RECENTI SVILUPPI METODOLOGICI NELL'ANALISI DEGLI ALBERI DI GUASTO

S. Contini
Commissione Europea, Centro Comune Ricerche
Istituto dei Sistemi dell'Informatica e della Sicurezza
TP 650, 21020 Ispra, Varese

SOMMARIO

La presente memoria descrive gli sviluppi metodologici piu` recenti nell'analisi logica e probabilistica degli alberi di guasto. Viene presentato un nuovo approccio per la manipolazione di una funzione booleana rappresentata sotto forma di grafo aciclico direzionato. L'analisi probabilistica eseguita sul grafo consente di ottenere i valori esatti di indisponibilita`, frequenze di guasto e riparazione del sistema, oltre gli indici di importanza probabilistica dei componenti. I risultati dell'applicazione del software ASTRA-FTA ad alcuni alberi di sistemi reali mostrano la maggiore efficienza del nuovo approccio rispetto ai precedenti.

1. INTRODUZIONE

La metodologia dell'Albero dei Guasti (Fault Tree) trova ampio impiego nell'analisi di affidabilita` e disponibilita` di sistemi complessi. Proposta agli inizi degli anni '60, essa ha avuto uno sviluppo notevole in campo nucleare, in particolare a partire dalla pubblicazione, nel 1969, della teoria probabilistica (KInetic Tree Theory, KITT) [1]. Questa teoria, arricchita negli anni '70, consente di calcolare "upper bounds" per l'indisponibilita`, la frequenza di guasto, il numero medio di guasti e gli indici di importanza dei componenti.

L'applicazione della KITT richiede la conoscenza dei modi di guasto minimi del sistema (Minimal Cut Sets, MCS), ossia delle combinazioni minime di componenti il cui guasto implica il guasto del sistema. Sono stati pertanto messi a punto numerosi metodi per la determinazione degli MCS, basati su diversi approcci: manipolazione di cut sets con riduzione dell'albero di tipo Top-down, Bottom-up e Ibrida, pattern recognition ed altri.

Il problema della determinazione degli MCS e` alquanto complesso, basti pensare che un albero con qualche decina di porte logiche e di eventi primari puo` contenere anche decine di migliaia di MCS. Diverse tecniche sono state messe a punto per ridurre i tempi di calcolo, tra cui le piu` importanti sono la modularizzazione (l'albero viene scomposto in tanti sottoalberi indipendenti) e le tecniche di taglio (al fine di limitare il calcolo al sottoinsieme di MCS probabilisticamente e/o strutturalmente piu` significativi). Il fatto di limitare il calcolo ai soli MCS significativi implica la necessita` di stimare accuratamente l'errore di troncamento Pe (problema di notevole complessita`).

Il contributo del CCR-Ispra a queste ricerche e' testimoniato dallo sviluppo della serie di programmi SALP [2, 3] basati sull'approccio Bottom-up e ISPRA-FTA [4, 5] basato su un approccio ibrido (Top-down + Bottom-up), appositamente sviluppato per ottenere una stima accurata di Pe.

Un recente approccio all'analisi degli alberi di guasto, che puo` essere considerato di gran lunga piu` efficiente dei precedenti, si basa sulla manipolazione di funzioni booleane rappresentate sotto forma di grafi aciclici direzionati, denominati Binary Decision Diagrams (BDD). Concettualmente, la rappresentazione in termini di BDD di una funzione booleana si ottiene manipolando un albero binario generato dall'applicazione del teorema di espansione di Shannon rispetto a tutte le variabili. I cammini dalla radice ai vertici terminali del grafo rappresentano i modi di guasto del sistema; essi sono disgiunti e di conseguenza risulta possibile calcolare esattamente le grandezze probabilistiche di interesse. Il grafo puo` essere ulteriormente manipolato fino ad ottenere un nuovo grafo contenente tutti e soli gli MCS.

I confronti effettuati fra algoritmi basati su BDD e sulla manipolazione di cut set, hanno mostrato l'elevata efficienza dei primi rispetto ai secondi. Alberi analizzati in passato solo mediante l'applicazione di tecniche semi automatiche o evitando la stima di Pe, sono stati analizzati con l'approccio BDD in pochi secondi senza introdurre alcuna approssimazione. A titolo di esempio, l'albero del sistema descritto in [6], contenente circa $1{,}2\ 10^7$ MCS e non analizzabile con i metodi tradizionali, e' stato analizzato con ASTRA-FTA, sviluppato presso CCR-ISIS, in circa un secondo su PC 486 a 200 Mhz.

La presente memoria descrive gli aspetti piu` importanti dei metodi di analisi logica e probabilistica basati sull'approccio BDD e la dimostrazione dell'efficienza di tale approccio mediante l'applicazione del programma di calcolo ASTRA-FTAad alcuni alberi complessi di sistemi reali.

2. RAPPRESENTAZIONE E MANIPOLAZIONE DI FUNZIONI BOOLEANE MEDIANTE I "BINARY DECISION DIAGRAMS"

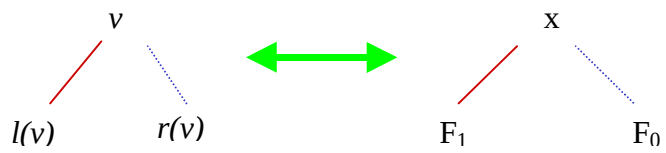
2.1 Generalita`

Una qualsiasi funzione booleana F (x_1, x_2, ..., x_n) puo` essere scritta nel modo seguente:

$$F(x_1, x_2, ..., x_n) = (x_i \wedge F_1) \acute{U} (\neg x_i \wedge F_0) \qquad (1)$$

dove:
$F_1 = F|_{xi = 1} = F(x_1, x_2, .. , x_{i-1}, 1, x_{i+1},..., x_n)$
$F_0 = F|_{xi = 0} = F(x_1, x_2, .. , x_{i-1}, 0, x_{i+1},..., x_n)$
sono i due residui di F, "$\wedge$" rappresenta l'operatore logico AND, "*Ú*" rappresenta l'operatore logico OR, e "$\neg$" l'operatore NOT.

