Métodos para a remoção de redundâncias de árvores de falhas

Paulo Renato Alves Firmino (UFPE) praf62@yahoo.com
Pedro Igor Carvalho Moreira (UFPE) pedroigor@click21.com.br
Rohgi Toshio Meneses Chikushi (UFPE) rohgi_toshio@yahoo.com.br
Enrique López Droguett (UFPE) ealopez@ufpe.br

Resumo

A análise de árvores de falhas é uma importante ferramenta de apoio às inferências quanto à confiabilidade que, por sua vez, é uma das bases da engenharia de produção. O método de BDDs (Diagramas de Decisão Binária) tem sido citado como uma alternativa às técnicas convencionais, que alia tanto maior precisão quanto menor esforço computacional. O grande problema para a aplicação de BDDs reside na necessidade de conversão da árvore de falhas para o seu formato. Os métodos de diagramas espirais, que assumem a função de conversores da árvore em BDD, requerem que não haja redundâncias na árvore; isto é, que ela esteja isenta de cortes não-mínimos. Neste artigo sugere-se, uma série de métodos de remoção de redundâncias de árvores de falhas, a princípio coerentes, inicialmente inspirada no método de redução de Faunet.

Palavras chave: Árvores de Falhas, Redundância, Diagramas Espirais.

1. Introdução

O estudo de uma árvore de falhas deduz todas as possíveis combinações de eventos que levam à falha do produto, são os chamados cortes da árvore. Entre os cortes da árvore pode-se separar aqueles que são mínimos; excluindo-se os cortes que contêm outros cortes; ou seja, as redundâncias. Esta exclusão tem o objetivo de resumir a estrutura lógica da árvore aos menores conjuntos de eventos que, se ocorridos, levam ao topo da árvore; isto é, à falha do sistema. O método de redução de Faunet (REAY & ANDREWS, 2002) esboça a lógica combinatória necessária para a eliminação das redundâncias; apesar de a sua intenção ser reduzir um específico subconjunto da árvore, dadas algumas características. Logo, partindo do método de redução de Faunet, é proposto, neste artigo, uma série de métodos capazes de remover todas as redundâncias da árvore, com o principal objetivo de permitir a aplicação posterior dos diagramas espirais (FIRMINO et al., 2004). Naturalmente, uma vez removidas todas as redundâncias, qualquer dos métodos de análise de árvores de falhas é favorecido, uma vez que será necessário apenas discriminar os cortes diretamente; porém, o direcionamento para diagramas espirais deriva da capacidade de conversão, da árvore para o formato BDD, dos seus métodos.

2. Definições

Para melhor compreensão do que é tratado neste artigo, segue alguns termos que serão bastante usados:

Conexão(ramificação): ligação de um evento básico, ou subsistema, a outro evento básico ou subsistema.

Subsistema: cada porta lógica da árvore de falhas constitue um subsistema, cujos elementos são as suas ramificações, sejam eles eventos básicos ou outros subsistemas.

Conector (topo): evento básico, ou subsistema, mais próximo da raíz da árvore de falhas, em relação àquele que é uma ramificação sua.

Diagrama Espiral: grafo cujas arestas (conexões ou ramificações) são direcionados para os topos.

3.Subsistemas

Como foi dito anteriormente, o que define quais serão os subsistemas são as portas lógicas presentes na árvore de falhas. Observando a figura 1, os subsistemas G_1 e G_2 comporiam os dois mais comlpexos subsistemas da árvore de falhas; os quais poderiam ser compostos por outros subsistemas, e assim por diante. Enfim, como os subsistemas formam a árvore de falhas, o topo é, também, um subsistema, com a particularidade de não ser uma ramificação de um outro subsistema.

