

一种结合线性时序逻辑和故障树的软件安全验证方法

王 飞¹ 沈国华¹ 黄志球¹ 马 琳¹ 刘 畅² 李海峰² 廖莉莉¹

(南京航空航天大学计算机科学与技术学院 南京 210016)¹

(中国航空综合技术研究所 北京 100028)²

摘 要 嵌入式软件在安全关键领域的广泛应用使得保障软件的安全性成为学界的研究热点。故障树技术是工业界常用的传统的安全分析方法之一。然而,传统的故障树无法精确描述安全关键系统中具有时序特征的系统故障。针对此问题,给出了一种结合线性时序逻辑和故障树的软件安全验证方法。该方法运用线性时序逻辑对故障树进行形式化规约,从中抽取出软件安全属性并用时序逻辑公式进行描述,用以支持对安全关键软件的模型检验。最后,以某机载控制系统软件数据处理故障模块的模型检验为例,来说明该方法的有效性和可行性。

关键词 故障树分析,模型检验,线性时序逻辑,安全关键系统,安全属性

中图分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.12.016

Method Combining Linear Temporal Logic and Fault Tree for Software Safety Verification

WANG Fei¹ SHEN Guo-hua¹ HUANG Zhi-qiu¹ MA Lin¹ LIU Chang² LI Hai-feng² LIAO Li-li¹

(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)¹

(China Aero-polytechnology Establishment, Beijing 100028, China)²

Abstract Embedded software is playing an important role in safety critical field. How to ensure the safety of safety-critical software has recently become a research focus. The fault tree technique is a safety analysis method which is commonly used in traditional industry. However, FTA (Fault Tree Analysis) itself lacks formal temporal semantics. To solve the problem, this paper proposed a method to verify the safety of embedded software combining linear temporal logic and FTA. Applying linear temporal logic to formal specification of the fault tree, it extracts the software safety properties from the formal fault tree and describes them with the temporal logic. Expert can use the extracted safety properties to do model checking of the safety-critical software, to analyze and verify its reliability and safety. The paper applied the model checking to a module of a safety-critical airborne software to demonstrate the method in detail.

Keywords Fault tree analysis, Model checking, Linear temporal logic, Safety-critical system, Safety property

1 引言

近年来,嵌入式软件在轨道交通、核工业、航空航天等安全关键领域的重大工程中得到广泛应用。为了提供更为复杂和可靠的服务,嵌入式软件变得越来越复杂,其规模也在不断扩大,使得软件对整个系统的影响逐渐占据了统治地位。安全关键系统中的软件失效会造成财产的损失、环境的破坏甚至人员的伤亡^[1]。因此,提高软件安全性已成为当前工业界和计算机领域共同关注的重要问题。

故障树分析法(Fault Tree Analysis, FTA)^[2]是目前在工业界应用最为广泛的安全性分析方法之一。近年来,随着FTA在各个领域特别是在各类软件系统中的广泛应用并日

益占据主导地位,原有故障树已无法描述某些不断出现的新系统故障。总体来说,传统故障树主要存在以下不足^[3-5]:建树困难,并且对很多特殊事件难以进行处理;底事件的失效数据存在不易收集和应用或无法应用等问题,这给定量分析带来了很大的困难;NP困难及其算法复杂性仍是故障树分析中面临的重大难题;已有的故障树辅助分析软件缺乏针对性或效率不高,且不支持对时序的分析;传统故障树的描述能力有限,无法描述动态系统、实时系统中的动态和时序故障,且对传统故障树的扩展工作缺乏统一规范等。

为了推广应用FTA技术,国内外研究者在故障树的建树、分析、理论扩充等方面做了大量工作。文献[6]通过在故障树中引入时序逻辑门而提出一种面向过去的时序故障树,

到稿日期:2015-02-12 返修日期:2015-06-13 本文受国家自然科学基金(61272083),国家高技术研究发展计划(863)(2009AA010307),国家国防科技工业局技术基础科研项目(Z052013B009,JSJC2013205C507)资助。

