(q_0, u) = q$. Se q è F_i-certo, allora $(\forall \omega \in P_L^{-1}(u)) \quad \Sigma_{fi} \in \omega$*

2. Se uno stato $q \in q_d$ è F_i-incerto, allora questo implica che $\exists s_1, s_2 \in L$ tale che $\Sigma_{fi} \in s_1, \quad \Sigma_{fi} \notin s_2, \quad P(s_1) = P(s_2)$, e $\delta_d[q_0, P(s_1)] = q$.

Introduciamo ora le nozioni di stati (F_i, I_i)-incerti e di cicli (F_i, I_i)-indeterminati.

Definizione 4.4.7. *Uno stato $q \in Q_d^I$ è (F_i, I_i)-incerto se $\exists (x, l), (x, l') \in q$, tale che $\{F_i, I_i\} \subseteq l \quad e \quad F_i \notin l'$.*

Lemma 4.4.3. *Se uno stato $q \in Q_d^I$ è (F_i, I_i)-incerto, allora questo implica che $\exists s_1 = p_1 t_1 \in L$ e $s_2 \in L$ tale che: $p_1 \in \Psi(\Sigma_{fi}), \quad I(\Sigma_{fi}) \in t_1, \quad \Sigma_{fi} \notin s_2, \quad P(s_1) = P(s_2), \quad e \quad \delta_d^I[q_0, P(s_1)] = q$.*

Il lemma appena enunciato dice semplicemente che la presenza di stati (F_i, I_i)-incerti in G_d^I corrisponde alla situazione in cui ci sono due tracce s_1 ed s_2 in L tali che s_1 contiene un guasto di tipo F_i seguito da un evento indicatore corrispondente a questo tipo di guasto, mentre s_2 non contiene un guasto F_i. Inoltre, le tracce s_1 ed s_2 producono la stessa sequenza di eventi osservabili. La dimostrazione di questo lemma segue direttamente dalla costruzione del diagnosticatore G_d^I.

Definizione 4.4.8. *Un insieme di stati $q_1, q_2, ..., q_n \in Q_d^I$ (F_i, I_i)-incerti forma un ciclo (F_i, I_i)-indeterminato se:*

1. $q_1, q_2, ..., q_n$ *formano un ciclo in* G_d^I *con* $\delta_d^I(q_l, \sigma_l) = q_{(l+1)}, \quad l = 1, 2, ..., n-1, \quad$ *e* $\quad \delta_d^I(q_n, \sigma_n) = q_1, \quad$ *dove* $\quad \sigma_l \in \Sigma_o, \quad l = 1, 2, ..., n, \quad$ *e*

2. $\exists (x_l^k, l_l^k), (y_l^r, \widetilde{l_l^r}) \in q_l, \quad l = 1, ..., n, \quad k = 1, ..., m \quad$ *e* $\quad r = 1, ..., m'$ *(x non necessariamente è distinto da y) tale che:*

   (a) $\{F_i, I_i\} \subseteq l_l^k, \quad F_i \notin \widetilde{l_l^r} \quad$ *per tutti gli* l, k, r*;*

   (b) *Le sequenze di stati* $\{x_l^k\}, \quad$ *con* $\quad l = 1, ..., n, \quad k = 1, ...m \quad$ *e* $\{y_l^r\}, \quad$ *con* $\quad l = 1, ..., n, \quad r = 1, ...m'$ *formano cicli in* G' *con:*

$$(x_l^k, \sigma_l, x_{(l+1)}^k) \in \delta_{G'}, \quad l = 1, ..., n-1, \quad k = 1, ..., m$$

$$(x_n^k, \sigma_n, x_1^{(k+1)}) \in \delta_{G'}, \quad k = 1, ..., m-1$$

e

$$(x_n^m, \sigma_n, x_1^1) \in \delta_{G'}$$

e

$$(y_l^r, \sigma_l, y_{(l+1)}^r) \in \delta_{G'}, \quad l = 1, ..., n-1, \quad r = 1, ..., m'$$

$$(y_n^r, \sigma_n, y_1^{(r+1)}) \in \delta_{G'}, \quad r = 1, ..., m'-1$$

e

$$(y_n^{m'}, \sigma_n, y_1^1) \in \delta_{G'}.$$

Un ciclo (F_i, I_i)-indeterminato in G_d^I indica la presenza in L di due tracce s_1 ed s_2 di arbitraria lunghezza, tali che da avere la stessa proiezione osservabile, ed s_1 contiene un guasto appartenente all'insieme Σ_{fi} seguito da un evento indicatore dell'insieme $I(\Sigma_{fi})$, mentre s_2 non contiene eventi appartenenti all'insieme Σ_{fi}.

Consideriamo il sistema rappresentato in figura 4.6. Un'occhiata attenta al diagnosticatore G_d^I del sistema rivela la presenza di un ciclo (F_i, I_i)-indeterminato.

Qui l'insieme $\{x_l^k\}$ della definizione precedente è $\{5, 6, 4\}$ (questi stati portano l'etichetta $\{F_1, I_1\}$ nel diagnosticatore G_d^I), l'insieme $\{y_l^r\}$ è $\{9, 11, 8\}$ (questi stati portano l'etichetta $\{N\}$), ed $m = m' = 1$.

Condizioni necessarie e sufficienti per la I-diagnosticabilità:

Teorema 4.4.2. *Un linguaggio L è I-diagnosticabile se e solo se il diagnosticatore G_d^I soddisfa la seguente condizione:*

C1) Non ci sono cicli (F_i, I_i)-indeterminati in G_d^I, per qualunque tipo di guasto F_i.

Notiamo che la condizione C1) insieme con l'assunzione che L sia vivo implica che L sia I-diagnosticabile , e ogni stato (F_i, I_i)-incerto conduca a uno stato F_i-certo in un numero limitato di transizioni del diagnosticatore. Inoltre notiamo che nel caso della I-diagnosticabilità, non siamo interessati agli stati F_i-incerti e ai cicli F_i-indeterminati, che non sono anche (F_i, I_i)-incerti ed (F_i, I_i)-indeterminati, rispettivamente.

Esempio 4.4.4. *La figura 4.6 è un esempio di sistema che non è I-diagnosticabile dal momento che il corrispondente diagnosticatore G_d^I contiene un ciclo (F_i, I_i)-indeterminato.*

4.5 UMDES

UMDES (Discrete Event Systems Modeled By Finite State Machine) è un software che consente l'implementazione di molte delle operazioni descritte in questo capitolo. E' stato sviluppato dal *gruppo Discrete Event Systems* dell'università del Michigan sotto la coordinazione di Stephane Lafortune e Demosthenis Teneketzis.

UMDES contiene comandi per la manipolazione di una macchina a stati finiti (circa 30 comandi), per la diagnosi dei guasti (circa 10 comandi) e per il controllo supervisivo (circa 10 comandi).
Per quanto riguarda la parte riguardante la diagnosi dei guasti vi sono comandi che consentono la costruzione del modello, la costruzione dei diagnosticatori e la verifica della diagnosticabilità (cicli indeterminati). Facciamo qualche esempio di questi comandi:

- *write-st*: questo programma, dato in ingresso l'automa a stati finiti, scrive in un opportuno file d'uscita tutti gli stati dell'automa dato;

- *compose* : questo programma, dati in ingresso l'automa a stati finiti, l'elenco degli eventi non osservabili dell'automa e la mappa dei sensori, genera un sistema composto, con le indicazioni dei sensori per ogni stato;
- *diag* : questo programma, dati in ingresso l'automa a stati finiti e la partizione dei guasti, genera in uscita il diagnosticatore per il sistema dato;
- *dcycle* : questo programma verifica la diagnosticabilità del sistema. Infatti, dati in ingresso l'automa a stati finiti e la partizione dei guasti, trova i cicli F_i-indeterminati nel diagnosticatore e li scrive in un opportuno file d'uscita.

Questo software e la libreria dei comandi (UMDES-LIB) è disponibile sul sito web `www.eecs.umich.edu/umdes`.

4.6 Modifiche all'approccio standard

Esiste una leggera modifica tra l'approccio presentato in [1] e l'approccio presentato in [7]. Si consideri un semplice esempio riportato in figura 4.7. Costruiamo il diagnosticatore con l'approccio standard in figura 4.8 e il diagnosticatore con l'approccio modificato in figura 4.9. La differenza è che dopo che osserviamo l'evento a nel diagnosticatore di fig.4.7 assumiamo che non si sia verificato alcun guasto, mentre nel diagnosticatore di fig.4.9 consideriamo la possibilità di essere nello stato x_2 a causa del verificarsi del guasto. La costruzione del secondo diagnosticatore è illustrata in [7]. Queste due tecniche sono equivalenti e tutte le proprietà che valgono per l'approccio standard valgono per quello modificato come spiegato in [8].

Il software UMDES utilizza l'approccio standard per la costruzione del diagnosticatore, mentre con il software **DESUMA**, sviluppato recentemente dal gruppo Discrete Event Systems, è possibile costruire il diagnosticatore con entrambe le tecniche.

D'ora in poi faremo riferimento per la costruzione del diagnosticatore all'approccio modificato.

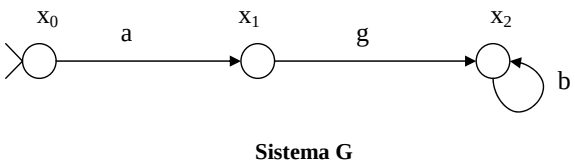


Figura 4.7: Esempio di un sistema semplice

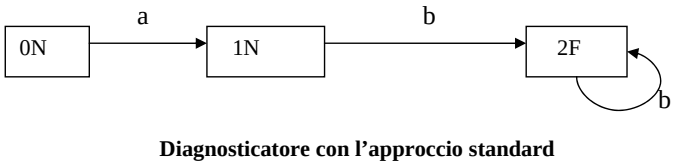


Figura 4.8: Costruzione del diagnosticatore con l'approccio standard

Figura 4.9: Costruzione del diagnosticatore con l'approccio modificato

Capitolo 5

Diagnosi mediante reti di Petri

5.1 Introduzione

In questo capitolo si presenta un efficiente approccio per la diagnosi dei guasti che utilizza le reti di Petri. Tale approccio è stato proposto da Alessandro Giua e Carla Seatzu in [2].

Altri approcci per la diagnosi dei sistemi ad eventi discreti basati sulle reti di Petri sono stati presentati anche in [4], [5], [6].

La diagnosi dei sistemi ad eventi discreti è un'area di ricerca che ha ricevuto moltissima attenzione negli ultimi anni a causa del bisogno pratico di assicurare un sicuro e corretto funzionamento dei sistemi molto complessi. Il modello reti di Petri è stato spesso usato in questo contesto: la natura intrinsecamente distribuita delle reti di Petri, dove la nozione di *stato* (marcatura) e *azione* (transizione) è locale, è stata spesso una risorsa per ridurre la complessità computazionale insita nella risoluzione dei problemi di diagnosi.
Nel seguito, così come proposto in [2], affrontiamo la diagnosi dei guasti dei sistemi ad eventi discreti modellati dalle reti posto/transizione. In particolare, assumiamo che i guasti siano modellati da transizioni non osservabili, ma che possano anche esistere altre transizioni che rappresentano un comportamento legale, ma che sono ugualmente non osservabili. Quindi assumiamo che l'insieme delle transizioni possa essere diviso in $T = T_o \cup T_u$, dove T_o è l'insieme delle transizioni osservabili, e T_u è l'insieme delle transizioni non osservabili. L'insieme delle transizioni di guasto è indicato con T_f e vale la relazione $T_f \subseteq T_u$.

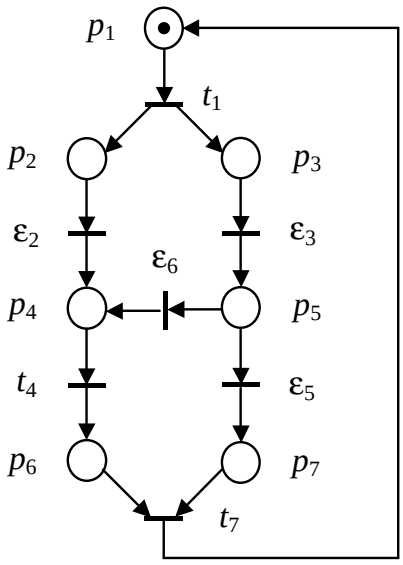


Figura 5.1: Una rete di Petri che descrive un sistema di comunicazione.

Come esempio consideriamo la rete in figura 5.1:

L'insieme delle transizioni osservabili è $T_o = \{t_1, t_4, t_7\}$. L'insieme delle transizioni non osservabili è $T_u = \{t_2, t_3, t_5, t_6\}$ dove, per essere più chiari abbiamo indicato ogni transizione non osservabile t_i con ε_i. L'unica transizione di guasto è la transizione t_6. Questa rete modella un sistema di comunicazione: i messaggi pronti per essere inviati sono divisi in due pacchetti (transizione t_1) per essere mandati su due canali separati (posto p_4 e p_5). I due pacchetti sono infine uniti e un avviso di ricevimento è mandato al mittente (transizione t_7).
Un guasto si verifica quando un pacchetto che dovrebbe viaggiare sul secondo canale è erroneamente spedito sul primo canale (transizione t_6). Come si può notare, la transizione t_6 non è osservabile, ma esistono anche altre transizioni non osservabili.
Ora, indichiamo con $\mathcal{C}(w)$ l'insieme delle marcature consistenti con una sequenza di scatto osservata $w \in T_o^*$. Per ogni sequenza osservata è possibile determinare una *marcatura di base* $M_{b,w}$, mentre l'insieme delle marcature nelle quali il sistema potrebbe essere è :

$$\mathcal{C}(w) = \{M \in \mathbb{N}^m \mid M_{b,w}[\sigma\rangle M, \sigma \in T_u^*\}$$

che rappresenta l'insieme di tutte le marcature raggiungibili dalla marcatura di base, con lo scatto di una sequenza di transizioni non osservabili.
Nel prossimo paragrafo introdurremo il concetto di *spiegazione minima*. Il termine spiegazione minima è usato per indicare la più piccola sequenza di tran-

sizioni non osservabili che possono essere scattate per dar luogo a una data osservazione. Per esempio, consideriamo nella rete in figura 5.1 la marcatura iniziale che assegna ai posti p_2 e p_3 un gettone, mentre gli altri posti sono vuoti. Se viene osservato lo scatto di t_4 allora il gettone richiesto per abilitare questa transizione potrebbe essere stato messo in p_4 o con lo scatto di ε_2 o con lo scatto di $\varepsilon_3\varepsilon_6$.

Nel seguito presenteremo dapprima un algoritmo per il calcolo delle spiegazioni minime. Secondariamente, presenteremo un'originale tecnica per costruire un osservatore per reti limitate. In particolare, definiamo per ogni osservazione w un insieme $\mathcal{M}(w)$ composto dalla coppia (M, y) dove: M è una marcatura di base corrispondente a w; y, che indichiamo come la sua *giustificazione*, è il vettore di scatto delle transizioni non osservabili che può essere scattato per raggiungere M.

Presenteremo inoltre un secondo algoritmo per la costruzione dell'*albero base di raggiungibilità* (BRT: basis reachability tree). Questo è un automa deterministico i cui archi sono etichettati con le transizioni osservabili, mentre un nodo raggiungibile dalla radice con una sequenza di scatto w è etichettato con l'insieme $\mathcal{M}(w)$. L'aspetto importante di questo approccio è che il BRT offre un'ottima caratterizzazione dell'insieme di raggiungibilità e del linguaggio della rete originale: l'insieme della marcature consistenti con un'osservazione w può essere determinata calcolando le marcature raggiungibili nella sottorete non osservabile partendo da ognuna delle marcature di base dell'insieme $\mathcal{M}(w)$. Se assumiamo che la sottorete non osservabile sia aciclica, questo calcolo può essere fatto risolvendo l'equazione di stato, mentre nella costruzione del BRT dobbiamo solo enumerare l'insieme delle marcature di base.
Nell'ultimo paragrafo di questo capitolo, applicheremo il BRT al problema della diagnosi dei guasti. In particolare lo useremo on-line per associare a ogni osservazione una diagnosi. E' anche possibile usare il BRT off-line per studiare le differenti proprietà di diagnosticabilità e determinare se in un dato sistema il verificarsi di un guasto è riconoscibile, ma questa parte non è trattata in questa tesi.

