

基于 SysML 的复杂机电系统设计模型形式化扩展与验证

曹悦¹⁾, 刘玉生^{1)*}, 赵建军²⁾, 叶晓平³⁾, 周书华⁴⁾

¹⁾(浙江大学 CAD&CG 国家重点实验室 杭州 310058)

²⁾(华中科技大学机械工程学院 武汉 430074)

³⁾(丽水学院工学院 丽水 324000)

⁴⁾(浙江经济职业技术学院汽车技术学院 杭州 310018)

(ysliu@cad.zju.edu.cn)

摘要: 形式化系统验证是保证系统设计正确性的一种重要手段. 如何针对复杂机电系统物理与软件相融合的特征, 对系统设计的动态特征进行验证, 是系统验证研究领域亟待解决的问题. 针对这一问题, 对系统工程标准建模语言 SysML 进行扩展, 提出了一套形式化系统模型验证方法. 首先, 以计算树逻辑和基于流的功能表示为形式化基础, 形成基于 SysML 的系统功能建模方法; 然后, 以混合自动机为基础, 建立基于 SysML 的系统行为建模方法; 最后, 针对物理与软件子系统的不同动态特征, 借助 NuSMV 模型校验器, 以层次化方式实现系统模型的自动验证. 以移动机器人系统为例, 展示了复杂机电系统设计模型的自动验证过程.

关键词: 机电一体化系统; 基于模型的系统工程; 系统设计; 系统验证; 模型校验

中图分类号: TP391.41 DOI: 10.3724/SP.J.1089.2019.17911

Formal Extension and Verification of System Design Models for Complex Mechatronic Systems Based on SysML

Cao Yue¹⁾, Liu Yusheng^{1)*}, Zhao Jianjun²⁾, Ye Xiaoping³⁾, and Zhou Shuhua⁴⁾

¹⁾(State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou 310058)

²⁾(School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074)

³⁾(School of Engineering, Lishui University, Lishui 324000)

⁴⁾(School of Automotive Technology, Zhejiang Technical Institute of Economics, Hangzhou 310018)

Abstract: Formal system verification is an important method to guarantee the correctness of system design. Because of the physical-software synergetic characteristics of complex mechatronic systems, how to verify the system design from the dynamic perspective is a challenging issue in this research area. To this end, we proposed a formal system model verification method by extending SysML, which is the standard modeling language for systems engineering. Specifically, first, we combined the computational tree logic (CTL) and flow-based functional representation as the formal basis and proposed a system functional modeling method in SysML; then, we proposed a system behavioral modeling method in SysML based on hybrid automata (HA); finally, specific to the different dynamics of physical and software subsystems, we conducted the system design model verification in a layered manner by leveraging the model checker NuSMV. We used the mobile robot system as a case study to illustrate the system verification process.

收稿日期: 2019-06-19; 修回日期: 2019-07-22. 基金项目: 国家重点研发计划项目(2018YFB1700901); 国家自然科学基金(61672247, 61772247, 61873236); 浙江省自然科学基金(LY18F030001). 曹悦(1986—), 女, 博士研究生, 主要研究方向为模型驱动系统层设计; 刘玉生(1970—), 男, 博士, 教授, 博士生导师, 论文通讯作者, 主要研究方向为 MBSE, CAX; 赵建军(1973—), 男, 博士, 副教授, 硕士生导师, 主要研究方向为 MBSE, CAD/CAM; 叶晓平(1965—), 男, 学士, 教授, 主要研究方向为数字化设计与制造; 周书华(1963—), 男, 学士, 高级讲师, 主要研究方向为数字化设计、智能控制.