La (1) puo` essere graficamente rappresentata come albero binario nel modo seguente:



dove al nodo *v* e` associata la variabile x, *var(v)* = x, il discendente sinistro *l(v)* = F_1 e il discendente destro *r(v)* = F_0.
Ad esempio, il risultato dell'applicazione della (1) alla funzione F = *(a Ú (¬b Ù c)) Ù (b Ú c)* rispetto alle variabili *a*, *b* e *c*, puo` essere rappresentato graficamente con l'albero riportato in Figura 1, nel quale sono descritte, per maggior chiarezza, le espressioni dei residui ottenuti assegnando i valori 1 (linea continua) e 0 (linea trattegiata) alle variabili. I *nodi* dell'albero ai quali sono associate le variabili *a*, *b*, *c*, sono detti *non-terminali* e i nodi **0**, **1**, che rappresentano i valori della funzione F per diverse combinazioni di valori delle variabili, sono detti *nodi terminali*.

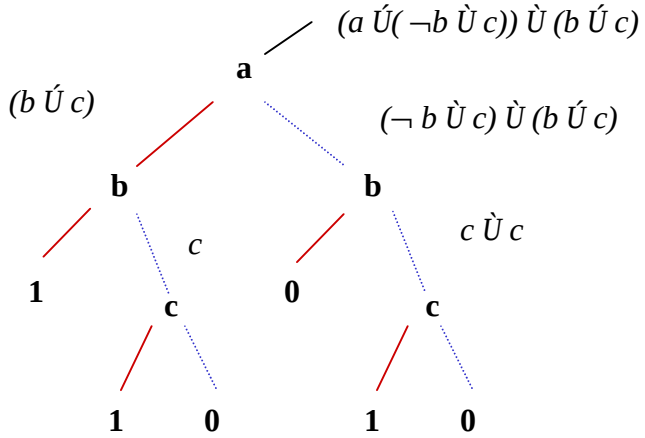


Figura 1. Rappresentazione sotto forma di albero binario della funzione F =*(a Ú (¬b Ù c)) Ù (b Ú c)*

Nell'albero, un generico cammino dal vertice **1** alla radice rappresenta un insieme di valori che soddisfa F. Per esempio, a = 1, b = 0, c = 1 e` un assegnamento che soddisfa F. Sia S l'insieme di tutti i cammini da 1 alla radice, S = {a b, a ¬b c, ¬a ¬b c}. Un generico elemento di S e` un implicante di F. L'insieme dei cut sets M = {a b, a c, c} si ottiene da S eliminando le variabili negate.

L'albero in Figura 1 puo` essere ridotto applicando le seguenti *regole di riduzione:*
R1: Raggruppamento dei nodi terminali aventi lo stesso valore (**0**, **1**).
R2: Raggruppamento di nodi non-terminali; due nodi *u* e *v* possono essere raggruppati in un unico nodo se:
var(u) = var(v), l(u) = l(v) e r*(u) = r(v).*

R3: Eliminazione dei nodi ridondanti; un nodo *v* e' ridondante se *l(v) = r(v).*

L'applicazione di queste regole all'albero di Figura 1 fornisce il grafo riportato in Figura 2, denominato Binary Decision Diagram, BDD.

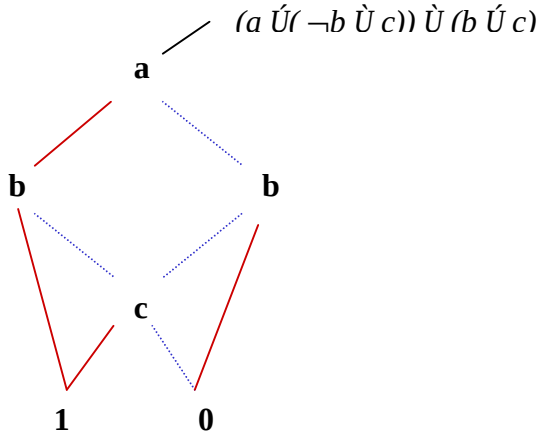


Figura 2. Grafo risultante dall'applicazione delle regole di riduzione all'albero binario di Figura 1

2.2 Ordinamento delle variabili

Il BDD in Figura 2 e' stato ottenuto espandendo la funzione F rispetto alle variabili secondo l'ordine *a, b* e *c*. Sia "<" il simbolo di ordinamento; x < y significa che la (1) viene applicata prima a x e poi a y e di conseguenza, nel grafo, x non potra` mai discendere da y. Si puo` dimostrare facilmente che cambiando l'ordinamento delle variabili cambia la complessita' del BDD risultante.

Ad esempio, nella funzione $F = (x_1 \wedge y_1) \vee (x_2 \wedge y_2) \vee \ldots \vee (x_n \wedge y_n)$ adottando l'ordinamento $x_1 < y_1 < x_2 < y_2 \ldots < x_n < y_n$ si ottiene un BDD con 2n vertici non terminali, mentre con l'ordinamento $x_1 < x_2 < \ldots x_n < y_1 < y_2 < \ldots < y_n$ il numero di vertici e` pari a $2(2^n - 1)$ [ 7 ]. L'ordinamento delle variabili deve pertanto essere scelto con cura in base al tipo di problema: un ordinamento non appropriato puo` ridurre notevolmente l'efficienza dell'approccio. La convenienza di generare grafi BDD ordinati (spesso indicati in letteratura come OBDD, Ordered Binary Decision Diagrams) e` dovuta al fatto che le operazioni di riduzione e minimizzazione risultano molto efficienti [ 7 ] e che il grafo risultante e` sempre in forma canonica.