5.2 Spiegazioni Minime

In questa sezione daremo alcune definizioni base che ci saranno utili nel seguito.

Definizione 5.2.1. *Data una marcatura M e una transizione osservabile $t \in T_o$, definiamo*

$$\Sigma(M, t) = \{\sigma \in T_u^* \mid M[\sigma\rangle M',\ M' \geq Pre(\cdot, t)\}$$

l'insieme delle spiegazioni di t data una marcatura M, e con

$$Y(M,t) = \{y \in \mathbb{N}^n \mid \exists \sigma \in \Sigma(M,t) \ : \ \pi(\sigma) = y\}$$

il corrispondente insieme di vettori di scatto. Cioè $\Sigma(M,t)$ è l'insieme delle sequenze non osservabili il cui scatto, partendo dalla marcatura M, è necessario ad abilitare la transizione t.

Fra le sequenze definite sopra, vogliamo selezionare quelle il cui vettore di scatto è minimo, che chiamiamo **spiegazioni minime**.

Definizione 5.2.2. *Data una marcatura M e una transizione $t \in T_o$, definiamo:*

$$\Sigma_{\min}(M,t) = \{\sigma \in \Sigma(M,t) \mid \ y = \pi(\sigma), \nexists \sigma' \in \Sigma(M,t) : \pi(\sigma') \lneqq y\}$$

l'insieme delle spiegazioni minime di t data una marcatura M, e

$$Y_{\min}(M,t) = \{y \in \mathbb{N}^n \mid \exists \sigma \in \Sigma_{\min}(M,t) \ : \ \pi(\sigma) = y\}$$

il corrispondente insieme dei vettori di scatto.

Facciamo un esempio per chiarire meglio questi due concetti.

Esempio 5.2.1. *Consideriamo la rete in figura 5.1. Indichiamo con M_0 la marcatura mostrata in figura. Allora, $\Sigma(M_0,t_1) = \{\varepsilon\}$, vale a dire la parola vuota, e $Y_{\min}(M_0,t_1) = \{\vec{0}\}$. Infatti t_1 è già abilitata da M_0 e non è necessario lo scatto di nessuna transizione non osservabile per abilitarla.*
Se consideriamo la transizione t_7, allora $\Sigma(M_0,t_7) = \emptyset$, così anche $Y_{\min}(M_0,t_7) = \emptyset$. Infatti t_7 non è abilitata da M_0, e non può abilitarla nessuna sequenza di transizioni non osservabili.
Consideriamo ora $M_1 = [0\ 1\ 0\ 0\ 0\ 0\ 0]^T$, allora $\Sigma(M_1,t_4) = \Sigma_{\min}(M_1,t_4) = \{\varepsilon_2\}$.
Consideriamo ora $M_2 = [0\ 1\ 1\ 0\ 0\ 0\ 0]^T$, allora $\Sigma(M_2,t_4) = \Sigma_{\min}(M_2,t_4) = \{\varepsilon_2, \varepsilon_3\varepsilon_6\}$.
Consideriamo infine $M_3 = [0\ 1\ 1\ 0\ 1\ 0\ 0]^T$, allora $\Sigma(M_3,t_4) = \{\varepsilon_2, \varepsilon_6, \varepsilon_3\varepsilon_6\}$, mentre $\Sigma_{\min}(M_3,t_4) = \{\varepsilon_2, \varepsilon_6\}$.

Definizione 5.2.3. *Una rete è detta **backward conflict-free** se tutte le transizioni non hanno posti in uscita comuni.*

Ad esempio, in figura 5.1 la sottorete non osservabile non è backward conflict-free, perché il posto p_4 ha in ingresso due transizioni non osservabili ε_2 ed ε_6.

Una volta data la precedente definizione possiamo enunciare il seguente teorema dimostrato formalmente in [3].

Teorema 5.2.1. *Consideriamo una rete di Petri* $N = (P, T, Pre, Post)$ *con* $T = T_o \cup T_u$. *Se la sottorete* T_u*-indotta è* aciclica *e* backward conflict-free*, allora* $|Y_{\min}(M, t)| = 1$.

Presentiamo ora un algoritmo che trova tutti i vettori in $Y_{\min}(M, t)$, se applicato a reti la cui sottorete T_u-indotta è aciclica. Questo algoritmo richiede delle semplici manipolazioni algebriche ed è ispirato alla procedura per il calcolo dei P-invarianti minimi proposta da Martinez e Silva [9]. Notiamo che il seguente algoritmo può anche essere applicato alle sotto-reti T_u-indotte che non sono acicliche. Però, in questo caso, l'algoritmo potrebbe ciclare: per garantire che l'algoritmo termini in un numero finito di passi abbiamo bisogno di aggiungere il vincolo dell'aciclicità.

Algoritmo 5.2.1. *[Calcolo di* $Y_{\min}(M, t)$*]*

1. *Consideriamo la matrice* $\Gamma := \left|\begin{array}{c|c} C_u^T & I_{n_u \times n_u} \\ \hline A & B \end{array}\right|$
dove $A^T := M - Pre(\cdot, t)$, $B := \vec{0}_{n_u}^T$.
2. *Fintanto che* $NOT(A \geq 0)$*,*
2.a *Scegli un elemento* $A(i^*, j^*) < 0$.
2.b *Calcola l'insieme* $\mathcal{I}^+ = \{i \mid C_u^T(i, j^*) > 0\}$.
2.c *Se* $\mathcal{I}^+ \neq \emptyset$*,*
2.c.i *Scegli un* $i \in \mathcal{I}^+$*, aggiungi ad* $[A \mid B]$ *una nuova riga calcolata come:* $[A(i^*, \cdot) + C_u^T(i, \cdot) \mid B(i^*, \cdot) + \vec{e}_i^T]$ *dove* $\vec{e}_i$ *è l'*i*-esimo vettore della base canonica.*
2.c.ii *Elimina la riga* $[A(i^*, \cdot) \mid B(i^*, \cdot)]$ *dalla matrice*
altrimenti se $I^+ = \emptyset$
2.c.iii *elimina la riga* $[A(i^*, \cdot) \mid B(i^*, \cdot)]$ *dalla matrice.*
3. *Elimina da* B *ogni riga che ne copre un'altra.*
4. *Ogni riga di* B *è un vettore dell'insieme* $Y_{\min}(M, t)$.

Esempio 5.2.2. *Consideriamo ancora una volta la rete in figura5.1. Vogliamo calcolare l'insieme* $Y_{\min}(M, t)$*, dove* $M = [0\ 1\ 1\ 0\ 1\ 0\ 0]^T$ *e* $t = t_4$*. La matrice di*

incidenza C è la seguente:

$$C = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 \end{bmatrix}$$

La matrice C_u è composta dalle colonne di C corrispondenti alle transizioni non osservabili:

$$C_u = \begin{array}{c} \begin{matrix} \varepsilon_2 & \varepsilon_3 & \varepsilon_5 & \varepsilon_6 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Essendo:

$$Pre(\cdot, t_4) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T,$$

otteniamo che

$$A = (M - Pre(\cdot, t_4))^T = \begin{bmatrix} 0 & 1 & 1 & -1 & 1 & 0 & 0 \end{bmatrix}.$$

Quindi la matrice Γ è uguale a:

$$\Gamma := \left[\begin{array}{ccccccc|cccc} 0 & -1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

Esiste un solo elemento negativo di A, e precisamente l'elemento $A(1,4)$.
Scelto l'elemento di A negativo, calcoliamo l'insieme I^+, ossia andiamo a vedere

se esistono sulla colonna 4 elementi maggiori di zero. Questi elementi esistono e si trovano nella riga 1 e nella riga 4, quindi otteniamo $\mathcal{I}^+ = \{1, 4\}$.
Sempre usando l'algoritmo 5.2.1 aggiungiamo le seguenti due nuove righe a Γ*:*

$$\left|\begin{array}{ccccccc|cccc} 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right|$$

e

$$\left|\begin{array}{ccccccc|cccc} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right|$$

ottenute aggiungendo alla prima riga di A *la prima e la quarta riga di* Γ*, rispettivamente.*
Infine, rimuoviamo la riga $\Gamma(5, \cdot)$ *dalla matrice e ci fermiamo perche tutti gli elementi di* A *sono non negativi.*

Poichè nessuna riga copre le altre, concludiamo che entrambe le righe di B*, che sono*

$$\left|\begin{array}{cccc} 1 & 0 & 0 & 0 \end{array}\right|$$

e

$$\left|\begin{array}{cccc} 0 & 0 & 0 & 1 \end{array}\right|$$

sono elementi di $Y_{\min}(M, t)$.

Questo risultato è in accordo con l'esempio 5.2.1, essendo $\Sigma_{\min}(M, t) = \{\varepsilon_2, \varepsilon_6\}$.

5.3 Marcatura di Base

Definizione 5.3.1. *Consideriamo una rete* $\langle N, M_0 \rangle$ *la cui sottorete non osservabile è backward conflict-free. Data un'osservazione* w*, la* ***marcatura di base*** $M_{b,w}$ *è la marcatura raggiunta dalla marcatura* M_0 *con lo scatto di* w *e di tutte quelle transizioni non osservabili che sono strettamente necessarie ad abilitare* w*.*

L'assunzione di backward conflict-free assicura l'unicità della marcatura di base $M_{b,w}$ per qualunque marcatura iniziale M_0 e per qualunque osservazione w.
Se questa assunzione è rilassata, la marcatura di base potrebbe non essere unica. Questo segue facilmente dalla semplice osservazione che, data una marcatura M e una transizione osservabile t, l'insieme delle spiegazioni minime di t data la marcatura M non è composto da un solo elemento. Ora, con lo scopo di generalizzare la nozione di marcatura di base, introduciamo la seguente definizione ricorsiva.

Definizione 5.3.2. *Consideriamo una rete* $\langle N, M_0\rangle$ *dove* $N = (P, T, Pre, Post)$ *e* $T = T_o \cup T_u$. *Indichiamo con:* $\mathcal{M}(\varepsilon) = \{(M_0, \vec{0})\}$ *e per* $\forall\ w \in T_o^*$, $\forall\ t \in T_o$ *indichiamo:*

$$\begin{aligned}\tilde{\mathcal{M}}(wt) = \ & \{(M, y) \in \mathbb{N}^m \times \mathbb{N}^{n_u} \ \mid \ \exists\,(M', y') \in \mathcal{M}(w),\\ & \exists\, y'' \in Y_{\min}(M', t)\ :\ y = y' + y'',\ M = M_0 + C(\cdot, t) + C_u y\}.\end{aligned}$$

Infine, $\forall\ w \in T_o^*$, *indichiamo con* $\mathcal{M}(w) \subseteq \tilde{\mathcal{M}}(w)$ *tale che*

$$\mathcal{M}(w) = \{(M, y) \in \tilde{\mathcal{M}}(w) \ \mid \nexists\,(M', y') \in \tilde{M}(w)\ : y' \lneq y\}.$$

Tutte le marcature M *tali che* $(M, y) \in \mathcal{M}(w)$ *sono chiamate* marcature di base *e i vettori* y *sono le corrispondenti* giustificazioni.

Perciò, per ogni osservazione w, $(M, y) \in \mathcal{M}(w)$ è una coppia (marcatura, vettore di scatto) tale che M può essere raggiunta da M_0 con lo scatto di una sequenza σ, tale che $L(\sigma) = w$ e $\pi(\sigma) = \pi(w) + y$. Chiaramente, quando non avviene alcuna osservazione, cioè $w = \varepsilon$, $\mathcal{M}(w)$ è composto da un solo elemento e $M = M_0$, $y = \vec{0}$.
Notiamo che ogni insieme $\mathcal{M}(w)$ contiene solo coppie (M, y) le cui giustificazioni sono minime, poichè $\mathcal{M}(w)$ è ottenuta da $\tilde{\mathcal{M}}(w)$ togliendo tutte le coppie le cui giustificazioni non sono minime.

Esempio 5.3.1. *Consideriamo la rete in figura 5.1 e assumiamo che la marcatura iniziale sia quella mostrata in figura.*

Consideriamo $w = t_1$. *Essendo* $Y_{\min}(t_1, M_0) = \{\vec{0}\}$, *se indichiamo con*

$$M_1 \ = M_0 + C_o\pi(t_1) = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T,$$

allora $\mathcal{M}(t_1) = \tilde{\mathcal{M}}(t_1) = \{(M_1, \vec{0})\}$, *e il vettore nullo è l'unica giustificazione di* $w = t_1$ *data la marcatura iniziale.*

Assumiamo ora che venga osservata t_4, *quindi* $w = t_1t_4$. *In questo caso* $Y_{\min}(M_1, t_4) = \{y_1, y_2\}$ *dove* $y_1 = \pi(\varepsilon_2)$ *e* $y_2 = \pi(\varepsilon_3\varepsilon_6)$. *Ora se indichiamo con:*

$$\begin{aligned} M_2 \ &= M_1 + C_o\pi(t_4) + C_u y_1 = M_0 + C_o\pi(w) + C_u y_1 = \\ &= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}^T \end{aligned}$$

e con

$$\begin{aligned} M_3 \ &= M_1 + C_o\pi(t_4) + C_u y_2 = M_0 + C_o\pi(w) + C_u y_2 = \\ &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}^T, \end{aligned}$$

allora

$$\mathcal{M}(t_1t_4) = \tilde{\mathcal{M}}(t_1t_4) = \{(M_2, y_1), (M_3, y_2)\}.$$

Assumiamo infine che scatti t_7, quindi $w = t_1t_4t_7$. Si trova che $Y_{\min}(M_2, t_7) = \{\pi(\varepsilon_3\varepsilon_5)\}$ e $Y_{\min}(M_3, t_7) = \emptyset$. Infatti, partendo dalla marcatura M_2 lo scatto di $\varepsilon_3\varepsilon_5$ abilita t_7, mentre la transizione t_7 non è abilitata dalla marcatura M_3 e nessuna sequenza di transizioni osservabili potrebbe abilitarla. Perciò,

$$\mathcal{M}(t_1t_4t_7) = \tilde{\mathcal{M}}(t_1t_4t_7) = \{(M_4, y_1 + y_3)\},$$

dove

$$\begin{aligned} M_4 &= M_2 + C_o\pi(t_7) + C_u y_3 = M_0 + C_o\pi(w) + C_u(y_1 + y_3) \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T = M_0. \end{aligned}$$

Il seguente teorema dimostra che questo approccio basato sulle marcature di base, con alcune osservazioni, è capace di caratterizzare completamente, l'insieme di raggiungibilità.

