doi: 10.3969/j.issn.1006-1576.2011.06.027

基于故障树的在轨服务目标航天器故障检测

邢晓辰[1]，蔡远文[2]，任江涛[1]
（1. 装备指挥技术学院 研究生队，北京 101416；2. 装备指挥技术学院 试验指挥系，北京 101416）

摘要：针对构建在轨故障检测知识库中知识获取和知识表示困难的问题，将故障树应用到在轨服务目标航天器故障检测知识库中。采用将专家经验知识建立成故障树的形式，通过对故障树的定性和定量分析，求出其最小割集和底事件重要度，减少知识库中的知识冗余，并以在轨运行的目标航天器上某负载的供电系统发生故障为例进行应用分析。分析结果验证了该方法的可行性。

关键词：故障树分析；知识库；知识获取；知识表示
中图分类号：TP277　　**文献标志码**：A

Fault Detection of On-Orbit Service Object Spacecraft Based on Fault Tree

Xing Xiaochen[1], Cai Yuanwen[2], Ren Jiangtao[1]
(1. Brigade of Graduate, Institute of Command & Technology of Equipment, Beijing 101416, China;
2. Dept. of Testing & Command, Institute of Command & Technology of Equipment, Beijing 101416, China)

Abstract: Knowledge acquisition and representation in constructing on orbit fault detection knowledge base are difficult. To solve this problem, experience knowledge can be used to establish fault tree. Through the qualitative analysis and quantitative analysis of the fault tree, minimum cut sets and importance of the bottom event are obtained, and knowledge redundancy is reduced. The method is demonstrated to be feasible by an example of electricity supply problems of on-orbit spacecraft.

Keywords: fault tree analysis; knowledge base; knowledge acquisition; knowledge representation

0　引言

在轨飞行器对目标航天器实施在轨故障检测，由于空间运行环境的影响，无法像地基故障检测那样采用大量精密仪器设备和计算机资源。基于故障检测知识库和推理机制的专家系统技术，是近年来智能故障检测领域兴起的一项尖端技术，不需耗费大量星载资源，就可对目标航天器实施故障检测。“知识库”是专家系统的核心，知识获取能力和知识表示水平的高低，直接影响推理机制推理的效率和知识库维护的方便性，能否解决好这一问题对于构造知识库至关重要[1]。

由于专家系统技术存在知识获取和知识表示困难的问题，并且难以调和处理多领域专家知识的矛盾。因此，笔者将专家经验知识建立成故障树的形式，直观地反映出故障与其产生原因之间的逻辑关系，并通过简化的故障树建立故障检测知识库。

1　故障树分析方法

故障树分析方法（fault tree analysis, FTA）是将系统故障的各种原因（包括硬件、环境、人为因素等），由总体至部分，按树枝状结构，自上而下逐层细化的分析方法。目前，故障树分析方法已渗透到各个工业应用领域，尤其是在航空、航天、核能等应用领域。该方法在可靠性和安全性分析中被公认为是简单、有效和最具有发展前途的分析方法。美国在导弹和运载火箭型号的研制中，规定必须要进行故障树分析[2]。

故障树模型是描述被检测对象结构、功能和关系的一种定性因果模型，包含顶事件、中间事件和底事件。用相应的符号代表这些事件，再用适当的逻辑门（与门、或门、异或门等）把这些事件连接成树状图，这种能体现故障传播逻辑关系的倒立的树状图形就称作故障树[3]。具体的符号如图 1。

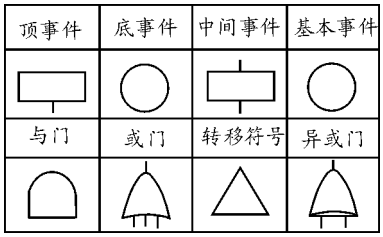

图 1　故障树符号图

收稿日期：2011-01-19；修回日期：2011-03-14
基金项目：国家高技术发展计划项目（2010AA7045007）
作者简介：邢晓辰（1988—），男，山东人，硕士研究生，从事自动化测试与控制研究。