2.3 Notazione ite

Per la descrizione delle operazioni logiche sui grafi e` conveniente introdurre la notazione ite (if-then-else). Un ite e` una tripla che rappresenta un nodo *v* del grafo:

ite(*var(v), l(v), r(v)*)

Secondo questa notazione, la (1) si scrive come: ite(x_i, F_1 , F_0).

Di seguito sono riportati alcuni esempi di semplici funzioni booleane in notazione ite, facilmente ricavabili applicando la (1) e le regole di riduzione:

F = x	ite(x, 1, 0)
F = x *Ú* y	ite(x, 1, ite(y, 1, 0))
F = x *Ù* y	ite(x, ite(y, 1, 0), 0)

L'espressione in notazione ite corrispondente al grafo di Figura 2 e` la seguente:

ite(a, ite(b, 1, ite(c, 1, 0)), ite(b, 0, ite(c,1 0)))

2.4 Composizione di funzioni

La procedura di costruzione del BDD attraverso l'albero binario non e` praticamente applicabile a funzioni con un numero significativo di variabili. E, quindi, essenziale definire regole di composizione di

funzioni secondo gli operatori binari dell'algebra booleana. Siano F e G due generiche funzioni booleane; esse possono essere composte in un'unica funzione mediante la seguente relazione:

$$F \langle op \rangle G = x\,(F|_{x=1} \langle op \rangle G|_{x=1}) + \neg x\,(F|_{x=0} \langle op \rangle G|_{x=0}) \qquad (3)$$

dove <op> rappresenta un generico operatore binario (AND, OR, XOR) e x una generica variabile.

Piu` specificatamente, indicando con $F = ite(x, F_1, F_0)$ e $G = (y, G_1, G_0)$, due funzioni booleane qualsiasi in notazione ite, possono verificarsi i seguenti casi:

x = y

$$ite(x, F_1, F_0) \langle op \rangle ite(x, G_1, G_0) = ite(x, F_1 \langle op \rangle G_1, F_0 \langle op \rangle G_0) \qquad (4)$$

x < y

$$ite(x, F_1, F_0) \langle op \rangle ite(y, G_1, G_0) = ite(x, F_1 \langle op \rangle ite(y, G_1, G_0), F_0 \langle op \rangle ite(y, G_1, G_0)) \qquad (5)$$

Per completezza si riporta l'operatore complementazione. Se $F = ite(x, F_1, F_0)$, allora $\neg F = ite(x, F_0, F_1)$. La complementazione si ottiene quindi invertendo i puntatori ai due residui.

2.5 Esempio di costruzione del BDD di una funzione booleana

Come esempio di applicazione della (3) e delle operazioni di riduzione, si mostra di seguito la costruzione del BDD relativo alla funzione $F = (a \vee (\neg b \wedge c)) \wedge (b \vee c)$.
Sia a < b < c l'ordinamento delle variabili.
La sequenza di costruzione del BDD, a partire dalle parentesi piu' iterne, e' mostrata nella seguenta tabella.

Espressione booleana	**Espressione in notazione *ite* da calcolare**	**Risultato del calcolo**
b	ite(b, 1, 0)	Ite(b,1,0)
$\neg b$	$\neg$ ite(b, 1, 0)	ite(b, 0, 1)
C	ite(c, 1, 0)	Ite(c,1,0)
$\neg b \wedge c$	ite(b, 0, 1) $\wedge$ ite(c, 1, 0)	ite(b, 0, ite(c, 1, 0))
$a \vee (\neg b \wedge c)$	ite(a, 1, 0) $\vee$ ite(b, 0, ite(c,1, 0))	ite(a,1, ite(b, 0, ite(c,1, 0)))
$b \vee c$	ite(b, 1, 0) $\vee$ ite(c, 1, 0)	ite(b, 1, ite(c, 1, 0))
$(a \vee \neg b \wedge c) \wedge (b \vee c)$	ite(a,1, ite(b, 0, ite(c,1, 0))) $\wedge$ ite(b, 1, ite(c, 1, 0))	ite(a, ite(b, 1, ite(c, 1, 0)), ite(b, 0, ite(c, 1, 0)))

Tabella 1. Costruzione del BDD della funzione $F = (a \vee (\neg b \wedge c)) \wedge (b \vee c)$

La rappresentazione in notazione ite delle singole variabili e' immediata, come pure quella relativa alla combinazione di funzioni semplici $(b \vee c)$ e $(\neg b \wedge c)$. La funzione $a \vee (\neg b \wedge c)$ si ottiene applicando la (5) in quanto a < b.

La funzione calcolata nell'ultima riga della tabella richiede i passaggi seguenti. Applicando la (5) con x=a, si ottiene:

$$ite(a,1, ite(b, 0, ite(c,1, 0))) \wedge ite(b, 1, ite(c, 1, 0)) = ite(a,1 \wedge ite(b, 1, ite(c, 1, 0)), ite(b, 0, ite(c,1, 0)) \wedge ite(b, 1, ite(c, 1, 0)))$$

L'analisi del primo residuo fornisce:

$$1 \wedge ite(b, 1, ite(c, 1, 0)) = ite(b, 1, ite(c, 1, 0))$$

Per il secondo residuo, applicando la (4) rispetto a b:

$$ite(b, 0, ite(c,1, 0)) \wedge ite(b, 1, ite(c, 1, 0)) = ite(b, 0 \wedge 1, ite(c,1, 0) \wedge ite(c, 1, 0))$$

Poiche` :

$0 \wedge 1 = 0$ e

$ite(c,1, 0) \wedge ite(c, 1, 0) = ite(c,1, 0)$

si ottiene:

$ite(b, 0, ite(c,1, 0))$

Il risultato finale e` il seguente:
ite(a, ite(b, 1, ite(c, 1, 0)), ite(b, 0, ite(c, 1, 0)))

3. APPLICAZIONE DEI BDD ALL'ANALISI DEGLI ALBERI DI GUASTO

L'applicazione dell'approccio BDD all'analisi degli alberi di guasto rappresenta il tentativo piu` recente di messa a punto di strumenti di calcolo in grado di analizzare alberi molto complessi, non analizzabili in modo soddisfacente con gli approcci tradizionali.