Key words: mechatronic systems; model-based systems engineering; system design; system verification; model checking

复杂机电系统是物理系统与信息技术的有机融合^[1]。随着系统复杂性、不确定性以及柔性的增加, 软件正在逐步成为复杂机电系统的核心竞争力之所在^[2]。然而, 传统的系统设计方法只有在机械、电气等物理部分设计完成后才开始软件部分的设计, 这会引入不必要的设计约束, 导致非最优系统设计结果的产生^[3]。基于模型的系统工程(model-based systems engineering, MBSE)^[4]为软件与物理的融合设计提供了解决思路。它采用模型从高度抽象、学科无关的角度对系统进行描述, 因此, 被广泛应用于复杂机电系统的早期设计中^[5-7]。

系统设计是 MBSE 过程中的一个重要环节, 该过程所产生的系统设计模型是后续软件、机械、电气等多学科详细设计的基础。因此, 系统设计模型的正确性对复杂机电系统开发的成败起到决定性作用。系统设计验证的一个重要方面是系统功能验证, 它是检验系统设计是否符合系统功能描述的过程。基于形式化方法的系统功能验证能够检验系统行为在所有可能场景下是否符合系统功能, 从而极大地提高验证的置信度, 也因此正逐渐受到工业界和学术界的重视^[8]。

形式化的系统设计模型是形式化验证的基础。然而, MBSE 的标准建模语言系统建模语言(systems modeling language, SysML)是一种领域无关的半形式化建模语言, 它并不具备面向复杂机电系统的形式化精确语义。为此, 本文针对复杂机电系统物理与软件相融合的特征, 对 SysML 进行扩展, 并提出了一套基于 SysML 的形式化验证方法。该方法的核心是根据复杂机电系统的特征、以形式化方法为基础对 SysML 进行扩展, 从而描述系统应满足的功能特征及系统行为; 之后, 有针对性地采取不同的验证手段对模型的各个部分进行验证。为实现这一目标, 本文首先以计算树逻辑(computational tree logic, CTL)为基础, 对 SysML 功能建模方法进行扩展; 然后, 以混合自动机(hybrid automaton, HA)为基础, 提出基于 SysML 的行为建模方法; 最后, 根据物理与软件子系统的不同特征, 采用层次化的方式自动验证行为模型是否符合功能模型所描述的功能特征。基于本文方法, 设计人员能够在设计早期发现系统设计缺陷, 以避免将其带入后续详细设计过程, 从而有效

地提高系统设计质量、降低系统开发成本。

1 相关工作

仿真是系统验证的一种重要手段。一些研究工作通过将 SysML 与系统仿真语言相结合, 实现了对系统模型的自动仿真验证。例如, Cao 等^[9]基于 SysML 提出了融合物理与控制系统动态特征的统一行为建模语言, 并将其映射到多种系统仿真平台以实现系统设计模型自动仿真; 周书华等^[10]分析了 SysML 与多领域仿真建模语言 Modelica 之间的映射关系, 实现了 SysML 系统设计模型与 Modelica 系统仿真模型之间的自动转换。然而, 仿真是一种非形式化的、非完全的验证手段, 它仅能对有限的使用场景进行验证, 而无法捕获一些未测试到的使用场景所导致的系统失效^[11]。

一些研究工作采用形式化方法对系统的静态特性进行验证。例如, Feldmann 等^[12]采用本体对系统设计不同阶段所产生的模型进行形式化描述, 并验证不同模型所描述的结构属性之间的一致性; Chen 等^[13]提出基于关系的静态特征描述语言以描述系统需求, 并将其与基于 SysML 的系统设计模型转换为概念图, 从而验证系统设计 with 系统需求之间的一致性。然而, 这些方法主要针对系统静态结构特征及属性, 而无法对系统动态特征进行描述和验证。

在系统动态特征验证方面, 柯文俊等^[14]提出了一种将 SysML 活动图模型自动转换为 Petri 网模型的机制, 并使用 Petri 网验证工具对系统设计进行自动验证。然而, 该方法仅能对系统的非功能性需求如安全性、活性以及无锁性进行验证, 而缺乏对系统功能需求的描述和验证。张琛等^[15]将 UML 状态机图转化为 Promela 模型, 并采用命题投影时序逻辑对系统交互性质进行描述, 通过将二者输入 Spin 模型检测器, 能够有效地判断系统模型是否满足系统期望的交互性质。然而, 该方法要求用户直接使用形式化语言对系统性质进行描述, 在易用性方面有所欠缺。此外, 上述方法主要面向软件系统, 而无法直接应用于同时具有软件与物理融合特征的复杂机电系统。Chapurlat^[16]提出了一个面向通用系统工程的模型验证框架, 对如何构

