DOI:10.19322/j.cnki.issn.1006-4710.2016.01.005

基于二元决策图的故障树底事件排序

黑新宏[1]，张阳阳[1]，钱富才[2]，谢　国[2]，何文娟[1]

(1.西安理工大学 计算机科学与工程学院，陕西 西安 710048；
2.西安理工大学 自动化与信息工程学院，陕西 西安 710048)

摘要：提出了一种底事件排序的新方法——最小深度子树法。该方法先将结构复杂的故障树化简为一棵简单的树，然后基于各子树的深度、节点数及节点间的位置关系对底事件进行静态排序，并根据排序结果动态构造BDD。最后，通过对航空发动机加速时喘振停车的故障树分析，证明该方法可快速构建BDD，而且构建的BDD产生的冗余节点数目较少。

关键词：故障树分析；二元决策图；底事件排序

中图分类号：TP206.3　　**文献标志码**：A　　**文章编号**：1006-4710(2016)01-0023-07

Variable ordering in fault tree analysis based on binary decision diagrams

HEI Xinhong[1], ZHANG Yangyang[1], QIAN Fucai[2], XIE Guo[2], HE Wenjuan[1]

(1. School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China;
2. School of Automation and Information Engineering, Xi'an University of Technology, Xi'an 710048, China)

Abstract: In this paper, a novel ordering method, namely the minimum depth of the sub-tree method, is proposed. This method reduces a complex fault tree to be a simple tree, and then makes static variable ordering based on the sub-tree's depth, node numbers and the positional relationship between the nodes, and finally dynamically construct a BDD according to the sorting result. In the end, the aviation engine acceleration surge parking fault tree shows that this method can quickly build a BDD, and the BDD has less redundant nodes.

Key words: fault tree analysis (FTA); binary decision diagram (BDD); variable ordering heuristics

故障树分析(Fault Tree Analysis，FTA)是用于处理大型复杂系统可靠性、安全性及风险评估的一种有效方法，在航空、航天、核能、化工等领域得到了广泛应用。传统基于割集的故障树分析[1]方法其时间及空间的复杂度较高且不易编程实现，而且在分析过程中会产生明显的"组合爆炸"，因而难以胜任对大型复杂系统的分析。为解决这一问题，Andrews提出了基于二元决策图(Binary Decision Diagrams，BDD)的故障树分析方法[2]，一次性实现割集的不交化。当故障树是最简割集BDD时，可以从不交化割集中直接析出最小割集[3]，避免了割集的不交化运算和化简。另外，BDD方法在模型定量分析时不需要近似和删减计算，在不能确定最小割集的情况下具有准确进行概率量化的可能性，能够正确处理负面逻辑(成功分支)[4]，减小了分析计算时的时间和空间复杂度。

对于大型复杂系统的故障树，可以利用BDD和计算机技术自动地进行分析与处理，但首先需要将故障树转换为BDD，然后再通过BDD对故障树进行定性与定量计算。而将故障树转换为BDD时，需要先对故障树进行预处理，即对其底事件进行排序，底事件节点顺序直接影响最后生成的BDD节点数量。因此，如何对故障树的底事件排序，成为了利用BDD对故障树进行分析的一个难题[5]。

近年来，国际上许多学者对故障树底事件排序进行了大量研究，主要有从上向下、从左到右法[6]，

收稿日期：2015-04-23

基金项目：国家自然科学基金资助项目(61273127，U1334211，U1534208)；高等学校博士学科点专项科研基金资助项目(20116118110008)

作者简介：张阳阳，男，硕士生，研究方向为可靠性分析。E-mail:1005763271@qq.com

通讯作者：黑新宏，男，博士，教授，研究方向为安全关键计算机系统、嵌入式系统及应用。E-mail:heixinhong@xaut.edu.cn

深度优先法[7]，事件重要度方法[8]，相邻底事件优先法[5]等等。但所有这些方法都只适用于一些简单的故障树或者某种特定类型的故障树，很难存在一种对所有类型的故障树都有效的排序方法。