Teorema 5.3.1. *Consideriamo un sistema di rete $\langle N, M_0\rangle$, la cui sottorete non osservabile è aciclica. Le due affermazioni che seguono sono equivalenti*

1. *Nella rete esiste un $\tilde{\sigma} \in T^*$ tale che $M_0[\tilde{\sigma}\rangle\tilde{M}$, con $L(\tilde{\sigma}) = w$ e $\pi(\tilde{\sigma}) = \tilde{y}$.*
2. *Nella rete esiste $(M, y) \in \mathcal{M}(w)$ e $\sigma'' \in T_u^*$ tale che $M[\sigma''\rangle\tilde{M}$ con $\tilde{y} = \pi(w) + y + \pi(\sigma'')$.*

Dimostrazione *Dimostriamo questo risultato per induzione sulla lunghezza della stringa osservata w.*

(passo base) *Per $w = \varepsilon$ i risultati sono ovvi.*

(passo induttivo) *Assumiamo che il risultato valga per w. Dimostriamo ora che questo vale per $w = vt$.*

Per prima cosa dimostriamo che $1) \Rightarrow 2)$. Infatti, se vale 1) esistono delle sequenze σ' e σ'' tali che

$$M_0[\sigma'\rangle M'[t\rangle M''[\sigma''\rangle\tilde{M}$$

dove $L(\sigma') = v$, e $\sigma'' \in T_u^*$. Per induzione, esiste $(M, y) \in \mathcal{M}(v)$ tale che

$$M_0[\sigma'_a\rangle M[\sigma'_b\rangle M'[t\rangle M''[\sigma''\rangle\tilde{M}$$

dove $L(\sigma'_a) = v$, $\pi(\sigma'_a) = \pi(v) + y$ *e* $\sigma'_b \in T_u^*$. *Ora esiste una spiegazione minima* $\sigma'_c \in \Sigma(M, t)$ *tale che* $\pi(\sigma'_c) \leq \pi(\sigma'_b)$ *e*

$$M_0[\sigma'_a\rangle M[\sigma'_c\rangle M'_c[t\rangle M'_d[\sigma'_d\rangle M''[\sigma''\rangle \tilde{M}$$

in cui $\pi(\sigma'_c) + \pi(\sigma'_d) = \pi(\sigma'_b)$ *e* $(M'_c, \pi(\sigma'_c)) \in \mathcal{M}(vt) = \mathcal{M}(w)$. *Ciò prova il risultato.*

Secondariamente dimostriamo che $2) \Rightarrow 1)$. *Infatti se vale 2) allora esiste* $\sigma' \in T^*$ *tale che* $M_0[\sigma'\rangle M[\sigma''\rangle \tilde{M}$ *con* $L(\sigma') = vt = w$ *e quindi* $M_0[\sigma\rangle \tilde{M}$ *con* $\sigma = \sigma'\sigma''$.

5.4 Progetto dell'osservatore basato sul BRT

In questo paragrafo focalizziamo l'attenzione sulle reti di Petri limitate e proponiamo un'originale tecnica per il progetto di un osservatore che verrà usato per la diagnosi dei guasti [2].

L'approccio proposto consiste nel disegno di un grafo deterministico, che chiameremo albero base di raggiungibilità (BRT).

Per prima cosa introduciamo le seguenti definizioni. Definiamo:

$$\mathcal{M}_b(w) = \{M \in \mathbb{N}^m \mid \exists y \in \mathbb{N}^{n_u} : (M, y) \in \mathcal{M}(w)\}$$

l'insieme delle marcature di base data l'osservazione w.

Poi, definiamo

$$\mathcal{O}(N, M_0) = \{w \in T_o^* \mid \exists \sigma \in T^*,\ M_0[\sigma\rangle,\ L(\sigma) = w\}$$

l'insieme delle parole osservabili della rete $\langle N, M_0\rangle$.

Indichiamo con:

$$\mathcal{O}_{\min}(N, M_0) = \{w \in \mathcal{O}(N, M_0) \mid \nexists\, w' \in \mathcal{O}(N, M_0) : w' \prec w,\ \mathcal{M}_b(w) = \mathcal{M}_b(w')\}$$

l'insieme delle parole osservabili di lunghezza minima alle quali corrisponde un diverso insieme di marcature di base.

L'albero base di raggiungibilità ha un numero di nodi pari alla cardinalità dell'insieme $\mathcal{O}_{\min}(N, M_0)$. Ogni nodo corrisponde ad un insieme differente $\mathcal{M}(w)$ e ogni arco è etichettato con una transizione osservabile. Più precisamente, il BRT è un automa che ha come alfabeto l'insieme delle transizioni osservabili T_o, il cui stato iniziale è $\mathcal{M}_0 = \mathcal{M}(\varepsilon)$, e se δ è la sua funzione di transizione, vale la relazione $\delta(\mathcal{M}_0, w) = \mathcal{M}(w)$ per ogni parola $w \in \mathcal{O}_{\min}(N, M_0)$. In altri termini, se $w \in \mathcal{O}_{\min}(N, M_0)$, allora esiste un percorso orientato etichettato w dal nodo radice $\mathcal{M}_0$ al nodo $\mathcal{M}(w)$.

L'albero base di raggiungibilità di un sistema di rete limitato $\langle N, M_0\rangle$ può essere costruito applicando il seguente algoritmo, nel quale verrà indicato con $\mathcal{M}_b$ (rispettivamente $\mathcal{M}'_b$, $\tilde{\mathcal{M}}_b$, $\bar{\mathcal{M}}_b$) l'insieme delle marcature di base relative all'insieme $\mathcal{M}$ (rispettivamente $\mathcal{M}'$, $\tilde{\mathcal{M}}$, $\bar{\mathcal{M}}$).

Algoritmo 5.4.1. *[Albero base di raggiungibilità]*

1. *Etichettiamo il nodo iniziale con* $\mathcal{M}_0 = \mathcal{M}(\varepsilon)$.
Questa sarà la radice dell'albero alla quale non assegnamo alcuna etichetta.
2. *Se esistono nodi senza etichetta, seleziona un nodo* $\mathcal{M}$ *senza etichetta e :*
2.1 *se* $\forall\, M \in \mathcal{M}_b$ *e* $\forall\, t \in T_o$, $Y_{\min}(M, t) = \emptyset$, *metti al nodo* $\mathcal{M}$ *l'etichetta "dead" e vai al punto2.*
2.2 $\forall\, t \in T_o : \{M \in \mathcal{M}_b \mid Y_{\min}(M, t) \neq \emptyset\} \neq \emptyset$
2.2.1 *considera* $\tilde{\mathcal{M}} = \emptyset$
2.2.2 $\forall (M, y) \in \mathcal{M}$
2.2.2.1 $\forall \tilde{y} \in Y_{\min}(M, t)$
2.2.2.2 *calcola* $M' = M + C_o\pi(t) + C_u\tilde{y}, \quad y' = y + \tilde{y}$
2.2.2.3 *considera* $\tilde{\mathcal{M}} = \tilde{\mathcal{M}} \cup \{(M', y')\}$
2.2.2.4 *considera* $\mathcal{M}' = \{(M, y) \in \tilde{\mathcal{M}} \mid \nexists (M', y') \in \tilde{\mathcal{M}} \,:\, y' \lneq y\}$
2.4 *aggiungi nel grafo un nuovo nodo* $\mathcal{M}'$ *e un arco* t *dal nodo* $\mathcal{M}$ *al nodo* $\mathcal{M}'$
2.5 *se* $\exists$ *già nel grafo un nodo* $\bar{\mathcal{M}}$ *tale che* $\bar{\mathcal{M}}_b = \mathcal{M}'_b$, *metti al nuovo nodo l'etichetta "dup".*

Esempio 5.4.1. *Il BRT della rete rappresentata in figura 5.1 è riportato in figura 5.2. Guardando questo grafo troviamo tutti i risultati già discussi nell'esempio 5.3.1.*

Dobbiamo fare un'importante osservazione.

Dalla costruzione dell'albero, i percorsi orientati del BRT che iniziano dal nodo radice sono etichettati con parole $w \in \mathcal{O}_{\min}(N, M_0)$. Al contrario non esistono

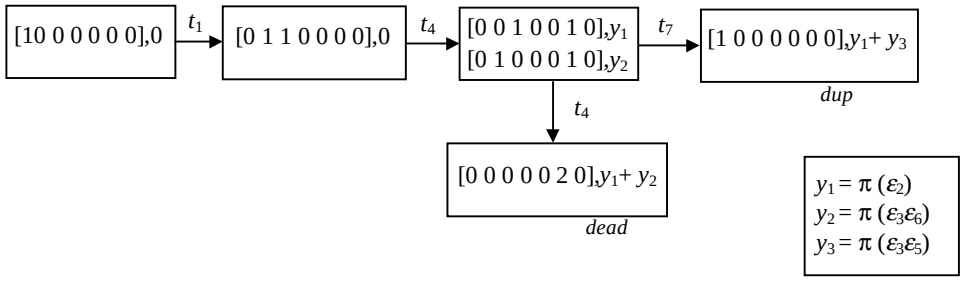


Figura 5.2: L'albero base di raggiungibilità della rete in figura 5.1.

percorsi orientati etichettati con parole $w \in \mathcal{O}(N, M_0) \setminus \mathcal{O}_{\min}(N, M_0)$. Notiamo però, che questo non riduce l'informazione contenuta nel BRT. Per rendere questo concetto più chiaro, presentiamo il seguente esempio.

Esempio 5.4.2. *Consideriamo l'albero base di raggiungibilità presentato in figura 5.2 relativo alla rete di Petri in figura 5.1. Consideriamo la parola* $w = t_1t_4t_7t_1t_4$*, tale che* $w \in \mathcal{O}(N, M_0) \setminus \mathcal{O}_{\min}(N, M_0)$*. Il più lungo prefisso di* w *in* $\mathcal{O}_{\min}(N, M_0)$ *è* $w' = t_1t_4t_7$ *ed esiste un percorso orientato dal nodo radice al nodo* $\mathcal{M}(w')$*, che è etichettato "dup". Infatti, l'insieme* $\mathcal{M}_b(w)$ *(che in questo caso è composto da un solo elemento) coincide con l'insieme* $\mathcal{M}_b(\varepsilon)$*, ossia coincide con la marcatura di base del nodo radice. Se teniamo traccia del fatto che* w' *è stata osservata quando le transizioni* $t_1,\ t_4$ *sono scattate la seconda volta, guardando il BRT possiamo concludere immediatamente che:*

$$\begin{aligned}\mathcal{M}(w) = \ & \{([0\,0\,1\,0\,0\,1\,0]^T, 2y_1 + y_3),\\ & ([0\,1\,0\,0\,0\,1\,0]^T, y_1 + y_2 + y_3)\}.\end{aligned}$$

Conclusioni analoghe possono essere fatte per qualunque altra parola osservata w *che non appartiene all'insieme* $\mathcal{O}_{\min}(N, M_0)$*.*

Facciamo un'ultima osservazione sul BRT. Nella costruzione standard di un grafo di copertura/raggiungibilità, dopo che si è costruito l'albero, per ottenere il grafo, si fondono i nodi uguali in un unico nodo. Il grafo potrebbe contenere cicli. Nel caso dell'albero base di raggiungibilità, la costruzione del grafo non ha senso, in quanto due nodi potrebbero corrispondere allo stesso insieme di marcature di base, ma avere diverse giustificazioni. Chiariamo meglio quest'ultimo concetto con un esempio.

Prendiamo come esempio la rete in figura 5.1 e il suo BRT rappresentato in figura 5.2. Le parole ε, $t_1t_4t_7$, $(t_1t_4t_7)^2$, ..., corrispondono tutte alla stessa marcatura

di base $M_0 = [\,1\,0\,0\,0\,0\,0\,]^T$, ma hanno diverse giustificazioni $\vec{0}$, y_1+y_3, $2y_1+2y_3$, ... Infatti, a ogni ciclo $M_0[t_1t_4t_7\rangle M_0$ la giustificazione cresce di una quantità $y_1 + y_3$.

Quindi, dobbiamo considerare l'albero così com'è, e per calcolare l'insieme $\mathcal{M}(w)$ per una parola w di arbitraria lunghezza, dobbiamo ricordarci che ogni volta che viene raggiunta una foglia, dobbiamo continuare la produzione dal nodo progenitore corrispondente allo stesso insieme di marcature di base, e dobbiamo aggiungere la corrispondente giustificazione, ogni volta che viene ripetuto il ciclo.

5.5 Diagnosi

Il formalismo riguardante le marcature, descritto nelle sezioni precedenti, può essere usato per progettare un diagnosticatore. Definiamo per prima cosa l'insieme delle sequenze di scatto consistenti con un'osservazione $w \in T_o^*$:

$$\mathcal{L}(w) = \{\sigma \in T^* \mid M_0[\sigma\rangle, L(\sigma) = w\}.$$

Definizione 5.5.1. *Un* diagnosticatore *è una funzione* $\Delta : T_o^* \times T_f \to \{0, 1, 2, 3\}$ *che associa a ogni osservazione* w *e a ogni transizione di guasto* $t_f \in T_f$ *uno stato di diagnosi.*

$\Delta(w, t_f) = 0$ *se per ogni* $\sigma \in \mathcal{L}(w)$ *vale la relazione* $t_f \notin \sigma$. *In questo caso il guasto non può essere avvenuto in quanto non esiste alcuna sequenza di scatto contenente* t_f *e consistente con l'osservazione.*

$\Delta(w, t_f) = 1$ *se esiste un* $\sigma \in \mathcal{L}(w)$ *tale che* $t_f \in \sigma$, *ma per qualunque coppia* $(M, y) \in \mathcal{M}(w)$, *la giustificazione* y *della marcatura di base* M *è tale che* $y(t_f) = 0$. *In questo caso il guasto potrebbe essere accaduto, ma non mentre è stata raggiunta una marcatura di base.*

$\Delta(w, t_f) = 2$ *se esiste una coppia* $(M, y) \in \mathcal{M}(w)$ *tale che* $y(t_f) > 0$. *In questo caso il guasto potrebbe essere accaduto mentre viene raggiunta una marcatura di base.*

$\Delta(w, t_f) = 3$ *se per qualunque* $\sigma \in \mathcal{L}(w)$ *vale la relazione* $t_f \in \sigma$. *In questo caso il guasto deve essere accaduto perché tutte le sequenze di scatto consistenti con l'osservazione contengono* t_f.

Gli stati di diagnosi 1 e 2 corrispondono entrambi ai casi in cui il guasto potrebbe essere accaduto, ma non è necessariamente accaduto. Il motivo principale per distinguere tra questi due casi è il seguente. Nello stato 1 il comportamento osservato non suggerisce il fatto che un guasto sia avvenuto, mentre nello stato 2 almeno una delle giustificazioni corrispondenti al comportamento osservato implica che il guasto sia accaduto.
Gli stati di diagnosi associati a ogni osservazione w possono essere facilmente calcolati usando l'albero base di raggiungibilità.
Presenteremo ora una serie di risultati importanti.

Ricordiamo il fatto che il BRT è un automa che ha come alfabeto l'insieme degli eventi osservabili T_o. Lo stato iniziale è $\mathcal{M}_0 = \{(M_0, \vec{0})\}$, e se indichiamo con δ la sua funzione di transizione segue che $\delta(\mathcal{M}_0, w) = \mathcal{M}(w)$.

Proposizione 5.5.1. *Consideriamo una parola osservata $w \in T_o^*$.*

$\Delta(w, t_f) \in \{0, 1\}$ *se* $\forall\ (M, y) \in \mathcal{M}(w)$ *vale* $y(t_f) = 0$.

$\Delta(w, t_f) = 2$ *se esiste* $(M, y) \in \mathcal{M}(w)$ *ed* $(M', y') \in \mathcal{M}(w)$ *tale che* $y(t_f) = 0$ *e* $y'(t_f) > 0$.

$\Delta(w, t_f) = 3$ *se* $\forall\ (M, y) \in \mathcal{M}(w)$ *vale* $y(t_f) > 0$.

Il BRT contiene tutte le informazioni necessarie per assegnare a una sequenza osservata uno stato di diagnosi 2 o 3. Però non ci permette di distinguere immediatamente tra uno stato di diagnosi 0 o 1. Per questa distinzione è necessaria un'ulteriore analisi, che verrà spiegata nella seguente proposizione.

Proposizione 5.5.2. *Consideriamo una parola osservata $w \in T_o^*$ tale che $\forall\ (M, y) \in \mathcal{M}(w)$ vale $y(t_f) = 0$.*

$\Delta(w, t_f) = 0$ *se per qualunque* $(M, y) \in \mathcal{M}(w)$ *non esiste una sequenza* $\sigma \in T_u^*$ *tale che* $M[\sigma\rangle$ *e* $t_f \in \sigma$.