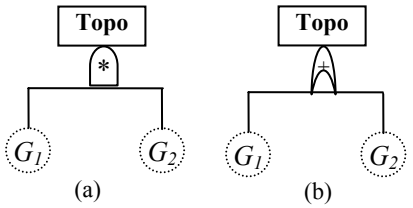

Figura 1- Subsistemas de uma árvore de falhas.

4. Incoerência, Inconsistência e Redundância

Os cortes inconsistentes são provenientes de árvores de falhas inconsistentes, mal construídas às vezes pela falta de conhecimento sobre o sistema ou pela falta de experiência do profissional responsável por sua construção. Em geral, a inconsistência se apresenta quando existem subsistemas compostos por eventos básicos que representam a não-falha dos seus componentes. Neste trabalho, têm sido tratados apenas os casos em que não se verifica a presença de inconsistência; já que são tratadas somente árvores coerentes.

De acordo com Jung et al. (2004), considerando a função booleana, da árvore, representada de forma linear para qualquer evento básico x_i, $f_{x_i} = a \cdot \bar{x}_i + bx_i + c$, onde a, b e c, são funções booleanas das outras variáveis conectadas a x_i e $\bar{x}_i$ é o evento complementar a x_i, a árvore de falhas é dita coerente se a função booleana a é sempre vazia; isto é, a pode ser descartada, e é este tipo de árvore de falhas que está sendo tratado neste artigo. Com isso, $f_{x_i} = bx_i + c$.

A redundância (ou repetição desnecessária) de eventos básicos é mais freqüente em grandes árvores de falhas. Torna-se cada vez mais difícil evitá-la, à medida que o nível de detalhamento do sistema aumenta.

Considerando como evento original da redundância, aquele mais próximo do topo da ocorrência da redundância (a raíz da qual o evento original e o redundante são descendentes) e considerando como eventos redundantes aqueles mais distantes do topo da ocorrência da redundância, são estudados quatro tipos de ocorrência da redundância:

1. *O evento original e os redundantes pertencem à primeira geração do topo da ocorrência da redundância*. Nesse caso a solução para a redundância é simples, basta excluir todas as redundâncias e manter o evento original; pois, em álgebra booleana, $A + B + A = A + B$.

Considerando, na figura 1, que G_1 e G_2 são o mesmo evento básico, este tipo de ocorrência de redundância se verifica.

2. *Apenas o evento original pertence à primeira geração do topo da ocorrência da redundância.* Aquí ocorrem duas lógicas de matemática combinatória. Se os topos dos eventos redundantes possuirem portas lógicas diferentes da do topo do evento original (que é também o topo da ocorrência da redundância), o topo das redundâncias deve ser eliminado da árvore; observe a figura 2. Porém, Se os topos dos eventos redundantes possuirem portas lógicas iguais à do topo do evento original, apenas os eventos redundantes devem ser eliminados dos seus topos. Observe a figura 3. É importante que se comente que, dada a ocorrência da redundância e a sua resolução, G_1 passa a possuir apenas uma ramificação e deve ser excluído da árvore. Sua ramificação deve passar a ser uma ramificação do seu topo. Veja que a árvore descrita na figura 2 também passa por este processo. Firmino et al., 2004 detalham este e outros tratamentos.

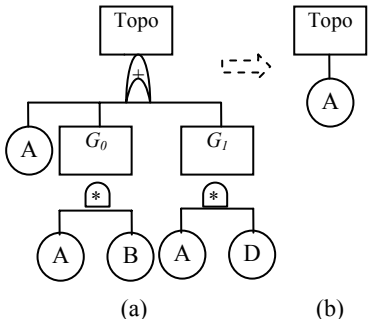

Figura 2- Ocorrência de redundâncias, onde apenas o evento original pertence à primeira geração do topo da ocorrência da redundância.

