文章编号:1674-8190(2023)04-177-12

面向航电软件的Safety_SysML一致性验证器设计与实现

尹伟[1],王辉[1],孙海英[2],丁郭欢[2],康介祥[1],刘静[2]
(1.中国航空无线电电子研究所,上海 200241)
(2.华东师范大学 软件工程学院,上海 200062)

摘 要: 民用飞机航空电子系统的高度综合化导致机载安全关键软件规模的成倍增加,带来航电系统软件需求的来源众多且不一致,航电系统软件各层次需求传递存在二义性等问题,因此,如何确保航电软件的一致性成为系统研发过程中亟待解决的核心问题之一。基于Safety_SysML状态机的语法,设计Safety_SysML一致性验证器,包括静态数据检测和动态数据检测;通过设计测试用例对核心算法及系统进行单元测试与集成测试,结合错误推断及边界,设计并执行功能测试,用于发现验证器存在的缺陷。结果表明:Safety_SysML一致性验证器能够有效识别航电系统软件中存在的二义性问题,对于提高航电软件的可靠性具有重要意义。

关键词: Safety_SysML状态机;一致性验证器;动态数据检测;功能测试
中图分类号: V243 **文献标识码:** A
DOI: 10.16615/j.cnki.1674-8190.2023.04.19

Design and implementation of Safety_SysML consistency verifier for avionics software

YIN Wei[1], WANG Hui[1], SUN Haiying[2], DING Guohuan[2], KANG Jiexiang[1], LIU Jing[2]
(1. China Aeronautical Radio Electronics Research Institute, Shanghai 200241, China)
(2. Software Engineering Institute, East China Normal University, Shanghai 200062, China)

Abstract: The high integration of the civil aircraft avionics systems will lead to an exponential rise in the size of airborne safety-critical software, and cause the numerous and inconsistent sources of its requirements, the transfer of requirements at each level of the avionics system software has duality and other problems. Therefore, how to ensure the consistency of the avionics software has become one of the core issues to be solved during the development of the system. On the basis of the syntax of Safety_SysML state machine, the Safety_SysML consistency verifier is designed, including static data detection and dynamic data detection. The test cases are designed for unit and integration test of the core algorithm and system. Based on the error inference and boundaries, the functional tests are designed and executed to find the defects in the verifier. The results show that the Safety_SysML consistency verifier can effectively identify the problem of duality in avionics system software, and is of significant importance for improving the reliability of the avionics software.

Key words: Safety_SysML state machine; consistency verifier; dynamic data detection; functional test

收稿日期: 2022-08-15; **修回日期:** 2023-01-14
基金项目: 工信部民机预研项目(MJ-2018-S-29)
通信作者: 尹伟, yinw008@avic.com
引用格式: 尹伟, 王辉, 孙海英, 等. 面向航电软件的Safety_SysML一致性验证器设计与实现[J]. 航空工程进展, 2023, 14(4): 177-188.
YIN Wei, WANG Hui, SUN Haiying, et al. Design and implementation of Safety_SysML consistency verifier for avionics software[J]. Advances in Aeronautical Science and Engineering, 2023, 14(4): 177-188. (in Chinese)

0 引 言

随着计算机科学在不同行业的广泛应用，软件系统的安全性越来越受到关注。在国防建设和国民经济建设的诸多领域中，如航空航天控制系统、轨道交通运行控制系统、核能核电安全控制系统、医疗设备控制系统、武器装备系统、医疗电子系统、汽车电子系统等，这类系统一旦出现不确定行为，会导致系统失效，从而严重威胁相关人员的生命安全和财产安全，造成不可预估的灾难性后果。近年来，随着各类安全关键系统功能的日益增加和需求的不断增多，传统的系统工程开发技术和安全性分析技术受到了很大的挑战。

基于模型的安全性分析（Model Based Safety Analysis，简称MBSA）是近年来出现的一类对安全关键系统建模并基于模型实现自动或半自动的安全分析及验证的技术方法。MBSA已成为系统建模和分析领域的一个热点研究问题。系统建模语言（Systems Modeling Language，简称SysML）是一种支持对系统进行规约、设计的图形化建模语言[1]，具有用于建模系统需求、行为、结构和参数的语义基础，目前在上述安全攸关领域有了广泛的应用，作为系统工程的标准建模语言，由于缺少严格的形式化定义语义，目前尚缺少有效的模型分析方法[2]。

针对航电软件的安全性需求的建模、验证和测试，国内外研究者从不同角度进行了深入研究，提出了许多理论和技术。B. S. Medikonda等[3]综合软件故障树分析（Software Fault Tree Analysis，简称SFTA）和软件失效模式与影响分析（Software Failure Modes and Effects Analysis，简称SFMEA），提出了一种新的方法，用于识别潜在的航电系统软件故障；H. G. Gürbüz等[4]提出了安全视角（Safety Perspective），用于确保安全性需求从架构的角度被正确地提出，可以用来辅助航电系统和软件架构师设计、分析和交流安全性需求；W. Elkholy等[5]采用条件承诺的计算树逻辑对自治代理之间的通信进行建模并跟踪其进程，解决了建模、验证和测试智能关键航空电子系统的问题；J. Barnat等[6]基于功能需求可以用时序逻辑表示的事实，提出了新的健全性检查技术，该技术可以自动检测航电系统中的缺陷并建议改进给定的需求；Wang H等[7]引入了一种新的安全分析方法，用于集成模块化航空电子系统，通过使用遍历算法，模型检查可以以有条理的数学方式搜索所有系统状态，使用此分析过程可以自动化并且减少人类经验的要求。这些安全性分析方法具有简单、工程难度低、分析结果生成快的优点，随着复杂系统的要求越来越高，功能越来越复杂，人类的认知能力很难分析出模型的所有可能存在的行为，且对分析人员的能力和经验要求较高，不同人员对于系统的理解可能存在误差，开发安全高效的模型安全性分析工具具有十分重要的意义[8]。

作为自动机理论当前发展的最高表现形式，行为状态机（Behavior State Machines，简称BSM）因其直观的图形化建模方式和丰富的模型构建元等优势而被安全关键系统研发组织广泛采用。但是，对象管理组织（OMG）为了降低学习成本而选择采用自然语言定义语义的方式，使得BSM语义说明中存在诸多不一致、二义性、不完整和不明确的地方，极易导致模型存在不易被发现的潜在错误，威胁着安全模型的正确性。