Una dimostrazione delle maggiori potenzialita' dei BDD rispetto ai metodi basati sulla manipolazione di cut sets (riduzione top-down o bottom-up dell'albero) e` fornita in [ 8 ]. Il programma ISPRA-FTA, nella sua versione originale e` stato confrontato con una versione modificata nella quale, per la fase di analisi piu` pesante (analisi top-down senza possibilita' di utilizzo della tecnica di cut-off), si e` fatto uso dello stesso algoritmo top-down basato pero` sull'approccio BDD. Le prove effettuate su un insieme di alberi di diversa complessita` hanno mostrato una riduzione dei tempi di calcolo fino a due ordini di grandezza per il calcolo degli MCS significativi (come prevedibile, per alberi meno complessi, le differenze non sono risultate significative).

Nel seguito vengono mostrati i metodi utilizzabili per l'analisi di alberi di guasto rappresentati da funzioni monotone (*fault tree coerenti*) e non monotone (*fault tree non coerenti*).

3.1 Analisi di funzioni booleane monotone

Un fault tree riconducibile ad una funzione contenente i soli operatori AND, OR rappresenta una funzione monotona crescente.

La procedura di analisi di tale tipologia di funzioni secondo l'approccio BDD, si articola nelle seguenti fasi:

1. definizione dell'ordinamento delle variabili;
2. costruzione della rappresentazione BDD dell'albero dei guasti;
3. riduzione del BDD;
4. esecuzione dell'analisi probabilistica;
5. determinazione del BDD minimo contenente tutti gli MCS;
6. estrazione degli MCS significativi.

La prima fase e` relativa alla definizione dell'ordinamento degli eventi primari (l'ordinamento secondo il metodo di visita dell'albero in ordine differito si e` rivelato soddisfacente). La seconda fase consiste nella costruzione del BDD mediante l'applicazione della (3) rispetto a tutti gli eventi primari. La fase di riduzione del grafo (applicazione delle regole R1, R2 e R3) viene in realta` eseguita, per motivi di efficienza, durante la costruzione stessa del grafo. Sull'albero ridotto viene eseguita l'analisi probabilistica. Successivamente si calcola il BDD minimo contenente i soli MCS e, infine, in base ai valori di soglia definiti in input, vengono determinati gli MCS.

Le prime tre fasi sono gia` state descritte: di seguito vengono brevemente descritte le fasi relative all'analisi probabilistica ed al calcolo del BDD minimo.

Analisi probabilistica applicata al BDD ridotto

La quantificazione del BDD consente di ottenere i *valori esatti* dell'indisponibilita` del sistema $Q_S(t)$ e della frequenza di guasto $\omega_S(t)$.
Sia ite(x, F_1, F_0) il generico nodo del grafo. L'indisponibilita` si ottiene applicando la seguente formula a tutti i nodi del BDD, partire dai nodi terminali e procedendo verso la radice:

$$Q_U(t) = Q_X(t)\, Q_1(t) + [1 - Q_X(t)]\, Q_0(t) \qquad (6)$$

dove $Q_X(t)$, $Q_1(t)$ e $Q_0(t)$ sono rispettivamente le indisponibilita' di x, F_1e F_0.
Per i nodi terminali **1** e **0**: $Q_1(t) = 1$ e $Q_0(t) = 0$.

La frequenza non condizionata di guasto del sistema, $\omega_S(t)$, ossia la probabilita` che lo stato del sistema definito dal top event si verifichi nell'intervallo infinitesimo t,t+dt, si ottiene semplicemente applicando la seguente espressione in corrispondenza di ciascun nodo ite(x, F_1, F_0):

$$\omega_U(t) = \omega_X(t)\,[Q_1(t) - Q_0(t)] + \omega_1(t)\,Q_X(t) + [1 - Q_X(t)]\,\omega_0(t) \qquad (7)$$

dove $\omega_X(t)$, $\omega_1(t)$ e $\omega_0(t)$ sono rispettivamente le frequenze di guasto di x, F_1 e F_0.
Per i nodi terminali **1** e **0**: $\omega_1(t) = \omega_0(t) = 0$.

Il numero aspettato di volte, $W_S(T)$, che il top event si verifica nell'intervallo di missione 0 - T e' dato da:

$$W_S(T) = \int_o^T w_S(t)\,dt$$

$W_S(T)$ e` spesso utilizzato anche come limite superiore per l'inaffidabilita`.

Equazioni simili possono essere scritte per la frequenza non condizionata di riparazione $n_S(t)$, il cui integrale sull'intervallo di missione rappresenta il numero aspettato di riparazioni $V_S(T)$.

E` altresi` possibile calcolare gli indici di importanza dei componenti con algoritmi molto rapidi e di complessita` paragonabile al calcolo di $Q_S(t)$ e $\omega_S(t)$. Questi algoritmi sono in corso di informatizzazione nel modulo ASTRA-SDI per l'analisi di criticita` di sistemi complessi, eseguibile contemporaneamente su tutti gli alberi di guasto del sistema.

Calcolo del BDD minimo

Un BDD ridotto contiene l'insieme degli implicanti primi. Nell'analisi degli alberi di guasto e` molto importante anche la conoscenza degli MCS. Per funzioni monotone e` possibile ottenere un BDD contenente tutti e soli i cut sets minimi applicando l'algoritmo descritto in [9].

Si consideri una generica funzione monotona Z = ite(w, F, G). Sia M_F l'insieme dei cammini di F e M_G l'insieme dei cammini di G. L'insieme dei cammini di Z, indicato con M_Z , e` dato da:

$$M_Z = w \wedge (M_F \text{ senza i cammini non minimi rispetto ai cammini in } M_G) \cup M_G$$

A partire dal BDD ordinato e ridotto e` possibile ricavare un nuovo grafo contenente solo gli MCS semplicemente eliminando dal ramo sinistro di ciascun nodo i cammini non minimi rispetto ai cammini contenuti nel ramo destro.