在对前人方法研究的基础上，本文提出了一种新的底事件排序方法——最小深度子树法。该方法是基于各子树的深度、节点数以及底事件节点间的位置关系来为其排序，能够在很大程度上减少构造 BDD 时冗余节点的产生，静态排序动态构造[9]。实例结果证明，利用该方法构造的 BDD，其节点数目大多少于现有的其它方法所构造的 BDD，而节点数量更少的 BDD 可有效提高对故障树分析计算的性能。

1 BDD 结构

BDD 是一有向无环图，它主要由顶节点、中间节点、叶子节点及连接两节点间的分支组成，其结构如图 1 所示。

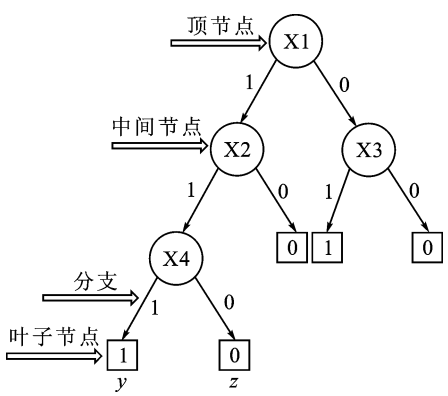

图 1 BDD 结构图

Fig. 1 BDD diagram

BDD 中的顶节点(X1)和中间节点(X2、X3、X4)代表故障树中的底事件，即系统中的基本事件，叶子节点(y、z)表示对应故障树顶事件的状态。由图 1，顶节点和中间节点包含有 1 和 0 两个分支，分别表示基本事件的发生和不发生；而叶子节点为 1 和 0 两种状态之一，用以表示顶事件的发生与否，即系统是否发生故障。

在构造好 BDD 后，从顶节点开始遍历该 BDD，到达叶子节点时，若叶子节点的值为 1，则其所在路径上的所有非终结点的集合组成一个割集；若叶子节点的值为 0，则向上继续遍历其它未被遍历过的节点。当整个 BDD 遍历完成后，可找到所有值为 1 的叶子节点所在的路径，即可得到 BDD 所对应故障树的所有最小割集。由此可知，图 1 中 BDD 所对应故障树的最小割集为{X1, X2, X4},{X3}。

与传统方法相比，通过 BDD 技术求解的结果就是最小割集，不需进行多次近似和删减计算，可极大提高分析计算效率。而利用传统基于割集的上行法[10]、下行法[10]、删去留下法[11]等方法分析求解大规模故障树时，在分析计算过程中会产生明显的“组合爆炸”，且在求解最小割集时还需要进行多次近似和删减计算，造成了时间和空间复杂度的增加，降低了分析计算的效率。

2 最小深度子树法

在现实中，一个大型复杂系统其各个子系统之间常常会受到大量重复事件的影响，即一个基本事件可能会出现在多个子系统中，若该事件发生故障，则可能会导致多个子系统失效，而这些重复事件对于故障树的求解及从故障树到 BDD 的转换都带来了诸多困难，所以在故障树转换为 BDD 之前，化简故障树，删除故障树中的重复事件就变得非常必要。

同一个故障树在转换为 BDD 时，不同的底事件排序会得到不同的 BDD，则遍历该图所消耗的时间和空间复杂度也会有所不同，其所对应的定性、定量分析的效率也会不同。

本文提出的最小深度子树法是基于子树深度和节点数的排序方法，继承了结构式方法的特点，着重强调了树的结构特征及节点之间的相互关联关系，减小了因故障树画法的不同而对构造 BDD 的影响。