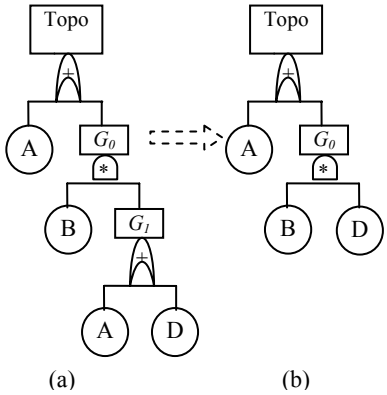

Figura 3- Ocorrência de redundâncias, onde apenas o evento original pertence à primeira geração do topo da ocorrência da redundância.

Este tipo de redução utiliza-se do seguinte fato: Seja *f* a função booleana referente à árvore de falhas apresentada na figura 3(a), em álgebra booleana, $f = A + B \cdot (A + D) = A + B \cdot D$; e, em se tratando da árvore de falhas referente à figura 2(a), $f = A + A \cdot B + A \cdot D = A$.

3. *Tanto o evento orginal, quanto os eventos redundantes, pertencem à segunda geração de ramificações do topo da ocorrência da redundância.* É neste tipo de ocorrência de redundância que o método de redução de Faunet (REAY & ANDREWS, 2002) se aplica. Utilizando álgebra booleana, $(A + B) \cdot (A + C) = A + B \cdot C$; assim como, $A \cdot B + A \cdot C = A \cdot (B + C)$; onde, do lado direito destas duas igualdades a redundância é eliminada, como pode ser visto na figura 4.

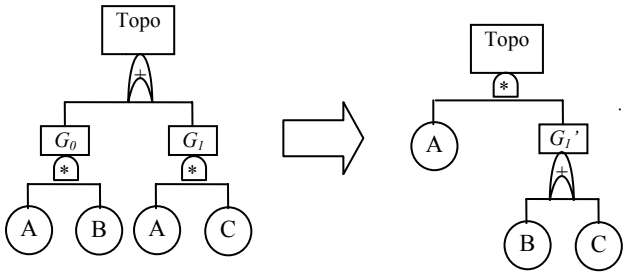

Figura 4- Ocorrência de redundâncias, onde tanto o evento orginal, quanto os eventos redundantes, pertencem à segunda geração de ramificações do topo da ocorrência da redundância.

4. *Não há qualquer das relações de parentesco citadas anteriormente e a porta lógica do topo da ocorrência da redundância é do tipo E.* Este tipo de ocorrência de redundância é o que requer mais esforço computacional, para ser solucionado, de todos os tipos apresentados. O método de resolução utilizado é o método espiral de eliminação de cortes não-mínimos (MEEC), proposto por este artigo. O MEEC utiliza-se, do fato de que $(G_i + G_j)\ (G_k + G_l) = G_i\ (G_k + G_l) + G_j\ (G_k + G_l)$ [1],

onde G_i, G_j, G_k, G_l são subsistemas da árvore de falhas. Considerando que a primeira parcela do lado direito da igualdade não contém o evento original, e que, conseqüentemente, a segunda parcela o contém, esta lógica de igualdade é aplicada recursivamente na segunda parcela, até que o evento original seja uma ramificação do, então, topo da ocorrência da redundância; quando isto acontece é possível aplicar a lógica de solução para o caso em que apenas o evento original pertence à primeira geração do topo da ocorrência da redundância, chegando-se, assim, à solução possível. Em seguida, aplica-se a lógica inversa à primeira parcela do lado direito da igualdade de [1], correspondente á última chamada da recursividade. Esta lógica inversa remove o topo do evento repetido da primeira parcela do lado direito da igualdade de [1], quando apenas o evento repetido é removido da segunda parcela, e remove apenas o evento repetido da primeira parcela do lado direito de [1], quando o topo do evento repetido é removido da segunda parcela.

Existe um específico caso em que a definição de evento original e a aplicação da lógica inversa são alteradas. Quando o evento redundante pertencer à terceira geração do topo da ocorrência da redundância, este deve passar a ser considerado como o evento original, a não ser que o evento original também possua esta característica. Além disso, quando o evento original possuir a característica de pertencer à terceira geração do topo da ocorrência da redundância, a lógica inversa não deve ser aplicada à primeira parcela do lado direito de [1], mantendo-a inauterada.