$\Delta(w, t_f) = 1$ *se esiste almeno una coppia* $(M, y) \in \mathcal{M}(w)$ *e almeno una sequenza* $\sigma \in T_u^*$ *tale che* $M[\sigma\rangle$ *e* $t_f \in \sigma$.

Se la sottorete non osservabile è *aciclica*, la raggiungibilità di questa può essere caratterizzata dall'equazione di stato ed esisterà nella sottorete non osservabile una sequenza contenente una transizione t_f che può scattare partendo dalla marcatura M se e solo se l'insieme di vincoli relativo a variabili intere (ICS: integer

constraint set) ammette soluzione:

$$\begin{aligned} & M + C_u z \geq \vec{0} \\ & z(t_f) > 0 \\ & z \in \mathbb{N}^{n_u} \end{aligned} \tag{5.1}$$

In questo modo, abbiamo il seguente risultato.

Proposizione 5.5.3. *Per una rete di Petri la cui sottorete non osservabile è* aciclica*, consideriamo una parola $w \in T_o^*$ tale che per qualunque $(M, y) \in \mathcal{M}(w)$ vale $y(t_f) = 0$.*

$\Delta(w, t_f) = 0$ *se per qualunque $(M, y) \in \mathcal{M}(w)$ l'insieme dei vincoli* (5.1) *non ammette soluzione.*

$\Delta(w, t_f) = 1$ *se esiste una coppia $(M, y) \in \mathcal{M}(w)$ tale che l'insieme dei vincoli* (5.1) *ammette soluzione.*

Esempio 5.5.1. *Come esempio consideriamo la rete in figura 5.1 e associamo all'albero base di raggiungibilità già costruito in figura 5.2 gli stati di diagnosi appena introdotti. Come si può notare al nodo iniziale contenente la marcatura iniziale $M_0 = [\ 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\ ]$abbiamo associato uno stato di diagnosi pari a zero in quanto non si è verificato alcun guasto. Al secondo nodo $M_1 = [\ 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0\ ]$ abbiamo associato uno stato di diagnosi pari a uno in quanto il guasto potrebbe essere accaduto ma non mentre viene raggiunta la marcatura iniziale. Al terzo nodo del BRT abbiamo associato uno stato di diagnosi pari a due in quanto il guasto potrebbe essere accaduto mentre viene raggiunta una marcatura di base. Infatti il nodo contiene due marcature di base, la prima $M_2 = [\ 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0\ ]$ con spiegazione minima nulla e la seconda $M_3 = [\ 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0\ ]$ avente una spiegazione minima contenente il guasto. Il nodo con etichetta* dead *ha stato di diagnosi pari a tre in quanto come si può notare dalla giustificazione il guasto è sicuramente avvenuto, mentre il nodo con etichetta* dup *ha stato di diagnosi pari a zero, in quanto il guasto non è sicuramente avvenuto.*

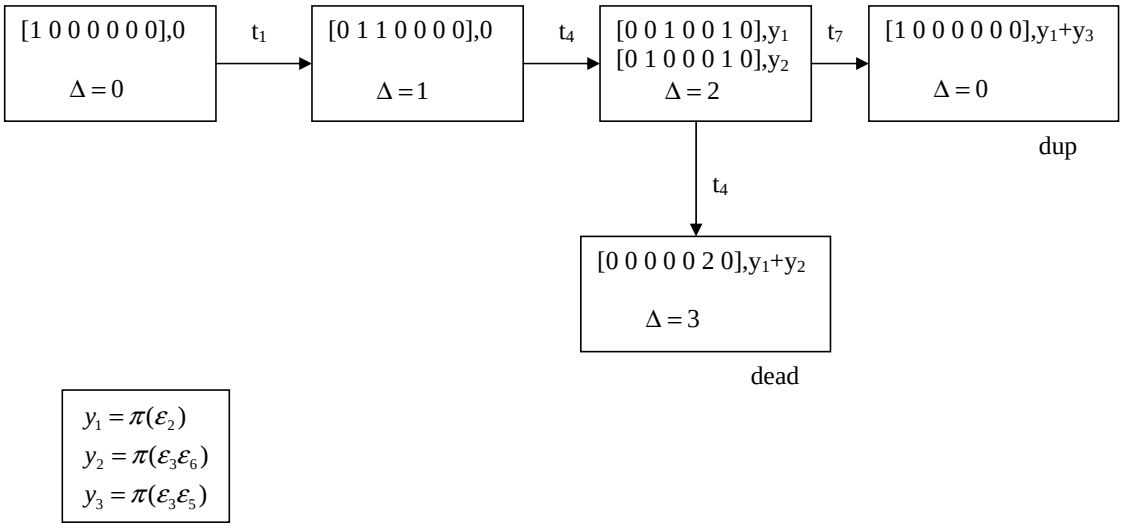


Figura 5.3: L'albero base di raggiungibilità della rete in figura 5.1 con gli stati di diagnosi associati.

Capitolo 6

Toolbox di MATLAB

6.1 Introduzione

In questo capitolo verranno illustrate le due funzioni che costituiscono il pacchetto Matlab e verranno forniti degli esempi sul funzionamento delle due funzioni.

6.2 Calcolo delle spiegazioni minime

In questa sezione presentiamo la funzione di matlab **miny** da me sviluppata per il calcolo delle spiegazioni minime. Ricordiamo che l'insieme delle spiegazioni minime è l'insieme delle sequenze minime non osservabili il cui scatto, partendo dalla marcatura M, è necessario ad abilitare la transizione t. Trattandosi di una funzione Matlab la chiamata alla funzione è soggetta alla sintassi del linguaggio, `[OUTPUT]=NOMEFUNZIONE[INPUT]`, quindi nel nostro caso:

[B]=miny[Pre,Post,M,t,nu].

Tra gli input troviamo le matrici *Pre* e *Post* che caratterizzano completamente la rete, la marcatura M e la transizione t che vogliamo considerare per il calcolo della spiegazione minima e lo scalare *nu* che indica il numero delle transizioni non osservabili della rete di Petri considerata. Le matrici Pre e $Post$ dovranno contenere nelle prime colonne le transizioni osservabili e nelle ultime quelle non osservabili. L'uscita è una matrice B che contiene nelle sue righe le spiegazioni minime e quindi ha dimensione $i \times nu$ dove i è il numero delle spiegazioni minime

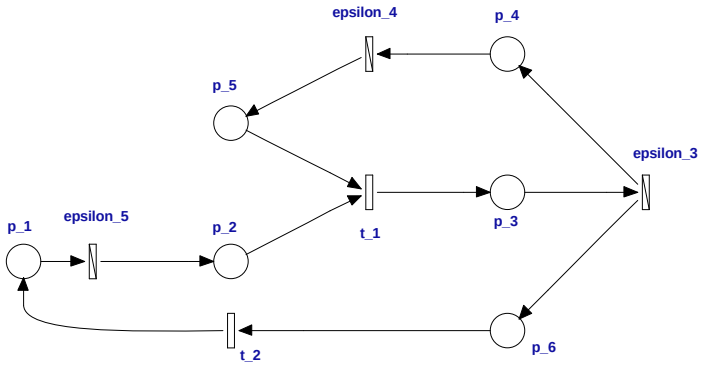


Figura 6.1: esempio di una rete di Petri

per la marcatura e la transizione data, mentre nu è il numero delle transizioni non osservabili della rete.

La spiegazione di come opera la funzione e tutti i chiarimenti sul codice sono riportati nel manuale che si trova nell'appendice della tesi, dove si trova inoltre il codice del programma.

Per illustrare il funzionamento della funzione facciamo un paio di esempi.

Esempio 6.2.1. *Consideriamo la rete di Petri illustrata in figura 6.1. Inseriamo in un file.m che chiamiamo* dati.m *gli ingressi della funzione* miny. *Eseguiamo la funzione dati.m e mostriamo l'output in figura 6.2.*

Abbiamo scelto come marcatura quella che assegna ai posti p_2 e p_5 un gettone. Vogliamo sapere se la transizione t_1 può venir abilitata dallo scatto di transizioni non osservabili e quale è la sua spiegazione minima. Per saperlo eseguiamo la funzione miny *e presentiamo l'output in figura 6.3. Come si vede facilmente dalla figura 6.1 la marcatura che assegna ai posti p_2 e p_5 un gettone abilita la transizione t_1 subito senza richiedere lo scatto di transizioni non osservabili e dunque la sua spiegazione minima è il vettore nullo.*
Se invece consideriamo la marcatura che assegna un gettone al posto p_1 e un gettone al posto p_4, la transizione t_1 può essere abilitata solo dopo lo scatto delle transizioni non osservabili ε_4 ed ε_5. Vediamo l'output in figura 6.4. Giustamente la funzione miny *stampa a video il vettore $B = [\ 0 \quad 1 \quad 1\ ]$ che ci dice che sono scattate le transizioni non osservabili ε_4 ed ε_5.*
Se consideriamo la stessa marcatura M e la transizione t_2, questa volta la funzione miny *dovrebbe dare come risultato l'insieme vuoto, ossia nessuna spiegazione minima può abilitare la transizione data a partire dalla marcatura M. L'output è mostrato in figura 6.5.*

```
MATLAB Command Window                                                Page 1
17 ottobre 2005                                                   10.59.52
>> dati

Pre =

     0     0     0     0     1
     1     0     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     1     0     0     0     0
     0     1     0     0     0


Post =

     0     1     0     0     0
     0     0     0     0     1
     1     0     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     1     0     0


M =

     0
     1
     0
     0
     1
     0


t =

     1


nu =

     3

>>
```

Figura 6.2: Finestra dei comandi di Matlab

```
MATLAB Command Window                                                Page 1
17 ottobre 2005                                                   11.01.14
>> B = miny(Pre,Post,M,t,nu);
le spiegazioni minime sono:
     0     0     0

cioè la transizione scelta è subito abilitata dalla
marcatura data senza lo scatto di transizioni non osservabili
>>
```

Figura 6.3: Finestra dei comandi di Matlab

```
MATLAB Command Window                                                  Page 1
17 ottobre 2005                                                     11.10.30

>> dati

M =

     1
     0
     0
     1
     0
     0


t =

     1

>> B = miny(Pre,Post,M,t,nu);
le spiegazioni minime sono:
     0     1     1

>>
```

Figura 6.4: Finestra dei comandi di Matlab

```
MATLAB Command Window                                                  Page 1
17 ottobre 2005                                                     11.40.17

>> dati

M =

     1
     0
     0
     1
     0
     0


t =

     2

>> B = miny(Pre,Post,M,t,nu);
la transizione t non può essere abilitata dalla marcatura data
>>
```

Figura 6.5: Finestra dei comandi di Matlab

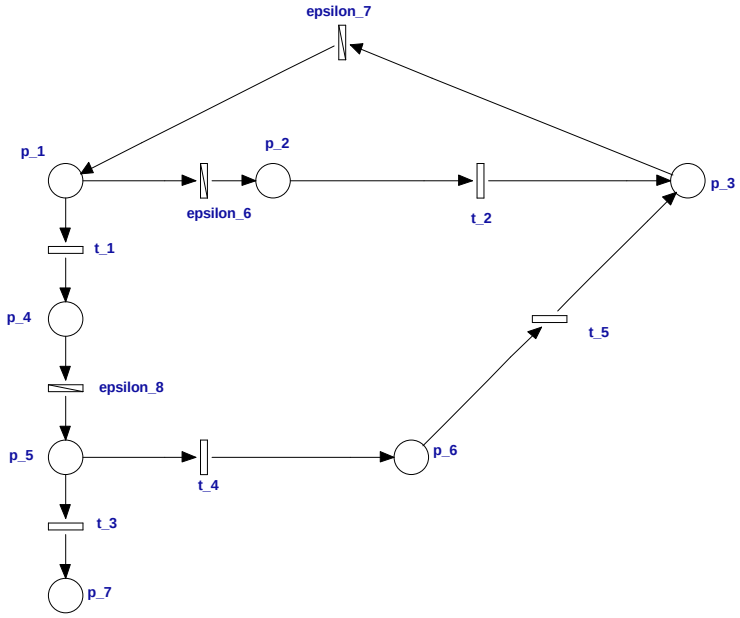


Figura 6.6: esempio di una rete di Petri

Facciamo un secondo esempio.

Esempio 6.2.2. *Consideriamo la rete di Petri illustrata in figura 6.6. Inseriamo gli ingressi della funzione* miny *in un file.m che chiamiamo* dati.m*. Eseguiamo la funzione dati.m e mostriamo l'output in figura 6.7.*

Abbiamo scelto come marcatura quella che assegna al posto p_1 un gettone. Vogliamo sapere se la transizione t_2 può venir abilitata dallo scatto di transizioni non osservabili e quale è la sua spiegazione minima. Per saperlo eseguiamo la funzione miny *e presentiamo l'output in figura 6.8. Come si vede facilmente dalla figura 6.6 la marcatura che assegna al posto p_1 un gettone abilita la transizione t_2 e ha come spiegazione minima $B = [\ 1 \quad 0 \quad 0\ ]$, ossia la transizione t_2 sarà abilitata dopo lo scatto della transizione non osservabile ε_6.*
Se invece consideriamo la marcatura che assegna un gettone al posto p_4, la transizione t_4 non può venir abilitata, in quanto il peso dell'arco in ingresso a t_4 è pari a 2. Vediamo l'output in figura 6.9. Se consideriamo la stessa transizione t_4 e una marcatura che assegna ora due gettoni a p_4, questa volta la funzione miny *dovrebbe dirci che la transizione è abilitata, e dovrebbe darci come spiegazione minima $B = [\ 0 \quad 0 \quad 2\ ]$, ossia la transizione t_4 sarà abilitata dopo che la transizione non osservabile ε_8 è scattata due volte. L'output è mostrato in figura 6.10.*

```
MATLAB Command Window                                                  Page 1
18 ottobre 2005                                                     14.16.45

>> dati

Pre =

     1     0     0     0     0     1     0     0
     0     1     0     0     0     0     0     0
     0     0     0     0     0     0     1     0
     0     0     0     0     0     0     0     1
     0     0     1     2     0     0     0     0
     0     0     0     0     1     0     0     0
     0     0     0     0     0     0     0     0


Post =

     0     0     0     0     0     0     1     0
     0     0     0     0     0     1     0     0
     0     2     0     0     1     0     0     0
     1     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     1
     0     0     0     1     0     0     0     0
     0     0     1     0     0     0     0     0


t =

     2


nu =

     3


M =

     1
     0
     0
     0
     0
     0
     0

>>
```

Figura 6.7: Finestra dei comandi di Matlab

```
MATLAB Command Window                                                  Page 1
18 ottobre 2005                                                     14.19.01

>> B = miny(Pre,Post,M,t,nu);
le spiegazioni minime sono:
     1     0     0

>>
```

Figura 6.8: Finestra dei comandi di Matlab

```
MATLAB Command Window                                                Page 1
18 ottobre 2005                                                   14.27.43

>> dati

t =

     4


M =

     0
     0
     0
     1
     0
     0
     0

>> B = miny(Pre,Post,M,t,nu);
la transizione t non può essere abilitata dalla marcatura data
>>
```

Figura 6.9: Finestra dei comandi di Matlab

```
MATLAB Command Window                                                Page 1
18 ottobre 2005                                                   14.33.00

>> dati

t =

     4


M =

     0
     0
     0
     2
     0
     0
     0

>> B = miny(Pre,Post,M,t,nu);
le spiegazioni minime sono:
     0     0     2

>>
```

Figura 6.10: Finestra dei comandi di Matlab

Mostriamo infine un esempio in cui vi è più di una spiegazione minima.