为了建立标准行为状态机的形式化语义，M. V. Cengarle等[9]将完整UML/P状态图转换为简化但语义上等价的UML/P状态图，并将其映射到一个被称为“系统模型”的对象系统的通用数学模型中来给出语义；Zhou Y等[10]研究了UML状态图的底层语义，结合实时嵌入式系统建模和分析规范MARTE中的时间元素，提出了分层时间自动机的形式化操作语义；蒋慧等[11]采用将状态映射到项代数的方式，再利用结构操作语义（SOS）规则给出满足组合性的状态图语义；H. Fecher等[12]利用更少的设计特性和一个精确的语法推导出核心状态机（Core State Machines），并赋予其一个正式的语义；Liu S等[13]提出了一个正式的操作语义，涵盖了状态机的所有特性，使用标记迁移系统作为语义模型，支持状态机之间的同步和异步通信。这些语义最大的优势是便于实现，但是对于航电软件来说，上述语义缺乏描述响应式系统的能力，并不能保证系统行为的确定性和安全关键软件的安全性[14-16]。

对安全关键系统进行高效准确的安全性分析具有十分重要的意义，模型一致性作为安全性的前提显得尤为重要[17-19]。为了支持安全关键系统的建模需求，本文在支持需求一致性验证的半形式化建模语言Safety_SysML状态机（Safety_Sys-

ML State Machines，简称S²MSM）的基础上，设计并构造基于S²MSM的一致性验证器，用于验证安全关键系统建模过程中的一致性，通过设计测试用例对基于S²MSM的一致性验证器进行验证，并以飞机主动控制技术中边界控制为例说明其应用场景及效果。

1　Safety_SysML状态机语法

S²MSM用于安全关键软件需求建模，支持需求一致性验证的半形式化建模语言[20]，采用具有确定语义定义并有严格语法的语言表达的规范风格。

1.1　状　态

S²MSM中的状态（State）与BSM的状态保持一致。状态是对可持续存在条件的抽象定义，当处于某个状态时，系统会响应消息池派发的事件并且根据接收的事件所触发的迁移转变为其他状态。S²MSM中状态分为简单状态、复合状态和子状态机状态。S²MSM的状态模型元素如图1所示，包含简单状态和复合状态。

初始状态是S²MSM的一个状态，表示初始伪状态通过迁移指向的状态。

结束状态是S²MSM的一个状态，表示所在区域行为的结束或整个状态机行为的结束。

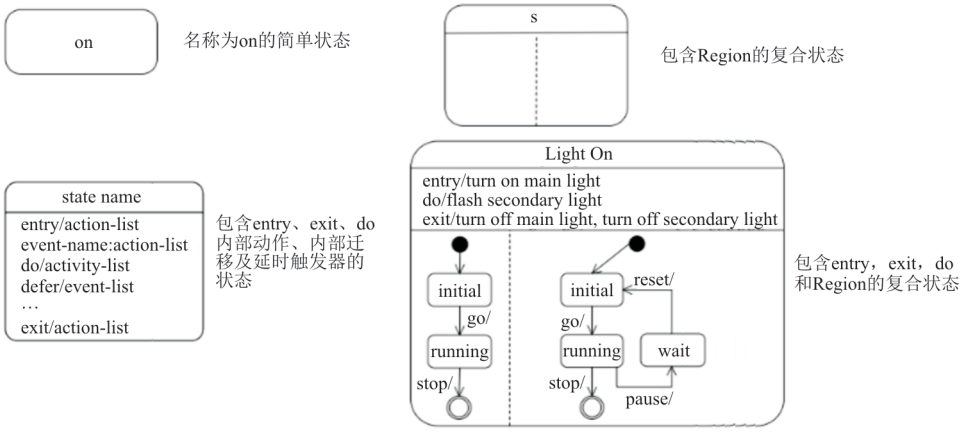

图1　状态建模元素说明

Fig. 1　Illustration of state modeling elements

状态内可以定义状态内行为有以下三项。

entry：如果定义，则表示由外部迁移进入该状态时执行的行为。

exit：如果定义，则表示退出该状态时执行的行为。

doActivity：如果定义，则表示状态处于活动态时执行的行为，在entry行为之后执行。

1.2　迁　移

S²MSM中的迁移（Transitions）如图2所示，迁移可以有标签，标签由迁移优先级、触发器、卫士条件和迁移输出行为构成，其语法结构为：priority：trigger［guard］/effect。其中，触发器、卫士条件和迁移输出行为都是可选项。不包含trigger的迁移，被称为完成迁移。不包含标签的迁移被称为空迁移（Empty Transition）。

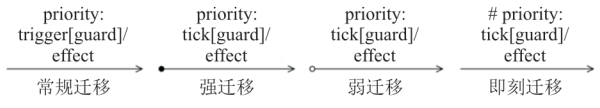

图2　迁移建模元素

Fig. 2　Trigger modeling elements

迁移优先级（priority）：S²MSM通过定义迁移优先级解决迁移发生冲突问题。迁移的优先级是一个整型变量，包含在迁移源状态的标签中，数字越大，优先级越低。数字0表示优先级最高的迁移；priority缺省值为－1，表示迁移没有定义优先级。

触发器（trigger）：迁移的触发器是一个特定点，表示某个事件发生可以触发迁移输出行为的执行。同步触发器是一种特殊的触发器，用于支持S²MSM的同步计算模式。

卫士条件（guard）：是一个布尔表达式。可以

是一个简单布尔,表达式也可以是由与、或、非连接的复合布尔表达式。

迁移输出行为(effect):迁移发生时执行的动作。

1.3 复合状态

复合状态(Composite States)是指包含其他S²MSM自动机的状态,如图1所示。每个被包含的状态机是一个子状态机,属于某个区域。一个复合状态可以有多个子状态机,而子状态机可以认为是其父状态的精化。

当一个复合状态包含若干子状态机时,需要考虑这些子状态机之间的组合方式及其语义。这些子状态机通过并行组合(Parallel Composition)进行协作。并行组合可以采用同步语义也可以采用异步语义。为保证确定性,在S²MSM中,每个子状态机必须包含一个初始状态和结束状态。

1.4 伪状态

S²MSM支持初始、junction、浅历史、深历史、进入点、退出点等伪状态。这些伪状态与BSM中相应的伪状态一致,消除了其中不确定和不完整的部分。S²MSM中的伪状态如图3所示。

标识	标志名	标识	标志名
●	初始伪状态	Ⓗ	浅历史伪状态
○	进入伪状态	Ⓗ*	深历史伪状态
⊗	退出伪状态		连接伪状态

图3 S²MSM中的伪状态

Fig. 3 Pseudo-state in S²MSM

junction伪状态用于建模不同迁移所具有的相同的触发器部分。在S²MSM中,如果有一个以上的junction伪状态的迁出迁移触发器为真,则根据优先级选择触发哪条输出迁移。为避免所有迁移触发条件都不满足而导致状态机无法运行,S²MSM的junction伪状态必须包含else迁移,否则,判定模型错误。

