第55卷第5期 南京航空航天大学学报 Vol. 55 No. 5
2023年10月 Journal of Nanjing University of Aeronautics & Astronautics Oct. 2023

DOI:10.16356/j.1005-2615.2023.05.003

自动飞行模式转换逻辑的形式化建模与验证

李俊安[1],胡 军[1,2],王立松[1,2],黄志球[1],蔡 鑫[1]

(1.南京航空航天大学计算机科学与技术学院,南京 211106;2.软件新技术与产业化协同创新中心,南京 210007)

摘要: 自动飞行控制系统(Automatic flight control system,AFCS)是现代飞机中重要的安全关键系统之一,飞行引导控制系统(Flight guidance control system,FGCS)是其重要的组成部分。FGCS中的飞行模式有数十种,模式转换逻辑十分复杂,在各个模式间转换时易出现模式混淆等问题,难以对其安全性和正确性进行验证。而利用计算机科学中的形式化方法,通过对安全关键系统进行形式化建模和验证,可以提高系统的正确性和安全性。本文以典型FGCS中的自动飞行模式转换逻辑作为研究对象,采用自主研制的软件工具ART(Avionics requirement tool)对其进行形式化建模与验证,并与Matlab/Simulink中的Design Verifier工具进行了验证能力和效率的对比分析。实例研究结果表明,采用形式化方法对FGCS的自动飞行模式转换逻辑进行建模、验证可行,所研制的软件平台具有更完善的验证能力和更好的验证效率。

关键词: 计算机软件与理论;飞行制导控制系统;基于模型的安全性分析;模型检测;安全关键系统

中图分类号: TP311.5 **文献标志码:** A **文章编号:** 1005-2615(2023)05-0768-12

Formal Modeling and Verification of Automatic Flight Mode Transition Logic

LI Junan[1], *HU Jun*[1,2], *WANG Lisong*[1,2], *HUANG Zhiqiu*[1], *CAI Xin*[1]

(1. College of Computer Science and Technology, Nanjing University of Aeronautics & Astronautics, Nanjing 211106, China;
2. Collaborative Innovation Center of Novel Software Technology and Industry, Nanjing 210007, China)

Abstract: Automatic flight control system (AFCS) is one of the most important safety-critical systems in modern aircraft, and flight guidance control system (FGCS) is an important part of it. There are dozens of flight modes in FGCS, and its mode conversion logic is very complex, which is prone to pattern confusion and other problems in the conversion of various modes, making it difficult to verify their safety and correctness. However, formal modeling and verification of safety-critical systems can improve the correctness and safety of the system by using formal methods in computer science. This paper takes the automatic flight mode conversion logic of typical FGCS as the research object, uses the software tool ART (Avionics requirement tool) independently developed by the author team to carry out formal modeling and verification, and compares the verification ability and efficiency with the Design Verifier tool in Matlab/Simulink. The case study results show that it is feasible to model and verify the automatic flight mode conversion logic of FGCS using formal methods. Meanwhile, our software platform has more complete verification capability and better verification efficiency.

Key words: computer software and theory; flight guidance control system; model-based safety analysis; model checking; safety-critical system

基金项目: 国家自然科学基金(U2241216)。

收稿日期: 2023-03-14;**修订日期:** 2023-05-11

通信作者: 胡军,男,副教授,E-mail:hujun@nuaa.edu.cn。

引用格式: 李俊安,胡军,王立松,等. 自动飞行模式转换逻辑的形式化建模与验证[J]. 南京航空航天大学学报,2023,55(5):768-779. LI Junan, HU Jun, WANG Lisong, et al. Formal modeling and verification of automatic flight mode transition logic[J]. Journal of Nanjing University of Aeronautics & Astronautics, 2023, 55(5):768-779.

现代飞机中的自动飞行控制系统（Automatic flight control system，AFCS）是保障飞机安全和高效的完成飞行任务的关键系统。自动飞行控制系统的飞行模式多达数十种，并且在各个飞行阶段对应多种工作模式，转换逻辑及方式十分复杂[1]。自动飞行模式转换逻辑是自动飞行控制系统重要功能组成部分，通常以机载关键软件的形式来承载[2]。飞行模式转换逻辑和飞行控制律算法共同组成自动飞行控制系统中的飞行导引系统（Flight guidance control system，FGCS）。飞行模式转换软件功能需要确保在自动飞行过程中，飞机在每一个飞行阶段都处于安全有效的运行模式控制中。尤其是在进近着陆等阶段，安全风险高且系统模式转换复杂，所以对飞行模式转换软件功能进行严格的安全性分析与验证很有必要。

基于模型的安全性分析（Model based safety analysis，MBSA）方法[3-5]近年来开始应用在航空、铁路等领域的安全关键系统[6]的安全性分析与验证。MBSA 的目标是首先构建一套系统模型，然后在系统开发过程中以系统模型为核心进行安全分析和验证，从而完成系统的安全评估过程。MBSA 通过在设计层面的安全分析，可以消除系统中的安全隐患及潜在风险导致的后果，提高整个系统的安全性[7]。航空无线电技术委员会（Radio Technical Commission for Aeronautics，RTCA）在最新机载软件适航认证标准 DO-178C[8]中给出了机载软件开发过程中各阶段软件制品所要达到的安全目标[9]，并强调机载软件的安全性保证是以高级需求和低级需求为核心来展开多层级的分析与验证工作。其补充标准 DO-331《基于模型的开发与验证指南》[10-11]以及 DO-333《DO-178C 和 DO-278A 的形式化方法补充》[12]中提供了基于模型的机载软件开发过程，以及其中应用形式化方法验证的相关指导。

近年来，基于计算机科学理论的形式化验证方法已被工业界逐步接受用于安全关键系统的验证。形式化验证[13]是使用准确、严格和有效的数学方法来验证关键系统行为的正确性和安全性。常见的形式化验证工具包括：SPIN[14]、SCADE[15]、MATLAB Simulink design verifier（SDV）[16]、UPPAAL[17]等。在 MBSA 框架中，形式化方法能够充分发挥基于模型的分析验证能力。文献[18，19]使用 MATLAB/Simulink 分别对飞机液压控制逻辑及自动飞行模式转换逻辑进行仿真，并对仿真结果进行了验证。前者通过仿真结果输出信号图进行验证，后者通过在 MATLAB 中搭建的 GUI 界面显示仿真结果进行验证，但两者都没有使用 Design Verifier 进行系统需求层面的验证，仅通过仿真结果进行人工验证。NuSMV 是基于符号模型检测技术的形式化验证工具，有许多研究将其他形式化语言模型转换为 NuSMV 能识别的 SMV 模型并进行验证。文献[20]提出了从 AltaRica3.0 模型到 NuSMV 模型的转换规则及算法，使得在 AltaRica 中不能验证的时序属性在 NuSMV 中得以验证。文献[21]研究了基于 NuSMV 的 AADL 模型形式化验证方法，提出由 AADL 到 NuSMV 的模型转化方法，并证明了转换后语义的正确性。文献[22]设计了从 SCR 模型到 NuSMV 模型的自动转换框架，并给出了相关规则的证明及有效性分析。也有部分研究直接面向安全关键系统的需求[23-24]，提出面向需求的形式化建模与验证方法，并针对真实的安全关键系统实例进行建模及验证。但上述研究均涉及到模型转换算法，会增加模型建模验证及验证工作的学习成本。