Esempio 6.2.3. *Consideriamo la rete di Petri illustrata in figura 5.1. Inseriamo gli ingressi della funzione* miny *in un file.m che chiamiamo* dati.m*. Eseguiamo la funzione dati.m e mostriamo l'output in figura 6.11. Come detto prima inserisco nelle matrici Pre e $Post$ prima le transizioni osservabili e poi quelle non osservabili. La figura 6.11 mostra $t = 2$, ossia si sceglie di considerare la seconda colonna che è quella assegnata alla seconda transizione non osservabile t_4.*

Abbiamo scelto come marcatura quella che assegna un gettone ai posti p_2, p_3 e p_5. Vogliamo sapere se la transizione t_4 può venir abilitata dallo scatto di transizioni non osservabili e quale è la sua spiegazione minima. Per saperlo eseguiamo la funzione miny *e presentiamo l'output in figura 6.12. Come si vede facilmente dalla figura 5.1 la marcatura che assegna un gettone ai posti p_2, p_3 e p_5 può abilitare la transizione t_4 con lo scatto delle seguenti transizioni non osservabili: ε_2,ε_6, oppure $\varepsilon_3\varepsilon_6$. Si vede subito analizzando l'insieme delle spiegazioni che la spiegazione $\varepsilon_3\varepsilon_6$ non è minima. Quindi l'insieme delle spiegazioni minime che abilita la transizione t_4 è:*

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ossia la transizione t_4 sarà abilitata dopo lo scatto della transizione non osservabile ε_2 o dopo lo scatto della transizione non osservabile ε_6.

6.3 Costruzione del BRT

In questa sezione presentiamo la funzione di matlab **BRT** da me sviluppata per la costruzione dell'albero di raggiungibilità delle marcature di base, più brevemente detto albero base di raggiungibilità. Anche in questo caso trattandosi di una funzione Matlab la chiamata alla funzione è soggetta alla sintassi del linguaggio, `[OUTPUT]=NOMEFUNZIONE[INPUT]`, quindi nel nostro caso:

[T]=BRT[Pre,Post,M_0,nu,B].

Tra gli input troviamo le matrici *Pre* e *Post* che caratterizzano completamente la rete, la marcatura iniziale M_0, lo scalare *nu* che indica il numero delle transizioni non osservabili della rete di Petri e la matrice delle spiegazioni minime B associata alla marcatura M_0. L'uscita è una matrice T che contiene nelle sue righe tutte le

```
MATLAB Command Window                                              Page 1
2 novembre 2005                                                    8.20.05
>> dati

Pre =

     1     0     0     0     0     0     0
     0     0     0     1     0     0     0
     0     0     0     0     1     0     0
     0     1     0     0     0     0     0
     0     0     0     0     0     1     1
     0     0     1     0     0     0     0
     0     0     1     0     0     0     0


Post =

     0     0     1     0     0     0     0
     1     0     0     0     0     0     0
     1     0     0     0     0     0     0
     0     0     0     1     0     0     1
     0     0     0     0     1     0     0
     0     1     0     0     0     0     0
     0     0     0     0     0     1     0


t =

     2


nu =

     4


M =

     0
     1
     1
     0
     1
     0
     0

>>
```

Figura 6.11: Finestra dei comandi di Matlab

```
MATLAB Command Window                                          Page 1
2 novembre 2005                                              8.27.36

>> B = miny(Pre,Post,M,t,nu)
le spiegazioni minime sono:
     1     0     0     0
     0     0     0     1


B =

     1     0     0     0
     0     0     0     1

>>
```

Figura 6.12: Finestra dei comandi di Matlab

informazioni per disegnare l'albero base di raggiungibilità. T ha un numero di righe pari al numero di nodi del BRT e un numero di colonne pari a 6. Il contenuto delle colonne è il seguente:

1. la prima colonna contiene uno scalare che rappresenta il numero del nodo;
2. la seconda colonna contiene una matrice che racchiude nelle sue righe le marcature di base del nodo considerato;
3. la terza colonna contiene una matrice che racchiude nelle sue righe le giustificazioni relative alle marcature di base del nodo considerato;
4. la quarta colonna contiene una matrice con un numero di righe pari al numero di marcature del nodo considerato ed un numero di colonne pari al numero delle transizioni osservabili. La quarta colonna indica le transizioni che vengono abilitate dal nodo;
5. la quinta colonna contiene una matrice con un numero di righe pari al numero di marcature del nodo considerato ed un numero di colonne pari al numero delle transizioni osservabili. La quinta colonna indica le marcature che vengono raggiunte dal nodo ;
6. la sesta ed ultima colonna contiene uno scalare che rappresenta l'etichetta del nodo. L'etichetta indica che il nodo è stato esplorato (ed in questo caso vale 1), oppure che il nodo è morto (ed in questo caso vale -1), oppure che il nodo è duplicato (ed in questo caso vale 2).

La spiegazione di come opera la funzione e tutti i chiarimenti sul codice sono riportati nel manuale che si trova nell'appendice della tesi, dove si trova inoltre il codice del programma.

```
>> datiBRT

M0 =

    0
    1
    0
    0
    1
    0

>> T = BRT(Pre,Post,M0,nu,B)
T =

    [1]    [1x6 double]    [1x3 double]    [1x2 double]    [1x2 double]    [1]
    [2]    [1x6 double]    [1x3 double]    [1x2 double]    [1x2 double]    [1]
    [3]    [1x6 double]    [1x3 double]    [1x2 double]    [1x2 double]    [1]
    [4]    [1x6 double]    [1x3 double]    [1x2 double]    [1x2 double]    [2]
```

Figura 6.13: Finestra dei comandi di Matlab

Per illustrare il funzionamento della funzione facciamo un paio di esempi.

Esempio 6.3.1. *Consideriamo la rete in figura 6.1. Consideriamo come marcatura iniziale la marcatura* $M_0 = [\ 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0\ ]^T$. *Inseriamo i dati di input nel file* datiBRT.m *ed eseguiamo la funzione BRT. L'output della funzione è mostrato in figura 6.13. Le informazioni subito visibili sono rappresentate dalla prima e dall'ultima colonna che indicano rispettivamente che l'albero è costituito da quattro nodi e che il quarto nodo è un nodo duplicato. Per visualizzare le informazioni contenute nelle altre colonne della matrice* T *usiamo l'istruzione* $T\{:, 2:5\}$ *che stampa a video i contenuti della seconda, poi della terza ed infine della quarta colonna, come riportato in figura 6.14.*

Ciascun nodo è composto da una sola marcatura di base. Le prime quattro righe costituiscono le marcature di base contenute nei quattro nodi e le righe dalla cinque alla otto sono le rispettive giustificazioni. La nona riga, corrispondente al primo nodo, ci dice che il primo nodo abilita la transizione t_1*, la decima ci dice che il secondo nodo abilita la transizione* t_2*, l'undicesima ci dice che il terzo nodo abilita la transizione* t_1*, mentre il quarto nodo non abilita alcuna transizione, perchè come si legge dall'ultima colonna è un nodo duplicato (ha etichetta 2). Infine le ultime quattro righe ci indicano le marcature che vengono raggiunte dai rispettivi nodi: il nodo 1 raggiunge il nodo 2 con lo scatto della transizione* t_1*, il nodo 2 raggiunge il nodo 3 con lo scatto della transizione* t_2 *e il nodo 3 raggiunge il nodo 4 con lo scatto della transizione* t_1*.*
Con tutte queste informazioni siamo in grado di disegnare il BRT, che si trova in figura 6.15.

Esempio 6.3.2. *Consideriamo la rete in figura 6.6. Consideriamo come marcatura iniziale la marcatura* $M_0 = [\ 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\ ]^T$. *Inseriamo i dati di input nel file* datiBRT.m *ed eseguiamo la funzione BRT. L'output della funzione è*

```
>> T{:,2:5}

ans =

    0    1    0    0    1    0


ans =

    0    0    1    0    0    0


ans =

    1    0    0    1    0    0


ans =

    0    0    1    0    0    0


ans =

    0    0    0


ans =

    0    0    0


ans =

    1    0    0


ans =

    1    1    1


ans =

    1    0


ans =

    0    1


ans =

    1    0


ans =

    0    0


ans =

    2    0


ans =

    0    3


ans =

    4    0

ans =

    0    0
```

Figura 6.14: Finestra dei comandi di Matlab

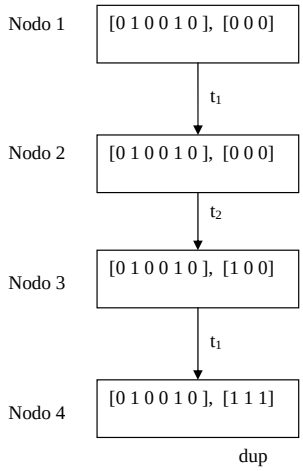


Figura 6.15: Albero base di raggiungibilità della rete in figura 6.1

mostrato in figura 6.16. Le informazioni subito visibili sono rappresentate dalla prima e dall'ultima colonna che indicano rispettivamente che l'albero è costituito da sei nodi e che il quarto nodo è un nodo morto, mentre il quinto e il sesto nodo sono dei nodi duplicati. Per visualizzare le informazioni contenute nelle altre colonne della matrice T usiamo l'istruzione $T\{:, 2:5\}$ che stampa a video i contenuti della seconda, poi della terza ed infine della quarta colonna, come riportato in figura 6.17.

Ciascun nodo è composto da una sola marcatura di base. Le prime sei righe costituiscono le marcature di base contenute nei sei nodi e le righe dalla sette alla dodici sono le rispettive giustificazioni. La tredicesima riga, corrispondente al primo nodo, ci dice che il primo nodo abilita la transizione t_1 e la transizione t_2, la quattordicesima riga ci dice che il secondo nodo abilita la transizione t_3, la quindicesima ci dice che il terzo nodo abilita le transizioni t_1 e t_2, mentre il quarto, il quinto e il sesto nodo non abilitano alcuna transizione, perchè come si legge dall'ultima colonna il quarto è un nodo morto (ha etichetta -1), mentre il quinto e il sesto sono due nodi duplicati (hanno etichetta 2). Infine le ultime sei righe ci indicano le marcature che vengono raggiunte dai rispettivi nodi: il nodo 1 raggiunge il nodo 2 con lo scatto della transizione t_1 ed il nodo 2 con lo scatto della transizione t_2, il nodo 2 raggiunge il nodo 4 con lo scatto della transizione t_3 e il nodo 3 raggiunge i nodi 5 e 6 con lo scatto rispettivamente delle transizioni t_1 e t_2.

Con tutte queste informazioni siamo in grado di disegnare il BRT, mostrato in figura 6.18.

```
>> datiBRT

M =

    1
    0
    0
    0
    0
    0
    0
>> T = BRT(Pre,Post,M0,nu,B)
T =

   [1]   [1x7 double]   [1x3 double]   [1x5 double]   [1x5 double]   [ 1]
   [2]   [1x7 double]   [1x3 double]   [1x5 double]   [1x5 double]   [ 1]
   [3]   [1x7 double]   [1x3 double]   [1x5 double]   [1x5 double]   [ 1]
   [4]   [1x7 double]   [1x3 double]   [1x5 double]   [1x5 double]   [-1]
   [5]   [1x7 double]   [1x3 double]   [1x5 double]   [1x5 double]   [ 2]
   [6]   [1x7 double]   [1x3 double]   [1x5 double]   [1x5 double]   [ 2]

>>
```

Figura 6.16: Finestra dei comandi di Matlab

```
>> T{:,2:5}

ans =

   1   0   0   0   0   0   0

ans =

   0   0   0   1   0   0   0

ans =

   0   0   1   0   0   0   0

ans =

   0   0   0   0   0   0   1

ans =

   0   0   0   1   0   0   0

ans =

   0   0   1   0   0   0   0

ans =

   0   0   0

ans =

   0   0   0

ans =

   1   0   0

ans =

   0   0   1

ans =

   1   1   0

ans =

   2   1   0

ans =

   1   1   0   0   0

ans =
```

```
     0     0     1     0     0

ans =

     1     1     0     0     0

ans =

     0     0     0     0     0

ans =

     0     0     0     0     0

ans =

     0     0     0     0     0

ans =

     2     3     0     0     0

ans =

     0     0     4     0     0

ans =

     5     6     0     0     0

ans =

     0     0     0     0     0

ans =

     0     0     0     0     0

ans =

     0     0     0     0     0

>>
```

Figura 6.17: Finestra dei comandi di Matlab

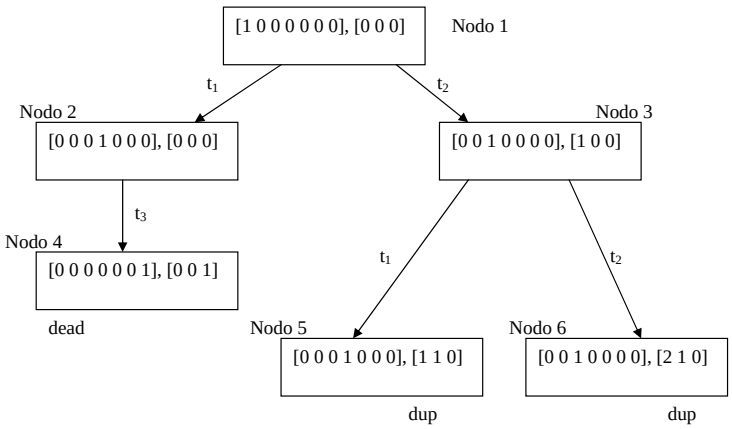


Figura 6.18: Albero base di raggiungibilità della rete in figura 6.6

Capitolo 7

Confronto fra i due approcci e casi di studio

7.1 Confronto fra i due approcci

In questo capitolo vengono confrontate le ipotesi e le peculiarità dei due approcci di diagnosi dei guasti trattati finora.

I due approcci utilizzano un modello del sistema diverso, ma si basano sulla stessa ipotesi di limitatezza del modello: l'approccio di Lafortune utilizza gli automi, mentre l'approccio di Giua-Seatzu utilizza le reti di Petri, ma così come gli automi si suppongono sempre a stati finiti, così le reti di Petri si suppongono limitate.

La seconda assunzione comune ai due modelli è che i guasti siano modellati in un caso con eventi non osservabili (automi) , nell'altro con transizioni non osservabili (reti di Petri). Inoltre in entrambi i casi vengono usate le stesse scomposizioni: nel caso degli automi l'insieme degli eventi viene diviso in eventi osservabili ed eventi non osservabili, e questi ultimi contengono al loro interno l'insieme degli eventi che caratterizzano il guasto:

$$\Sigma = \Sigma_o \cup \Sigma_u, \quad \Sigma_f \subseteq \Sigma_u.$$

Allo stesso modo le transizioni delle reti di Petri vengono divise in osservabili e non osservabili; queste ultime contengono al loro interno le transizioni che caratterizzano il guasto:

$$T = T_o \cup T_u, \quad T_f \subseteq T_u.$$

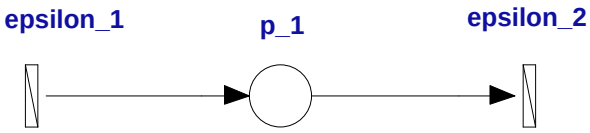


Figura 7.1: Rete di Petri aciclica

La terza ipotesi che hanno in comune i due approcci, oltre alle prime due riguardanti i modelli, è quella di aciclicità. Le due ipotesi nascono da esigenze diverse: nell'approccio con gli automi la condizione di aciclicità nasce dall'esigenza di voler evitare che l'automa G che modella il sistema contenga cicli di eventi non osservabili; nell'approccio con le reti di Petri la condizione di aciclicità nasce dal fatto di poter calcolare la raggiungibilità a partire dalle marcature di base con l'equazione di stato.