入口伪状态是状态机或复合状态的入口点。在状态机或复合状态的每个区域中,入口伪状态只有一条指向同层的某个节点的迁移。

出口伪状态是状态机或复合状态的出口。在由子机状态引用的复合状态或状态机的任何区域内输入退出点意味着该复合状态或子机状态的退出以及触发以该出口点为源状态的迁移。

浅历史伪状态是保存该伪状态直接所有者的最后活动的子状态本身。

深历史伪状态是保存该伪状态直接所有者的最后活动的状态配置。

1.5 强迁移和弱迁移

强迁移和弱迁移是S²MSM扩展的建模元,在同步模式下,处理状态内行为的两种不同方式。

1) 强迁移是当状态处于活动时,限制状态内行为执行的迁移。

2) 弱迁移是当状态处于活动时,等待状态内行为执行完成才发生的迁移。

强迁移和弱迁移的差别仅在于迁移发生时,是否允许状态内行为执行完毕。当迁移能够发生时,强迁移不会等待状态内行为执行完毕,而弱迁移则相反,会等待状态内行为执行完毕。

1.6 即刻迁移

即刻迁移是S²MSM扩展的建模元,是在同步模式下,跳过迁移源状态而直接发生的迁移,其目的在于加速迁移行为的发生,即迁移行为在当前同步触发器tick中就可发生,而不是要等到下一次同步触发器tick的发生。

2 一致性验证器的设计与实现

Safety_SysML一致性验证器主要包括数据读取与连接,静态数据检测,动态数据检测的设计实现。一致性验证器的整体架构如图4所示。

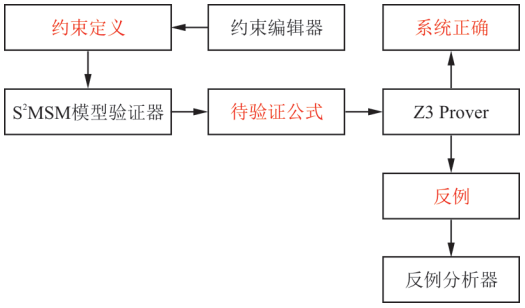

图4 一致性验证器整体架构

Fig. 4 Overall architecture of consistency verifier

一致性验证器首先读取约束编辑器定义的约束条件,模型验证器根据数据检测算法生成待验证公式;然后使用Z3验证器进行验证,并输出结果。

2.1　数据读取与处理

从 states 文件，trans 文件，variables 文件，constrains 文件中读取相应数据，分别储存在 state_list，trans_list，value_list，constrain_list 数据结构中。储存格式如下：

1）＜S，T，V，C＞

S 表示 S²MSM 中的状态集合，其中 T 表示 S²MSM 中的转移集合，其中 V 表示 S²MSM 中的变量集合，其中 C 表示 S²MSM 中的约束集合。

s∈S. s＝＜id，Rid，Rbid，st，act＞

其中 id 表示状态标识符（具有唯一性），Rid 表示该状态所属区域的 id（具有唯一性），Rbid 表示该状态在所属区块的块 id，st 表示状态类型如默认状态、初始状态、正常状态、终止状态，act 表示该状态的行为。

2）t∈T. t＝＜idt，idi，ide，tt，p，g，e＞

idt 表示转移标识符，idi 表示转移的起始状态或区域，ide 表示转移的终止状态或区域，tt 表示转移的类型如常规迁移、强迁移、弱迁移、即刻迁移等，p 表示转移的优先级，g 表示转移 guard，e 表示转移 effect。

3）v∈V. v＝＜idv，type，min，max，hv，nv＞

idv 表示变量标识符，type 表示变量类型，min 表示变量最小值，max 表示变量最大值，hv 表示该变量是否有明确的确定值，nv 表示该变量当前的确定值。

4）c∈C. c＝＜E＞

E 表示表达式 e 的集合，而 e 则是 string 类型的约束表达式。

数据读取与处理过程如图 5 所示，成功读入建模元数据后，按照 state 和 trans 的连接关系，将其一一连接。根据 state 的数据来构建相应的区域状态（region），region 内的 state 对应的 trans 与 region 连接，并确认 region 内的初始状态。

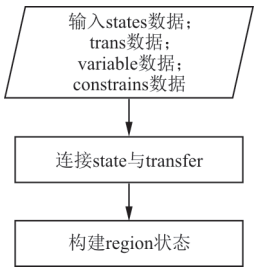

图 5　数据读取与处理

Fig. 5　Data reading and processing

2.2　静态数据检测

数据遍历与迁移流程如图 6 所示，遍历储存转移的列表 trans_list，检测是否存在 t1，t2∈trans_list：t1. idi＝＝t2. idi，即 t1，t2 的转移起始状态或区域一致，并且 t1. p＝＝t2. p∧t1. g∩t2. g≠NULL，即 t1，t2 的优先级一致，并且他们的 guard 集合相交，当转移起始状态或区域一致的两条 trans 优先级与触发器一致时，将他们的 guard 约束条件放入 Z3 求解器读取的数据结构中，通过求解器来求解，若存在解，则表示存在某一种特定的取值空间使得这两个转移均可激活，说明模型存在不确定性。