本文针对自动飞行控制系统对其模式逻辑功能要求进行形式化验证，具体而言，本文进一步完善了变量关系模型（Variable relationship model，VRM）理论模型，并给出一系列工程实用的安全属性构造模板。依托这两项理论研究在自主研制软件工具 ART（Avionics requirement tool）中增加了形式化模型验证的功能。同时提出了一种针对系统需求的形式化建模与验证的完整流程，并以自动飞行控制系统为实例，分别在 ART 与 MATLAB/Simulink 中对系统进行分层次的形式化建模与安全属性验证。最后对二者的验证能力及效率进行了对比分析。

1　自动飞行控制系统

自动飞行控制系统（Automatic flight control system，AFCS）可以接受飞行员手动设置、飞行管理系统（Flight management system，FMS）发送的指令以及相关传感器的输入信号等，按设定的姿态、航迹、空速对飞行进行自动控制，而 FGCS 是其中的重要组成部分，它包括自动驾驶（Autopilot，AP）、飞行指引（Flight director，FD）等功能。下面对 FGCS 及其相关联系统的功能和 FCGS 模式划分进行简要介绍。

1.1　FGCS 及其关联系统功能

1.1.1　FGCS

FGCS 通过大气数据系统、惯性基准系统、无线电高度表、飞行管理系统等提供的飞机姿态、位置与姿态偏差、控制指令等与期望中的飞机状态进行比较，并产生俯仰和滚转制导指令，从而提供 AP 和 FD 功能。AP 可以自动通过控制飞机的俯

仰、滚转等操纵面来改变其飞行航迹、高度等姿态，从而实现机动飞行；FD根据选择的FGCS工作模式在主飞行显示器(Primary flight display, PFD)上向飞行员展示飞行引导指令，同时通过自动计算向AP或飞行员手动飞行提供所需的操纵量，并显示在PFD上。

FGCS内部逻辑可分为模式逻辑和飞行控制率两部分，飞行控制率接受飞机当前和期望状态的信息，并生成制导指令。模式逻辑则确定任意给定系统运行时刻飞机的各方向上哪些模式处于预位状态、哪些模式处于激活状态。

1.1.2 飞行模式控制面板

飞行模式控制面板(Flight mode control panel, FMCP)是飞行员与自动飞行控制系统的主要人机交互设备，其位于飞机驾驶舱内的遮光罩上，其布局如图1所示。

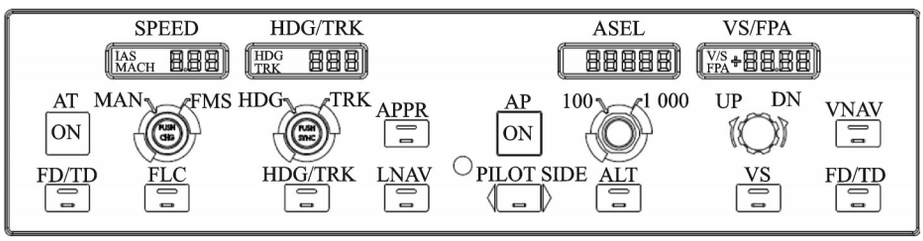

图1 飞行模式控制面板

Fig.1 Flight mode control panel

面板上的部件按功能可分为4类：FGCS管理功能、垂直引导、水平引导、推力/速度控制。FGCS管理功能按钮包括：用于接通/断开自动驾驶的AP、显示/取消飞行指引的FD和选择数据来源侧的PILOT SIDE；垂直、水平引导按钮/旋钮用于飞行员手动选择垂直、水平模式或输入相关数据；通过推力/速度控制按钮可以进行飞机推力和速度模式的选择。

1.2 FGCS工作模式划分

AC25.1329(Approval of flight guidance systems)咨询通告对FGCS工作模式进行了规范说明，其中“飞行模式及特性”章节分析了常见的飞行工作模式，并将飞机飞行工作模式规范为3种：横向模式、垂直模式和多轴模式。各类别飞行模式如图2所示。

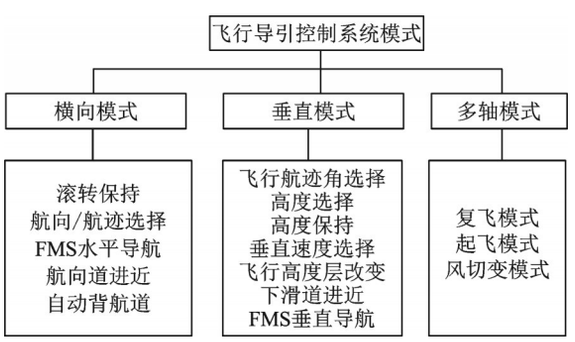

图2 FGCS工作模式

Fig.2 Flight modes of FGCS

横向模式用来控制飞机水平通道上的航迹和航向等；垂直模式用来控制包括起飞、巡航、进近着陆等阶段飞机的高度和俯仰通道上的姿态；多轴模式用来在飞机起飞、复飞或遭遇风切变等情况下综合控制飞机俯仰和水平通道上的方向与姿态控制。

2 验证工具

2.1 ART工具链平台

ART(Avionics requirement tool)，即航电需求建模与分析工具链平台，是一个自主设计实现的面向民机机载软件领域的自然语言需求形式化建模与分析的软件平台[25-26]。下面从该工具的理论模型、安全属性模板构建以及应用流程等方面进行简要介绍。

2.1.1 VRM模型

ART工具链平台所采用的理论模型为VRM。VRM模型以四变量模型[27]为基础，其保留了四变量模型中同时具备表格化和形式化语义的核心特征。另外，由于ART是针对于航电领域的工具链平台，所以VRM模型相较于四变量模型而言，会根据目前机载软件领域特征进行相应的调整。例如：根据机载软件对航空数据处理的特性，弱化了原四变量模型中监视变量和控制变量这两类系统级变量集合。具体来说，VRM模型是一种基于具有严格形式化定义语义的二维表格的需求结构。这种表格化的呈现方式不同于常见的利用数学符号和逻辑公式构建的形式化方法，可以被专业领域工程人员较为轻松地理解和接受。虽然表面上VRM模型是一张二维表格结构，但实际还是经过关系演算逻辑定义过的形式化模型。这里对VRM模型的规约元组做简要介绍，VRM规约六元组为：{SV, C, E, F, TS, VR}，各项具体定义如下：

SV：所有变量的集合，包括输入、输出变量、模

式集和中间变量，其中模式是指不同系统状态对应的等价类，模式集则是交集为空的模式的集合。

C：条件，用逻辑表达式来表示一个条件。

E：事件，通用表达式为 event(S)modifier(D)，其中 S 为前置事件，D 为守卫条件。event 表示事件类型共有 3 种：@T、@F 和@C。用_S 表示状态的前置取值，@T 事件语义为!_S & S、@F 事件语义为_S & !S、@C 事件语义为(!_S & S)||(_S &!S)。modifier 表示修饰操作符共有 3 种：WHEN、WHILE、WHERE。WHEN 表示条件 D 上一时刻为真、WHILE 表示当前时刻 D 为真、WHERE 表示上一状态和当前状态 D 皆为真。

F：表格函数，涵盖了模型中的模式表、条件表、状态转换表等表格内容。

TS：数据类型，包含系统内所有变量所属的数据类型。

VR：变量函数，用来表示状态变量的所有取值范围。

结合文献[28]中的简化版本的波音 737 自动驾驶模式控制面板系统给出 VRM 模型中 3 类表格模型的示例及其形式化语义。

表 1 是一个简单的条件表示例，表格中条件栏目下两列分别代表满足不同条件时，输出量的取值情况。直观语义为：若 tCASpresel 为 true 那么 cCASdiaplay 取值为 mCASdesired，否则 cCASdiaplay 取值为 mCASactual。