Ci chiediamo ora se nel secondo approccio il sistema possa contenere cicli di transizioni non osservabili. In generale l'aciclicità della rete non implica che la rete non possa contenere sequenze di transizioni non osservabili di lunghezza infinita. Consideriamo come esempio la rete in figura 7.1. Le transizioni ε_1 ed ε_2 sono non osservabili. Tale rete è aciclica, ma posso avere come sequenza di scatto $\sigma = (\varepsilon_1\varepsilon_2)^n$ dove n potrebbe essere infinito. Quindi in generale l'aciclicità consente alla rete di poter contenere sequenze di transizioni non osservabili di lunghezza infinita.

Tuttavia l'approccio di Giua-Seatzu ipotizza che la rete di Petri sia limitata. Presenteremo ora un teorema che dimostra che se una rete di Petri è limitata e la sua sottorete non osservabile è aciclica allora non possono esistere sequenze di scatto contenenti solo transizioni non osservabili di lunghezza maggiore ad un certo limite superiore che chiameremo b.

Teorema 7.1.1. *Si consideri una rete di Petri* $\langle N, M_0\rangle$ limitata *e tale che la sua sottorete non osservabile sia* aciclica*. Si dimostra che esiste un intero* $b \in \mathbb{N}$ *tale che:*

$$\forall M \in R(N, M_0), \forall \sigma \in T_u^* : M[\sigma\rangle \Longrightarrow |\sigma| \leq b.$$

Dimostrazione. Data una sottorete aciclica, possiamo dividere le sue transizioni in livelli.

Date due transizioni t e t' appartenenti alla sottorete non osservabile si definisce la relazione:

$$t \prec t'$$

se, nella sottorete non osservabile, esiste un percorso orientato che va da t a t'.

Possiamo ora definire il primo livello di transizioni non osservabili come:

$$T_1 = \{t \in T_u \mid (\nexists t' \in T_u) : \ t' \prec t\}.$$

Il primo livello è costituito da transizioni sorgente, e si dimostra facilmente che tale insieme non è vuoto. Se così non fosse vorrebbe dire che ogni transizione è preceduta da un'altra transizione, ossia poichè la rete è stata assunta limitata le transizioni non osservabili formano un ciclo, ma questo violerebbe l'ipotesi di aciclicità.

Ricorsivamente possiamo definire per $k = 2, \ldots, r$ i livelli

$$T_k = \{t \in T_u \mid (\nexists t' \in T_u \setminus \cup_{i=1}^{k-1} T_i) : \ t' \prec t\}$$

sino a che $T_1 \cup \cdots \cup T_r = T_u$.

Abbiamo quindi dimostrato che possiamo dividere le transizioni della sottorete non osservabile in livelli. Dimostriamo ora che la lunghezza di tutte le sequenze di scatto che contengono solo transizioni non osservabili è limitata da un intero b.

Data una transizione $t \in T_u$ definiamo la funzione:

$$\varrho(t) = \min_{p \in {}^\bullet t} \left\lfloor \frac{bound(p)}{Pre(p,t)} \right\rfloor,$$

dove $bound(p)$ indica il numero massimo di gettoni che possono trovarsi nel posto p. L'intero $\varrho(t)$ indica il numero massimo di volte in cui la transizione t può scattare nella sottorete non osservabile a partire da una qualunque marcatura raggiungibile nella rete di partenza.

Si verifica facilmente che, essendo la rete non osservabile aciclica, e non potendo quindi i suoi posti sorgente ricevere gettoni dallo scatto di transizioni non osservabili, b sarà minore o uguale della somma degli interi $\varrho(t)$ calcolati per ogni transizione non osservabile appartenente alla rete, ossia

$$b \leq \sum_{t \in T_u} \varrho(t).$$

□

Il teorema afferma che se una rete è limitata e la sua sottorete non osservabile è aciclica, la lunghezza di tutte le sequenze di scatto che contengono solo transizioni

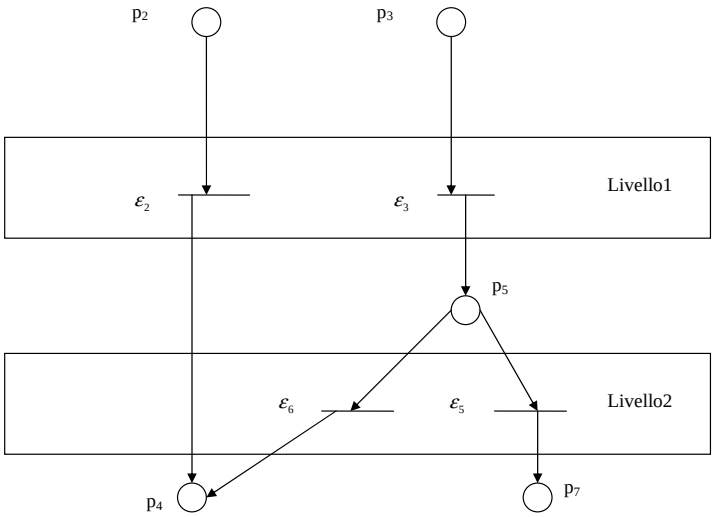


Figura 7.2: sottorete aciclica non osservabile (divisa in livelli) della rete in figura 5.1

non osservabili è limitata da un intero b. Ciò implica che la definizione di aciclità data da Lafortune in [1] e quella data da Giua-Seatzu in [2] è sostanzialmente equivalente.

Esempio 7.1.1. *Consideriamo la rete in figura 5.1. Tale rete è limitata e ha come sottorete non osservabile, la sottorete aciclica mostrata in figura 7.2. Tale rete ha due livelli* $T_1 = \{\varepsilon_2, \varepsilon_3\}$ *e* $T_2 = \{\varepsilon_5, \varepsilon_6\}$ *e ha tutti i posti 1-limitati. Facendo un semplice calcolo si trova che l'intero* b *è pari a quattro e la sequenza* σ *più lunga di transizioni non osservabili è pari a 3. E' quindi rispettata la condizione del teorema* $|\sigma| \leq b$*.*

Nell'approccio con le reti di Petri viene fatta l'ulteriore assunzione, che anche se non citata viene condivisa anche dal primo approccio, che un'etichetta non possa essere associata a più di una transizione. Questa ipotesi è stata fatta per non introdurre ulteriore non determinismo oltre a quello introdotto dalle $\varepsilon-$transizioni.

In entrambe le teorie viene costruito un diagnosticatore. Nel primo caso si tratta di un automa a stati finiti deterministico G_d con stato iniziale $q_0 = \{(x_0, \{N\})\}$, mentre nel secondo caso il diagnosticatore è una funzione che associa a ogni osservazione ω e ad ogni transizione di guasto $t_f \in T_f$ uno stato di diagnosi. Gli stati di diagnosi associati ad ogni osservazione ω possono essere facilmente calcolati usando l'albero base di raggiungibilità. Quest'ultimo è rappresentato da un automa a stati finiti deterministico, con stato iniziale $\mathcal{M}_0 = \{(M_0, \vec{0})\}$.

Possiamo notare tre ulteriori analogie tra le due rappresentazioni:

- Uno stato q_d di G_d è della forma

  $$q_d = \{(x_1, l_1), ..., (x_n, l_n)\}$$

  dove x_i appartiene a X_o e l_i appartiene a Δ; l_i è della forma $l_i = \{N\}$, $l_i = \{A\}$, $l_i = \{F_{i_1}, F_{i_2}, ...F_{i_k}\}$, o $l_i = \{A, F_{i_1}, F_{i_2}, ...F_{i_k}\}$ dove negli ultimi due casi $\{i_1, i_2, ...i_k\} \subseteq \{1, 2, ...m\}$. Un nodo dell'albero è della forma $\mathcal{M}(w) = (M, y)$ dove M è la marcatura di base associata all'osservazione ω e y è la corrispondente giustificazione.
  Così come nel primo approccio le etichette attaccate agli stati portano informazioni sui guasti e i guasti sono diagnosticati controllando queste etichette, così anche le giustificazioni indicano lo stato di diagnosi e quindi servono a capire anche in questo caso se il guasto è accaduto o meno.

- Agli stati di diagnosi introdotti nel quinto capitolo potremmo associare gli stati del diagnosticatore.
  Uno stato q appartenente a Q_d F_i-certo, cioè tale che $\forall (x, l) \in q \quad F_i \in l$ può essere associato allo stato di diagnosi $\Delta(\omega, t_f) = 3$, che vale se per qualunque $\sigma \in \mathcal{L}(w) \quad t_f \in \sigma$.
  Allo stesso modo uno stato q appartenente a Q_d tale che $\forall (x, l) \in q \quad l = \{N\}$, può essere associato allo stato di diagnosi $\Delta(w, t_f) = 0$, che vale se per ogni $\sigma \in \mathcal{L}(w)$ vale la relazione $t_f \notin \sigma$.
  Inoltre uno stato q appartenente a Q_d F_i-incerto, cioè se $\exists (x, l), (y, l') \in q$, tale che $F_i \in l$ e $F_i \notin l'$, può essere associato allo stato di diagnosi $\Delta(w, t_f) = 2$, che vale se esiste una coppia $(M, y) \in \mathcal{M}(w)$ tale che $y(t_f) > 0$.
  Possiamo, forzando un po' le cose, associare allo stato di diagnosi $\Delta(w, t_f) = 1$, che vale se esiste un $\sigma \in \mathcal{L}(w)$ tale che $t_f \in \sigma$ (ma per qualunque coppia $(M, y) \in \mathcal{M}(w)$, la giustificazione y della marcatura di base M è tale che $y(t_f) = 0$) , uno stato q appartenente a Q_d tale che $\forall (x, l) \in q \quad l = \{N\}$.
  Infine uno stato q appartenente a Q_d ambiguo, cioè tale che $\exists (x, l) \in q$ dove $A \in l$, lo potrei associare al caso di due nodi $(M, y_1), (M, y_2)$ del BRT aventi la stessa marcatura di base M e $y_1(t_f) = 0$ mentre $y_2(t_f) > 0$.

- Una terza analogia riguarda la *propagazione delle etichette* applicata nel primo approccio. Ricordiamo che la funzione di propagazione delle etichette

$LP : X_o \times \Delta \times \Sigma^* \longrightarrow \Delta$ è definita come segue:

$$LP(x,l,s) = \begin{cases} \{\text{N}\}, & \text{se } l = \{N\} \wedge \forall i[\Sigma_{fi} \notin s] \; ; \\ \{\text{A}\}, & \text{se } l = \{A\} \wedge \forall i[\Sigma_{fi} \notin s] \; ; \\ \{F_i : F_i \in l \vee \Sigma_{fi} \in s\}, & \text{altrimenti .} \end{cases}$$

Allo stesso modo nel secondo approccio le giustificazioni vengono $''$ propagate $''$ nel BRT. Cioè se sono nel nodo 1 e vado nel nodo 2 con lo scatto della transizione t, la nuova giustificazione sarà calcolata come la somma della giustificazione dello stato 1 più la giustificazione associata alla transizione t.

Si possono notare altre cose molto importanti.

Mentre l'approccio di Giua-Seatzu tiene conto di quante volte il guasto si è verificato nel sistema settando opportunamente il valore della giustificazione, l'approccio di Lafortune trattato nel quarto capitolo tiene conto solo del fatto che il guasto si sia verificato o meno.

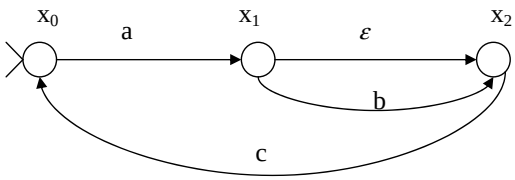
Il calcolo delle marcature di base è concorde con l'approccio standard per la costruzione del diagnosticatore presentato nel quarto capitolo, mentre la diagnosi che viene affrontata nelle reti di Petri è più simile all'approccio modificato presentato in [7].

Con l'approccio di Lafortune inoltre guardando uno stato del diagnosticatore non ho immediatamente l'informazione del guasto che si è verificato, infatti questa informazione sparisce con l'ambiguità. Questo fatto non si verifica con l'approccio con le reti di Petri, infatti le etichette non vengono rimpiazzate in quanto non esiste il concetto di ambiguità.

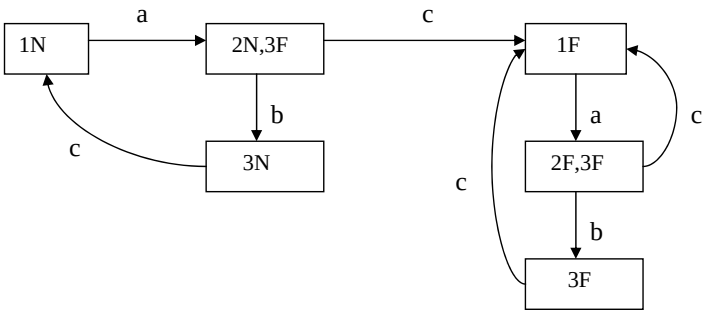
7.2 Esempi

Illustriamo ora due esempi di sistemi che studieremo con i due approcci.

Esempio 7.2.1. *Come primo esempio consideriamo un sistema semplice, cratterizzato da un insieme di eventi $\Sigma = \{a, b, c, \varepsilon\}$. L'insieme degli eventi osservabili è pari a $\Sigma_o = \{a, b, c\}$, mentre l'insieme degli eventi non osservabili Σ_u è costituito dal solo evento di guasto con etichetta ε. Rappresentiamo in figura 7.3 l'automa a stati finiti G e il suo corrispondente diagnosticatore G_d. Allo stesso modo rap-*



Il sistema G



Il diagnosticatore G_d

Figura 7.3: Esempio di un automa e del suo diagnosticatore G_d

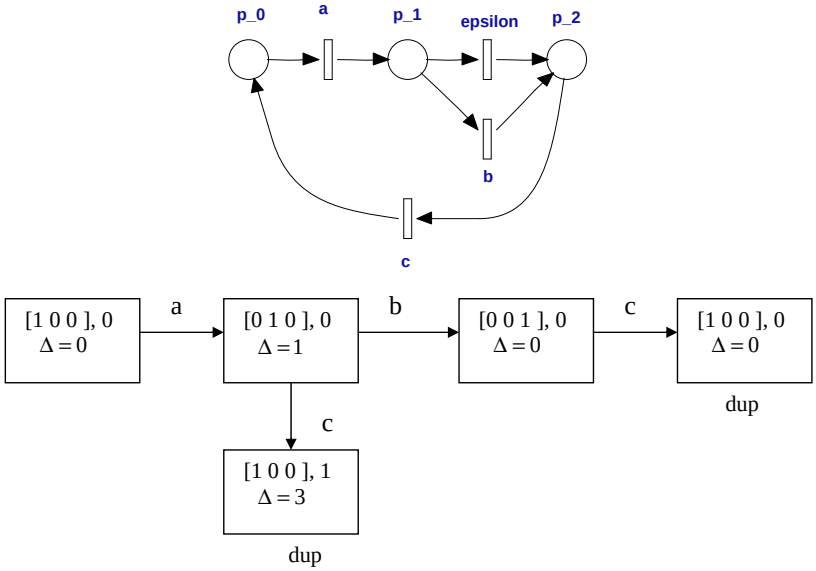


Figura 7.4: Esempio di une rete di Petri e del suo BRT

presentiamo lo stesso sistema con una rete di Petri. Come prima le transizioni osservabili sono pari a $T_o = \{a, b, c\}$ *e l'insieme delle transizioni non osservabili* T_u *è costituito dal solo elemento* ε*. In figura 7.4 vediamo rappresentata la rete di Petri e l'albero base di raggiungibilità.*

Si vede chiaramente in questo esempio che mentre l'approccio di Lafortune mira a scoprire se si è verificato o meno il guasto, l'approccio Giua-Seatzu oltre alla scoperta del guasto tiene anche conto di quante volte il guasto si è verificato. Infatti una volta osservata la parola $\omega = \{acacacac\}$ *nel diagnosticatore mi trovo nello stato* $\{1F\}$ *che mi da l'informazione che il guasto è accaduto; nell'albero base di raggiungibilità invece mi trovo nello stato* $[\ 1 \quad 0 \quad 0\ ]$ *con giustificazione* $y = 4$ *e stato di diagnosi* $\Delta = 3$*. Lo stato di diagnosi* $\Delta = 3$ *mi dice che il guasto è avvenuto e la giustificazione* y *mi dice che il guasto si è verificato quattro volte.*

Presentiamo ora un secondo esempio, un po' più complicato del primo, con il quale vorremmo far notare come l'approccio con le reti di Petri sia molto più vantaggioso nel caso in cui si considerino sistemi concorrenziali.