王 飞(1990—),男,硕士生,CCF 学生会会员,主要研究方向为安全工程、需求工程、本体匹配、Web 服务, E-mail: wangfei2014@nuaa.edu.cn; 沈国华(1976—),男,博士,副教授,硕士生导师,CCF 高级会员,主要研究方向为领域工程、软件度量、语义 Web、描述逻辑;黄志球(1965—),男,教授,博士生导师,CCF 杰出会员,主要研究方向为软件工程;马 琳(1988—),女,硕士,主要研究方向为软件工程、形式化方法、软件安全性分析等;刘 畅(1979—),男,博士,高级工程师,主要研究方向为软件安全性工程、软件工程、需求验证等;李海峰(1981—),男,博士,高级工程师,主要研究方向为软件可靠性工程、软件测试、软件评估等;廖莉莉(1989—),女,硕士生,主要研究方向为本体匹配、语义 Web、描述逻辑。

增加了故障树自身的表达能力,但这种时序逻辑缺乏工具的支持且故障描述能力有限;文献[7]通过在故障树中引入动态门构建动态故障树,用以描述动态实时系统中故障事件之间的依赖关系;文献[8]用区间时序逻辑规约的故障树描述了实时系统中故障事件之间的时序依赖。但这些工作主要集中于对故障树的研究,而对软件安全性的验证工作则有所欠缺。文献[9]运用 TCTL 描述故障树顶事件的性质,并用 UP-PAAL 对这些性质进行验证。文献[10]提出用模型检验的验证方法来对故障树进行验证,以保证故障树分析的正确性,但工程量非常庞大,难以在实际工作中推广应用。文献[11]和文献[12]则运用 CTL 对故障树进行形式化规约,并将其运用到核反应堆控制系统的 SMV 验证中。然而这些工作在待验证性质的提取方面仍缺乏相应的研究。

基于上述已有研究,本文提出一种将线性时序逻辑与故障树相结合的方法,并从中抽取相关系统安全属性进行建模与验证,以此来改善模型检验中安全属性全凭经验生成的现状。本文将从以下方面进行阐述:(1)提出一种将线性时序逻辑与传统故障树相结合的方法来构建线性时序逻辑故障树;(2)在此基础上,对安全属性进行提取并形式化建模,然后对构建的模型进行安全属性验证;(3)最后,本文将结合一个飞机襟缝翼控制软件系统进行案例分析,以说明本文方法的有效性和可行性。

2 线性时序逻辑故障树

2.1 线性时序逻辑理论

线性时序逻辑(Linear Temporal Logic, LTL)^[13]是一种形式化的符号,主要用于对计算进行推理。LTL 以时间的线性顺序来描述单一的计算路径,并将路径作为命题的判断对象。它主要用于描述安全性和活跃性,其中安全性描述某不良性质永远不会发生,活跃性描述某良好性质最终会发生。

在 LTL 中主要包含 X、G、F、U、R 和 W 6 个时序连接词,其中 X(next)表示“下一状态”;G(Globally)表示“所有未来状态”;F(Future)表示“某个或某些未来状态”;U(Until)表示“直到”;R(Release)表示“释放”;W(Weak-until)表示“弱-直到”。LTL 运用这些时态连接词将时间建模成一系列“状态”的序列,并向未来无限延伸,这个状态序列称为计算路径(简称“路径”)。LTL 公式一般用来描述并发程序的性质,通过检测程序是否具有由 LTL 公式描述的性质可以验证程序的解决方案是否符合需求。

2.2 结合时序逻辑的故障树构建

随着实时软件系统的出现并不断向安全关键领域渗透,事件之间的时序因素日益成为系统故障树分析过程中的一个主要因素,而传统故障树并不具备直观描述这种时序因素的能力。为了增强传统故障树的表达能力,使其能够更直观地表达故障事件间的时序关系,本文借鉴文献[6],运用 LTL 对传统故障树进行扩展,并进行形式化规约,提出了线性时序故障树(Linear Temporal Fault Tree, LTFT)^[14]的概念。