表 1　条件表示例

Table 1　Example of condition table

变量名(Variable)	条件(Condition)	
tCASpresel	True	False
cCASdiaplay	mCASdesired	mCASactual

表 1 所对应的形式语义可以表达为如下的逻辑公式：

$$\text{cCASdisplay} = F(\text{tCASpresel}) = \begin{cases} \text{mCASdesired} & \text{tCASpresel} = \text{true} \\ \text{mCASactusl} & \text{tCASpresel} = \text{false} \end{cases}$$

表 2 给出了一个事件表示例，该事件表的先导条件是当 *mcStatus* 取值为 ATTmode 或 FPAmode 时，基于两个新旧状态依赖关系集合{mALTsw, tALTpresel, tNear, mALTsw′, tALTpresel′, tNear′}和{mcStatus, mcStatus′}最后定义出中间变量 tARMED 的值。

表 2　事件表示例

Table 2　Example of event table

模式(Mode)	事件(Event)	
ATTmode,FPAmode (mcStatus)	@T(mALTsw=on) WHEN (tALTpresel AND NOT tNear)	@F(mcStatus=FPAmode)
tARMED	True	False

表 2 对应的形式语义可以表达为如下逻辑公式

$$\text{tARMED} = F(\text{mALTsw}, \text{tALTpresel}, \text{tNear}, \text{mALTsw}', \text{mcStatus}, \text{mcStatus}') = \begin{cases} \text{True} & (\text{mcStatus} = \text{ATTmode} \wedge \text{mALTsw}' = \text{on} \wedge \\ & \text{mALTsw} = \text{off} \wedge \text{tALTpresel} = \text{true} \wedge \text{tNear} = \text{false}) \vee \\ & (\text{mcStatus} = \text{FPAmode} \wedge \text{mALTsw}' = \text{true} \wedge \text{mALTsw} = \text{false} \wedge \\ & \text{tALTpresel} = \text{true} \wedge \text{tNear} = \text{false}) \\ \text{False} & (\text{mcStatus} = \text{true} \wedge \text{mALTsw}' = \text{false}) \end{cases}$$

表 3 给出了一个模式转换表的示例，该表定义了 mcStatus 模式随对应事件发生而转换到其他模式的变化特性。表 3 对应的形式语义可以表达为如下逻辑公式

表 3　模式转换表示例

Table 3　Example of mode transition table

源模式(Source mode)	事件(Event)	目的模式(Destination mode)
ALTmode	@T(mFPAsw=on)	FPAmode
ATTmode	@T(mALTsw=on) WHEN(tALTpresel AND tNear)	ALTmode
ATTmode	@T(mFPAsw=on) OR @T(mALTsw=on)WHEN (tALTpresel AND NOT tNear)	FPAmode
FPAmode	@T(mFPAsw=on) WHEN(tALTpresel AND tNear) OR @T(tNear) WHEN tARMED	ALTmode

$$\text{mcStatus}' = F\begin{pmatrix} \text{mcStatus}, \text{mFPAsw}, \text{mFPAsw}', \text{mALTsw}, \text{mALTsw}', \\ \text{tNear}, \text{tNear}', \text{tALTpresel}, \text{tARMED} \end{pmatrix} =$$

$$\begin{cases} \text{FPAmode} & \text{mcStatus}=\text{ALtmode}\wedge\text{mFPAsw}=\text{off}\wedge\text{mFPAsw}'=\text{on} \\ \text{ATTmode} & \text{mALTsw}=\text{off}\wedge\text{mALTsw}'=\text{on}\wedge\text{tALTpresel}=\text{true}\wedge\text{tNear}=\text{false} \\ \text{ATTmode} & (\text{mFPAsw}=\text{off}\wedge\text{mFPAsw}'=\text{on})\vee \\ & (\text{mALTsw}=\text{off}\wedge\text{mALTsw}'=\text{on}\wedge\text{tALTpresel}=\text{true}\wedge\text{tNear}=\text{false}) \\ \text{FPAmode} & \big(\text{mFPAsw}=\text{off}\wedge\text{mFPAsw}'=\text{on}\wedge\text{tALTpresel}=\text{true}\wedge\text{tNear}=\text{false}\big)\vee \\ & (\text{tNear}'=\text{true}\wedge\text{tNear}=\text{false}\wedge\text{tARMED}=\text{true}) \end{cases}$$

2.1.2 安全属性模板构造

ART 工具链中验证功能部分使用线性时序逻辑或计算树逻辑表达式进行安全属性描述。为了方便研究人员将自然语言描述的属性转换为对应的逻辑表达式,在 ART 中通过定义模式集合以及作用域集合,对它们进行组合后得出了多种系统属性描述模板。模型验证人员可以选择的合适模板将自然语言描述的属性中的语素映射为对应的模型变量,之后将变量填入模板中状态及事件位置,从而大大减轻了验证步骤的学习成本。下面将分别介绍作用域集合、模式集合和安全属性模板。

安全属性通常是描述一个系统在运行过程中的某个阶段需要满足的性质或根据某个条件而做出的反应。即每条属性都有一个自己所属的执行范围。本文定义了4种作用域,如图3所示,其中阴影部分表示执行区间。

全局(Global):包含整个系统运行过程范围;之前(BeforeP):包含给定状态或事件P之前的范围;之后(After):包含给定状态或事件P之后的范围;直到(AfterPUntilQ):包含从给定状态或事件P发生后到Q发生前的范围。

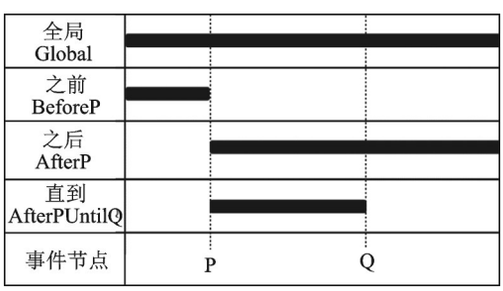

图3 作用域集合图示

Fig.3 Diagram of action scope set

模式的定义来源于属性的具体语义结构,在本文中定义了如下4种模式,前两者和后两者可以根据具体情况进行复合。同时,为简洁起见,描述中“指定状态/事件出现”指代的是“指定状态公式为真的状态”和“指定的若干事件的析取事件发生”:

不存在模式 Absence(P):用于描述一个系统的不存在行为,表示在作用域中状态/事件P不会出现;

存在模式 Existence(P):用于描述一个系统存在性行为,表示在作用域的某一部分中,状态/事件P必会出现;

偏序模式 Precedence(P,Q):用于描述一个系统的偏序关系的行为,表示在作用域中,状态/事件P必然出现在状态/事件Q出现之前;

因果模式 Response(P,Q):用于描述一个系统的因果关系行为,表示在作用域中,状态/事件P出现后总是会出现状态/事件Q。

通过前面介绍的作用域集合和模式集合定义,我们在ART中定义出了若干安全属性模板,下面介绍几个最常用模板:

(1) 全局因果模板:全局作用域与因果模式相结合,自然语言语义描述为:在整个系统运行过程中,若状态/事件P出现后必然出现状态/事件Q,逻辑表达式为:AG(P→Q);

(2) 全局存在因果模板:全局作用域与存在模式、因果模式相复合。自然语言描述为:在整个系统运行过程中存在某一时刻状态/事件P出现后状态/事件Q出现,逻辑表达式为:EF(P→Q);