造模型验证工具给出了系统的流程及相应表示语言。然而,该工作并没有阐述针对特定领域(如复杂机电系统设计)对其框架进行定制的方法。

综上所述,当前研究工作还缺乏对复杂机电系统设计的动态特征进行有效验证的方法。这主要由以下 3 个原因所造成:缺少对物理与软件融合的功能特征的形式化表示,以及图形化建模方法;缺少物理与软件融合系统行为建模方法;缺少同时考虑物理与软件动态特征的自动模型验证方法。

2 基于 SysML 的物理与软件融合功能建模

功能模型描述系统应满足的功能需求。形式化的功能模型是对系统设计进行功能验证的基础。当前,形式化功能建模的主流方法是基于流的功能表示^[17]。然而,这种方法仅能描述机械、电气等物理对象的功能,而无法对软件功能进行描述。因此,本文首先对这种经典的功能表示方法进行分析;之后,根据软件功能的特性对该表示进行扩展,使其能够将物理与软件功能进行融合表示;最后,基于形式化的功能表示对 SysML 进行扩展,以支持系统功能的图形化建模。

2.1 基于流的功能表示及其问题分析

根据基于流的功能表示,一个对象的功能被描述为该对象对其输入流进行转换并产生输出流的过程。在此基础上,Hirtz 等^[18]对功能的流及转换进行标准化并定义了功能基。在功能基中,流被分为物料流、能量流和信号流 3 大类及其子类;功能对流的操作被分为分支、转换、连接、控制等 9 大类及其子类。由于功能基将功能对流的操作简单表示为动词,缺乏计算机可理解的语义,因此,一些研究人员对功能的操作进行了形式化表示。例如,Chen 等^[19]将其表示为对流属性的约束,Yuan 等^[20]采用功能效应来表示可观察的、客观的流属性变化等。

从上述基于流的功能表示的相关研究可以看出,为了表示系统功能,需要从系统的处理对象和预期行为 2 个方面进行描述。物料流、能量流和信号流涵盖了物理和软件系统的处理对象,这也为物理与软件融合功能表示提供了基础。然而,对于系统预期行为,即功能语义的描述,则主要面向传统的物理占主导的机电一体化系统,而无法将软件所承担的功能进行融合表示。这主要是由于软件具有与物理部分截然不同的行为特性。与物理

部分所执行的物理转换不同,软件的主要功能是对不同的物理过程的执行顺序进行协调,这体现出离散而非连续的动态特征。因此,形式化的功能表示必须能够描述这种离散的时序和逻辑关系。

2.2 基于 CTL 的形式化功能表示

考虑到软件部分所承担的功能,复杂机电系统的功能模型不仅需要描述连续的物理转换,即物理功能,还需要说明这些转换之间的复杂逻辑,即软件功能。具体来讲,一方面,物理转换的执行需要遵循一定的顺序;另一方面,多个物理转换可以根据控制模式按照不同的顺序执行。

为了将物理功能与软件所主导的复杂的高层控制逻辑进行融合表示,本文将传统的功能模型扩展为 3 个层次:

(1) 功能效应层 $T = \{t_1, t_2, \dots, t_n\}$ 。该层描述系统需要执行的所有物理转换。每一个物理转换表示为一个功能效应。功能效应建立在定性推理的基础之上,它将物理转换描述为对流属性的定性改变。例如,“加热气体”这一功能的操作语义描述为将气体的温度从低升高,其中,“气体”是转换的对象流,“温度”是流属性,“低”和“高”是属性的定性值,“升高”是属性值的变化趋势。功能效应的形式化定义及其应用参见文献[20]。