在本文提出的方法中，最重要的就是要减小重复事件对于构造 BDD 的影响。首先根据故障树的结构特征，化简该故障树，最大程度地删除树中的重复节点、冗余节点及中间节点，使树结构变得更加简单，然后再根据化简之后得到的故障树，利用子树深度及其所含节点个数对底事件进行排序并构造 BDD。在构造 BDD 时，文献[9]介绍了渐进式的排序方法。该方法基于静态排序动态构造这一思想，即对于故障树的所有底事件采用一定的规则排序，而在构造 BDD 时，根据各个分支的不同特点采用不同的底事件顺序，从而使得构造的 BDD 节点数目极大减少。受此启发，本文提出的方法也采用了静态排序动态构造这一思想。

在对故障树分析之前，根据图 2 对下文中提到的父节点、子节点等名词做一简单介绍。在该故障树中 T1 为该树的根节点，也称为整棵树的顶事件，同时 T1 是 X1、X2 和 T2 的父节点，X1、X2、T2 为 T1 的子节点，X1、X2、T2 互为兄弟节点，以 T2 为根节点的树称为以 T1 为根节点的树的子树。同理，X3、X4 为 T2 的子节点，T2 是它们的父节点，X3、X4 互为兄弟节点。在该树中节点 X1、X2、X3、

X4 均无子节点，将其称为整棵树的底事件。

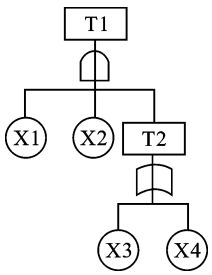

图 2 故障树概念模型

Fig. 2 Fault Tree conceptual model

用本文提出的最小深度子树法对故障树分析时，可分为故障树的化简、BDD 排序以及构造 BDD 三个阶段。

2.1 故障树的化简

在对故障树的底事件排序之前，首先对该树进行化简，删除树中与顶事件无关的节点事件。故障树的化简大多基于吸收律(1)、(2)和幂等律(3)进行。

$$A + A * B = A \tag{1}$$

$$A * (A + B) = A \tag{2}$$

$$A * A = A \tag{3}$$

具体化简步骤如下。

1) 遍历该故障树，若在树中子节点 G2 的逻辑门类型与其父节点 G1 的逻辑门类型相同，则可将 G2 节点下的所有子节点添加到节点 G1 中，使 G2 节点的子节点成为 G1 节点的子节点，并在故障树中删除以 G2 节点为根节点的子树，如图 3 所示。

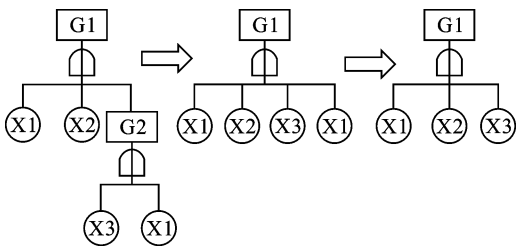

图 3 同门合并示例

Fig. 3 Example of combining the same gate

2) 在整棵树中，若以 G1 为根节点的子树下只有一个子节点 X1，则将 X1 节点添加到 G1 节点的父节点中，使其成为 G1 的父节点的子节点，删除以 G1 为根节点的子树，如图 4 所示；若 G1 节点下只有一个以节点 G2 为根节点的子树再无其它子节点，则将以 G2 为根节点的子树添加到节点 G1 的父节点中，使其成为节点 G1 的父节点的一个子树，删除以 G1 为根节点的子树，如图 5 所示。