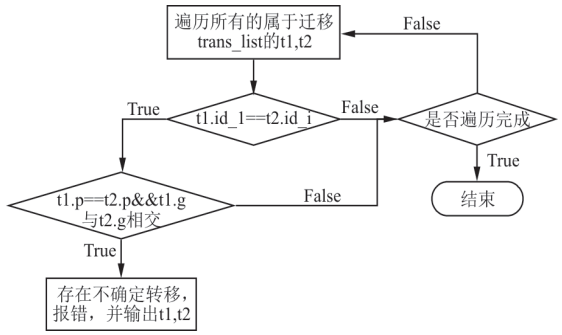

图 6　数据遍历与迁移

Fig. 6　Data ergodic and transfer

存在冲突的迁移流程如图 7 所示，到达状态 2 后，检测到其两条迁移的优先级、触发器一致，监护条件有交集，这两条转移均可激活，此时函数会指出其不确定性。

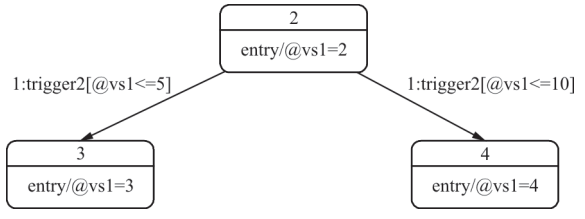

图 7　存在冲突的迁移

Fig. 7　Conflict trigger

2.3　动态数据检测

待静态检测通过后，初始状态的动态数据检测如图 8 所示，执行初始状态，检测该初始状态是否满足约束 constrains，若不满足则返回不安全状态报错。

满足约束 constrains 则对该状态的所有 trans 进行遍历，如图 9 所示，检测 trans 是否满足约束

constrains和已激活的guard条件，不满足则返回状态不可达报错，满足则将该trans到达的终止状态加入待执行列表。

然后遍历待执行列表，如图10所示，对列表中的状态及其trans执行以上检测，直至所有区块执行完毕，说明全图遍历完成。

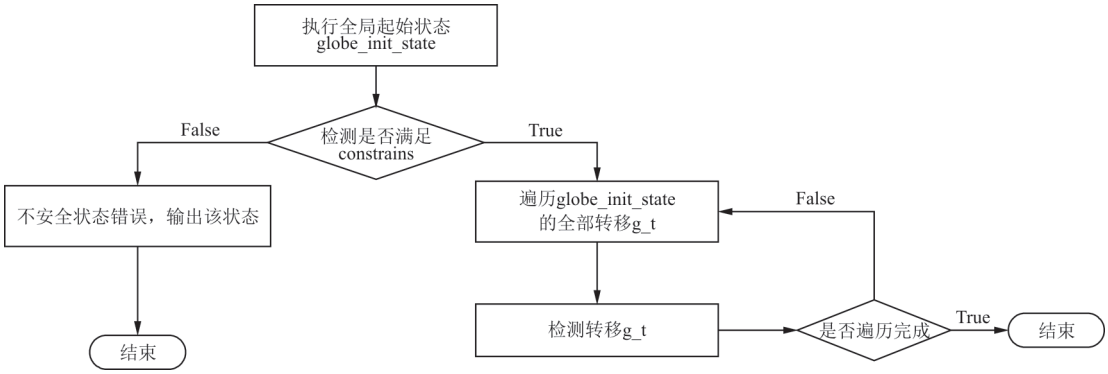

图8 初始状态的动态数据检测

Fig. 8 Dynamic data detection of initial state

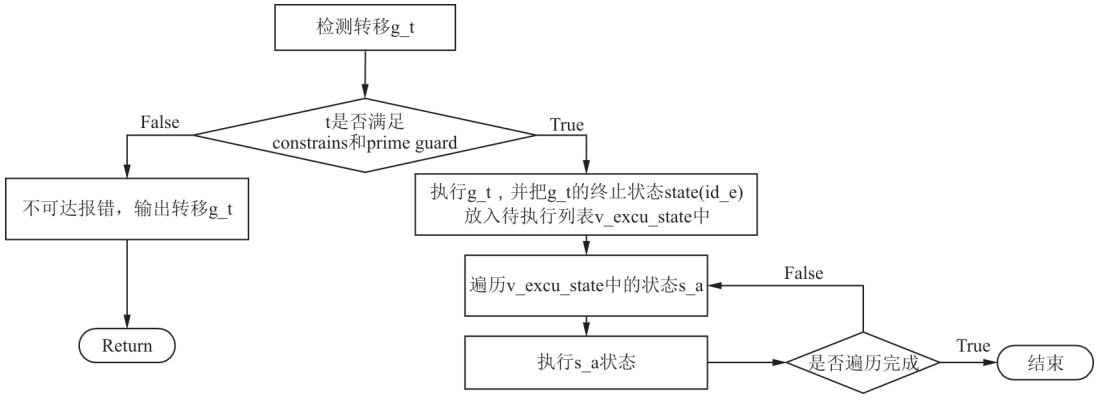

图9 转移状态的动态数据检测

Fig. 9 Dynamic data detection of transfer state

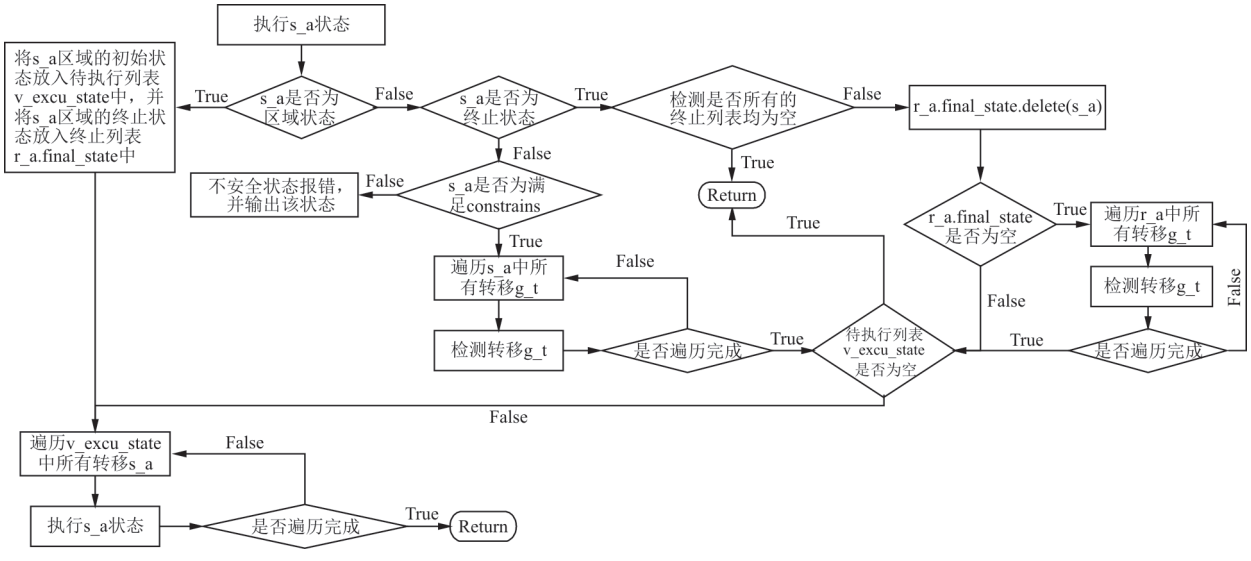

图10 动态数据检测的待执行列表

Fig. 10 Executing list of dynamic data detection

3　功能测试

对于状态机中的某个状态下的迁移，影响其迁移的因素有迁移上的优先级、触发器和卫士条件。通过控制变量的方法，分别设置同一状态下的两条迁移设置相同或不同的优先级、触发器和卫士条件来设计测试用例，在给定相同约束条件下，对状态机中变量的赋值不变，测试验证器在状态机到达相同状态时，对于不同迁移条件时的检测功能是否与预期一致。

3.1　简单状态模型

对于优先级、触发器、卫士条件各不相同的 S^2MSM 而言，根据该测试用例设计思想，需要保证所有迁移的影响因素互不相同，可构造 S^2MSM 如图 11 所示，将变量 a 定义为 int 型，规定其最小值为 −10，最大值为 10，初始值为 2，状态 State 的一条迁移的优先级为 0，触发器为 trigger1，卫士条件为 a=2，另一条迁移的优先级为 1，触发器为 trigger2，卫士条件为 a>2。根据所设计的模型，到达状态 State，由于终点 State2 的迁移优先级为 0，优先级高于终点为 State1 的迁移，故此模型应不存在迁移冲突。在测试环境中，执行一致性验证后，验证器提示该模型不存在迁移冲突，验证器实际运行结果与预期结果一致表明该测试用例通过。