(2) 操作层 $P = \langle T, Q_s \rangle$ 。该层描述功能效应之间所有可能的执行序列。每个执行序列被称为一个操作,每个操作表示为功能效应通过时序算子相连所形成的表达式。 $Q_s = \{\&, |, \cup\}$ 为时序算子集合,代表顺序、并发以及无时序 3 种可能的执行方式。例如, $p = t_1 \& t_2 \& t_3$ 表示 3 个功能效应 t_1 , t_2 和 t_3 顺序执行的一种操作。

(3) 规约层 $S = \langle AP, O_{CTL} \rangle$ 。该层描述了系统依据不同的控制模式对操作的选择。每种控制模式通过一个 CTL 规约描述,每个规约由原子命题通过 CTL 算子连接而成。

原子命题集合(atomic proposition, AP)的形式化定义为 $AP = SP \cup EP \cup CP$ 。其中,状态命题集合(state proposition, SP)为形如 $state = p$ 的状态命题集合,变量 $state$ 用于记录系统状态,当操作 p 正在执行时,则该命题为真;事件命题集合(event proposition, EP)为形如 $event = e$ 的事件命题集合, $event$ 表示系统中的事件,当事件 e 发生时,则该命题为真;条件命题集合(condition proposition, CP)为形如 $p = v$ 的条件命题集合,当某属性 p 取值为 v

时, 则该命题为真.

$O_{CTL} = O_t \cup O_q \cup O_l$ 表示 CTL 算子. 其中, 时态算子 $O_t = \{F, G, X, U\}$ 对状态的时间关系进行修饰, 例如, F 表示某未来状态、 G 表示所有未来状态; 路径量词 $O_q = \{A, E\}$ 对时间路径进行修饰, 其中, A 表示对所有路径, E 表示存在一条路径; 基本逻辑算子 O_l 由与、或、非以及蕴含等组成.

在扩展后的功能模型中, 功能效应层描述了复杂机电系统的物理功能, 操作层与规约层对软件功能进行描述, 因此, 该模型能够支持复杂机电系统的物理与软件融合功能表示.

以下通过一个例子对上述形式化功能描述进行说明. 例如, 某数控弯管机系统的送料系统需要执行送料与转料 2 个动作, 当模式为 1 时, 2 个动作顺序执行; 当模式为 2 时, 2 个动作并发执行. 基于上述文字化功能描述, 可构建层次化的功能描述:

(1) 功能效应层包含送料和转料 2 个物理过程, 它们分别由 2 个功能效应 t_m 和 t_r 表示;

(2) 操作层包含 2 个操作, $op_1 = t_m \& t_r$ 表示送料与转料顺序执行, $op_2 = t_m | t_r$ 表示送料与转料并发执行;

(3) 规约层包含 2 条 CTL 规约, 即 $AG(state=init \& m=1 \rightarrow EF state=op_1)$ 以及 $AG(state=init \& m=2 \rightarrow EF state=op_2)$, 它们分别表示在 2 种模式下系统所执行的不同操作过程.

2.3 基于参数图的图形化功能建模

采用上述形式化功能表示, 可以将复杂机电系统的物理与软件功能进行统一的表示. 然而, 由于 CTL 语法的复杂性, 用户难以直接使用这种形式化方法来对系统功能进行描述. 为解决形式化功能建模方法的易用性问题, 本文基于上述形式化语法对 SysML 进行扩展, 使用户可以采用图形化的方式创建具有形式化语义的、可验证的系统功能模型.

由于基于 CTL 的功能描述本质上是对系统预期行为及其执行顺序的一种约束, 因此, 在 SysML 模型元素中, 与其语义最为相近的是约束模块. 约束模块是一种特殊的 SysML 模块, 它对布尔型的约束表达式进行封装, 表达式中所涉及的变量通过约束模块的约束参数来描述. 通过对约束模块进行实例化, 可以将这些参数与系统中模块的属性进行绑定, 从而对系统应符合的特征进行说明. 这一绑定过程可以在 SysML 参数图 (parametric diagram, PAR) 中进行图形化建模.