3) 利用幂等律和吸收律对步骤 1)和 2)处理后的故障树再次化简，如图 3 所示。

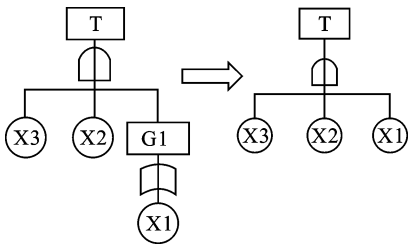

图 4 单个子节点化简示例

Fig. 4 Example of a single child node simplified

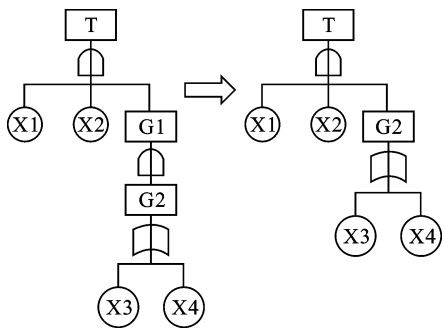

图 5 单个子树化简示例

Fig. 5 Example of a single sub-tree simplified

2.2 BDD 排序

在对故障树化简后，即可利用以下规则对化简后的故障树底事件进行排序，其具体排序规则如下。

1) 从故障树的顶事件开始，遍历整棵故障树，统计树中所有底事件重复出现次数。

2) 从树的根节点开始，计算以根节点的各个子节点为根节点的子树深度，首先，将所有子树深度为 1 的节点即底事件节点在已经排好的序列中查找，若该节点存在于已排好的序列中，则无需重复加入，若该节点尚未参与排序，则按其重复出现的次数，重复出现次数多者优先加入到已排好的序列中；其次，将所有深度非 1 的子树，按其深度由小到大排序，并依次重复规则 2)。

3) 若以同一父节点下的两个子节点为根节点的两棵子树的深度相同，则子树中所含底事件节点数目少者优先参与排列，如果底事件节点数目相同，则按从左向右的顺序依次重复规则 2)。

4) 在同一颗子树中，对其所有底事件节点按其重复出现次数排序，重复出现次数多者优先排列，若重复出现次数相同，则按从上向下、从左向右的顺序排序。

当整棵树中的所有底事件都参与排序后，最后即可得到底事件的有序的集合 $A_1, A_2, \cdots, A_n$。在得到底事件的有序集合的过程中，对每个节点的访问操作都进行了两遍，该算法的复杂度是线性$O(n)$。

2.3 构造 BDD

当排序结束得到有序集合 $A_1, A_2, \cdots, A_n$ 后，则

需通过该序列集合和化简后的故障树逻辑结构来构造 BDD。根据得到的序列集合,依次取得该序列集合中的节点,并对所取得的节点依据故障树的逻辑结构做逻辑运算。因为在排序过程中,一个父节点其所有的子节点的顺序是相邻的,所以有以下结论。

1) 若节点所代表的事件发生:如果该节点的父节点的逻辑门为“与”门,则需对该节点的其它兄弟节点做逻辑运算;如果该节点的父节点的逻辑门为“或”门,则无需对该节点的其它兄弟节点做逻辑运算。

2) 若节点所代表的事件没有发生:如果该节点的父节点的逻辑门为“与”门,则无需对该节点的其它兄弟节点做逻辑运算;如果该节点的父节点的逻辑门为“或”门,则需对该节点的其它兄弟节点做逻辑运算。

以上对本文提出的最小深度子树法做了详细的阐述,即每次首先对深度最小的子树中的节点排序,若子树深度相同,则先对所含节点数目少的排序。利用该方法分析故障树时,对原始故障树进行化简,删除树中的冗余节点,可以提高对底事件排序的效率。在构造 BDD 时采用动态构造的方法,极大减少了 BDD 中冗余节点的产生,简化了所构造 BDD 的结构,因而该方法不仅在某些方面提高了分析计算的效率,同时也易于用软件实现。

2.4 软件实现流程

为了验证该方法的可行性与可编程实现性,对该方法做了深入研究,并以 Microsoft Visual Studio 2010 为平台,以 C# 为主要编程语言,利用该方法编写了一个故障树软件,该算法在软件中的具体流程描述如图 6 所示。