LTFT 保留了传统故障树中的所有逻辑门和事件,同时借鉴了 PLTLP 时序故障树^[6]的思想,将与 LTL 时态连接词相对应的线性时序逻辑门引入到传统的故障树中,使得 LTFT 能够对复杂的时序故障进行描述和规约。LTFT 中的各时序逻辑门所表达的含义如表 1 所列。

表 1 LTFT 中时序逻辑门与 LTL 时态连接词映射表

LTL 的时态连接词	LTFT 时序逻辑门	说明
G	GATE  “G”门	“G”门有一个输入事件和一个输出事件。当且仅当“G”门下的输入故障事件在整个系统运行周期内永真,“G”门连接的上层输出故障事件才会发生。
F	GATE  “F”门	“F”门有一个输入事件和一个输出事件。仅当“F”门下的输入故障事件在整个系统运行周期内某一时刻为真,“F”门连接的上层输出故障事件发生。
U	GATE  “U”门	“U”门有两个输入事件(A,B)和一个输出事件(C),当“AUB(即事件 A 为真,直到事件 B 为真)”成立时,输出事件 C 发生。

在构建 LTFT 时,需要对传统的故障树构建方式作出一些改进,使其能够表示含有时序相关的信息。具体而言,对于已有的半形式化传统故障树,需要对其事件节点进行形式化命名(例如:将“顶事件”命名为 TopEvent,将“基本事件”命名为 Event 或 BaseEvent 等),同时引入时序逻辑门描述事件之间的因果关系,并使用 LTL 公式(例如: $p \wedge q, \neg q U p$)来描述故障树中的基本事件和各个事件发生的条件,同时构建一张“基本事件—LTL 基本命题公式”映射表。

下面以机载软件的数据输入错误模块的建模为例,说明如何根据现有故障树构建 LTFT。其中图 1(a)为传统故障树建模,图 1(b)为 LTFT 建模,表 2 为“基本事件—LTL 基本命题公式”的映射表。

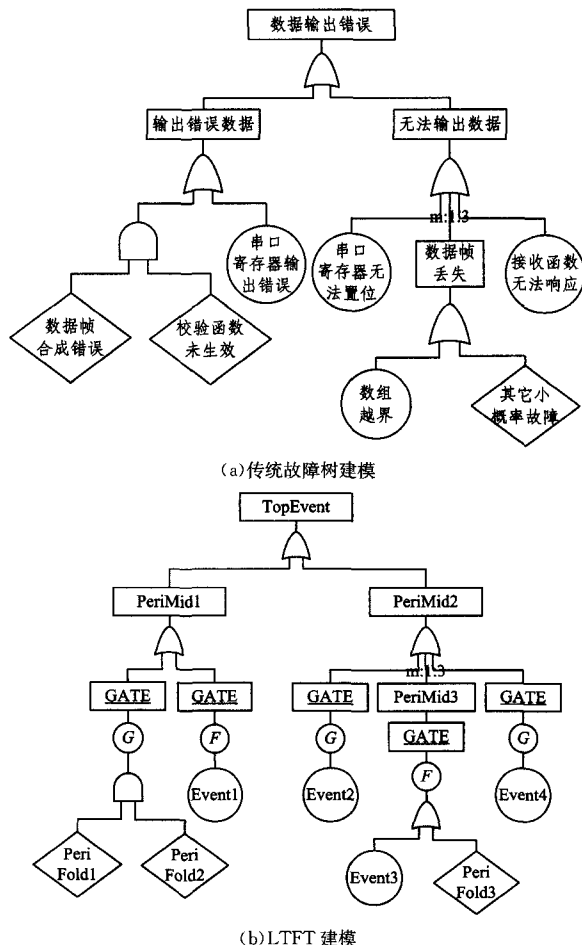


图 1

表2 图1(b)中LTFT“基本事件-LTL基本命题公式”映射表

叶节点基本事件	基本命题	LTL表达式
串口寄存器 输出错误	Event1	$\neg(g_SCITXBUF \neq 0 \wedge g_SCITXBUF \neq 1 \wedge g_SCITXBUF \neq 2)$
串口寄存器 无法置位	Event2	$\neg(g_SCITXBUF == 0)$
数组越界	Event3	$(g_dRegister > MAX_REG_LENGTH)$
接收函数无法响应	Event4	$\neg(g_SciaRx_Ready == 1)$