由于约束模块具有可重用性, 因此, 可以将功能模型中的常用操作、原子命题以及 CTL 规约定义为可重用的约束模块, 用户可以在 PAR 中对这些已定义的约束模块进行配置和实例化, 以实现图形化的功能建模过程. 采用该方法可以将 CTL 的底层复杂语法进行隐藏, 从而提高建模方法的易用性与模型的准确性.

基于上述思想, 本文采用如下步骤实现基于 SysML 的功能建模:

(1) 采用轻量级扩展方式, 在 SysML 已有模型元素基础上定义版型, 以创建与形式化语法元素相对应的模型元素. 版型是 UML 三大扩展机制之一, 它通过在已有元类或版型基础上进行扩展或继承, 来创建具有领域相关语义的新的模型元素. 如图 1 所示, 在约束模块基础上定义 «FunctionalEffect», «Operation», «Proposition» 以及 «CTLSpec» 版型来表示功能模型中的功能效应、操作、命题以及 CTL 规约等概念. «Proposition» 又进一步细化为 «StateProposition», «EventProposition» 以及 «ConditionProposition», 以描述 3 种特定的命题. «FunctionalEffect» 具有 FlowObject, FlowProperty 以及 Trend 这 3 个标签, 用于描述功能效应的作用对象、作用属性和执行语义.

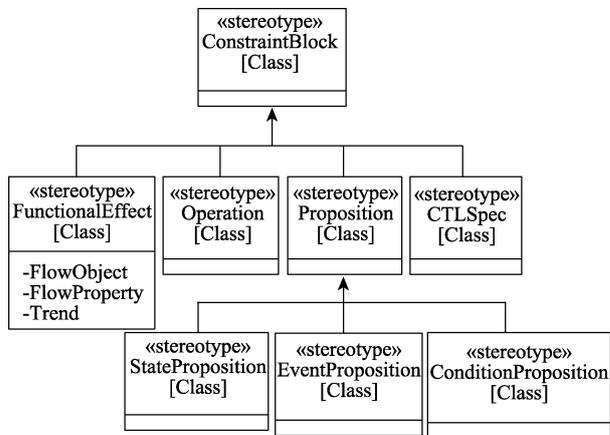


图 1 基于 CTL 的功能建模扩展包

(2) 采用上述版型, 创建表示常用功能效应、操作、命题及 CTL 规约的约束模块, 形成可重用模块库. 例如, ConditionalActivation 是一个 CTL 规约, 它的约束表达式为 $AG(StateInitial \& Condition \rightarrow EF StateFinal)$; 其中, $StateInitial$, $Condition$ 与 $StateFinal$ 为该约束模块的约束参数, 分别表示状态命题或条件命题.

(3) 通过将已定义的约束模块实例化, 并在

PAR 中将相应的约束参数与系统属性进行绑定, 可以完整表达基于 CTL 的功能模型. 与形式化定义相对应, 图形化的 SysML 功能模型同样分为 CTL 规约层、操作层与功能效应层 3 个层次. 图 2 展示了第 2.2 节中数控弯管机示例 CTL 规约的图形化模型. 模型的最上层为一个 CTL 规约模板

ConditionalActivation 的实例 $spec_1$, 该规约所涉及的具体的命题通过其参数与 3 个命题实例 $initState$, $modeCon$, $finalState$ 相连; 每个命题所使用的操作或属性又进一步与操作或属性实例相连. 例如, $finalState$ 所指定的操作为 op_1 , 它表示 2 个功能效应 t_m 与 t_r 顺序执行.