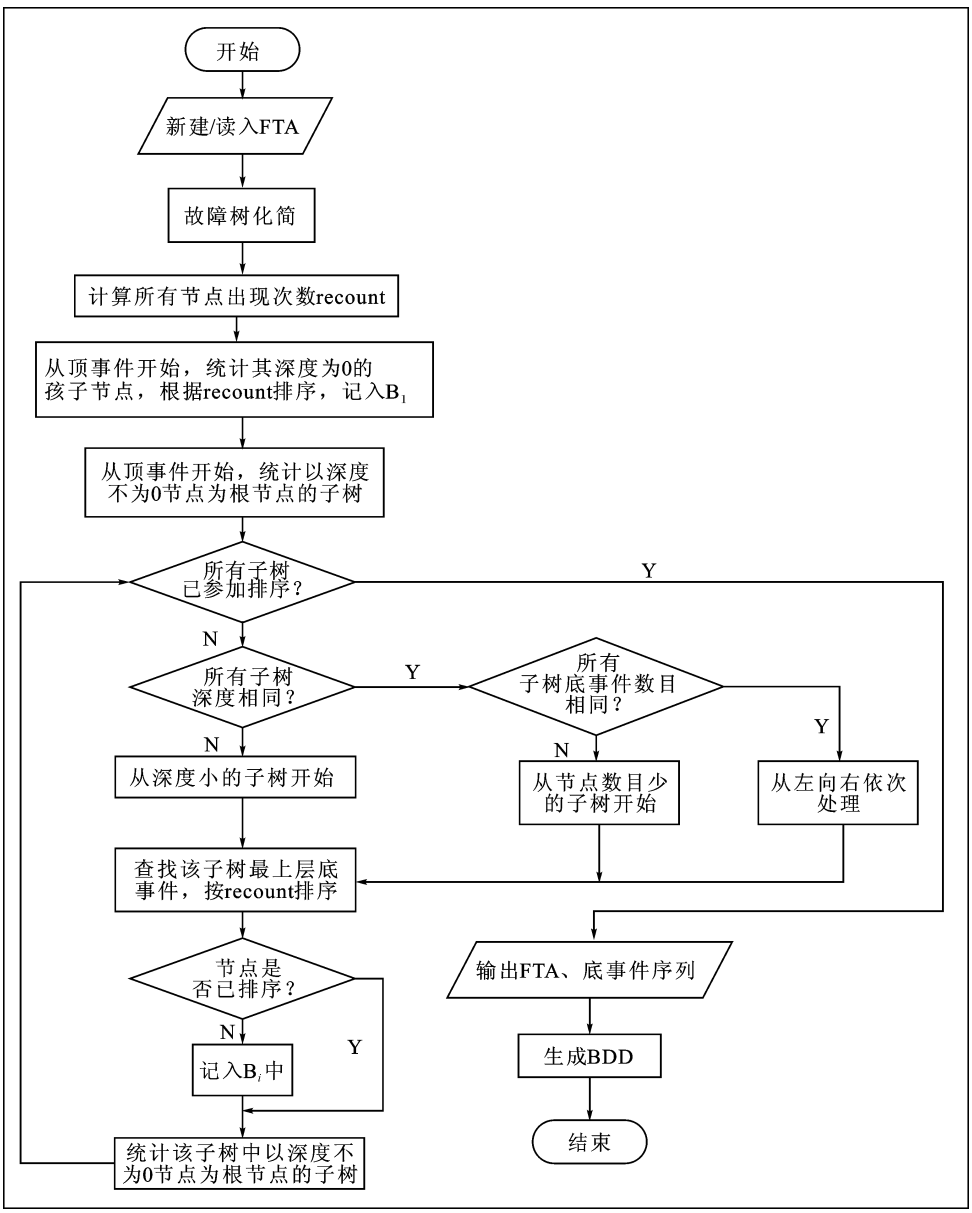

图 6 计算机排序流程图

Fig. 6 The flow chart of the computer sorting

利用该方法编写的软件可以非常方便地将一个复杂系统的故障树化简为一简单的树,通过化简得到的新故障树很容易实现对底事件的排序以及构造 BDD。

为了验证该方法的高效性,本文选取了 31 棵各种类型的故障树,如子节点和父节点的逻辑门不相同但树中有重复节点、子节点和父节点的逻辑门相同但树中无重复节点等等,与用相邻底事件优先法[5]构造的 BDD 做了对比。