Sia Z = (w, F, G) con F = (x, F_1, F_0) and G = (y, G_1, G_0). La minimizzazione si esegue applicando le seguenti regole:

1) se $x < y \Rightarrow F = ite(x, F_1 \setminus G, F_0 \setminus G)$
2) se $x > y \Rightarrow F = ite(x, F_1 \setminus G_0, F_0 \setminus G_0)$
3) se $x = y \Rightarrow F = ite(x, F_1 \setminus G_1, F_0 \setminus G_0)$

dove " \" rappresenta l'operatore che elimina da F i cammini non minimi rispetto ai cammini contenuti in G.

Ad esempio, sia Z = ite(a, ite(b, 1, ite(c, 1, 0)) , ite(c, 1, 0)), dove:
F = ite(b, 1, ite(c, 1, 0)), $F_1 = 1$, F_0= ite(c, 1, 0)
G = ite(c, 1, 0), $G_1 = 1$, $G_0 = 0$
Ordinamento delle variabili: a < b < c.

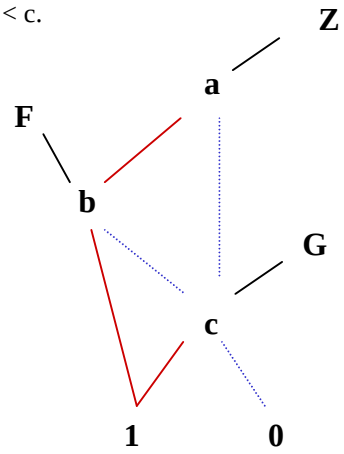


Figura 3. BDD ridotto della funzione Z = a ∧ (b ∨ c) ∨ c

I cut sets contenuti nel BDD sono: M = {a b, a c, c}.
Il grafo in Figura 3 viene visitato secondo l'ordine differito c, b, a. Come si puo` facilmente verificare, dall'esame di c e b non si ottiene alcuna modifica del grafo. Si consideri ora la variabile a. Poiche` b < c, si applica il caso # 1:

ite(x, $F_1 \setminus G$, $F_0 \setminus G$), con $F_1 = 1$, F_0 = ite(c, 1, 0) e G = ite(c, 1, 0).

Pertanto, $F_1 \setminus G = 1$ e $F_0 \setminus G = 0$; di conseguenza F = (b, 1, 0).

Il grafo risultante, contenente gli MCS {c, a b }, e` il seguente: Z = ite(a, ite(b,1,0), ite(b,1,0))

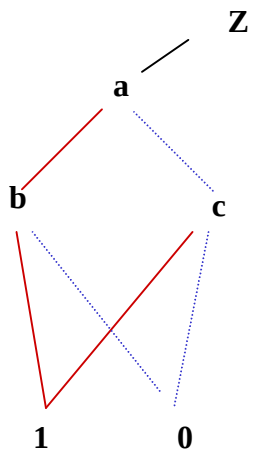


Figura 4. BDD ottenuto minimizzando il grafo in Figura 3

Determinazione degli MCS significativi

Il calcolo degli MCS significativi contenuti nel BDD minimo viene eseguito in base ai valori di soglia assegnati sull'ordine e sulla probabilita` degli MCS. L'algoritmo realizzato in ASTRA-FTA e` molto efficiente e fa uso di un nuovo cut-off , N_{max}, basato sul numero di MCS (piu` significativi) da estrarre.

3.2 Analisi di funzioni non monotone

Quanto descritto per le funzioni monotone e` applicabile anche a funzioni non monotone, ossia a funzioni nelle quali il top event puo` verificarsi a seguito della riparazione di un componente. Ovviamente, cio` non significa che il sistema si degradi a seguito di una riparazione, ma semplicemente che non vi sono piu` le condizioni che verificano l'evento Top [10]. Le funzioni non monotone sono necessarie per modellizzare:

- modi di guasto mutualmente esclusivi;
- sequenze dell'albero degli eventi;
- top-events condizionati allo stato di funzionamento/guasto di sottosistemi e componenti.

I nuovi operatori logici utilizzati sono pertanto l'OR esclusivo (XOR) e la complementazione (NOT). Per funzioni non-monotone il concetto di MCS deve essere sostituito con quello di implicante primo (Prime Implicant, PI), ossia di una congiunzione di variabili negate e non negate non contenuta in nessun altro implicante.

E` immediato mostrare che rimuovendo gli eventi negati dall'insieme dei PI, e minimizzando il risultato, si ottiene la forma approssimata espressa come insieme degli MCS. Di conseguenza la quantificazione probabilistica fornisce risultati conservativi (infatti la rimozione degli eventi negati equivale ad assumere che la probabilita` di funzionamento sia uguale a 1) e, in pratica, l'approssimazione introdotta e` accettabile. Ne consegue che la necessita` di considerare eventi negati e` limitata all'analisi logica per l'identificazione ed eliminazione delle combinazioni impossibili, nelle quali uno stesso evento e` presente in entrambe le forme, negata e non [10].
I vantaggi dell'applicazione di questo approccio semplificato sono notevoli:

- riduzione significativa del tempo di calcolo;
- possibilita` di utilizzare la teoria probabilistica applicabile a funzioni monotone;
- piu` chiara interpretazione dei modi di guasto del sistema.

L'approccio semplificato all'analisi di strutture non monotone e` tuttora largamente impiegato nei programmi di analisi di fault trees basati sugli approcci tradizionali, ed e` conveniente anche con l'approccio BDD per quanto concerne il calcolo degli MCS.

La rappresentazione di funzioni non monotone mediante BDD segue le stesse regole precedentemente descritte per funzioni monotone. Fissato l'ordinamento delle variabili, si ottiene una sola forma ridotta sulla quale puo` essere eseguita l'analisi probabilistica.