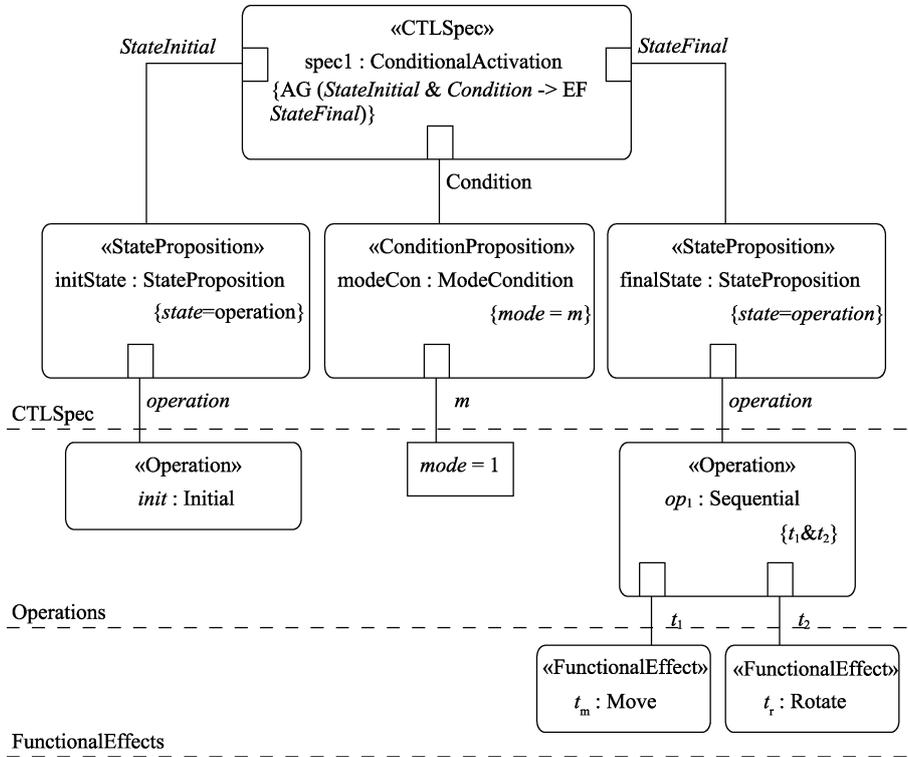


图 2 图形化 CTL 规约示例

3 基于 SysML 的物理与软件混合行为建模

系统功能验证的目的是判断系统行为是否与系统功能所描述的预期执行过程相符. 因此, 除了功能特征外, 还需要对系统行为进行形式化描述. 复杂机电系统可涉及 3 种行为, 即时间连续行为、基于事件的离散行为以及二者的混合行为. 时间连续行为是指系统的状态变量随时间发生连续变化; 离散行为是指系统状态根据事件的激发或条件的改变而发生跳变; 混合行为是上述二者行为的结合, 系统状态在连续变化的同时, 又可能发生离散的跳变. 为了表示复杂机电系统的行为, 与大部分行为建模方法类似, 本文采用 HA^[21]作为理论基础. 由于 HA 为一种理论模型, 因此, 还需要合理选取或扩展 SysML 语言中的模型元素来支持混合行为的图形化建模.

3.1 HA 理论模型

HA 主要由状态及其之间的转换构成, 其定义为 $HA = \langle X, Loc, Edge, \Sigma, init, inv, flow, jump \rangle$. 其中, $X = \{x_1, x_2, \dots, x_n\}$ 为系统状态变量的集合, 这些状态变量的取值随时间发生连续变化或受事件驱动发生离散跳变; $Loc = \{l_1, l_2, \dots, l_n\}$ 为系统状态的集合; $Edge \subseteq Loc \times \Sigma \times Loc$ 表示状态之间受事件驱动的离散转换; $\Sigma = \{e_1, e_2, \dots, e_n\}$ 表示系统接收或发出的事件; $init$, inv 以及 $flow$ 为 3 个函数, 它们将 3 个布尔值函数 $init(x)$, $inv(x)$ 以及 $flow(x)$ 赋予系统的每一个状态, 以描述状态变量在该状态下的初始条件、应满足的条件以及所遵循的连续变化; $jump$ 是一个函数, 它为每一个转换赋予一个布尔值函数 $jump(x)$, 以描述转换发生时状态变量的离散跳变.