通过大量试例分析,可以清晰地看到本文所提方法的优势,即在排序之前尽可能地删除了树中的冗余节点,使得排序过程中极大地避免了重复节点的影响,构造的 BDD 在很大程度上减少了冗余节点的出现,并且结构简单。

通过两种方法构造的 BDD 节点数目对比统计值如表 1 所示。

表 1 BDD 节点数目对比统计值

Tab. 1 Comparative statistics of the number of BDD nodes

BDD 节点数对比	故障树个数	故障树个数所占百分比
最小深度子树法构造的 BDD 节点数 ＞相邻底事件优先法构造的 BDD 节点数	6	19.4%
最小深度子树法构造的 BDD 节点数 ≤相邻底事件优先法构造的 BDD 节点数	25	80.6%

通过表 1 也可清晰直观地看到,在对 31 棵各种类型的故障树分别使用两种方法构造 BDD 后,使用最小深度子树法构造的 BDD 中有 80.6% 的 BDD 所含节点数少于或等于相邻底事件优先法构造的 BDD。同时通过软件的自动化处理使得本文中提出的新方法不管是对底事件进行排序还是构造 BDD,相比以前的方法在效率上都有了很大提升。

3 实例分析

为更加直观清晰地理解这一方法,以航空发动机 WP7 主燃油系统的子系统——加速调节系统中的加速时喘振停车[12]为例,说明本文所提方法的应用。

为方便建立故障树,将基本事件进行了编码,如表 2 所示。

表 2 加速时喘振停车故障事件

Tab. 2 Failure event of the accelerated surge parking

代码	名称	代码	名称	代码	名称
Top	加速时喘振停车	B7	活塞右室油压上升较快	X5	活门磨损泄漏
B1	升压限制器控制的加速时间短	B8	开门力量较小	X6	2 号节流器流量较大
B2	延迟控制器的加速时间短	B9	油压阻力小	X7	1 号节流器流量较大
B3	升压限制器活门关闭早	X1	回输节流器流量大	X8	延迟节流器流量较大
B4	活门开门力量小	X2	升压限制器活门卡在左边	X9	回输节流器流量较大
B5	升压限制器活塞右室油压上升快	X3	延迟节流器流量大		
B6	延迟器活塞和随动活塞左移较快	X4	分配器弹簧松		

加速时喘振停车 Top 是由模块 B1 升压限制器控制的加速时间短或 B2 延迟控制器的加速时间短引起的。其中:

1) 模块 B1 的失效是由 B3、B4、B5、X1、X2 其中任意一个的失效而引发;

2) 模块 B3 是因 B7 和 B8 的共同失效而失效的,而 B7 的失效是由 X6 或 X7 的失效造成的,B8 的失效是由 X4 或 X5 的失效引发的;

3) X4 和 X5 的共同失效造成了 B4 的失效;

4) B5 的失效是由于 X6、X7、B9 中的某一个发生失效而引起的,而 B9 是由于 X6 和 X7 的共同失效而引发的;

5) 模块 B2 的失效是由 X3、X1、B6 中任意一个的失效引发的,而 B6 的失效是由于 X8 或 X9 的失效而造成的。

基于以上分析,建立了图 7 航空发动机加速时喘振停车的故障树。

1) 故障树的化简

① 在故障树中,由于顶事件 Top 与其子节点 B1、B2 的逻辑门均相同为“或门”,所以可将 B1 子树、B2 子树下的子节点添加到顶事件 Top 的子节点中,同时删除中间节点 B1、B2 及其逻辑门;