无冲突的迁移流程如图 11 所示，对于存在相同优先级，但触发器、卫士条件各不相同的 S^2MSM 而言，根据该测试用例设计思想，需要保证模型中存在某一状态的迁移中存在相同优先级，但触发器、卫士条件各不相同的迁移，可将图 11 所示的 S^2MSM 中，状态 State 的一条迁移为 1 的迁移优先级改为 0，其他因素不变。

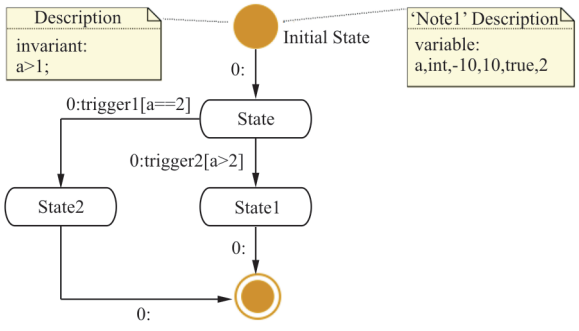

图 11　无冲突的迁移

Fig. 11　Conflict-free trigger

根据所设计的模型，到达状态 State，由于终点为 State2 的触发器与终点为 State1 的迁移触发器不一致，迁移顺序由触发器决定，故此模型不存在迁移冲突。

对于存在相同触发器，但优先级、卫士条件各不相同的 S^2MSM 而言，根据该测试用例设计思想，需要保证模型中存在某一状态的迁移中存在相同触发器，但优先级、卫士条件各不相同的迁移，可将图 11 所示的 S^2MSM 中，状态 State 的一条触发器为 trigger2 的迁移触发器改为 trigger1，其他因素不变。根据所设计的模型，到达状态 State，由于终点为 State2 的迁移的优先级为 0，优先级高于终点为 State1 的迁移，故此模型不存在迁移冲突。

对于存在相同的卫士条件，但优先级、触发器各不相同的 S^2MSM 而言，根据该测试用例设计思想，需要保证模型中存在某一状态的迁移中存在相同的卫士条件，但优先级、触发器各不相同的迁移，可将图 11 所示的 S^2MSM 中，状态 State 的一条卫士条件为 a>2 的迁移触发器改为“a==2”，其他因素不变。根据所设计的模型，到达状态 State，由于终点为 State2 的迁移的优先级为 0，优先级高于终点为 State1 的迁移，故此模型不存在迁移冲突。

对于简单状态的两条迁移的迁移优先级冲突引起不一致的场景而言，根据该测试用例设计思想，需要保证模型中存在某一状态的迁移中存在互相冲突的迁移且由相同优先级引起，可将图 11 所示的 S^2MSM 中，状态 State 的两条迁移优先级均改为 0，其余影响因素置为空。根据所设计的模型，到达状态 State，状态 State 的两条迁移存在相同优先级 0，此时对于同一输入可能存在不同输出，导致不一致，一致性验证器应指出出现冲突的 trans，终点为 State2 的迁移与终点为 State1 的迁移存在冲突。

对于简单状态的两条迁移的触发器冲突引起不一致的场景而言，根据该测试用例设计思想，需要保证模型中存在某一状态的迁移中存在互相冲突的迁移且由相同触发器引起，可将图 10 所示的 S^2MSM 中，状态 State 的两条迁移优先级均改为 0，触发器均改为 trigger1，其余影响因素置为空。根据所设计的模型，到达状态 State，状态 State 的两条迁移触发器相同且存在相同优先级 0，此时对于同一输入可能存在不同输出，导致不一致，一致性验证器应指出出现冲突的 trans，终点为 State2 的迁移与终点为 State1 的迁移存在冲突。

对于简单状态的两条迁移的卫士条件冲突引起不一致的场景而言，根据该测试用例设计思想，

需要保证模型中存在某一状态的迁移中存在互相冲突的迁移且由有交集的卫士条件引起，可将图11所示的S²MSM中，状态State的两条迁移优先级均改为0，卫士条件分别改为“$a<=5$、$a<3$”，其余影响因素置为空。根据所设计的模型，到达状态State，状态State的两条迁移的卫士条件存在交集且存在相同优先级0，此时对于同一输入可能存在不同输出，导致不一致，一致性验证器应指出出现冲突的trans，终点为State2的迁移与终点为State1的迁移存在冲突。

对以上测试用例分别进行测试，记录的测试结果如表1所示，用于测试一致性验证功能是否与预期一致。

表1 简单状态测试结果

Table 1 Simple state test results

简单状态迁移测试	一致性验证结果
优先级、触发器和卫士条件各不相同	没有迁移冲突
存在相同优先级，但触发器、卫士条件各不相同	没有迁移冲突
存在相同触发器，但优先级、卫士条件各不相同	没有迁移冲突
存在相同的卫士条件，但优先级、触发器各不相同	没有迁移冲突
简单状态的两条迁移的迁移优先级冲突引起的不一致	指出冲突的trans
简单状态的两条迁移的触发器冲突引起的不一致	指出冲突的trans
简单状态的两条迁移的卫士条件冲突引起的不一致	指出冲突的trans

3.2 复合状态模型

含有复合状态的S²MSM行为较仅有简单状态的S²MSM更为复杂，在实际建模过程中，复合状态也出现的更为频繁，基于以上出现的迁移冲突，对含有复合状态的S²MSM进行验证。

对于复合状态的两条迁移的迁移优先级冲突引起不一致的场景而言，根据该测试用例设计思想，需要保证模型中存在某一复合状态的迁移中存在互相冲突的迁移且由相同优先级引起，可将S²MSM中状态cs1的两条迁移优先级均改为0，其余影响因素置为空，如图12所示。根据所设计的模型，到达状态cs1，状态cs1的两条迁移存在相同优先级0，此时对于同一输入可能存在不同输出，导致不一致，一致性验证器应指出出现冲突的迁移，终点为cs2的迁移与终点为s3的迁移存在冲突。

对于复合状态的两条迁移的触发器冲突引起的不一致的场景而言，根据该测试用例设计思想，需要保证模型中存在某一复合状态的迁移中存在互相冲突的迁移且由相同触发器引起，S²MSM如图12所示，状态cs1的两条迁移优先级均改为0，触发器均改为trigger1，其余影响因素置为空。根据所设计的模型，到达状态cs1，状态cs1的两条迁移存在相同触发器且存在相同优先级0，此时对于同一输入可能存在不同输出，导致不一致，一致性验证器应指出出现冲突的迁移，终点为cs2的迁移与终点为s3的迁移存在冲突。