从上述形式化定义可以看出, HA 是一种特殊

的有限状态机(finite state machine, FSM), 它由状态及其之间的转换所构成. 转换的 *jump* 函数描述了混合行为中的离散部分, 状态的 *flow* 函数则描述了状态变量的连续变化情况.

3.2 基于状态机图的图形化行为建模

为了支持基于 HA 的系统行为的图形化建模, 需要选择 SysML 中的相应模型元素来对 HA 中的概念进行表示. 由于 HA 是一种特殊的 FSM, 因此, SysML 中的状态机是与其语义最为接近的模型元素. 状态机是 SysML 中描述模块行为的模型元素之一, 它用于表示模块由事件所驱动的状态改变. 每个状态机至少包含一个区域, 区域中包含状态和转换. 每个状态具有 3 种内部行为: *entry* 和 *exit* 表示进入和离开该状态时模块的行为, *do* 表示模

块在该状态下所执行的行为. 转换表示从一个状态到另一个状态的改变, 转换只有在其上的触发器事件发生且满足 *guard* 时才会被触发, 转换触发时会执行其上的 *effect*. 状态机可以通过 SysML 的状态机图(state machine diagram, STM)进行图形化建模.

为了表示 HA 的特殊语义, 需要对 SysML 状态机的相关模型元素进行扩展, 该扩展包所包含的版型如图 3 所示. 状态机的版型«HybridAutomaton»用于表示整个 HA, 它是一种仅包含单一区域且区域中的状态均为简单状态的特殊状态机. 简单状态是指不包含子状态的状态, HA 中的状态和转换分别通过状态的版型«HALocation»与转换的版型«HAEdge»表示.

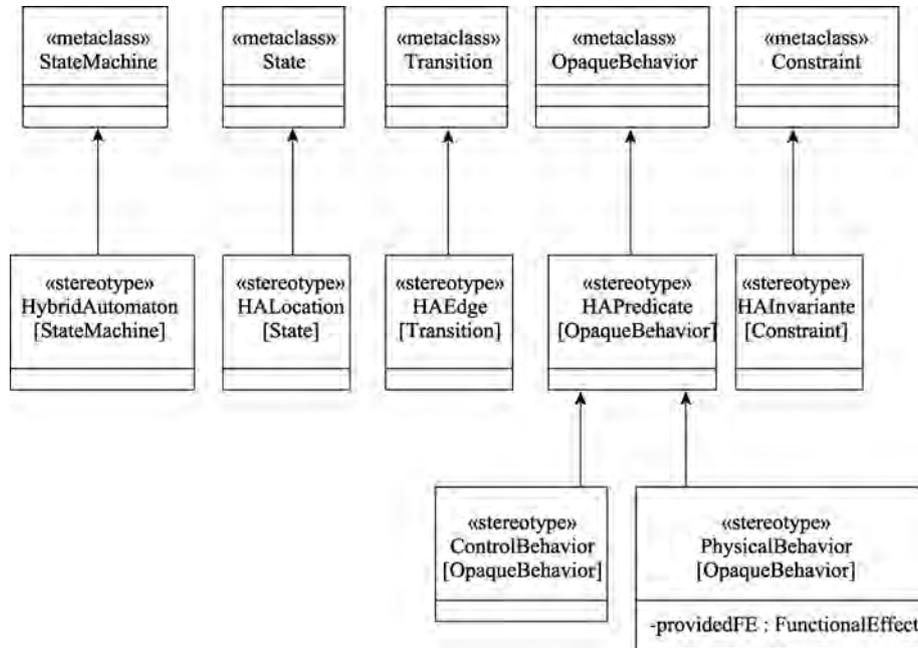


图 3 基于 HA 的行为建模扩展包

HA 所执行的具体行为通过不透明行为的版型«HAPredicate»来表示. 不透明行为是一种描述行为的模型元素, 它可以采用任意语言, 如 Modelica, Java 等来描述系统的具体行为. 根据«HAPredicate»所描述行为语义的不同, 又将其进一步划分为«ControlBehavior»与«PhysicalBehavior». 前者描述对系统状态变量进行离散赋值的控制行为, 后者以微分代数方程的形式描述系统的连续物理行为. 物理行为还具有 *providedFE* 标签, 用于记录其所能实现的功能效应.