2.3 LTFT 约简

为了简化安全属性的提取过程,需要依据安全属性的提取需求对构建的 LTFT 进行约简。本文采用目前国际公认的对故障树和网络图结构进行约简的规则——FAUNET^[15]对 LTFT 进行约简。

FAUNET 约简规则主要包括删除规则、收缩规则、吸收规则以及提取公因式 4 个约简规则。重复进行这 4 步约简规则,直到故障树不能被再次约简为止,即可得到紧凑表示系统故障树的极小形式。

结合传统故障树约简经验,对支持安全属性验证的 LTFT 约简大致可以划分为如下步骤。

1) 对所有“未展开事件”进行剪枝或转换

通过分析,对 LTFT 中的所有“未展开事件”大致可以分为 4 类,分别为:故障原因不明确的事件,发生概率微小到可以忽略的事件,在测试阶段被检测出的故障的事件以及需要进一步规约的程序进程切换时可能出现的问题的事件。

对 LTFT 进行约简的目的是提取支持安全性验证的安全属性。故在 LTFT 约简过程中,对于前三类“未展开事件”可以直接进行剪枝处理,而由于最后一类事件是安全性验证的关注重点,因此应当依据其数据完整性,将其转换为“基本事件”再进行 LTL 规约或剪枝。

2) 规范特殊逻辑门

按照 FAUNET 约简规则将带有优先“与”门、表决门、禁门和异或门等特殊逻辑门的故障树转换为只含有逻辑“与”门(以下简称“与门”)、逻辑“或”门(以下简称“或门”)、非门和时序逻辑门的故障树。具体而言,有以下几条转换规则:

- 将异或门等效转换为与门、或门和非门的组合;
- 将表决门等效转换为与门和或门的组合;
- 对禁门进行最坏假设情况处理;
- 将优先“与”门等效转换为逻辑“与”门;
- 将时序逻辑门尽量向故障树的底端移动。

3) 遵循 FAUNET 规则对“规范化故障树”自底向上逐层约简

依据 FAUNET 约简规则对图 1(b)所示的 LTFT 进行约简,得到最终简化的 LTFT,如图 2 所示。

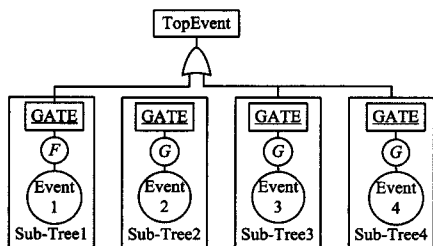


图2 约简后的LTFT

3 安全属性的提取与验证

3.1 安全属性的提取

对故障树进行定性分析的主要目的是确定系统的故障模

式和安全模式,从而定位系统的薄弱环节。其中系统的全部故障模式可以通过割集分析得出,实际分析过程中我们关注的只是这些割集中的最小割集,在求解故障树最小割集的众多算法中,最著名的是 Fussell-Vesely 算法(“下行法”)和 Semanderes 算法(“上行法”)。

本文针对 Semanderes 算法进行改进和扩充,在此基础上提出了 LTL-Semanderes 算法。该算法可以计算出表示逻辑门输出故障事件的 LTL 公式,其基本思想是:从基本故障事件开始,自底向上逐级规约。对于故障树中的每个节点,重复上述布尔化简原则,直到故障树中所有中间节点均被处理。该方法的递归算法见算法 1。