对故障树可以进行定性分析和定量分析。对故障树进行定性分析的目的主要是找出它的所有最小割集。计算系统故障树的最小割集，常采用上行法和下行法。下面以图 2 所示故障树为例，简要分析求最小割集的 2 种方法。下行法即从故障树顶事件开始，由上至下顺次把上一级事件置换为下一级事件，遇到与门将输入事件横向并列写出，遇到或门将输入事件竖向写出，直到把全部逻辑门都置换成底事件为止，此时最后一列代表所有割集，将之简化吸收得最小割集，如表 1。

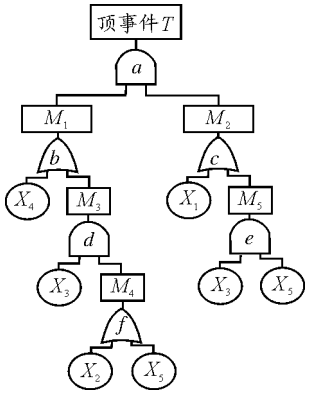

图 2 故障树示意图

表 1 下行法求解过程

分析步骤								最小割集
1	2	3	4	5	6	7	8	
T	M_1M_2	X_4M_2	X_4X_1	X_4X_1	X_4X_1	X_4X_1	X_4X_1	X_4X_1
		M_3M_2	X_4M_5	$X_4X_3X_5$	$X_4X_3X_5$	$X_4X_3X_5$	$X_4X_3X_5$	$X_3X_2X_1$
			M_3X_1	$X_3M_4X_1$	$X_3X_2X_1$	$X_3X_2X_1$	$X_3X_2X_1$	X_3X_5
			M_3M_5	$X_3M_4M_5$	$X_3X_5X_1$	$X_3X_5X_1$	$X_3X_5X_1$	
					$X_3X_2M_5$	$X_3X_2X_3X_5$	$X_2X_3X_5$	
					$X_3X_5M_5$	$X_3X_5X_3X_5$	X_3X_5	

上行法即从故障树最底层开始，利用逻辑与门和或门运算法则，顺次向上，将中间事件用底事件表示，直到顶事件为止，得到割集，简化吸收后得最小割集，运算步骤如下：

$$M_4=X_2+X_5$$

$$M_3=X_3M_4=X_3(X_2+X_5)=X_2X_3+X_3X_5$$

$$M_1=X_4+M_3=X_4+X_2X_3+X_3X_5$$

$$M_5=X_3X_5$$

$$M_2=X_1+X_3X_5$$

$$T=M_1M_2=(X_4+X_2X_3+X_3X_5)(X_1+X_3X_5)=$$
$$X_1X_4+X_1X_2X_3+X_1X_3X_5+X_3X_4X_5+X_2X_3X_5+$$
$$X_3X_5=X_1X_4+X_1X_2X_3+X_3X_5$$

故该故障树的最小割集为$\{X_1,X_4\}$，$\{X_1,X_2,X_3\}$，$\{X_3,X_5\}$。

对故障树进行定量分析的目的是求出系统的不可靠度和各底事件的重要度，重要度包含概率重要度和关键重要度[4]。故障树的不可靠度表示的是顶事件发生即系统故障的概率，底事件的重要度表明了各个底事件对系统故障的影响和作用大小。通过对底事件重要度的分析，可以改善系统的设计，明确系统在运行过程中需要重点监控的部位，为确定系统的故障检测方案发挥重要作用。

求出故障树的最小割集后，为防止直接对故障树定量分析时会产生组合爆炸问题，对最小割集采取不交化处理，通过不交化方法得到不交和，就可求得顶事件发生概率，即系统不可靠度。设故障树有 n 个底事件，每个底事件发生概率为 F_i（$1\le i\le n$），顶事件发生概率为 F_s，则第 i 个底事件的概率重要度定义为 $\frac{\partial F_s}{\partial F_i}$ 或 $\frac{\partial R_s}{\partial R_i}$，关键重要度定义为 $\frac{\partial F_s}{\partial F_i}\bullet\frac{F_i}{F_s}$ 或 $\frac{\partial R_s}{\partial R_i}\bullet\frac{1-R_i}{1-R_s}$。