(3) 直到存在因果模板:直到作用域与存在、因果模式相复合。自然语言描述为:在系统运行过程中,直到状态/事件R出现前,存在状态/事件P出现导致状态/事件Q出现的因果关系,逻辑表达式为:E(P→Q)U(R);

(4) 全局复合因果模板:全局作用域与因果模式复合。全局因果模板的扩充版本,自然语言描述为:在整个系统运行过程中,若出现状态/事件P,则下一时刻状态/事件Q发生且会导致状态/事件R发生。逻辑表达式为:AG(P→AX(Q→R))。

2.1.3 ART 工具链平台应用流程

ART 是一个面向 DO-178C/DO-333 机载软件适航标准的软件需求分析与验证工具平台。ART工具链可以接受自然语言描述的安全关键系统的设计需求,通过这些系统安全需求条目建立包含条件、事件和模式等核心要素。以此实现一套工程实用的形式化需求模型,即VRM模型的完整建模方法,ART 工具平台设计并应用的场景中的5个阶段如图4所示。

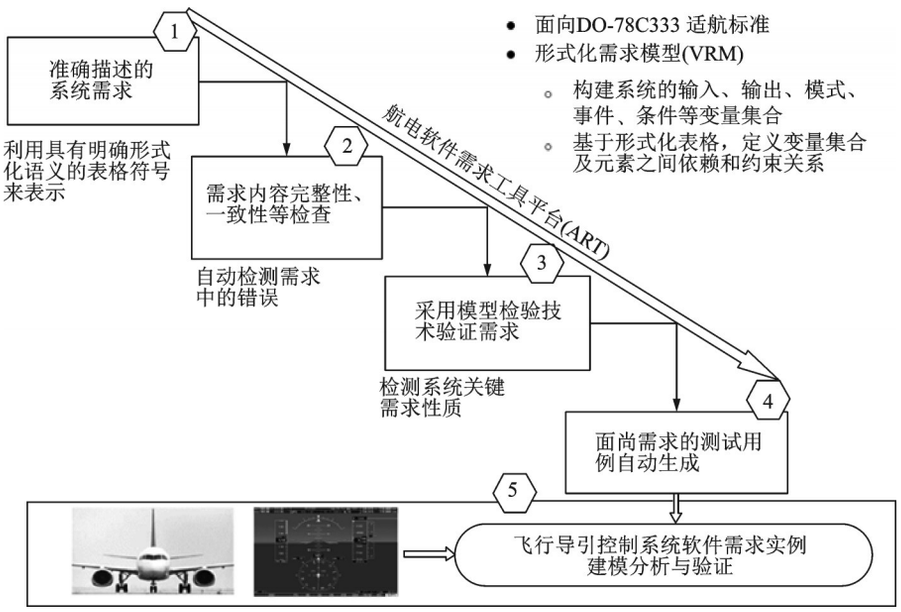

图 4　ART 应用流程

Fig.4　ART utilization proce

2.2　MATLAB/Simulink/Statesflow

Matlab 由 MathWorks 公司开发和发行，是一款用于数值计算的软件。Simulink 是 MATLAB 的一种辅助设计工具，常用于用于设计、模拟航空电子系统[29]。这些系统在 Matlab 中被建模为信号和状态变量之间基于时间关系的一系列框图。一个 Simulink 模块可以由一个或多个其他 Simulink 模块组成，因此可以构建分层框图。Stateflow 支持创建状态转移模型，这是有限状态机表示法的一种变体[30]。Stateflow 模型可以集成到 Simulink 中，数据可以在 Simulink 和 Stateflow 模型之间交互。Design Verifier 是 Simulink 中的一个扩展工具，采用内置的形式化方法识别模型中隐藏的设计错误，检测模型中导致整数溢出、数组访问越界和除以零的块；其同时具备形式化验证的能力来验证系统是否符合安全性需求，但仅提供黑盒验证功能，其内置引擎相关资料公开甚少，相关技术并不可控。

3　面向需求的形式化建模与验证

3.1　建模与验证流程框架

本文将分别使用 ART 工具链平台与 Simulink design verifier(SDV)图形化建模验证工具对系统需求进行建模验证。基于 MBSA 方法给出的一套整体流程如图 5 所示。

整体流程主要分为两部分，前半部分的内容是根据系统需求进行建模。首先，对自然语言描述的系统设计层面的需求进行细化和分层。其次，根据层次化结果，按照系统每一层次的需求转化为形式化模型。后半部分是将系统中的设计功能和安全属性转换成两种验证工具都能识别的属性输入。由于 ART 工具内置了部分领域的领域概念库，在一定程度上简化了系统功能和需求的转换。只需在工具中选择对应的模板，然后填写对应的属性即可将安全属性转化为逻辑表达式。而对于 SDV 而言，属性验证模块包含在系统的 Simulink 模型中，两者密切相关。因此，在建模之前需要进行预处

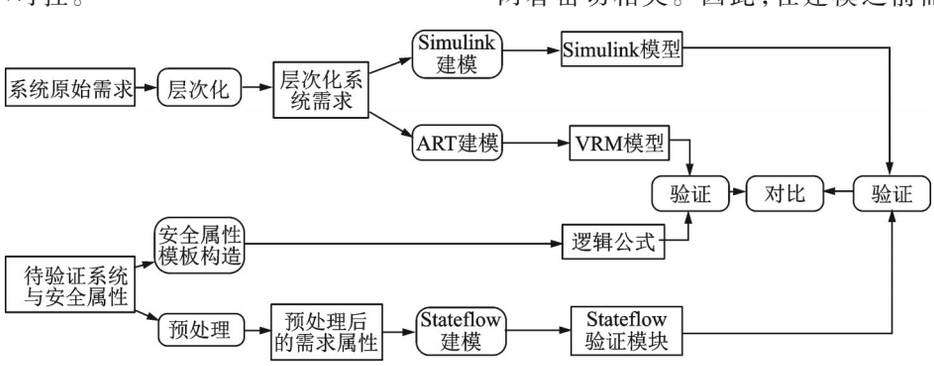

图 5　面向系统需求的建模与验证框架

Fig.5　System requirements-oriented modeling and verification framework

理。对于一些特殊的动作,需要在验证模块中编写若干新的子模块。

3.2 建模与验证流程框架中的关键问题

本节将对上一节提出的流程图中的关键问题进行展开介绍,并以FGCS中的典型飞行模式转换功能需求进行说明。

3.2.1 层次化建模

通常,安全关键型复杂系统的内部结构、设计要求和功能要求都非常复杂。这使得整个系统的架构十分庞大。如果不按照某种方式对系统进行层次化结构划分,同时进行建模,会大大增加建模工作量,建模结果也会变得模糊不清,对后期的验证工作也非常不利,很可能会出现状态空间爆炸等问题。因此,有必要对安全关键复杂系统进行分层建模。

根据FGCS系统中各模式与相关组件的关联性,将系统分为4个层次,如表4所示。

表4 FGCS层次划分

Table 4 FGCS hierarchy

层级	水平模式	垂直模式	边缘组件
第1层	ROL HDG/TRK	FPA VS	FD、AP FMCP
第2层	—	ALT ASEL	PF、ADS Navigation
第3层	LNAV	FLC	—
第4层	BC LOC	GS VNAV	Independent operation