算法1 LTL-Semanderes 算法

```

calLTL(Node a)
{
    if(a 是叶子节点) return a.LTL;
    //若 a 是叶子节点,直接返回描述 a 的 LTL 公式
    else if(a 是逻辑门){
        //若 a 是逻辑门,则对 a 的每个子节点 b 递归调用 calLTL() 函数
        for(each b in a.children){
            calLTL(b);
        }
        //得到 a 的所有子节点的 LTL 公式后,计算描述 a 的 LTL 公式
        cal(a);
    }
    else{
        //若 a 是中间事件节点,则计算 a 的孩节点,即逻辑门的 LTL 公式并赋值给 a
        calLTL(a.children);
        a=a.children;
        return a.LTL;
    }
}

//对逻辑门节点计算 LTL 表达式的方法
cal(Node a){
    if(a 的节点类型为“OR”)
        将 a 的所有子节点用逻辑“或”连接;
    else if(a 的节点类型为“AND”)
        将 a 的所有子节点用逻辑“与”连接;
    else if(a 的节点类型为“NOT”)
        将 a 的子节点取反;
    else if(a 的节点类型为“G”)
        a 的表达式“G”和“calLTL(a.child)”的连接;
    else if(a 的节点类型为“F”)
        a 的表达式“F”和“calLTL(a.child)”的连接;
    else
        a 的表达式“calLTL(a.child1)”、“U”和“calLTL(a.child2)” 3 项的连接;
}

```

运用该算法对图 2 中简化后的 LTFT 进行割集分析,得到 LTFT 的每个割集的 LTL 公式,如表 3 所列,其中“formula()”表示每个叶节点包含的基本命题,在实例分析中要根据实际情况进行替代。

通过对 LTFT 的定性分析,实现用 LTFT 的基本事件完成对故障树顶事件的 LTL 形式化规约(即提取系统待验证安全属性),解决了软件安全性验证过程中待验证属性难以提取的问题。该方法同样适用于对规模较大的 LTFT 进行分析和安全属性提取。


```

④<>(g_SciaRx_R eady==1)
⑤[]!(ifMsgError==true || ifMsgUseless==true)&&
(g_dRegister>MAX_REG_Length))
⑥<>((ifMsgError==true || ifMsgUseless==true)
&&(g_dRegister>MAX_REG_Length))) || -(g_dReg-
Length<MAX_REG_LENGTH)U(ifMsGError==true ∨ if-
MsGUseless==true))

```

将上述公式组合输入到 ESpin 中,对数据处理模块的 Promela 模型进行检验,在验证第 2 条公式时,可以得到反例“assertion violated-invalid array index”(即存在不合法数组索引),如图 4 所示。通过对源程序进行跟踪分析,发现数据处理模块的数据接收函数出现“数组越界错误”。这是由于数据接收函数对数组下标处理不当而造成的软件故障。在安全关键软件系统中的数组越界错误是非常危险的,这种情况不仅会导致数据的丢失,而且还可能会修改系统关键区域的代码,造成一系列无法预知的安全隐患,甚至是灾难性事件的发生。因此,对数组越界故障的验证是加强软件安全性的不可或缺的环节。

```

warning:-i and -I need -DREACH to work accurately
warning:never claim + accept labels requires -a flag to fully verif
warning:for p. o. reduction to be valid the never claim must be stutter
-invariant
t
<never claims generated from LTL formulae are stutter-invariant>
pan;1;assertion violated - invalid array index <at depth 196>

```

```

pan;wrote DataEx. pml. trail
pan;reducing search depth to 197
pan;wrote DataEx. pml. trail
pan;reducing search depth to 179

```

图 4 存在不合法数组索引

针对上述情形,只需将数据接收函数中多余的数组下标操作语句删除,即可解决该对软件安全性造成重大隐患的软件缺陷。

此外,在对第⑤条公式进行验证时,还可发现程序意外终止错误,在此不做详细分析。该问题的出现可追溯到数据处理模块中的输入缓冲区溢出,这是由于在软件设计时对缓冲区长度设置不当造成的。“缓冲区溢出”故障也是安全关键软件中普遍存在的安全隐患,可能会导致一系列的软件系统问题,需要引起足够的重视。在实际中,缓冲区的长度应当由领域专家对真实环境进行评估后慎重设定。