Esempio 7.2.2. *Consideriamo i due automi finiti non deterministici* G' *e* G''*, che rappresentano un sistema concorrenziale, mostrati in figura 7.5. Vogliamo determinare il diagnosticatore di questo sistema. Per prima cosa operiamo la composizione concorrente dei due automi che rappresentiamo in figura 7.6. Abbiamo ottenuto un automa* G *avente 25 stati. Ogni stato* x *di* G *è una coppia* (x', x'')

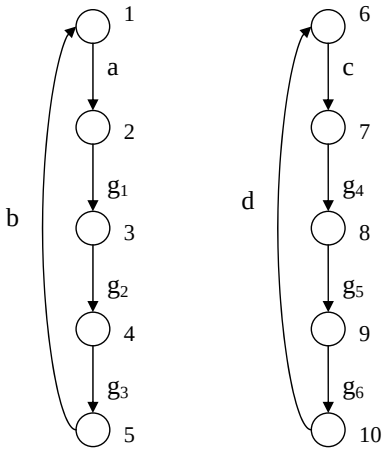


Figura 7.5: I due automi finiti non deterministici G' e G''

dove x' è uno stato appartenente all'insieme degli stati di G' e x'' è uno stato appartenente all'insieme degli stati di G''. L'insieme degli eventi osservabili del sistema G è l'unione dell'insieme degli eventi osservabili di G' e di G'' ed è pari a $\Sigma_o = \{a, b, c, d\}$, allo stesso modo l'insieme degli eventi non osservabili di G è l'unione dell'insieme degli eventi non osservabili di G' e di G'' ed è pari a $\Sigma_u = \{g_1, g_2, g_3, g_4, g_5, g_6\}$. Consideriamo che ogni guasto abbia la sua partizione e cioè $\Sigma_{f1} = g_1$, $\Sigma_{f2} = g_2$, $\Sigma_{f3} = g_3$, $\Sigma_{f4} = g_4$, $\Sigma_{f5} = g_5$ e $\Sigma_{f6} = g_6$. Costruiamo ora il diagnosticatore e lo rappresentiamo in figura 7.7.

Come prima, consideriamo ora l'equivalente rete di Petri rappresentata in figura 7.8. Come per l'automa l'insieme delle transizioni osservabili è pari a $T_o = \{a, b, c, d\}$ e l'insieme degli eventi non osservabili è pari a $T_u = \{g_1, g_2, g_3, g_4, g_5, g_6\}$. Calcoliamo l'albero base di raggiungibilità e associamo ad ogni nodo uno stato di diagnosi. Il BRT è rappresentato in figura 7.9.

Come si può notare dal numero di stati la costruzione del diagnosticatore è molto più semplice con l'approccio di Giua-Seatzu. Inoltre nel caso degli automi se considero un sistema concorrenziale con più canali devo sempre prima fare la composizione concorrente. Se avessi avuto lo stesso sistema con tre canali invece che due, il sistema risultante G avrebbe avuto ben 125 stati e un numero poco inferiore il diagnosticatore. Il numero e la complessità cresce quindi al crescere del numero dei canali e cresce ulteriormente anche al crescere dei gettoni per posto.

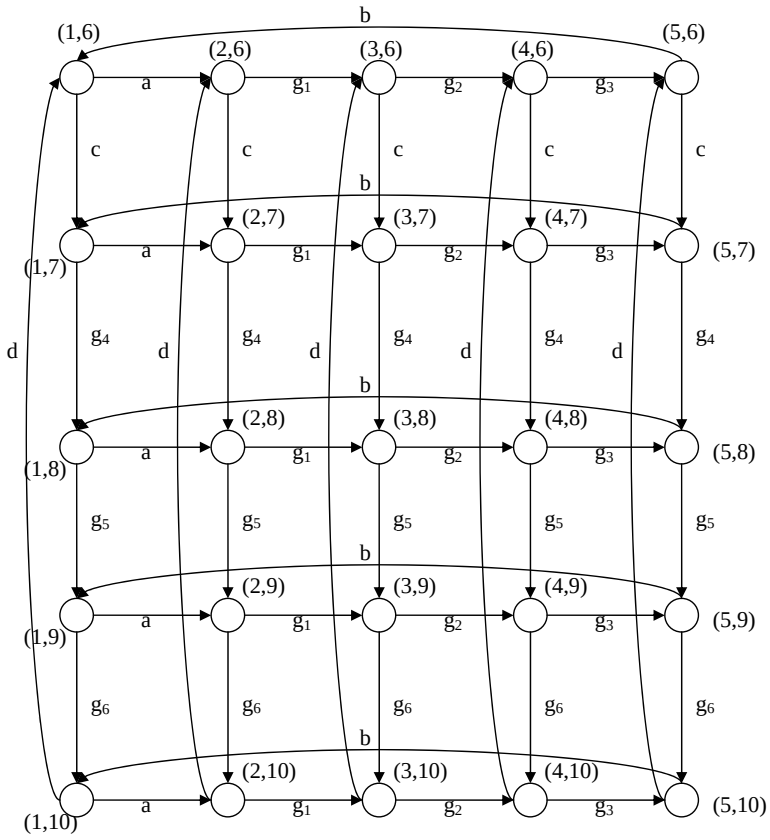


Figura 7.6: Automa G ottenuto dalla composizione concorrente di G'e G''

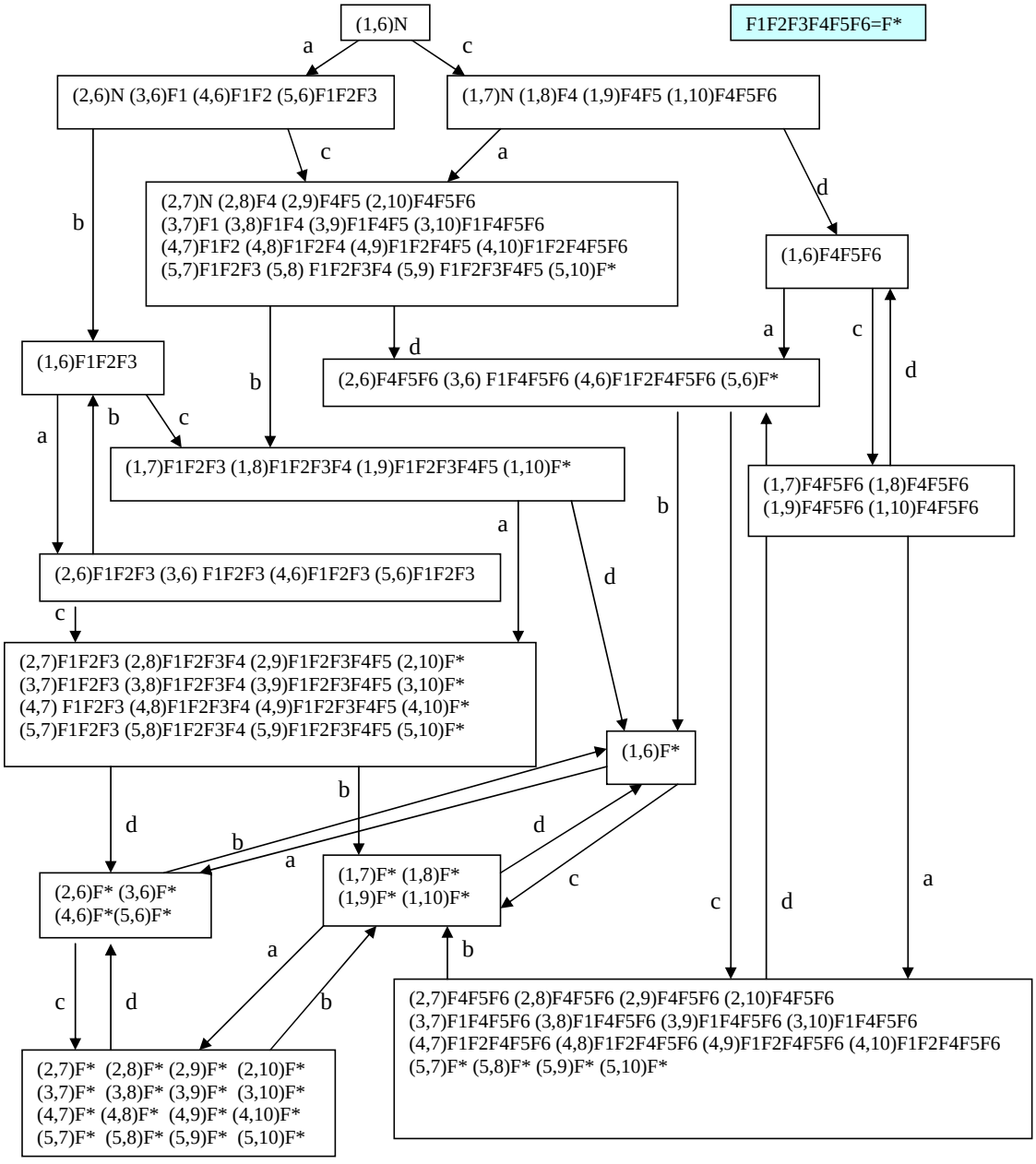


Figura 7.7: Diagnosticatore del sistema G

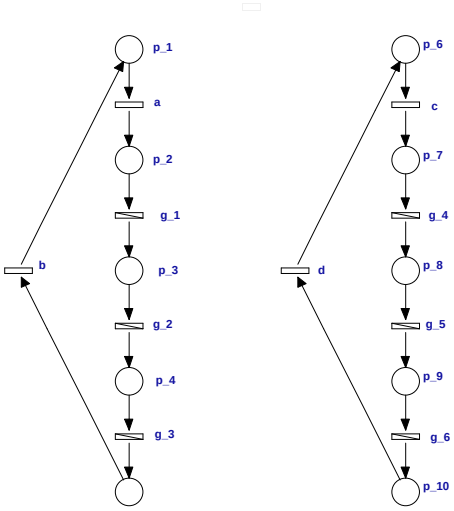


Figura 7.8: rete di Petri di un sistema concorrenziale

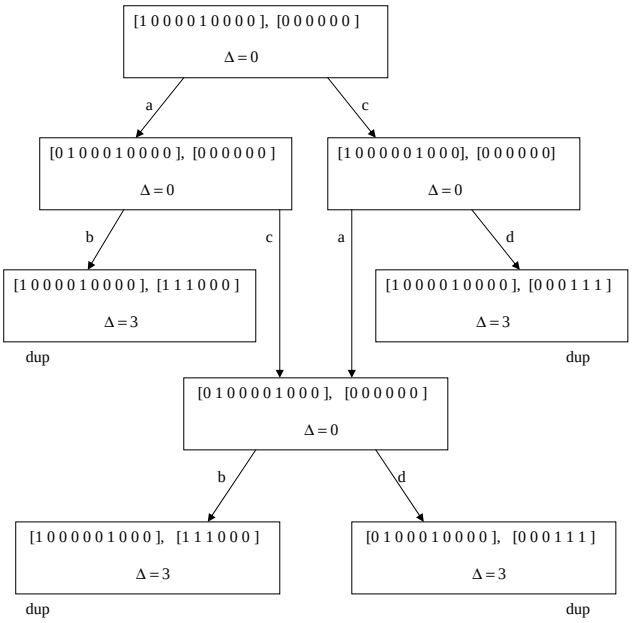


Figura 7.9: BRT della rete di Petri di figura 7.8

Capitolo 8

Conclusioni

8.1 Conclusioni

In questa tesi abbiamo affrontato il problema della diagnosi dei guasti. Abbiamo esposto l'approccio di Lafortune, che si basa sugli automi, introducendo le nozioni di diagnosticabilità e di $I-$diagnosticabilità di un linguaggio. Abbiamo dato le regole per la costruzione del diagnosticatore e presentato le condizioni necessarie e sufficienti per la diagnosticabilità e la $I-$diagnosticabilità. Abbiamo poi esposto l'approccio di Giua-Seatzu, basato sulle reti di Petri. Abbiamo caratterizzato le sequenze di scatto corrispondenti a una data osservazione, con le nozioni di marcature di base e giustificazioni. Abbiamo proposto un algoritmo per il calcolo delle marcature di base e lo abbiamo usato per determinare un automa deterministico, che abbiamo chiamato albero base di raggiungibilità, che può essere usato come diagnosticatore. Inoltre abbiamo confrontato i due approcci trovando delle importanti similarità e arrivando ad un importante risultato, che è stato accompagnato da un originale teorema. Questo teorema dimostra che le ipotesi di aciclicità imposte dai due approcci, pur partendo da esigenze diverse, sono sostanzialmente equivalenti. Infine abbiamo presentato nel penultimo capitolo un toolbox MATLAB per la diagnosi mediante reti di Petri.

Come si è fatto osservare nei precedenti capitoli l'approccio con le reti di Petri risulta più conveniente nel caso di un sistema concorrenziale, infatti con questo approccio non dobbiamo enumerare tutti gli stati. Inoltre se siamo interessati non solo alla conoscenza della presenza di un guasto, ma anche al fatto di quante volte esso si è manifestato nel sistema dobbiamo per forza adottare l'approccio con le

reti di Petri in quanto quello con gli automi non ci fornisce tale informazione.

8.2 Sviluppi futuri

I possibili sviluppi di questo lavoro sono tanti. L'approccio con le reti di Petri presenta molte possibilità di sviluppo.

Sarebbe utile costruire un programma che assegna ad ogni nodo del diagnosticatore la sua etichetta di guasto.

Sarebbe interessante provare a vedere se togliendo l'ipotesi di limitezza della rete è ancora possibile costruire un diagnosticatore. In questo caso ci aspettiamo che al posto dell'albero di raggiungibilità possa essere costruito un grafo di copertura.

Sarebbe utile formulare un algoritmo per il calcolo del Δ introdotto nella definizione 5.5.1 e dare una funzione di propagazione delle etichette sulla base del Δ.