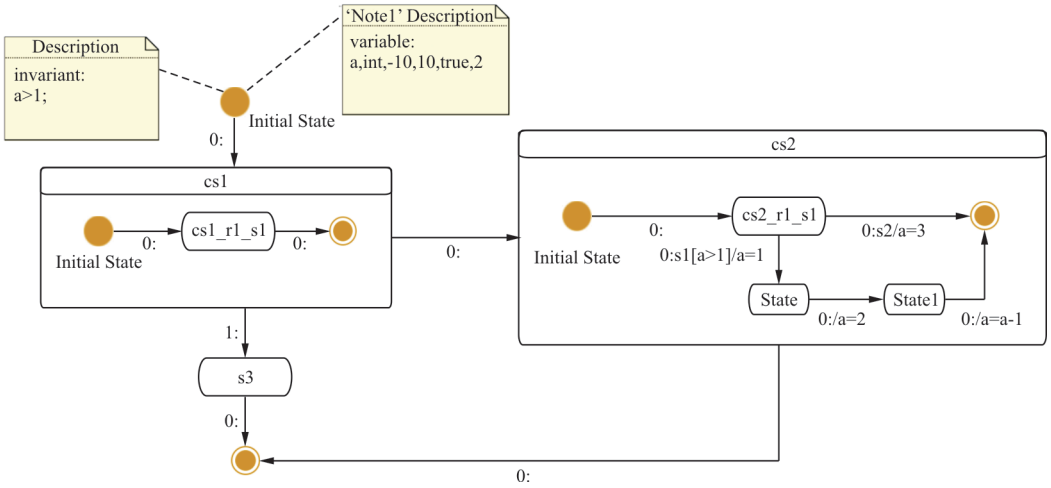

图12 含有复杂状态的模型

Fig. 12 Model with complex states

对于复合状态的两条迁移的卫士条件冲突引起不一致的场景而言,根据该测试用例设计思想,需要保证模型中存在某一复合状态的迁移中存在互相冲突的迁移且由有交集的卫士条件引起,可将图11所示的S²MSM中,状态State的两条迁移优先级均改为0,卫士条件分别改为“$a<=5$、$a<3$”,其余影响因素置为空。根据所设计的模型,到达状态cs1,状态cs1的两条迁移的卫士条件存在交集且存在相同优先级0,此时对于同一输入可能存在不同输出,导致不一致,一致性验证器应指出出现冲突的迁移,终点为cs2的迁移与终点为s3的迁移存在冲突。

对以上测试用例分别进行测试,记录的测试结果如表2所示,用于验证复杂状态下功能是否与预期一致。

表2　复合状态测试结果
Table 2　Composite state test results

简单状态迁移测试	一致性验证结果
复合状态的两条迁移的迁移优先级冲突引起的不一致	指出冲突的trans
复合状态的两条迁移的触发器冲突引起的不一致	指出冲突的trans
复合状态的两条迁移的卫士条件冲突引起的不一致	指出冲突的trans

3.3　典型建模错误

在实际建模过程中,随着模型构建复杂程度越来越高,可能会出现环路、迁移无法转移导致存在反复到达某个状态,或某个状态不可达的情形,本文对常见的导致不一致的情况进行测试。

对于存在环路的S²MSM引起不一致的场景而言,根据该测试用例设计思想,需要保证模型中存在环路,可构造S²MSM如图13所示,模型中变量a定义为int型,规定其最小值为-10,最大值为10,初始值为2,状态State1的两条迁移分别为“0:和1:”。根据所设计的模型,到达状态State1,状态State1的一条迁移指向State,此时存在环路State->State2->State1->State,导致不一致,一致性验证器应指出出现冲突的环路。

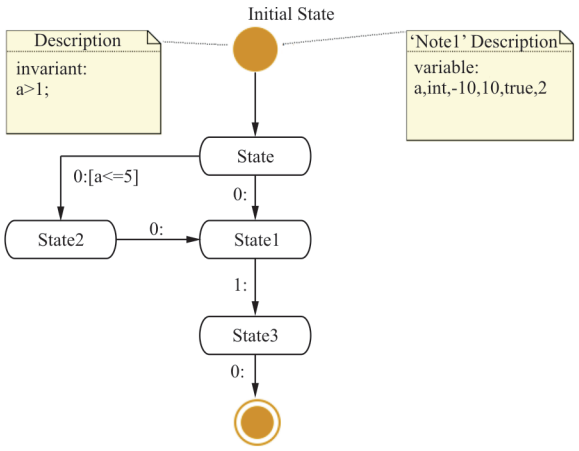

图13　存在环路的S²MSM引起不一致的模型
Fig. 13　Model of inconsistency caused by S²MSM in the loop

对于存在死锁的S²MSM引起状态不可达的场景而言,根据该测试用例设计思想,需要保证模型中存在某一状态不满足其所有迁移的卫士条件,可构造的S²MSM如图14所示,变量a定义为int型,规定其最小值为-10,最大值为10,初始值为2,状态S2的一条迁移的优先级为0,迁移触发器为sig2,卫士条件为“var1>12 || var1<-12”,另一条迁移的优先级为0,迁移触发器为sig1,卫士条件为“var1 < 6 && var1 >11”,迁移效果为“var1=var1+2”。根据所设计的模型,到达状态S2后,var1的值为2,不满足其两条迁移的卫士条件,从而停留在状态S2,造成死锁,验证器应指出不可达状态。

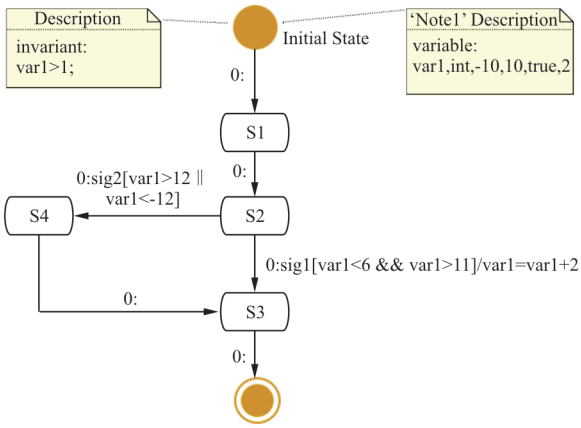

图14　存在死锁的S²MSM
Fig. 14　S²MSM with deadlock

4 案例应用

航电系统作为航空领域的重要组成，其适航问题是安全关键系统关注的重要问题之一。1992年，南方航空B737-3Y0型号客机在备降时，右发自动油门故障未能随动，导致左右发动机的推力不一致，最终飞机高度突然下降，在广西境内撞山造成粉碎性解体[21]。由此可以看出，保障软件开发满足适航标准的重要性。飞机的控制软件中包括了各种各样的参数，如高度参数，该参数的显示可以让飞行员将飞机控制在安全范围，一旦失效就会造成不可挽回的结果。其余的各类参数也会在各方面影响飞机的安全以及驾驶员的安全驾驶。因此，研究和构造可信的开发方法成为安全关键系统建模的重要步骤。