Allo scopo occorre distinguere fra le negazioni introdotte durante la costruzione del BDD (a seguito dell'applicazione della (1)) e quelle che descrivono gli stati di funzionamento dei componenti, introdotti durante la costruzione del fault tree. Nel seguito le prime sono denominate *negazioni logiche* e le seconde *negazioni fisiche*.

Alcuni metodi di analisi di funzioni non monotone secondo l'approccio BDD prevedono il calcolo di un grafo contenente tutti gli implicanti primi mediante un algoritmo che fa uso di una struttura dati piu` complessa di quella richiesta per le funzioni monotone [11]. Il metodo sviluppato per ASTRA-FTA fa uso degli stessi algoritmi e della stessa struttura dati gia` utilizzata per le funzioni monotone, con l'aggiunta di alcune regole supplementari di espansione e semplificazione del grafo.

Questa scelta offre due notevoli vantaggi:

i) la possibilita` di utilizzare un solo metodo di gestione della memoria, indipendentemente dal tipo di funzione,

ii) di rendere di immediato utilizzo i metodi disponibili per il calcolo della frequenza di guasto di implicanti primi.

La procedura di analisi di funzioni non monotone si articola nelle seguenti fasi:

1. scelta dell'ordinamento delle variabili;
2. applicazione del metodo di sostituzione delle variabili agli eventi biformi ;
3. costruzione della rappresentazione BDD dell'albero dei guasti;
4. riduzione del BDD;
5. esecuzione dell'analisi probabilistica;
6. eliminazione delle variabili "negate fisicamente" e conseguente riduzione del grafo;
7. determinazione del BDD minimo contenente tutti gli MCS;
8. estrazione degli MCS significativi.

Calcolo del grafo ridotto

Ogni variabile biforme x (variabile che compare nella funzione in forma affermata e negata) viene sostituita da due variabili:

$x = x_1$ e $\neg x = x_0$ con ordinamento $x_1 < x_0$.

Data la mutua esclusivita` di x_1 e x_0:

$x_1 \wedge x_0 = 0$

$x_1 \wedge \neg x_0 = x_1$

$\neg x_1 \wedge x_0 = x_0$

$\neg x_1 \wedge \neg x_0 = 0$

La funzione risultante e` monotona e pertanto e` possibile utilizzare gli stessi algoritmi (e, cosa ancor piu` importante, la stessa struttura dati) definiti per le funzioni monotone.

Dopo l'espansione della funzione rispetto a x_1 e a x_0, vengono applicate le regole R4 e R5 per l'eliminazione rispettivamente dei rami per i quali $x_1 \wedge x_0 = 0$ e $\neg x_1 \wedge \neg x_0 = 0$:

R4: $ite(x_1, ite(x_0, A, B), C) = ite(x_1, B, C)$

R5: $ite(x_1, A, ite(x_0, B, C)) = ite(x_1, A, ite(x_0, B, 0))$

Dove A, B, C rappresentano generiche funzioni booleane.

Si consideri il seguente esempio di calcolo del BDD ridotto per $F = (x \vee y) \wedge (\neg x \vee (x \wedge \neg y))$.

Dopo la trasformazione di variabili: $F = (x_1 \vee y_1) \wedge (x_0 \vee (x_1 \wedge y_0))$.

L'ordinamento delle variabili e` quindi il seguente: $x_1 < x_0 < y_1 < y_0$
Espandendo F rispetto a x_1 si ottiene: F = ite(x_1, A, B),
dove A= $x_0 \vee y_0$ e B = $y_1 \wedge x_0$.
Espandendo sucessivamente rispetto a $x_{0:}$
ite(x_1, ite($x_{0,}$ 1, C), ite($x_{0,}$ D, 0)), con C = y_0 e D = $y_{1.}$
Applicando infine la regola R4 si ha F = ite(x_1, C, ite($x_{0,}$ D, 0));
Poiche` C = ite(y_0, 1, 0) e D = ite(y_1, 1, 0), si ottiene: F = ite(x_1, ite(y_0, 1, 0), ite($x_{0,}$ ite(y_1, 1, 0), 0).

Analisi probabilistica di una funzione non monotona

Si consideri un BDD ridotto ottenuto applicando il metodo precedentemente descritto. Al nodo non terminale x_1 (x in forma affermata) sono rispettivamente associati i valori $Q_x(t)$, $\omega_x(t)$, mentre al nodo x_0 (x in forma negata) i valori 1- $Q_x(t)$, $\nu_x(t)$, dove $\nu_x(t)$ rappresenta la frequenza non condizionata di riparazione.
Per il nodo terminale 1: $Q_1(t) = 1$, $\omega_1(t) = 0$, $\nu_1(t) = 0$; per il nodo terminale 0: $Q_0(t) = 0$, $\omega_0(t) = 0$, $\nu_0(t) = 0$.
. L'indisponibilita` $Q_S(t)$ si calcola semplicemente applicando la (6), con una variante dovuta alla presenza delle variabili $x_{0.}$
Dato un generico nodo ite(x, F_1 , F_0), si possono presentare le seguenti situazioni:

x = x_1, F_1 e F_0 qualsiasi

$$Q_U(t) = Q_X(t)\, Q_1(t) + [1 - Q_X(t)]\, Q_0(t) \qquad (8)$$

x = x_1, F_1 qualsiasi e F_0 = ite(x_0, A, B)

$$Q_U(t) = Q_X(t)\, Q_1(t) + Q_0(t) \qquad (9)$$

La (8) si rende necessaria in quanto Prob($\neg x_1 \wedge x_0$) = Prob(x_0).