结束语 本文致力于使用线性时序逻辑和故障树分析法解决安全关键系统中具有时序特征的系统故障问题,总结如下:(1)本文提出了一种能够精确描述系统时序故障的形式化故障树——LTFT,并具体阐述了其构建与约简规则;(2)在 Semanderes 算法的基础上进行修改扩充,提出一种能够从 LTFT 中提取描述时序故障的安全属性的算法——“LTL—Semanderes 算法”,然后对使用该方法提取的安全属性进行建模与验证;(3)运用本文提出的方法对某飞机襟缝翼控制软件系统的数据处理模块故障进行案例分析,说明了本文方法的可行性和有效性。此外,本文所提方法对 LTFT 进行约简和安全属性的提取需要依靠人工进行,下一步研究将关注其自动化实现。

- [1] Peled D A. Software reliability methods[M]. Springer Science & Business Media, 2013
- [2] Lee W S, Grosh D, Tillman F A, et al. Fault Tree Analysis, Methods, and Applications-A Review[J]. IEEE Transactions on Reliability, 1985, 34(3): 194-203
- [3] 苏晓勤. FTA 故障树分析算法改进研究与实现[D]. 天津:河北工业大学, 2005
Su X Q. Research on Improving and Realizing Arithmetic of Fault Tree Analysis [D]. Tianjin: Hebei University of Technology, 2005
- [4] 龚海里. 故障树计算机辅助分析优化算法研究与应用[D]. 大连:大连理工大学, 2004
Gong H X. Research and Application of Computer Aided Analysis Optimization Algorithm in Fault Tree [D]. Dalian: Dalian University of Technology, 2004
- [5] 陈建军. 关于故障树分析中联合重要度的研究[D]. 大连:大连交通大学, 2009
Chen J J. Research of Joint Failure Importance in Fault Tree Analysis [D]. Dalian: Dalian Jiaotong University, 2009
- [6] Palshikar G K. Temporal fault trees[J]. Information and Software Technology, 2002, 44(3): 137-150
- [7] Čepin M, Mavko B. A dynamic fault tree[J]. Reliability Engineering & System Safety, 2002, 75(1): 83-91
- [8] Gluchowski P. Duration Calculus for Analysis of Fault Trees with Time Dependencies[C]//2nd International Conference on Dependability of Computer Systems, 2007: 107-114
- [9] 刘磊. 软件时序故障树建模与分析技术研究[D]. 长沙:国防科学技术大学, 2011
Liu L. Research on Modeling and Analysis Techniques for Software Temporal Fault Tree [D]. Changsha: National University of Defense Technology, 2011
- [10] Schäfer A. Combining real-time model-checking and fault tree analysis[M]//FME 2003: Formal Methods, 2003: 522-541
- [11] Koh K Y, Seong P H. SMV model-based safety analysis of software requirements[J]. Reliability Engineering & System Safety, 2009, 94(2): 320-331
- [12] Cha S, Yoo J. A safety-focused verification using software fault trees[J]. Future Generation Computer Systems, 2012, 28(8): 1272-1282
- [13] Pnueli A. The temporal logic of programs [C]// 18th Annual Symposium on IEEE, 1977: 46-57
- [14] 马琳, 黄志球, 徐丙凤, 等. 支持模型检测的故障树生成方法研究[J]. 计算机与数字工程, 2013, 40(2): 257-260
Ma L, Huang Z Q, Xu B F, et al. A Fault Tree Analysis Method Supporting Model Checking [J]. Computer & Digital Engineering, 2013, 40(2): 257-260
- [15] Platz O, Olsen J V. FAUNET: a program package for evaluation of fault trees and networks[J]. Research Establishment Risk Report, 1976, 348(41): 97-103
- [16] 吕威, 黄志球, 陈哲, 等. ESpin: 基于 SPIN 的 Eclipse 模型检测环境[J]. 计算机工程与应用, 2013, 49(7): 45-51
Lv W, Huang Z Q, Chen Z, et al. ESpin: SPIN-based Eclipse model checking environment. Computer Engineering and Applications [J]. Computer Engineering and Applications, 2013, 49(7): 45-51