3.2.2 系统状态机分类

FGCS中的状态机共可以分为3类,最简单的一类模式(非预位模式)仅包含两种状态:清除和选中,前者表示该模式没有被选中,后者则表示该模式被选中;其次,某些模式需要满足某些条件时激活(预位模式),此类条件则需要在选中状态中添加两个子状态:预位和激活,预位表示该模式处于待命中,当满足激活条件时,将立即由预位状态转换为激活状态。一些模式根据其功能需求,还需要将激活状态进一步区分为捕获和追踪两种子状态(捕获/追踪模式)。这种模式的状态机如图6所示。

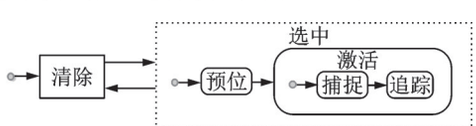

图6 捕获/跟踪模式

Fig.6 Capture/track mode

捕获/追踪模式一旦处于激活状态,便会自动进入捕获子状态,通过操纵飞机将判断目标与导航源或参考对象是否一致来捕获目标。一旦两者一致,模式就会转换到跟踪子状态,在该子状态下将使得飞机保持在目标上。

3.2.3 ART系统及属性建模

使用ART工具建模首先需要将系统原始需求条目导入到ART工具中,这些原始需求一般由自然语言描述,所以需要领域需求建模人员为建模的目标系统创建相应且合适的专业领域概念库。具体而言,建模人员需要从原始需求中提取关键变量、常量及其数据类型、模式集、专有名词。变量根据其使用位置和功能的不同分为输入变量、中间变量和输出变量3种类型。每个变量在定义时都需要指定变量名、初始值、变量类型和取值范围等。定义模式集时,除了指定名称和描述外,还需要给出模式表和模式转换表。前者描述了模式集中包含的所有模式,后者指定了模式集中每个模式的转换规则。每个模式集的内容不能重复。

在定义完专业领域概念库后就要完成对原始需求的规范化操作,ART工具中提供了4种模板,这里给出通用条件、通用事件这两种常用的模板:

通用条件:"当满足<条件>,<系统/设备>应能够<功能><对象>:<条件>"。

通用事件:"当发生<事件>,<系统/设备>应能够<功能><对象>:<事件>"。

根据待建模系统不同种类的原始需求,选取合适的模板进行规范化。规范化完成之后就可以使用ART工具内置算法自动生成VRM需求模型。

VRM模型生成完成后,可以选择使用ART工具的模型自动分析功能,此功能主要检查3个部分:模型的基本规范、完整性规范以及一致性规范。利用此功能可以确保上一步生成的模型中不存在语法错误、语义错误和需求不完整等错误。当得到正确无误的VRM模型后就可以使用工具自动生成验证引擎可以识别的模型文件,ART工具链平台中使用的是NuSMV验证引擎,所以VRM工具会对应生成SMV模型文件。

建模工作完成后,需要将待验证的需求属性转换为验证引擎可以识别的语言,ART工具中使用的NuSMV引擎对应的语言为线性时序逻辑(Linear temporal logic, LTL)或计算树逻辑(Computing tree logic, CTL)表达式。利用ART内置的安全属性模板功能可以将自然语言描述的待验证需求属性转化为LTL或CTL表达式。

3.2.4 SDV系统及属性建模

为了与所自主研制的工具相对比,结合Simulink、Stateflow的功能特点与FGCS系统的实际需求,构建如图7所示的Simulink建模架构图。

框架左侧的输入包括FGCS系统中FMCP面

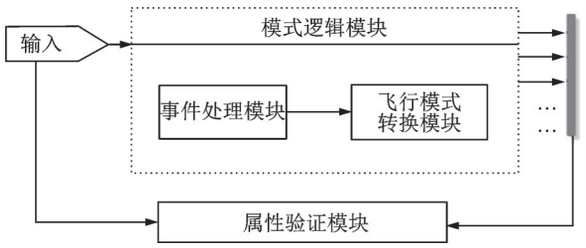

图 7　FGCS 系统 Simulink 建模设计框架

Fig.7　SDV modeling design framework of FGCS system

板上的各个控制按钮、预位条件捕捉信号、超速信号等。这些输入信号将首先逐个传入逻辑转换模块，模式逻辑模块中包含事件处理和逻辑转换两个子模块。事件处理模块在传入的事件和条件之间建立优先级，并确保当多个事件或条件同时发生时，仅将较高优先级的事件和条件输出到逻辑转换模块。逻辑转换模块是模式逻辑的核心，包含 FD、模式通知、垂直模式和水平模式 4 个状态机组件。FD、PFD 状态机确定飞行指引、模式通知是否显示在 PFD 上。垂直和水平模式状态机进一步分解为 FGCS 中各个横向和垂直模式的状态机。输入信号经由模式逻辑模块处理后变为输出变量传入验证模块。验证模块中包含若干以 Stateflow 建模形式给出的待验证属性。待验证的属性通常会由自然语言预处理得到含有系统内变量及逻辑运算符的符号语言，再将符号语言对应到验证模块中，调用输入、输出变量及逻辑运算符，最终得到安全属性的 Stateflow 模型。

Simulink 内置的逻辑运算符可以满足大多数属性的描述需求，但对 FGCS 而言，需要额外创建若干逻辑模块。例如，FMCP 面板上有若干模式按钮，验证属性中常见“当某模式处于清除状态时按下按钮”等涉及到前后相邻两个时间步的操作。传统的逻辑运算符并不能满足上述操作的需求，所以本文定义了下降沿和上升沿两个模块，如图 8 所示。下降沿模块的作用是捕获某信号的下降沿，如图 8(a)所示。模块的输入信号首先经过 DELAY 延时模块得到上一时刻的输入信号值，其次将此信号值与取反的输入值相与，最后将相与结果输出。若当前时刻输入型号值为 False 且前一时刻信号值为 True，模块输出信号为 True，此时输入信号的变化正好满足下降沿信号的需求。此模块可以良好的适配如“当某模式处于激活状态下按下该模式按钮”等属性描述。上升沿模块的原理与之类似，其功能则是捕获某信号的上升沿，内部结构如图 8(b)所示。

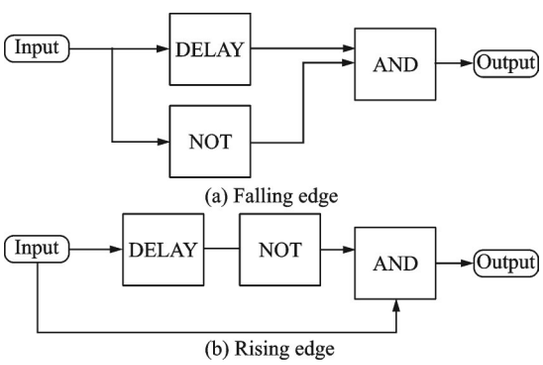

图 8　下降沿和上升沿模块的内部结构

Fig.8　Internals of falling edge and rising edge

4　FGCS 实例建模与安全属性验证

本文选择了某飞机上 AFCS 中的 FGCS，并根据第 3 节中介绍的流程，在两个工具中对此 FGCS 的前 3 层进行分层建模，并验证其相关的安全属性。

4.1　实例系统建模

飞行引导系统的前 3 层包括 5 个垂直模式和 3 个横向模式。表 5 给出了前 3 层中 8 种飞行模式的描述。

表 5　8 种飞行模式的描述

Table 5　Descriptions of the eight flight modes

模式	描述
ROL	水平方向的基本模式，将飞机保持在固定的倾斜角度
HDG/TRK	捕获并保持 FMCP 上的航向
LNAV	捕获并跟踪用于航路导航和非精密进近的横向引导
FPA	垂直方向的基本模式，将飞机保持在固定的俯仰角度
VS	将飞机保持在垂直速度参考值
ALT	获取并跟踪设置的高度参考
ALTSEL	捕获并跟踪预选的海拔高度
FLC	获取跟踪指示或马赫空速，并爬升或下降到预选高度