Appendice A

Listati dei programmi

A.1 Calcolo delle spiegazioni minime

```
function B = miny(Pre,Post,M,t,nu)

% Help[...]


[p,tn]=size(Pre);
[x,c]=size(M);
%controllo delle grandezze in ingresso
if (size(Post)==size(Pre))&(c==1)&(p==x),
    elseif size(Post)~=size(Pre)
     fprintf('\n ERRORE!! Le matrici Pre  e Post hanno
               dimensione diversa!\n')
     fprintf('\n Inserisci nuovamente i dati\n')
      else
      fprintf('\n ERRORE! le dimensioni della
                 marcatura sono sbagliate!\n')
end

C=Post-Pre;
Cu=(C(:,(tn-nu+1):tn));
A=M-Pre(:,t);
[nu,p]=size(Cu');
```

```
Q=[Cu' eye(nu);A' zeros(1,nu)];
[m,n]=size(Q);
S=Cu';
h=[];


while ~isempty(find(Q(nu+1:m,1:p)<0)) % se tutti gli
         % elementi di A sono >=0 esci dal ciclo

h=find(Q(nu+1:m,1:p)<0);% dà la posizione in cui sono
                  % contenuti gli elementi <0 in A
  [q,w]=size(h);

% scelgo l'elemento < 0 di A in maniera casuale
  w1=w*rand+.001;
  w1=ceil(w1);
  w1;
  if w1>w,
   w1=w;
  end;

 for j=w1:w1

o=h(q,j);
u=ceil(o/(m-nu));% mi seleziona la colonna dove ho
            %trovato l'elemento minore di zero
i= o-((m-nu)*(u-1))+nu;% mi seleziona la riga dove
            %ho trovato l'elemento minore di zero
y1=find(S(:,u)>0);% y1 è l'insieme contenente
           % gli indici di riga per  cui Cu'(i,j*)>0
[l,g]=size(y1);

    if  isempty(y1)
     Q(i,:)=[]; % rimuovo la riga i
     Q;
     [m,n]=size(Q);

      else

        for c=1:l
```

```
f=y1(c,g);% selez. gli elementi del vettore colonna y1
k=ceil(abs(Q(i,u)/Q(f,u) ));%trova l'intero inferiore
      %tale che A(i*,j*)+k*Cu'(i,j*)>=0
Z=[Q(i,:)]+k*[Q(f,:)];% nuova riga da aggiungere
Q=[Q ;Z];%  aggiungo la nuova riga (Z)
[m,n]=size(Q);

        end
            Q(i,:)=[];
            Q;
            [m,n]=size(Q);

     end
 end
end

% quando esco da questo ciclo sono
%nella condizione in cui tutti
%gli elementi di A sono >=0
%devo ora rimuovere da B ogni riga
%che copre le altre righe il
%risultato saranno le spiegazioni
%minime Ymin(M,t).


B = Q(nu+1:m,p+1:n); % matrice contenente le spiegazioni
%(potrebbero già essere minimali ma bisogna verificarlo)


if   isempty (B)
   fprintf('la transizione t non può essere
       abilitata dalla marcatura data \n')
end

[s,t]=size(B); m=1; C==zeros(m,1);
%controllo che le spiegaz. trovate siano minime
for i=1:s
   for j=1:s
      if i~=j
      P=B(i,:)-B(j,:);
```

```
        if P>=0
        k=i;% indici di riga della matrice B da eliminare

        C(m,1)=k;
        m=m+1;

        end
       end
  end
end


if C>0
B(C(:,1),:)=[];
  fprintf ('le spiegazioni minime sono:\n')
  disp(B)
    else  if ~isempty(B) % se le spiegazioni
     %erano già minime e quinci C=0
     fprintf ('le spiegazioni minime sono:\n') disp(B)
    end
end

if B==zeros(s,nu)
   fprintf('cioè la transizione scelta è subito
   abilitata dalla\n marcatura data senza lo
   scatto di transizioni non osservabili')

end
```

A.2 Calcolo dell'albero base di raggiungibilità

```
function T = BRT(Pre,Post,M0,nu,B)

% Help[...]
```

```
Mx=M0';
[m,n]=size(Post);
[c,p]=size(Mx);
C=Post-Pre;
Co=C(:,1:n-nu);
Cu=(C(:,(n-nu+1):n));


%verifica dell'esattezza delle
%dimensioni dei dati in ingresso
if (size(Post)==size(Pre))&(c==1)&(p==m),
     elseif size(Post)~=size(Pre)
   fprintf('\n ERRORE!! Le matrici Pre
          e Post hanno dimensione diversa!\n')
   fprintf('\n Inserisci nuovamente i dati\n')
      else
   fprintf('\n ERRORE! le dimensioni della
         marcatura sono sbagliate!\n')
end


T=[{1} {[Mx]} {[zeros(1,nu)]} {[zeros(1,(n-nu))]}
{[zeros(1,(n-nu))]} {[0]}];
%prima riga della matrice che rappresenta il BRT
c=cell2mat(T(:,6));
  d=find(c==0);

while ~isempty(d)
% fintanto che esiste almeno un nodo senza tag
d=find(c==0);% mi da il numero del nodo da esplorare
for j=1:(n-nu)%for sulle transiz. osservabili

v=1;%mi serve per creare il nuovo nodo
%considero l'insieme delle marcature del
%nodo non esplorato
MM=T{(d(1,1)),2};
[a,b]=size(MM);
B1=[];

     for i=1:a
     % prendo ad una ad una le marcature del nodo
```

```
f=MM(i,:);
f1=f';

   B = miny(Pre,Post,f1,j,nu); %calcolo le spiegazioni
   %minime per una data transizione per tutte le M del nodo
if ~isempty(B)
  B1=B;%B1 mi serve affinchè B non sia sovrascritto
  %dall'insieme vuoto (capita quando la prima
  %marcatura del nodo ha un certo B, mentre
  %la seconda ha come spiegaz. minima l'insieme vuoto)
end

      [s,t]=size(B);
      if isempty(B)

      else % la transizione è abilitata
     % caso di più giustificazioni per una data marcatura
       for h=1:s


       Mprimo=  f + Co(:,j)'+ (Cu*B(h,:)')';
       % nuova marcatura T{d,2}(i,:)
       yprimo=T{d(1,1),3}(i,:)+B(h,:);
       %nuova giustificazione

       P(v,1:m)=Mprimo; % P è la matrice che
       %contiene nelle sue righe le marcature del nodo
       Y(v,1:nu)=yprimo;% Y è la matrice che contiene
      %nelle sue righe le spiegazioni minime
      %delle marcature del nodo
      v=v+1;

       T{d(1,1),4}(i,j)=1; % setto a uno il bit
   %della transizione se questa è abilitata dal nodo

         end
      end
 end

   if ~isempty(B)   | ~isempty(B1)
   e1=cell2mat(T(:,1));
```

```
        e=max(e1);%numero di nodi già creati

 %controllo che all'interno del nodo
 %non ci siano più marcature identiche
        [o,m]=size(P);
        [o,nu]=size(Y);

        r=1;
C=zeros(r,1);

        for z=1:o
            for w=1:o
                if z~=w
                    D=P(w,:)-P(z,:);
                    if D==zeros(1,m)
                        E=Y(z,:)-Y(w,:);
                        if E==zeros(1,nu)
                            k=i;
                            C(r,1)=k;
                            r=r+1;
                        end
                    end
                end
            end
        end

if C==0  %non ci sono marcature
%identiche all'interno del nodo

elseif ~isempty(C)
%C indica la riga da eliminare
%perchè già contenuta nel nodo
P(C(:,1),:)=[]; Y(C(:,1),:)=[]; else
  break
  end

% devo riordinare le righe del nodo
%per il confronto successivo dei nodi duplicati
  U=sortrows([P Y]);
  P1=U(:,1:m);
  Y1=U(:,m+1:m+nu);
```

```
  T{d(1,1),5}(i,j)=e+1;

T=[T ; {e+1} {[P1]} {[Y1]} {[zeros(1,(n-nu))]}
 {[zeros(1,(n-nu))]} {0}];

    end
  end


  % controllo che il nodo non sia dead.
  x=cell2mat(T(d(1,1),4));
   if x(1,:)==[zeros(1,(n-nu))]
    T{d(1,1),6}(1,1)=-1;%il nodo ha il tag 'dead'
      else
      T{d(1,1),6}(1,1)=1;  % il nodo ha il tag
      'vecchio' per indicare che quel nodo
      è già stato esplorato
   end


   %controllo che il nodo non sia duplicato
  [nn,mm]=size(d);
  for i=1:nn
     for  f=d(i,1)+1:e+1

      if f<=e+1
           MM1=T{f,2};
           [a1,b]=size(MM1);

       for q=1:d
            MM=T{q,2};
            [a,b]=size(MM);
          if (a==a1) % se i nodi hanno
          %diverso numero di marcature
          %saranno sicuramente diversi
            r=max(max(MM~=MM1));
              if r==0 % il nodo che sto
              %considerando è duplicato
             T{f,6}=2;%il nodo ha il tag 'dup'
              end
          end
```

```
        end
     end
   end
end

% metto il contenuto dell'ultima
%colonna di T in un vettore
    c=cell2mat(T(:,6));
    d=find(c==0)  ;

end
```

Appendice B

Manuali dei programmi

B.1 Manuale del programma ″Calcolo delle spiegazioni minime″

La funzione **miny** calcola le spiegazioni minime di una rete di Petri. Una spiegazione minima indica la più piccola sequenza di transizioni non osservabili che devono scattare per spiegare una data osservazione. La sintassi di questa funzione è `B=miny(Pre,Post,M,t,nu)`. Questa funzione prende in ingresso le matrici Pre, $Post$, una marcatura M, una transizione t e il numero di transizioni non osservabili nu e restituisce una matrice B che contiene nelle sue righe le spiegazioni minime della rete data. Le matrici Pre e $Post$ dovranno contenere nelle prime colonne le transizioni osservabili e nelle ultime quelle non osservabili.

La funzione provvede nella prima parte dell'algoritmo alla verifica dell'esattezza delle dimensioni dei dati in ingresso:

- le matrici Pre e $Post$ dovranno avere la stessa dimensione $m \times n$;
- la marcatura M, che sarà un vettore colonna, dovrà avere lo stesso numero di righe delle matrici Pre e $Post$.

Se queste condizioni non sono verificate verranno stampati dei messaggi di errore nella finestra dei comandi di Matlab quando la funzione verrà lanciata.

Per prima cosa viene definita la matrice su cui lavorerà il programma:

$$\Gamma := \left|\begin{array}{c|c} C_u^T & I_{n_u \times n_u} \\ \hline A & B \end{array}\right|.$$

C_u sono le colonne della matrice di incidenza corrispondenti alle transizioni non osservabili, mentre la matrice A è pari a $A^T := M - Pre(\cdot, t)$.
Il programma ha inizio con un ciclo *while* che termina quando tutti gli elementi di A sono maggiori o uguali a zero. Il programma seleziona le righe di A aventi almeno un elemento minore di zero. Scelta una riga e considerato l'elemento minore di zero (i^*, j^*) (o se ce n'è più di uno ne prendo uno a caso), guardo nella colonna corrispondente della matrice C_u^T se esistono elementi maggiori o uguali a zero. Se non ne esistono elimino direttamente la riga di A selezionata. Se invece ne esistono cerco l'intero minimo k tale che $A(i^*, j^*) + k * Cu'(i, j^*) \geq 0$ e sommo le due righe; il risultato della somma sarà una nuova riga di A, mentre la riga di A appena considerata viene eliminata. Quando esco dal ciclo *while* la matrice B conterrà le spiegazioni che stiamo cercando. L'ultima operazione che fa il programma è la verifica che tali spiegazioni siano minime.
La funzionalità del programma è stata testata ed alcuni esempi sono stati proposti nel capitolo 7.

B.2 Manuale del programma ″Calcolo del BRT″

La funzione **BRT** calcola l'albero di raggiungibilità delle marcature di base di una rete di Petri. La sintassi di questa funzione è `T = BRT(Pre,Post,M0,nu,B)`. Questa funzione prende in ingresso le matrici Pre, $Post$, la marcatura iniziale M_0, il numero di transizioni non osservabili nu e e la matrice delle spiegazioni minime B associata alla marcatura M_0. L'uscita è una matrice T che contiene nelle sue righe tutte le informazioni per disegnare l'albero base di raggiungibilità. Questo programma utilizza per la matrice T un array di celle, ossia ogni elemento di questa matrice che viene detto ″cella″ a sua volta può contenere un array. T ha un numero di righe pari al numero di nodi del BRT e un numero di colonne pari a 6. Il contenuto delle colonne è il seguente:

1. la prima colonna contiene uno scalare che rappresenta il numero del nodo;

2. la seconda colonna contiene una matrice che racchiude nelle sue righe le marcature di base del nodo considerato;

3. la terza colonna contiene una matrice che racchiude nelle sue righe le giustificazioni relative alle marcature di base del nodo considerato;

4. la quarta colonna contiene una matrice con un numero di righe pari al numero di marcature del nodo considerato ed un numero di colonne pari al numero delle transizioni osservabili. La quarta colonna indica le transizioni che vengono abilitate dal nodo;

5. la quinta colonna contiene una matrice con un numero di righe pari al numero di marcature del nodo considerato ed un numero di colonne pari al numero delle transizioni osservabili. La quinta colonna indica le marcature che vengono raggiunte dal nodo;

6. la sesta ed ultima colonna contiene uno scalare che rappresenta l'etichetta del nodo. L'etichetta indica che il nodo è stato esplorato (ed in questo caso vale 1), oppure che il nodo è morto (ed in questo caso vale -1), oppure che il nodo è duplicato (ed in questo caso vale 2).

La funzione provvede nella prima parte dell'algoritmo alla verifica dell'esattezza delle dimensioni dei dati in ingresso:

- le matrici Pre e $Post$ dovranno avere la stessa dimensione $m \times n$;
- la marcatura M, che sarà un vettore colonna, dovrà avere lo stesso numero di righe delle matrici Pre e $Post$.

Se queste condizioni non sono verificate verranno stampati dei messaggi di errore nella finestra dei comandi di Matlab quando la funzione verrà lanciata.

La prima cosa che fa il programma è creare il primo nodo dell'albero base di raggiungibilità che sarà composto dalla marcatura iniziale e avrà tutti i campi uguali a zero:

$$T = [\{1\} \quad \{[Mx]\} \quad \{[zeros(1, nu)]\} \quad \{[zeros(1, (n - nu))]\}$$

$$\{[zeros(1, (n - nu))]\} \quad \{0\}]$$

La marcatura Mx non è altro che M_0^T. Le istruzioni per la costruzione dell'albero sono tutte racchiuse all'interno di un ciclo *while* dal quale esco solo se tutti i nodi hanno un'etichetta, ossia se l'ultima colonna ha tutti gli elementi diversi da zero. Ogni volta che seleziono un nodo senza etichetta vado a vedere quali transizioni

sono abilitate dal nodo, quale è il nodo di destinazione, quali sono le spiegazioni minime del nodo e inserisco tutte queste informazioni all'interno della matrice T. Seguono poi nel programma una serie di controlli. Il primo controllo verifica che all'interno del nodo non vi siano più marcature identiche. Il secondo e il terzo controllo verificano rispettivamente che il nodo non sia un nodo morto o duplicato. In questo caso nell'ultima colonna dovrò inserire rispettivamente il valore -1 o 2. La funzionalità del programma è stata testata ed alcuni esempi sono stati proposti nel capitolo 7.

Bibliografia

[1] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, D. Teneketzis. *Diagnosability of Discrete-Event Systems*. In *Proc. IEEE Transaction on Automatic Control , 9 settembre 1995.*

[2] A. Giua, C. Seatzu. *Fault detection for discrete event systems using Petri nets with unobservable transitions*. In *Proc. IEEE 44rd Int. Conf. on Decision and Control (Siviglia), dicembre 2005.*

[3] D. Corona, A. Giua, C. Seatzu. *Marking estimation of Petri nets with silent transitions*. In *Proc. IEEE 43rd Int. Conf. on Decision and Control (Atlantis, The Bahamas), dicembre 2004.*

[4] A. Aghasaryan, E. Fabre, A.Benveniste, R. Boubour, C. Jard. *Fault detection and diagnosis in distribuited systems: an approuch by partial stochastic Petri nets*. In *Discrete Events Dinamical Systems , giugno 1998.*

[5] R.K.Boel, G. Jiroveanu. *Distributed contextual diagnosis for very large systems*. In *Proc. IFAC WODES'04: 7th Work. on Discrete event systems (Reims, France) , settembre 2004.*

[6] R.K.Boel, J.H. Shuppen. *Decentralized failure diagnosis for discrete event systems with costly communication between diagnosers*. In *Proc. IFAC WODES'02: 6th Work. on Discrete event systems (Zaragoza, Spain) , ottobre 2002.*

[7] C. G. Cassandras, S. Lafortune. *Introduction to Discrete Event Systems*. In *Kluwer Academic Publisher 1999.*

[8] S. Lafortune *Comunicazione personale, novembre 2005.*

[9] J. Martinez, M. Silva. *An simple and fast algorithm to obtain all invariants of a generalized Petri net*. In *Girault, C. e Reisig, W., editors. 1982.*