2 FTA 在知识获取和知识表示中的应用

知识库通常是由计算机工程师将所获得的人类检测专家的知识与经验，编制成计算机可读的一系列规则，存入计算机而形成，目前应用最广泛的是“若-则”（if…then…）形式的产生式规则。

知识获取就是从知识丰富的源头收集、整理、形式化知识，加入到知识库中，或对已有的知识库进行知识求精。知识获取可以通过手工完成，也可以自动完成。手工方式是由知识工程师从一个或多个领域专家那里接收知识；自动模式则可使用机器学习系统，自主地学习和接收来自外界的知识[5]。知识表示就是研究用机器语言表示这些知识的可行性、有效性的一般方法，可看作是将知识符号化，并输入到计算机的过程和方法。在选择知识表示方法时，应从表示能力、推理效率、正确性及结构性等几个方面加以综合考虑。装备故障诊断系统中，

故障原因和故障现象之间表现出极其错综复杂的关系。同一种故障征兆往往对应着多种故障原因，同一种故障原因又对应着多种故障征兆。能否处理好这一复杂关系，决定了所构建专家系统的质量。

故障树的顶事件对应于专家系统要分析解决的任务，底事件对应于专家系统的推理结果，由顶至底的层次和逻辑关系则对应于专家系统的推理过程。故障树模型体现了故障传播的层次性和子节点（即下层故障）与父节点（即上层故障）之间的因果关系。对某一个节点而言，它对上层是故障源子节点，对下层是故障现象父节点，模型中各个子节点与其父节点构成了正向因果关系链，这种因果关系链与“if…then…”型故障检测规则本质上相一致，二者可以互换。

由于故障树的一个割集代表系统的一种失效模式，同时也对应着知识库中的一条规则；割集里的基本事件就是该失效模式的基本原因，即知识库中规则的结论（故障源或故障原因）；从顶事件到割集的路径，是该割集与其它割集相区别的中间条件，对应着知识库中规则的前提，因此，通过割集能够将故障树同检测专家系统的知识库相联系起来。

由于故障树的各节点知识之间有很强的逻辑和层次关系，因而在查询由故障树生成的知识库时，具有很强的针对性，并且诊断的模块性很强，较符合人类思维过程。通过构造故障树并对故障树进行定性和定量分析，即可解决构建在轨故障检测专家系统时经验知识获取和表示困难等问题。

3　应用实例

以在轨运行的目标航天器上某负载的供电系统发生故障为例，对故障树分析中的最小割集法在故障检测知识库中知识获取和知识表示方面的应用进行研究。图 3 为目标航天器上某负载的供电结构图，通过 2 条线路的冗余设计保证负载的不间断供电。