Fra i metodi per il calcolo di $\omega_S(t)$, presentati e confrontati in [12], in ASTRA e` stato applicato quello proposto da Liu-Pan [13], secondo una formulazione ridotta, in quanto ben si adatta alla struttura del grafo ottenuto applicando il metodo di cambiamento delle variabili. Il metodo Liu-Pan fornisce il valore esatto o approssimato di $\omega_S(t)$ a seconda che la struttura del BDD si riferisca ad una funzione monotona o non monotona. L'approssimazione e` comunque piu` che accettabile.
Il calcolo di $\omega_S(t)$ viene eseguito applicando le seguenti espressioni a ciascun nodo del grafo.
Dato un generico nodo ite(x, F_1 , F_0), si possono presentare le seguenti situazioni:

x = x_1, F_1 e F_0 qualsiasi

$$\omega_U(t) = \omega_X(t)\, Q_1(t) + \omega_1(t)\, Q_X(t) + [1 - Q_X(t)]\, \omega_0(t) - \omega_X(t)\, Q_0(t) \qquad (10)$$

x = x_1, F_1 qualsiasi e F_0 = ite(x_0, A, B)

$$\omega_U(t) = \omega_X(t)\, Q_1(t) + \omega_1(t)\, Q_X(t) + \omega_0(t) \qquad (11)$$

x = x_0, F_1 e F_0 qualsiasi

$$\omega_U(t) = \nu_X(t) Q_1(t) + \omega_1(t)\, Q_X(t) + [1 - Q_X(t)]\, \omega_0(t) - \nu_X(t) Q_0(t) \qquad (12)$$

Anche se non descritte per brevita`, e` possibile calcolare facilmente la frequenza di riparazione e la criticita` dei componenti con algoritmi di complessita'`pari a quelli visti per il calcolo della frequenza di guasto.

Calcolo degli MCS

Il calcolo degli MCS si esegue con gli stessi algoritmi visti per le funzioni monotone, ma solo dopo aver eliminato tutte le variabili "negate fisicamente" (variabili con pedice $_0$) applicando le seguenti regole di semplificazione:
R6: ite(x_1, A, ite(x_0, B, 0)) = ite(x_1, A, B)
R7: ite(y, ite(z_0, A, B), C) = ite(y, F, C)
R8: ite(y, C, ite(z_0, A, B)) = ite(y, C, F)
dove x_1, x_0 rappresentano una variabile biforme, z_0 una variabile monoforme negata, una variabile monoforme affermata e F = A $\vee$ B.
Il BDD risultante, opportunamente ridotto, rappresenta una funzione monotona che deve essere sottoposta a minimizzazione prima dell'estrazione degli MCS.

4. IL SOFTWARE ASTRA-FTA PER L'ANALISI DEGLI ALBERI DI GUASTO

ASTRA-FTA e` un nuovo software in ambiente Windows 95 e NT per l'analisi di alberi di guasto complessi basato sull'approccio BDD [14, 15 ]. Esso e` in grado di analizzare alberi contenenti gli operatori AND, OR, NOT, XOR, K/N e INH. L'efficienza di ASTRA-FTA puo` essere mostrata mediante i risultati ottenuti dall'analisi di un certo numero di alberi complessi. La tabella 2 contiene, per ogni albero considerato, il numero di operatori, il numero di eventi primari e il numero totale di MCS. Gli alberi 3 [ 6 ], 5 e 6 [ 9 ] sono in realta` semplificati nel numero di eventi e gates (sono alberi gia` modularizzati), mentre la logica di guasto e` complessa; essi presentano alcuni livelli di logiche maggioritarie in cascata e si riferiscono a sistemi di sicurezza di impianti nucleari.

Fault Tree	Numero di Gates	Numero di Eventi	Numero di MCS
1	104	143	76,785
2	132	215	746,754
3	68	58	$\sim 1.2\ 10^{7}$
4	169	102	583,973
5	81	139	$> 2.\ 10^{9}$
6	84	61	42,677

Tabella 2. Caratteristiche degli alberi di guasto di riferimento

L'applicazione di ASTRA-FTA agli alberi in Tabella 1, elaborati su PC 486 a 200 Mhz, ha fornito i risultati mostrati nelle tabelle seguenti. Nell'ultima colonna della Tabella 3 sono riportati i tempi di calcolo per la determinazione del valore esatto dell'indisponibilita` $Q_S(t)$, del grafo minimo contenente i soli MCS e del valore del primo bound $Q_S^1(t)$ (uguale alla somma delle indisponiilita' degli MCS).

Fault tree	Numero di gates	Numero di eventi	Numero totale di MCSs	$Q_S(t)$	Tempo di calcolo (sec)
1	104	143	276,785	$1.32\ 10^{-02}$	0.5
2	132	215	746,574	$1.05\ 10^{-01}$	0.58
3	68	58	$> 1.2\ 10^{7}$	$1.13\ 10^{-08}$	0.451
4	169	112	583,973	$3.98\ 10^{-03}$	0.481
5	81	139	$> 2.\ 10^{9}$	$2.00\ 10^{-03}$	0.491
6	84	61	46,188	$1.00\ 10^{-06}$	1.602

Tabella 3. Risultati dell'applicazione del metodo BDD agli alberi di cui alla Tabella 2.

La determinazione degli MCS viene eseguita applicando la tecnica di cut-off. In Tabella 4 sono riportati i risultati dell'analisi. Un MCS e` considerato significativo se la sua indisponibilita` $P \geq P_{lim}$, ossia non minore del valore di soglia stabilito dall'utente.