根据第 3 节介绍的建模方法，分别在两种工具中进行建模。由 ART 工具生成的 FGCS 系统 SMV 模型文件的一部分如图 9 所示，它所展示的是 ALT 模式选择模块。SDV 中前 3 层飞行模式转换模块如图 10 所示。

```
MODULE  Select_ALT
        (m_When_ALT_Button_Pressed_Seen,
         m_When_ALTSEL_Target_Altitude_Changed_Seen,
         m_Is_ALTSEL_Track, m_Modes)
VAR
    result: boolean ;
    ASSIGN
    init(result) := FALSE;
    next(result) :=
    case
        next(m_When_ALT_Button_Pressed_Seen.result)&(next(m_Modes.Modes)=On)
            : TRUE;
        next(m_When_ALTSEL_Target_Altitude_Changed_Seen.result)
            &(m_Is_ALTSEL_Track.result)&(next(m_Modes.Modes)=On)
            : TRUE;
        TRUE: result;
    esac;
```

图 9　FGCS 系统 XMV 模型

Fig.9　SMV model of FGCS

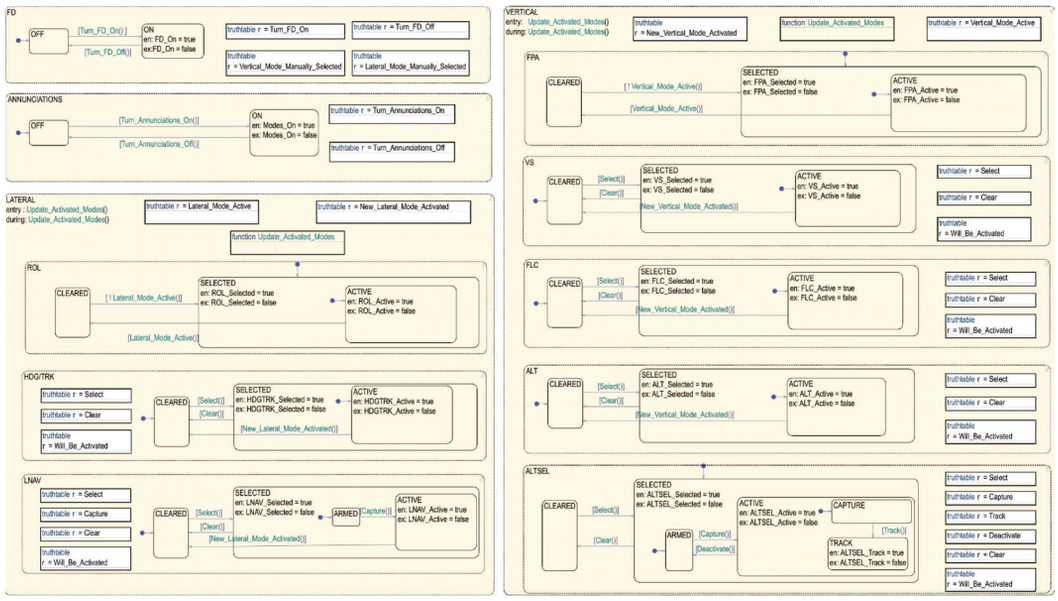

图 10 FGCS 系统前 3 层 Simulink 模型

Fig.10 The first three levels of Simulink model for FGCS

4.2 系统安全属性正向验证对比

本节从 FGCS 系统前 3 层中逐层选取 10 条安全属性，共计 30 条系统安全属性进行验证。

对于 ART，依次在工具中选取适当的安全属性模板对 30 条安全属性进行转化。例如，某一安全属性描述为“如果在 HDG 清除状态下按下 HDG 开关，并且没有按下更高优先级的按钮，则选中 HDG 模式”。选择工具中的全局复合因果模板。提取出属性描述中的语素，将他们与模型中的变量相对应，最后填入模板相应的位置，最后得到的计算树逻辑表达式为：AG((m_HDG.HDG=Cleared)→AX((m_When_HDG_Button_Pressed_Seen. result&m_No_Higher_Event_Than_HDG_Button_Pressed.result)→(m_HDG.HDG=Selected)))。30 条安全属性依次转换完成后，分别在模型的前 3 层中依次验证相应的属性。ART 内置的 NuSMV 引擎默认不计算可达状态数，而直接面向待验证的安全属性，仅对其中涉及到的状态变量等进行遍历验证，所有属性的验证结果都是正确的。

对于 SDV，需要对所有安全属性进行预处理，之后借助 Simulink 中的内置模块和自定义模块根据语义对安全属性依次建模，所有属性的验证结果都是正确的。因为 SDV 在验证前需要验证缓存的模型表示和确定模块是否符合 SDV 这两步预处理操作，其次再使用其内置的 Polyspace 引擎计算变量之间的关系，最后使用内置的 prover 引擎遍历模型可达状态。

表 6 总结了两种工具在不同层次的系统下的验证时间和内存，表中 SDV 工具验证时间部分的括号内分别表示的是预处理和验证时间。

表 6 两种工具验证的时间和内存

Table 6 Time and memory for verification of the two tools

系统层级	第 1 层		前 2 层		前 3 层	
参数	时间/s	内存/KB	时间/s	内存/KB	时间/s	内存/KB
ART	0.688	4.7×10^3	1.3	4.6×10^4	2.43	8.3×10^4
SDV	46(30+16)	1.86×10^6	69(37+32)	2.04×10^6	103(64+39)	2.32×10^6

4.3 系统错误需求属性验证对比

两种验证工具都具有生成反例的功能，本节选取上一节中的一个正确属性，将其修改为不正确的属性：“当 AP 断开且 FD 开关未按下时，FD 将关闭。”然后分别在两个工具中验证该条属性。

4.3.1 ART 验证

所选择的错误属性需要进行从自然语言到系统变量的映射。在 FGCS 系统的 VRM 模型中，当变量 m_When_AP_Disconnect_Button_Pressed_Seen.result 的值为 True 时，它表示 AP 断开连接。When_FD_Button_Pressed 表示了 FD 按钮被按下的动作。当变量 m_When_FD_Button_Pressed.result 的值为 True 时，表示 FD 按钮被按下。但是这一单一变量并不能清楚地表示 FD 从 True 到 False 的变化，所以需要借助 m_When_Turn_FD_On.result 来表示 FD 是否处于打开状态。通过分析这两个变量的值便可以表示 FD 变量变化的情况。例如，当 m_When_FD_Button_Pressed. result 和 m_When_Turn_FD_On.result 变量均为 True 时，则表示按下 FD 开关后 FD 被打开。

所选择的错误属性转换后得到的CTL逻辑表达式为：AG((m_When_Turn_FD_On.result = TRUE & m_When_AP_Disconnect_Button_Pressed_Seen.result = TRUE & (!(m_When_FD_Button_Pressed.result=TRUE)))→AX(! m_When_Turn_FD_On.result=TRUE))。

将上述公式导入ART工具进行验证，验证结果显示此属性是错误的，并给出了反例路径。反例路径共3个时间步，State 1.1表示初始状态，会列出所有变量的初始值，从State 1.2之后每个状态中展示出的则是在此状态出现变化的变量值。反之，未出现的变量的值直接继承上一状态的值。为了方便对比，将反例结果中关键变量值提取出来，总结如表7所示。