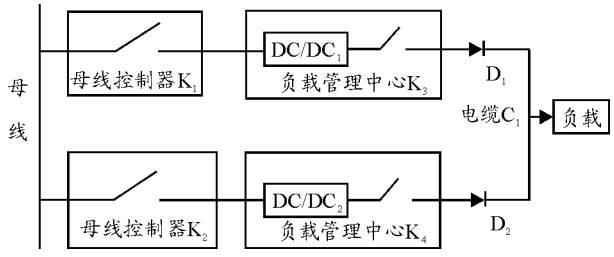

图 3　某负载的供电原理图

由图 3 可得该负载的无输入电压故障模式的故障树如图 4。

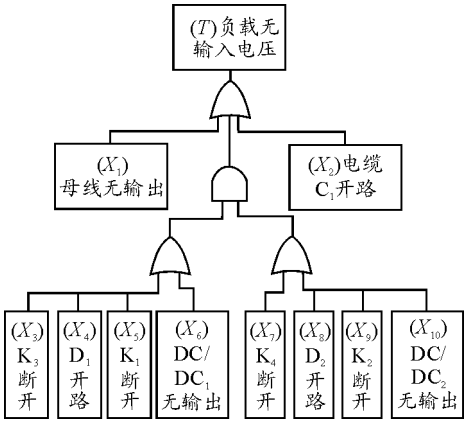

图 4　某负载无输入电压故障模式故障树

通过下行法或上行法可求得该故障树的最小割集为$\{X_1\}$，$\{X_2\}$，$\{X_3,X_7\}$，$\{X_3,X_8\}$，$\{X_3,X_9\}$，$\{X_3,X_{10}\}$，$\{X_4,X_7\}$，$\{X_4,X_8\}$，$\{X_4,X_9\}$，$\{X_4,X_{10}\}$，$\{X_5,X_7\}$，$\{X_5,X_8\}$，$\{X_5,X_9\}$，$\{X_5,X_{10}\}$，$\{X_6,X_7\}$，$\{X_6,X_8\}$，$\{X_6,X_9\}$，$\{X_6,X_{10}\}$。

由其最小割集可以看出，当该负载出现无输入电压这一故障时，其最基本的原因共有 18 种可能，显然为下一步专家系统对故障的推理分析带来难度。此时，可以对故障树采取定量分析，求故障树的不可靠度和底事件的概率重要度。在对故障树进行定量分析前，一般要作如下假设：一是底事件间相互独立；二是顶事件和底事件只考虑故障和正常 2 种状态。假设本文中故障树符合此条件。

示例故障树中各个底事件发生的概率（不可靠度）分别为：$F_1=F_2=0.15$，$F_3=F_4=F_5=F_6=F_7=F_8=F_9=F_{10}=0.12$。由于本案例中故障树最小割集的数目庞大，有 18 个之多，直接通过最小割集求不可靠度会很复杂。将故障树中与门变换为或门，或门变换为与门，各事件变为相应对立事件，就得到原故障树的对偶树。对偶树最小割集对应原故障树最小路集，通过对原故障树最小路集不交化求得系统可靠度。

顶事件的对立事件用 S 表示，则对偶树的最小割集即原故障树的最小路集表达式为：

$$\overline{T}=S=\overline{X_1X_2}(\overline{X_3X_4X_5X_6}+\overline{X_7X_8X_9X_{10}})=\overline{X_1X_2X_3X_4X_5X_6}+\overline{X_1X_2X_7X_8X_9X_{10}}$$

对其进行不交化得：

$$S=\overline{X_1X_2X_3X_4X_5X_6}+\overline{\overline{X_1X_2X_3X_4X_5X_6}}\times\overline{X_1X_2X_7X_8X_9X_{10}}=$$

$$\overline{X_1X_2X_3X_4X_5X_6}+(X_3+\overline{X_3}X_4+\overline{X_3X_4}X_5+\overline{X_3X_4X_5}X_6)\overline{X_1X_2X_7X_8X_9X_{10}}$$

顶事件不发生的概率为：

$$P(S)=R_1R_2R_3R_4R_5R_6+(F_3+R_3F_4+R_3R_4F_5+R_3R_4R_5F_6)R_1R_2R_7R_8R_9R_{10}=$$
$$0.85\times0.85\times0.88\times0.88\times0.88\times0.88+(0.12+0.88\times0.12+0.88\times0.88\times0.12+0.88\times0.88\times0.88\times0.12)\times0.85\times0.85\times0.88\times0.88\times0.88\times0.88\approx0.606\ 724$$

所以顶事件发生（系统发生故障）的概率为：

$$P(T)=1-P(S)=1-0.606\ 724=0.393\ 276$$

由于第 i 个底事件的概率重要度为 $\frac{\partial R_s}{\partial R_i}$，所以各底事件的概率重要度分别为：

$$\frac{\partial R_s}{\partial R_1}=\frac{\partial R_s}{\partial R_2}\approx0.713\ 793,$$

$$\frac{\partial R_s}{\partial R_3}=\frac{\partial R_s}{\partial R_4}=\frac{\partial R_s}{\partial R_5}=\frac{\partial R_s}{\partial R_6}=\frac{\partial R_s}{\partial R_7}=\frac{\partial R_s}{\partial R_8}=\frac{\partial R_s}{\partial R_9}=\frac{\partial R_s}{\partial R_{10}}\approx0.197\ 095\ 4$$