Para o tipo de ocorrência de redundância onde o MEEC se aplica, não é possível que se obtenha uma árvore de falhas sem repetições, mas é garantido que o método de diagramas espirais gere uma OBDD cujos caminhos (cortes) são mínimos e os eventos são otimamente ordenados. O MEEC pode ser melhor compreendido verificando-se a figura 5, onde a sua lógica elimina todos os cortes não-mínimos que, certamente, serão gerados, devido o topo da ocorrência da redundância ser uma porta lógica do tipo E. Como o evento básico B está presente em dois subsistemas distintos e de grau de parentesco elevado (neste caso, o topo da árvore é, também, o topo da ocorrência da redundância e tanto o evento original quanto o evento redundante pertencem à sua terceira geração), apenas o MEEC pode solucioná-lo. Neste caso específico, o evento original poderia ser qualquer uma das ocorrências do evento básico B; porém, quando as ocorrências possuem graus de geração distintos, o evento considerado como original será aquele com o menor grau de geração; ou seja, aquele mais

próximo do topo da ocorrência da redundância, como já dito anteriormente. O que o MEEC está fazendo de (a) para (b), na figura 5, é considerando que $(A+G_j)\cdot(D+G_l)=A\cdot(D+G_l)+B\cdot C\cdot(D+G_l)$. Como $B\cdot C\cdot(D+G_l)$ é o subsistema que possui a ocorrência da redundância, e como o evento original pertence à primeira geração do topo de tal ocorrência, G'_1; considerando que o evento básico original e o redundante são ramificações de subsistemas cujas portas lógicas são do mesmo tipo, apenas o evento básico redundante deve ser excluído do seu topo, em G'_1, e o seu topo não deve ser excluído em em G'_0 [o subsistema $A\cdot(D+G_l)$]; já que, neste caso, tanto o evento original quanto o redundante pertencem à terceira geração do topo da ocorrência da redundância; nos demais casos, G_l seria excluído. Observando a figura 5(c), percebe-se que, agora, os eventos básicos D, E e o prórpio B possuem uma repetição. Estas repetições são inevitáveis, devido às características da árvore proposta; porém, são necessárias para que seja garantida a geração de todos os cortes mínimos. Como os seus topos são subsistemas cujas portas lógicas são do tipo OU, não haverá problemas com suas permanências na árvore de falhas e suas repetições não devem ser consideradas como redundâncias. O maior problema, ao se tratar de árvores de falhas que possuem esta característica de redundância ou de repetição, é que é impossível que se gere um OBDD mínimo idealmente; ou seja, o OBDD sempre possuirá eventos básicos repetidos.