从表中可以看出验证引擎给出了一条否定所挑选的错误属性的正确路径。在State 1.2状态AP_Disconnect_Button（AP断开按钮）被按下，m_When_Disengage_AP.result也变为True，表示此时刻AP已经处于断开状态。同时，此时刻m_When_Turn_FD_On.result值为True，FD_Botton值为UNPress。表示FD处于打开状态，且并没有按下FD按钮。State 1.3中，AP继续处于断开状态，FD也处于打开的状态。此时，FD_Button值变为True，表示在此时刻按下FD按钮。显然，下一状态FD会被关闭。从State 1.2到State 1.3的操作正好符合所选需求属性："在断开AP时FD保持打开，除非手动按下FD按钮"。

表7　反例路径中部分变量值

Table 7　Some variable values in the counterexample

变量名	State 1.1	State 1.2	State1.3
AP_Disconnect_Button	UNPress	Press	Press
m_When_Disengage_AP.result	False	True	True
m_When_Turn_FD_On.result	False	True	True
FD_Button	UNPress	UNPress	Press

4.3.2　SDV验证

使用SDV对上述属性进行验证，结果表明FGCS系统模型不满足该属性。之后分析工具所生成的反例，SDV生成的默认反例只包含所有输入变量，所以还需要使用Debug Using Slicer对所有变量的状态进行单步调试，从而进一步分析错误来源。在T=0.304时刻，属性值为False，模块状态如图11所示。因为所选取的验证属性涉及到几个变量前后值的变化，所以记录由True变为False的时刻及其前面的两个时刻各变量状态如表8所示，其中最后1行为验证结果值。

表8　反例中三时刻各变量状态

Table 8　State of each variable at three steps

变量名	T=0.288	T=0.296	T=0.304
IS_AP_ENGAGED	True	True	False
FALLING(Is_AP_Engaged)	—	False	True
FD_SWITCH	True	True	True
RISING(FD_Switch)	—	False	False
Pre_FD_ON	—	True	True
FD_ON	True	True	True
FD_Stays_On_When_AP_Disengaged	True	True	False

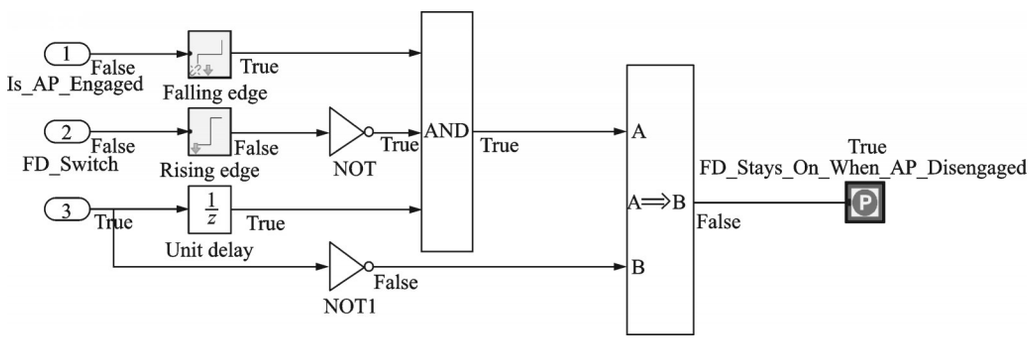

图11　T=0.304时刻错误属性模块状态

Fig.11　Status of incorrect property module at T=0.304

对于T=0.296时刻，前一时刻与当前时刻IS_AP_ENGAGED（自动驾驶接通）、FD_SWITCH（飞行指引按钮）的值均未发生改变，所以与两者相关的下降沿、上升沿变量的值均为False。T=0.288时刻FD_ON（飞行指引打开）值为True，所以T=0.296时刻Pre_FD_ON（前一时刻飞行指引打开）值也为True，最终的结果为True；同样，对于T=0.304时刻，FALLING（Is_AP_ENGAGED）、RISING（FD_Switch）和Pre_FD_ON为False，FD_ON的值为False，因此最终结果为False。经过逐步分析，可以发现错误出现在FD_On信号的NOT符号处。去掉这个符号后，所表达的意思与原来的正确性质一致。

4.4　含有"存在"语义的属性验证

假设FGCS第1层系统需要判断在运行时存在某一时刻HDG被激活。由于ART工具中使用CTL验证语言除了"全局（A）"还具有"存在（E）"这一路径量词，再配合上时态算子"有时（F）"，则该属性条件可直接表示为：EF(m_HDG.HDG=Selected)。在相应的FGCS模型中的验证结果也是正确

的,而在SDV中这种含有“存在”语义的属性无法准确表示,如图12所示,假设有两条相同的系统运行路径,在3状态HDG被激活,ART工具验证引擎对图12(a)路径从1开始遍历验证各状态,到3状态检测到HDG处于Selected状态,则断言此属性是正确的。而对于SDV,验证前后两个状态间的属性是有效的,如图12(b)路径需要给出2状态发生某种变化导致3状态HDG被激活的相关描述才能进行准确建模,否则建模表述的语义会变成“所有时刻HDG都被激活”,从而导致验证错误,所以SDV的形式化验证功能有一定的局限性。

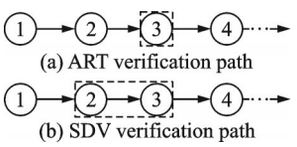

图12 两条验证路径

Fig.12 Verification paths in two tools

综上所述,可以看出两种工具都能够对安全关键系统及其属性进行建模,并具有准确验证属性的能力。但是从实验数据可以看出,对于FGCS系统前3层的验证,SDV验证耗时分别是ART工具验证耗时的67倍、53倍和42倍;SDV验证占用内存分别是ART工具验证占用内存的390倍、44倍和28倍,可以看出ART的验证效率是远好于SDV,依此可以推断出随着系统规模的增长,并发状态量也随之增加,SDV更易产生阻碍验证的状态空间爆炸的问题。从反例生成的角度来看,SDV生成的反例只包含所有输入变量。如果想知道其他变量的情况,需要手动逐步调试。但是,ART在验证引擎和内部反例结果处理算法的帮助下,输出的反例较为简洁。反例中的每个时间步中只包含该时间步中变化的所有变量,但不仅限于输入或输出变量。同时,一些安全属性在SDV中无法准确表示,则说明SDV不能对系统需求进行全覆盖验证,从而不能找出系统需求层级上的所有错误与漏洞,这也限制了它的功能。表9给出二者部分特性的比较。

表9 两种工具特性比较

Table 9 Feature comparison of two tools

特性	ART	SDV
系统建模及验证	支持	支持
验证内存占用	低	高
验证时间	短	长
反例信息完整度	完整	不完整
安全性质表达能力	强	弱

5 结 论

基于模型的安全分析方法,本文介绍了自主研发的ART工具所依托的理论模型的形式化定义及语义,针对于形式化验证功能,定义出一系列安全属性构造模板。同时,提出一套针对系统需求的形式化建模与验证流程,并详细介绍了分层建模和建模方法中的关键问题。选择某机型自动飞行控制系统中的飞行引导控制系统,在ART工具中对该系统的前3层依次建模。建模完成后,分别对不同层级的系统需求属性进行了验证。使用了MATLAB-SDV工具作为对照,在其中对同样的系统进行建模与验证。两种工具的验证结果一致且正确。对于相同规模的模型,可以得出SDV的验证效率低于ART工具的结论。同时,选择了一条具有代表性的错误属性,分别使用两个工具进行验证并得出反例。从生成的结果来看,ART工具给出的反例是完整且精确的,而SDV给出的反例中含有冗余信息且对比生成的反例不完整。最后分析了含有“存在”语义属性的验证问题。ART工具中使用的CTL语言可以准确地描述此类属性,而SDV中无法对此类准确建模。

本文的工作表明,对自动飞行系统这类复杂安全关键领域而言,自主研制的ART工具比商业的SDV工具具有更好的验证能力,在未来工作中,我们将提高ART工具的用户使用接口的设计,并在更大规模的国产机型系统中进行应用。

参考文献:

[1] 杨玉蕾. 民机自动飞行系统工作模式研究[D]. 南京:南京航空航天大学,2012.
YANG Yulei. Research on modes of automatic flight system of civil aircraft[D]. Nanjing: Nanjing University of Aeronautics and Astronautics,2012.