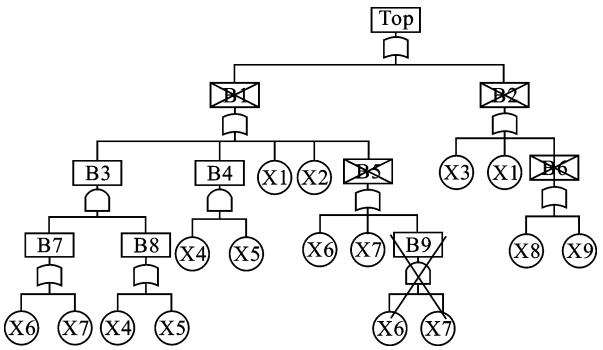

图 7 加速时喘振停车故障树

Fig. 7 Fault tree of accelerating surge parking

② 经过①处理后,以 B5 为根节点的树和以 B6 为根节点的树成为了根节点 Top 的子树,而节点 B5 和 B6 的逻辑门与其父节点 Top 的逻辑门相同,所以对其做与①相同的处理,并删除中间节点 B5、B6 及其逻辑门;

③ 相同逻辑门合并后,以 B9 为根节点的树成为了根节点 Top 的子树,而节点 B9 的孩子节点与 B9 的兄弟节点有若干个是相同的,所以利用吸收律运算,可以删除以 B9 为根节点的子树。

化简时最大程度地删除了树中的重复节点和冗余节点,使得化简后的树更加精炼,求解更加快速,在经过化简后,得到与原故障树相等价的新的故障树,如图 8 所示。

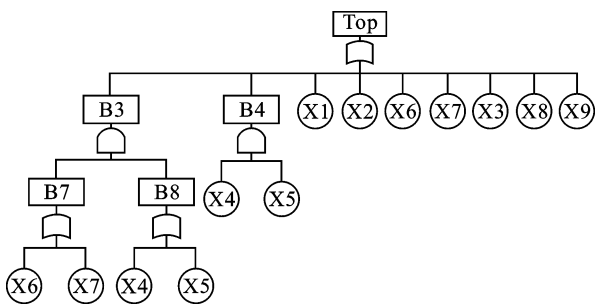

图 8 化简后的加速时喘振停车故障树

Fig. 8 Fault tree of accelerating surge parking after simplification

2) 对化简后的故障树底事件排序

① 在化简后的这棵新故障树中,从根节点 Top 开始遍历整棵故障树,计算出树中各底事件重复出现次数 recount,其中节点 X4、X5、X6、X7 出现次数均为 2,其它节点均为 1。

② 从树根节点 Top 开始,计算以根节点的子节点为根节点的子树深度,得到结果为以 B3 节点为根的子树深度为 3,以 B4 节点为根的子树深度为 2,其余节点均为底事件节点,其深度为 1。首先对深度为 1 的底事件节点按重复出现次数由大到小排序,若重复出现次数相同,则按从左向右的顺序排序。所以对这七个底事件节点排序结果为 A1(X6<X7<X1<X2<X3<X8<X9)。

③ 按子树深度,由小到大对其子节点排序,所以先对子树 B4 中的子节点排序。子树 B4 中含有 X4 和 X5 两个底事件且重复出现次数均为 2,所以按从左到右排序为 A2(X4<X5)。

④ 在经过③处理后,只剩下以节点 B3 为根节点的子树,经遍历可知该子树中的底事件均已参加排序,所以无需再次对其排序。

至此整棵树中的所有底事件节点均已参与排序,根据排序结果构造 BDD,得到如图 9 所示的 BDD。

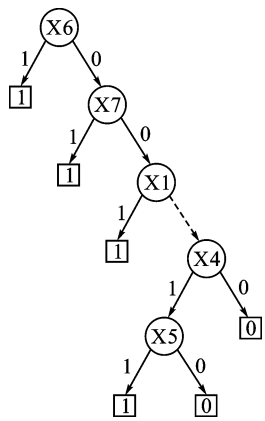

图 9 根据排序结果构造的 BDD

Fig. 9 BDD structure based on the sorting results

由遍历构造出的 BDD 可以很容易地得到对应故障树的最小割集为:{X6},{X7},{X1},{X2},{X3},{X8},{X9},{X4, X5 }共 8 个。