显然，当在轨运行的目标航天器上该负载的电力能源供应发生故障时，事件 X_1（母线无输出）和 X_2（电缆 C_1 开路）相比于其它底事件而言对发生故障的影响最大。

4 结束语

笔者通过构造故障树来获取和表示个性知识，并通过对故障树求最小割集、不可靠度和底事件概率重要度，解决了构建在轨故障检测知识库中的知识获取、知识表示问题和知识冗余现象，完善了故障检测知识库，为后续故障推理机制的建立打下了基础。通过对在轨运行的目标航天器上某负载的电力供应故障问题进行分析，验证了该方法的可行性。

参考文献：

[1] 杨军, 冯振声, 黄考利, 等. 装备智能故障诊断技术[M]. 北京: 国防工业出版社, 2003.

[2] 金星, 洪延姬. 系统可靠性与可用性分析方法[M]. 北京: 国防工业出版社, 2007.

[3] 唐志凌. 故障诊断专家系统在某武器系统中的应用[D]. 重庆: 重庆大学硕士学位论文, 2005.

[4] 刘帆. 基于混合智能的航天器故障诊断系统研究[D]. 北京: 北京科技大学信息工程学院, 2009.

[5] 郑晓霞, 王九龙. 航天器自主故障检测和识别的知识库设计研究[J]. 航天器工程, 2005, 14(4): 6-10.

****

（上接第 80 页）

```
    msgs[0].addr= HY_I2C_ADDRESS>>1;
    msgs[0].flags=I2C_M_DIBCOM_WR_RD;
     msgs[0].len     = 2;
     msgs[0].buf     = buffer;
     // send from DDC line
     return I2C_transfer(adap, msgs, 1) ;
}
```

具体的设备驱动程序完成之后，将 AXP192 设备驱动的配置选项添加到相应的 kconfig 文件中，在配置内核选项时就可以把 AXP192 设备驱动添加到内核中，也可以自行编写相应的上层应用程序，对 AXP192 驱动进行测试。

4 结束语

笔者设计的 I2C 设备的驱动方法，增加了系统的安全性，方便了管理。但针对 I2C 设备的驱动程序的编写并不一定要完全遵循 Linux I2C 驱动的体系结构，完全可以把它当成一个普通的字符设备，以省去一些繁琐的架构。本研究得到中国地质大学（北京）教学实验室开放基金支持，在此表示感谢。

参考文献：

[1] 范恩魁, 陈亚军, 彭钟钟, 等. 嵌入式 Linux2.6 I2C 总线及设备驱动分析与实现[J]. 微计算机应用, 2009, 30(6): 71-73.

[2] 宋宝华. Linux 设备驱动开发详解[M]. 北京: 人民邮电出版社, 2008: 333-360.

[3] 何亚军, 邓飞其. 嵌入式 Linux 中 I2C 总线驱动程序设计[J]. 计算机工程与设计, 2008, 29(10): 181-187.

[4] Linux 设备驱动程序[M]. 魏永明, 耿岳, 钟书毅, 译.北京: 中国电力出版社, 2006: 374-380.

[5] 张圣华, 喻晓峰, 石崇林. 基于 AT91RM9200 和嵌入式 Linux 的 I2C 总线驱动程序[J]. 兵工自动化, 2007, 26(7): 92-93.

[6] 孟令公, 朱宏, 杨忠孝等 Linux 下基于 I2 C 协议的 RTC 驱动开发[J]. 现代电子技术, 2009, 20: 32-35.

[7] 刘名博, 邓中亮. 基于 ARM 的嵌入式 Linux 操作系统移植的研究[J]. 计算机系统应用, 2006, 44(11): 87-88.

[8] 朱南浩, 李正祥. 嵌入式Linux中I2C设备驱动程序的研究与实现[J]. 微计算机信息, 2010, 26(4): 67-69.

[9] 冯国进. 嵌入式Linux驱动程序设计从入门到精通[M]. 北京: 清华大学出版社, 2008: 73-75.

[10] 李祥兵, 郑扣根. Linux 中 I2C 总线驱动程序的开发[J]. 计算机工程与设计, 2005, 26(1): 41-43.