[2] 王飞. 大型客机飞行导引控制系统工作模式仿真研究[D]. 天津:中国民航大学, 2020.
WANG Fei. Simulation research on operating mode of flight guidance control system for large passenger Aircraft[D]. Tianjin: Civil Aviation University of China , 2020.

[3] JOSHI A, HEIMDAHL M P E, MILLER S P, et al. Model-based safety analysis: NASA/CR-2006-213953 [R]. USA:Langley Research Center,2006.

[4] JOSHI A, MILLER S P, WHALEN M, et al. A proposal for model-based safety analysis[C]//Proceedings of Digital Avionics Systems Conference. Washington, Piscataway: IEEE, 2005:2-13.

[5] JOSHI A, HEIMDAHL M P E. Model-based safety analysis of simulink models using SCADE design verifier[C]//Proceedings of International Conference on Computer Safety, Reliability, and Security. Berlin, Heidelberg:Springer, 2005, 3688: 122-135.

[6] NABI F, YONG J, TAO X, et al. Concepts of safety critical systems unification approach & security assurance process[J]. Journal of Information Security, 2020,11:292-303.

[7] SANGO M, VALLÉE F, VIÉ A C, et al. MBSE and MBSA with Capella and safety architect tools [C]//Proceedings of the Seventh International Conference on Complex Systems Design & Management. Berlin, Heidelberg: Springer, 2017: 239-239.

[8] Radio Technical Commission for Aeronautics: DO-178 [S]. USA: RTCA Press, 2011.

[9] 徐丙凤，黄志球，胡军，等. 面向适航认证的模型驱动机载软件构件的安全性验证[J]. 航空学报，2012，33(5)：796-808.
XU Binfeng, HUANG Zhiqiu, HU Jun, et al. Model-driven safety dependence verification for component-based airborne software supporting airworthiness certification[J]. Acta Aeronautica et Astronautica Sinica, 2012, 33(5): 796-808.

[10] Radio Technical Commission for Aeronautics. DO-331 model-based development and verification supplement to DO-178C and DO-278A[M]. [S. l.]: RTCA Press, 2011.

[11] HUI J. Research on RTCA/DO-331 standard[J]. Civil Aircraft Design & Research, 2018, 130 (3): 124-129.

[12] COFER D D, MILLER S P. DO-333 certification case studies[C]//Proceedings of NASA Formal Methods Symposium. Berlin, Heidelberg: Springer, 2014.

[13] KHAN W K, MUHAMMAD N, SYED R K, et al. Formal verification of hardware components in critical systems[J]. Wireless Communications and Mobile Computing, 2020, 2020(8): 1-15.

[14] HOLZMANN G J. The SPIN model checker: Primer and reference manual[M]. Reading: Addison-Wesley, 2004.

[15] CAMUS J L, DION B. Efficient development of airborne software with Scade suite[J]. Esterel Technologies, 2003, 62, 1-2.

[16] DABNEY J B, HARMAN T L. Mastering simulink [M]. Upper Saddle River: Pearson/Prentice Hall, 2004.

[17] BENGTSSON J, LARSEN K G, LARSSON F, et al. UPPAAL—A tool suite for automatic verification of real-time systems[J]. Proceedings of the DIMACS/SYCON Workshop on Hybrid Systems III: Verification and Control: Verification and Control, 1996, 12: 232-243.

[18] 辛东华，李晶. 基于Stateflow的民机液压控制逻辑仿真与验证[J]. 民用飞机设计与研究，2019(2)：47-52.
XIN Donghua, LI Jing. Simulation and verification of civil aircraft hydraulic control logic based on stateflow [J]. Civil Aircraft Design & Research, 2019(2): 47-52.

[19] 周超，邵慧，刘文渊，等. 基于Stateflow的自动飞行模式转换逻辑研究[J]. 民用飞机设计与研究，2019(4)：86-91.
ZHOU Chao, SHAO Hui, LIU Wenyuan, et al. Mode transition logic of automatic flight control system based on stateflow [J]. Civil Aircraft Design & Research, 2019(4): 86-91.

[20] 陈朔，胡军，唐红英，等. 一种AltaRica3.0模型到NuSMV模型的转换方法[J]. 计算机科学，2020，47(12)：73-86.
CHEN Shuo, HU Jun, TANG Hongying, et al. Transformation method for AltaRica3.0 model to NuSMV model [J]. Computer Science, 2020, 47(12): 73-86.

[21] 刘畅，蒋永平，马春燕，等. 基于NuSMV的AADL模型形式化验证技术[J]. 航空学报，2022，43(3)：451-466.
LIU Chang, JIANG Yongpin, MA Chunyan, et al. Formal verification of AADL models by NuSMV [J]. Acta Aeronautica et Astronautica Sinica, 2022, 43 (1): 451-466.

[22] 张漾，胡军，王立松，等. 一种面向SCR需求模型的形式化验证方法研究[J]. 小型微型计算机系统，2022，43(1)：193-202.
ZHANG Yang, HU Jun, WANG Lisong, et al. Formal verification method for SCR requirement model [J]. Journal of Chinese Computer Systems, 2022, 43 (1): 193-202.

[23] 胡军，张维珺，李宛倩. 面向需求的安全关键系统形式化建模与验证方法研究[J]. 计算机工程与科学，2019，41(8)：1426-1433.
HU Jun, ZHANG Weijun, LI Wanqian. A requirement oriented formal modeling and verification method for safety critical systems, 2019, 41(8): 1426-1433.

[24] ZHANG W, HU J, LI W, et al. Case study of formal modeling analysis for safety-critical system requirements [J]. Journal of Frontiers of Computer Science and Technology, 2019, 13(8): 1295-1306.

[25] 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. A case study on natural language requirement based Formal modelling and analysis for an airborne display control software system [J]. Journal of Software, 2022, 33 (5): 1652-1673.

[26] HU J, HU J, WANG W, et al. Constructing formal specification models from domain specific natural language requirements [C]//Proceedings of 2020 6th International Symposium on System and Software Reliability (ISSSR). Washington, Piscataway: IEEE, 2020: 52-60.

[27] PATCAS L M, LAWFORD M, MAIBAUM T. From system requirements to software requirements in the four-variable model[J]. Automated Verification of Critical Systems, 2014, 66: 1-15.

[28] BHARADWAJ R, HEITMEYER C. Applying the SCR requirements specification method to practical systems: A case study[J]. Goddard Space Flight Center, 1997, 2: 1-16.

[29] DAVID H. Statecharts: A visual formalism for complex systems[J]. Science of Computer Programming, 1987, 8: 231-274.

[30] BRYANT R E. Graph-based algorithms for Boolean function manipulation[J]. IEEE Transaction Computers, 1986, 35(8): 677-691.

（编辑：刘彦东）