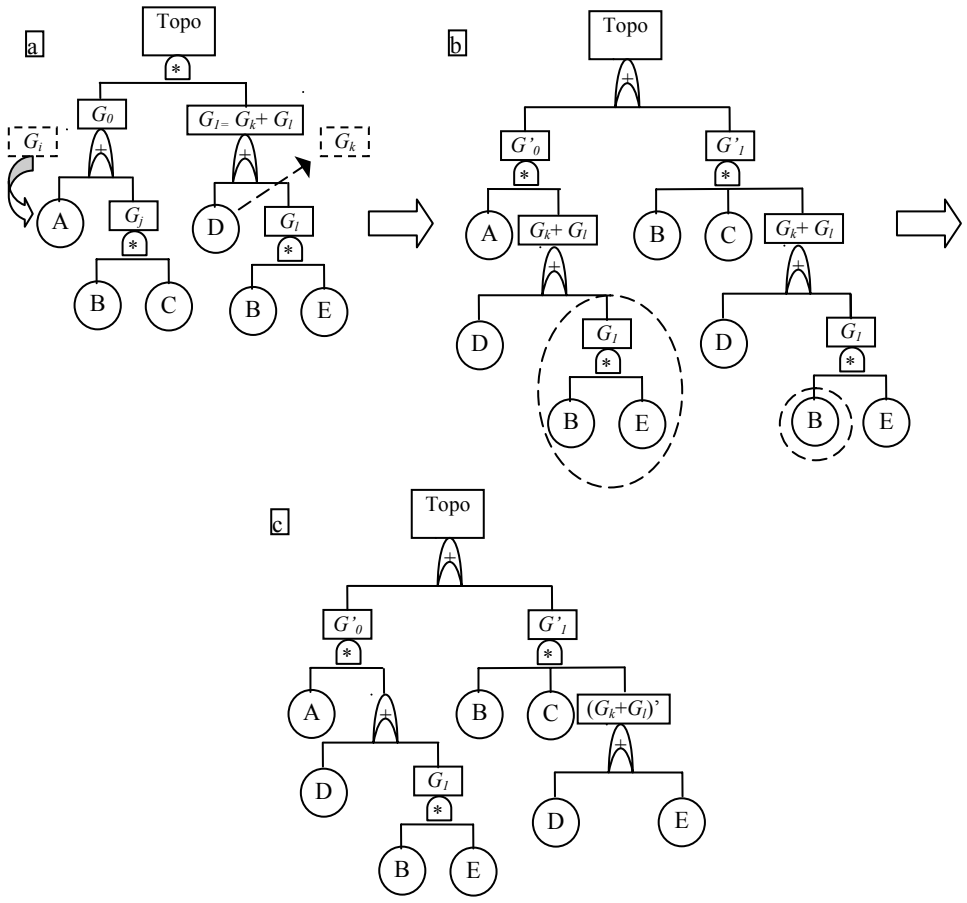

Figura 5- Ocorrência de redundâncias, onde não há qualquer relação de parentesco, evidente, entre o evento original e os redundantes.

Faz-se necessário salientar que apenas após todos os cuidados aqui citados, tanto de redução quanto de reestruturação da árvore de falhas, o método de diagramas espirais pode ser aplicado; caso isto não ocorra, os seus resultados são comprometidos. O método de diagramas espirais, que é responsável direto pela construção do OBDD, requer que a árvore esteja

ordenada, em relação aos tamanhos dos seus subsistemas, e isenta da presença de redundâncias.

11.Conclusões

Com o objetivo de preparar a árvore de falhas para a sua conversão para o formato BDD, os métodos de remoção de redundâncias da árvore, contribuem de maneira a não permitir que permaneçam, na árvore, os cortes não-mínimos. Este tratamento anterior à aplicação dos métodos de diagrams espirais garantem que a conversão resultará em um BDD com o menor tamanho possível; isto é, com o número mínimo de variáveis. Por enquanto, tais métodos de remoção de redundâncias, são adequados apenas a árvores coerentes; desafios futuros são aplicá-los a árvores de falhas incoerentes.

Referências

FIRMINO, P. R.; MOREIRA, P. I. & DROGUETT, E. L. (2004)- Diagramas espirais: eétodo auxiliar para a resolução ótima de árvores de falhas. Artigo submetido para este congresso.

JUNG, W. S.; HAN, S. H.& HA J. (2004)- A fast BDD algorithm for large coherent fault trees analysis. *Reliability Engineering and System Safety*. Vol. 83, p. 369-374.

MOREIRA, P. I.; FIRMINO, P. R. & DROGUETT, E. L. (2004)- Aplicação do método de diagramas espirais para resolução de árvores de falhas. Artigo submetido para este congresso.

REAY, K. A. & ANDREWS, J. D. (2002)- A fault tree analysis strategy using binary decision diagrams. *Reliability Engineering and System Safety*. Vol. 78, p. 45-56.