4 结 语

本文提出了一种对故障树底事件排序的新方法——最小深度子树法,利用该方法能够快速高效地将故障树转换为 BDD。该方法主要是将一棵子树作为一个整体进行分析,在简化原故障树的基础上利用子树中节点间的相互关系进行排序,然后再构造 BDD。该方法在故障树的化简过程中,针对不能将整棵树中的所有冗余重复节点都删除掉的问题,可以从重复节点位置逐层向上,找到包含所有相同重复节点的最小子树,可以通过在子树中提取该重复节点,重新构造该子树的方法予以解决。

参考文献:

[1] 郭永基. 可靠性工程原理[M]. 北京:清华大学出版社, 2002: 60-82.

[2] KAREN A R, ANDREWS J D. A fault tree analysis strategy using binary decision diagrams [J]. Reliability Engineering and System Safety, 2002, 78(1): 45-56.

[3] 周斌,黄元亮,黄威.基于模块化分解的故障树分析方法[J].计算机工程, 2015, 14(2): 141-144.
ZHOU Bin, HUANG Yuanliang, HUANG Wei. Fault tree analysis method based on modular decomposition [J]. Computer Engineering, 2015, 14(2): 141-144.

[4] IBANEZ-LLANO C, RAUZY A, MELENDEZ E, et al. A reduction approach to improve the quantification of linked fault trees through binary decision diagrams[J]. Reliability Engineering and System Safety, 2010, 95: 1314-1323.

[5] 孙艳,杜素果.一种二元决策图底事件排序的新方法[J].系统管理学报, 2008, 17(2): 210-216.
SUN Yan, DU Suguo. A novel ordering method of binary decision diagram[J]. Journal of Systems and Management, 2008, 17(2): 210-216.

[6] SINNAMON R M, ANDREWS J D. New approaches to evaluating fault trees[J]. Quality and Reliability Engineering International, 1997, 58(2): 89-96.

[7] RAUZY A. New algorithms for fault tree analysis[J]. Reliability Engineering and System Safety, 1993, 40(3): 203-211.

[8] BARTLETT L M, ANDREWS J D. An ordering heuristic to develop the binary decision diagram based on structural importance[J]. Reliability Engineering and System Safety, 2001, 72(1): 31-38.

[9] BARTLETT L M, Du S. New progressive variable ordering for binary decision diagram analysis of fault trees [J]. Quality and Reliability Engineering International, 2005, 21(4): 413-425.

[10] 徐宗昌.保障性工程[M].北京:兵器工业出版社, 2002.

[11] 邢冀.复杂事故树定性与定量分析算法研究与应用[D].昆明:昆明理工大学, 2009: 22-36.
XING Ji. Complex fault tree qualitative and quantitative analysis of the algorithm research and application [D]. Kunming: Kunming University of Technology, 2009: 22-36.

[12] 邓明,金业壮.航空发动机故障诊断[M].北京:北京航空航天大学出版社, 2012: 187-191.

[13] 苗祚雨,牛儒,唐涛.基于二元决策图的故障树最小割集求解算法研究[J].中北大学学报(自然科学版), 2014, 35(2): 141-146.
MIAO Zuoyu, NIU Ru, TANG Tao. Research on fault tree minimal cut set generation algorithm based on binary decision diagram[J]. Journal of North University of China(Natural Science Edition), 2014, 35(2): 141-146.

(责任编辑 王卫勋)