作为飞机主动控制技术中边界控制的一部分，飞机滚转角控制系统控制着飞机侧向机动的幅度，影响着飞机的飞行安全。边界控制的主要作用是限制飞机参数的范围，保证这些重要参数变化的稳定性。边界保护控制律的加入，保证了飞机飞行的稳定性，减少了人为因素导致的民航飞行事故。滚转角是边界控制的一个重要参数，展示了飞机相对于地面的姿态，飞行员可以通过相关数据了解飞机的飞行状态并控制飞机的飞行，实现安全驾驶。滚转角是指飞机与垂直面之间的夹角，如图15所示，该角度会影响飞机的安全以及乘客的舒适度，因此保证该角度在安全范围内，可以提升飞机飞行的质量和安全。

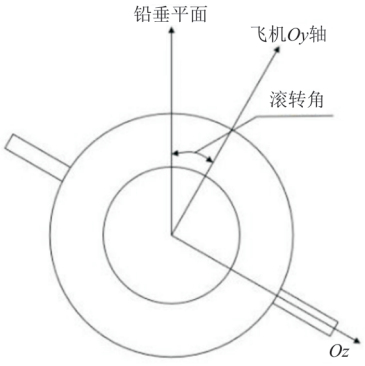

图15 飞机滚转角概述图

Fig. 15 Overview diagram of roll angle

飞机滚转角控制系统的组成部分以及每一部分对应的功能说明如表3所示，该系统主要分为飞机滚转角过大警告模块、滚转角当前模式检测模块和滚转角限幅模块。

表3 飞机滚转角控制系统组成部分功能说明

Table 3 Functional description of the components of the roll angle control system

简单状态迁移测试	一致性验证结果
飞机滚转角过大警告模块	当滚转角超过设定的滚转角警告门限值，则输出滚转角过大的警告信号
滚转角当前模式检测模块	用于对飞行器当前的滚转角状态进行判断
滚转角限幅模块	用于对操纵杆采集到的滚转角调整命令进行修改

在系统的设计中，飞机滚转角控制系统的三个主要组成模块通过传输信号和设置操作符来完成模块之间的连接，实现飞机滚转角的自动设定和调整。从安全的角度来看，民用飞机横向机动的范围很小，必须限制滚转角的范围。如果滚转角的角度过大，需要发出警告信号，以便及时调整滚转角的大小。

根据需求的分析结果，飞机滚转角控制系统的行为模型主要包括了滚转角限幅模块、飞机滚转角过大警告模块和滚转角当前模式检测模块，分别被建模为RollRateCalculate、RollRateWarning和RollRateMode状态，如图16中所示。

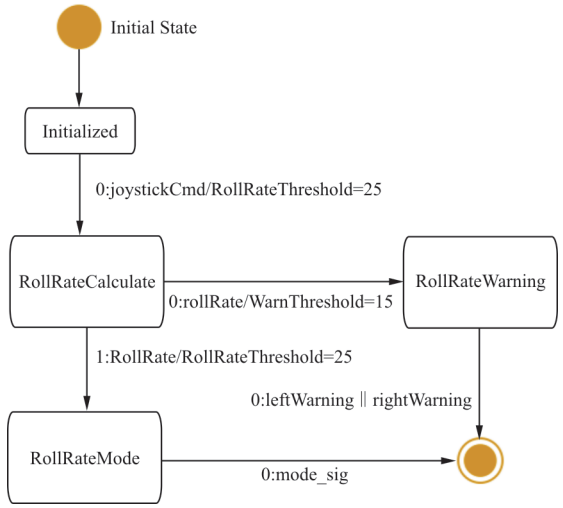

图16 飞机滚转角控制系统高层需求建模

Fig. 16 High-level requirements modeling of roll angle control system

模块之间的迁移关系可以解释为：

1）Initialized→RollRateCalculate：当系统启动后会处于Initialized状态，系统接收到操作杆对滚转角的调整命令，系统进入对调整命令进行修正程序。

2) RollRateCalculate→RollRateWarning:当系统处于RollRateCalculate状态,发出修正指令完成以及获取滚转角角度成功的信号后,则进入滚转角过大警告程序,根据飞机滚转角数据判断是否发出警告。

(3) RollRateCalculate→RollRateMode:当系统处于RollRateCalculate状态,接收到系统开关的状态以及获取滚转角角度成功的信号后,则进入滚转角模式检测程序。

(4) RollRateWarning→Final State:当系统处于RollRateWarning状态,发出相应的告警信号后,程序终止。

5) RollRateMode→Final State:当系统处于RollRateMode状态,显示相应的模式后,程序终止。

在系统行为执行的过程中,需要设定常量值RollRateThreshold和Warn-Threshold作为滚转角门限值,用于在系统执行过程中判定飞机当前滚转角是否超限。通过迁移中的effect属性对这些常量值进行赋值。

系统的可靠性主要包括模型的安全性、完整性、一致性和符合性四个方面,如表4所示,安全性是指系统满足Safety_SysML状态机定义的关于安全性的不变式,完整性是指模型中所有可能的执行路径可以涵盖Safety_SysML状态机定义的变量范围,一致性是指模型不存在迁移冲突,符合性是指变量的定义符合数据字典中对该变量的定义。

通过一致性验证器(基于Z3)分析了飞机滚转角控制系统的功能,验证结果表明模型符合上述性质。

表4　飞机滚转角控制系统验证结果

Table 4　Aircraft roll angle control system validation results

可靠性	验证结果
完整性验证	飞机滚转角控制系统总模型完整性验证结果,表明在模型中,所有可能的执行路径可以涵盖本文定义的变量的范围
安全性验证	飞机滚转角控制系统总模型安全性验证结果,对于每一个状态来说,由于操作杆的调整数值在[15,36]的范围,当调整的数值为25时左翼和右翼会同时报警,可以满足定义的不变式约束
一致性验证	飞机滚转角控制系统总模型一致性验证结果,表明模型对于同一状态在同一输入下,不存在迁移到多个不同状态的情况,或对于同一输入序列,不存在多条执行路径或者环路
符合性验证	飞机滚转角控制系统总模型符合性验证结果,表明变量的定义符合数据字典中对该变量的定义

5　结　论

1) Safety_SysML一致性验证器根据模型元数据的状态、迁移、变量和常数等进行数据的读取及处理,通过Z3证明器对S^2MSM模型的布尔逻辑进行求解,证明了系统的正确性。