Fault tree	Cut-off P_{lim}	Numero di MCS significativi	$Q_S^1(t)$	Pe	Tempo (sec)
1	$1.27\ 10^{-11}$	5779	$1.49\ 10^{-02}$	$4.6\ 10^{-08}$	0.675
2	$3.5\ 10^{-7}$	47,232	$1.88\ 10^{-01}$	$1.84\ 10^{-03}$	0.851
3	10^{-20}	204	$1.20\ 10^{-08}$	$1.12\ 10^{-09}$	0.941
4	10^{-12}	1412	$4.18\ 10^{-03}$	$3.84\ 10^{-10}$	0.591
5	10^{-12}	885	$2.04\ 10^{-03}$	$4.48\ 10^{-11}$	0.611
6	10^{-12}	4009	$1.68\ 10^{-06}$	$1.04\ 10^{-07}$	3.025

Tabella 4. Risultati dell'estrazione degli MCS significativi dai grafi minimi

Nell'attuale versione di ASTRA-FTA, l'analisi probabilistica viene eseguita sull'insieme degli MCS significativi, in quanto viene utilizzato lo stesso modulo di calcolo di ISPRA-FTA. Dal momento che l'analisi probabilistica viene eseguita solo su un sottoinsieme di MCS, l'informazione sull'errore di troncamento diventa importante; esso e` calcolato come somma delle indisponibilita` degli MCS aventi $P < P_{lim}$:

$Pe = Q_S^1(t) - \sum_i \text{Prob}(C_i)$ dove Ci rappresenta il generico MCS significativo.

5. CONCLUSIONI E SVILUPPI FUTURI

Nella presente memoria sono stati brevemente descritti i fondamenti teorici dell'approccio BDD all'analisi di fault trees complessi. L'efficienza di questo approccio e` stata mostrata mediante l'applicazione del programma ASTRA-FTA ad alcuni alberi di sistemi reali.

Grazie al nuovo approccio sara` possibile sviluppare strumenti di calcolo piu` potenti rispetto a quelli basati sugli approcci tradizionali. Presso CCR-ISIS e` in corso lo sviluppo di un nuovo modulo ASTRA per il miglioramento delle caratteristiche di affidabilita` e disponibilita` di sistemi complessi basato sull'analisi interattiva di criticita` dei componenti, applicata contemporaneamente a tutti gli alberi del sistema rappresentati mediante BDD. Tale modulo puo` essere ovviamente utilizzato anche per il miglioramento di sistemi, ad esempio nucleari, modellizzati mediante Event Trees e Fault Trees. Le potenzialita` dell'approccio BDD ne consentono l'applicazione anche al controllo on-line di sistemi complessi [16].

6. ELENCO DEI SIMBOLI

$Q_S(t)$ valore esatto dell'indisponibilita` del top event
$Q_S^1(t)$ primo bound dell'indisponibilita` del top event
$Q_X(t)$, $Q_1(t)$, $Q_0(t)$ indisponibilita` dell'evento x, del residuo F_1 e del residuo F_0
$\omega_X(t)$, $\omega_1(t)$, $\omega_0(t)$ frequenza istantanea non condizionata di guasto dell'evento x, del residuo F_1 e di F_0
$\nu_X(t)$, $\nu_1(t)$, $\nu_0(t)$ frequenza istantanea non condizionata di riparazione dell'evento x, del residuo F_1 e di F_0
$W_S(T)$ numero aspettato di guasti del top event
T tempo di missione
P_{lim} soglia associata alla indisponibilita` degli MCS
Pe errore di troncamento

7. BIBLIOGRAFIA

[1] W.E. Vesely, Analysis of Fault Trees by Kinetic Tree Theory
Idaho Nuclear Corporation, IN-1330, 1969

[2] M.Astolfi, S.Contini, C.Van den Muyzenberg, G.Volta, Fault Tree Analysis by List Processing, in Synthesis and Analysis Methods for Safety and Reliability Studies.
edited by G.Apostolakis, G.Garribba and G.Volta, Plenum Press, new York and London, 1980

[3] S.Contini, SALP-PC, A Fault Tree Analysis Package on Personal Computers. User Manual.
CEC-JRC Ispra Establishment, T.N. 1.87.165, 1986

[4] S. Contini, ISPRA-FTA. Interactive Software Package for Reliability Analysis. Fault Tree Analysis Tool for Personal Computers, *JRC-Ispra, EUR 13997 EN, 1992*

[5] S. Contini, A New Hybrid Method for Fault Tree Analysis,
Reliability Engineering and System Safety, 49, 1995

[6] L. Caldarola, W. Wichenhauser, The Boolean Algebra with Restricted Variables as a Tool for Fault Tree Modularization, *KfK 3190 / EUR 7056e, 1981*

[7] R.E. Bryant, Graph Based Algorithms for Boolean Functions Manipulation
IEEE Transactions on Computers, Vol. C-35, 1986

[8] S. Contini, G. de Cola (1996), A Top Down Approach to Fault Tree Analysis Using Binary Decision Diagrams, *European Journal of Automation, Vol. 30, n.8, 1996*

[9] A. Rauzy, New Algorithms for Fault Tree Analysis, *Reliability Engin. and Systems Safety, 40, 1993*

[10] A. Amendola, C.A. Clarotti, S. Contini, F. Spizzichino, Analysis of Complete Logical Structures in System Reliability Assessment, *JRC-Ispra, EUR 6886 EN, 1980*

[11] J.C Coudert, P. Madre, Metaprime: an Interactive Fault Tree Analyser with Binary Decision Diagrams
IEEE Transactions on Reliability, Vol. 43, 1994

[12] P. Gianotti, Rassegna dei metodi per la determinazione delle caratteristiche affidabilistiche di sistemi non coerenti, *CCR-ISIS, TN. I.97.131, 1997*

[13] J.C. Liu, Z.J. Pan, A new Method to Calculate the Failure Frequency of Not Coherent Systems,
IEEE Transactions on Reliability, Vol.39, N. 3, 1990.

[14] S.Contini, G. de Cola, M.Wilikens, G. Cojazzi, ASTRA, An Integrated Tool Set for Complex Systems Dependability Studies, Workshop on Tool Support for Systems Specification, Development and Verification, Malente, Germany, 1998.

[15] S.Contini, ASTRA Theoretical Handbook, JRC SP-report, in corso di pubblicazione.

[16] S.Contini, M. Wilikens, M. Masera, The Use of RAMS Probabilistic Analysis for On-line Maintenance and Decision Support, ESREL '96, Crete, Greece, 1996.