2) 针对航电系统,通过飞机滚转角的例子,设计测试用例,用试验性地执行程序的方式测试其功能,验证了安全性、完整性、一致性和符合性,验证结果符合对案例的期望结果。

3) 面向安全关键系统领域,一致性验证器(基于Z3)能够有效避免在建模过程中产生的二义性,保证系统的安全可靠性。

程序对不同数据的处理过程是不同的,而且数据的取值范围又十分广泛,使用测试方法穷尽程序的各种可能处理过程以确保程序的正确性,几乎是不可能实现的。对于不同的处理过程和广泛的输入数据取值,在未来可以通过不断提高测试用例的代码覆盖率和通过自动化测试实现对大部分测试用例的遍历等方式来对验证器的正确性进行有益补充。

参 考 文 献

[1] JOSHI A, HEIMDAHL M P E, MILLER S P, et al. Model based safety analysis: NASA/CR-2006-213953 [R]. US: NASA, 2006.

[2] FRIEDENTHAL S, MOORE A, STEINER R. A practical guide to SysM[M]. Amsterdam: Elsevier Inc., 2011.

[3] MEDIKONDA B S, RAMAIAH S. Software safety analysis to identify critical software faults in software-controlled safety-critical systems [C] // 48th Annual Convention of Computer Society of India. Visakhapatnam: AIAA, 2013: 1-5.

[4] GÜRBÜZ H G, TEKINERDOGAN B, PALA N E. Safety perspective for supporting architectural design of safety-critical systems[M]. Anonymous Cham: Springer International Publishing, 2014.

[5] ELKHOLY W, EL-MENSHAWY M, BENTAHAR J,

et al. Model checking intelligent avionics systems for test cases generation using multi-agent systems[J]. Expert Systems with Applications, 2020, 156: 113458.

[6] BARNAT J, BAUCH P, BENEŠ N, et al. Analysing sanity of requirements for avionics systems[J]. Formal Aspects of Computing, 2016, 28(1): 45-63.

[7] WANG H, ZHAO T, REN F, et al. Integrated modular avionics system safety analysis based on model checking [C]// 2017 Annual Reliability and Maintainability Symposium. [S.l.]: IEEE, 2017: 1-6.

[8] 车程, 刘轶斐. 基于模型的安全性分析技术研究[J]. 航空工程进展, 2016, 7(3): 369-373.
CHE Cheng, LIU Yifei. Reasearch on model based safety analysis[J]. Advances in Aeronautical Science and Engineering, 2016, 7(3): 369-373. (in Chinese)

[9] CENGARLE M V, GRÖNNINGER H, RUMPE B. System model semantics of statecharts[R]. US: Eprint Arxiv., 2014.

[10] ZHOU Y, LUCIANO B, MATTEO R. Towards a formal semantics for UML/marte state machines based on hierarchical timed automata[J]. Journal of Computer Science & Technology, 2013, 28: 188-202.

[11] 蒋慧, 林东, 谢希仁. Uml 状态机的形式语义[J]. 软件学报, 2002, 13(12): 2244-2250.
JIANG Hui, LIN Dong, XIE Xiren, The formal semantics of UML state machine[J]. Journal of Software, 2002, 13(12): 2244-2250. (in Chinese)

[12] FECHER H, SCHÖNBORN J. UML 2.0 state machines: complete formal semantics via core state machine[C]// International Workshop on Parallel and Distributed Methods in Verification. US: Springer, 2006: 244-260.

[13] LIU S, LIU Y, ANDRÉ E, et al. A formal semantics for complete UML state machines with communications[C]// International Conference on Integrated Formal Methods. US: Springer, 2013: 331-346.

[14] ANCONA D, FRANCESCHINI L, FERRANDO A, et al. RML: theory and practice of a domain specific language for runtime verification[J]. Science of Computer Programming, 2021, 205: 102610.

[15] LIU J, ZHANG Y, HAN J, et al. Intelligent Hazard-risk prediction model for train control systems[J]. IEEE Transactions on Intelligent Transportation Systems, 2020, 44(11): 4693-4704.

[16] LIU J, TENGFEI L I, DING Z, et al. AADL+: a simulation-based methodology for cyber-physical systems [J]. Frontiers of Computer Science, 2019, 13(3): 516-538.

[17] YUAN Z, CHEN X, JING L, et al. Simplifying the formal verification of safety requirements in zone controllers through problem frames and constraint-based projection[J]. IEEE Transactions on Intelligent Transportation Systems, 2018, 42(1): 1-12.

[18] BALLANCE W A, VILKOMIR S A, JENKINS W. Effectiveness of pair-wise testing for software with boolean inputs// 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation. US: IEEE, 2012: 580-586.

[19] KUHN R D. Fault classes and error detection capability of specification-based testing[J]. ACM Transactions on Software Engineering and Methodology, 1999, 8(4): 411-424.

[20] 胡军, 吕佳润, 王立松, 等. 一个机载软件需求形式化建模与分析实例研究[J]. 软件学报, 2022, 33(5): 1652-1673.
HU Jun, LYU Jiarun, WANG Lisong, et al. Case study on formal modeling and analysis of airborne software requirements[J]. Journal of Software, 2022, 33(5): 1652-1673. (in Chinese)

[21] 百度. 92南航桂林空难[EB/OL]. [2022-08-15]. https://baike.baidu.com/item/92%E5%8D%97%E8%88%AA%E6%A1%82%E6%9E%97%E7%A9%BA%E9%9A%BE/14105725?structureClickId=14105725&structureId=76f8efb947afcdcf8efed64b&structureItemId=24921b1a87ea0ce07289d9df.
Baidu. Southern Airlines Guilin Air Crash in 1992[EB/OL]. [2022-08-15]. https://baike.baidu.com/item/92%E5%8D%97%E8%88%AA%E6%A1%82%E6%9E%97%E7%A9%BA%E9%9A%BE/14105725?structureClickId=14105725&structureId=76f8efb947afcdcf8efed64b&structureItemId=24921b1a87ea0ce07289d9df. (in Chinese)

作者简介:

尹 伟(1979—),男,博士,高级工程师。主要研究方向:航空电子,控制科学与工程,软件工程,适航安全等。

王 辉(1979—),女,硕士,研究员。主要研究方向:航空电子,软件工程,软件验证等。

孙海英(1986—),女,博士,讲师。主要研究方向:软件工程,形式化方法,软件测试等。

丁郭欢(1997—),男,硕士研究生。主要研究方向:形式化方法。

康介祥(1969—),男,学士,研究员。主要研究方向:航空电子,软件工程等。

刘 静(1965—),女,博士,教授。主要研究方向:可信智能软件,人工智能。

(编辑:丛艳娟)