init 函数通过 HALocation 的 *entry* 属性表示, 它将离散的控制行为赋予状态. *flow* 函数通过

HALocation 的 *do* 属性来描述, 它将连续的物理行为赋予状态. *jump* 函数通过 HAEdge 的效应来表示, 它将离散控制行为赋予转换. *inv* 函数通过向 HALocation 上添加版型为«HAInvariante»的约束来实现.

4 物理与软件融合层次化自动模型验证

由于复杂机电系统的物理与软件部分呈现截然不同的动态特征, 因此, 其模型验证无法使用单一的验证方法, 而需要根据不同的特征有针对性地采用不同的验证策略来实现. 基于 CTL 的功能

模型与基于 HA 的行为模型本质上均采用层次化的方式来进行建模. 具体来讲, 在基于 CTL 的功能模型中, 功能效应描述物理系统功能, 操作与 CTL 规约描述软件系统功能; 在基于 HA 的行为模型中, 状态及其之间的转换描述软件系统所主导的离散行为, 而每个状态下的连续方程则描述了物理系统行为. 通过上述分析, 可以提取出如图 4 所示功能与行为模型各层次之间的对应关系. 基于这种对应关系, 可以在复杂机电系统的功能与行为模型之间自底向上开展包括物理验证、时序验证及模式验证 3 个层次的模型验证过程.

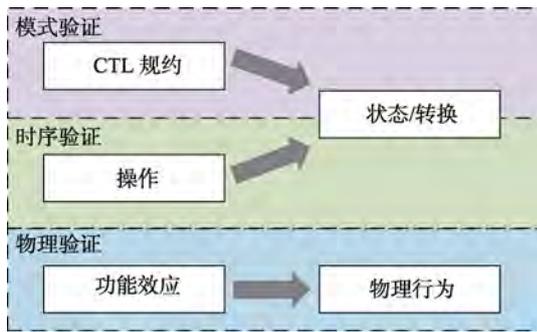


图 4 层次化模型验证方法

4.1 物理验证

物理验证的目的是判断系统行为模型中的物理行为是否能够实现系统功能模型所指定的功能效应. 由于行为模型中的物理行为记录了其所提供的功能效应, 因此, 物理行为验证可以基于功能效应的匹配来自动实现.

当需要实现的功能效应 t_{req} 与物理行为 b 所提供的功能效应 t_{prov} 满足以下 3 个条件时, 则判定物理行为 b 可以实现功能效应 t_{req} :

- (1) $t_{req}.FlowObject$ 与 $t_{prov}.FlowObject$ 相同或为其子类;
- (2) $t_{req}.FlowProperty$ 与 $t_{prov}.FlowProperty$ 相同;
- (3) $t_{req}.Trend$ 是 $t_{prov}.Trend$ 的子集.

当功能模型中的所有功能效应均得到满足, 则物理验证成功; 否则, 物理验证失败. 物理验证成功后, 需要在行为模型中标记每个状态所实现的功能效应, 作为后续时序验证的基础. 之后, 可进入下一步时序验证.

4.2 时序验证

时序验证的目的是判断系统行为模型中物理行为的执行顺序是否符合系统功能模型中操作所

指定的执行序列. 物理行为的执行顺序通过状态之间的转换来描述, 它可以表示为以状态为节点、以转换为边的图. 操作中的时序表达式所描述的执行序列可以转换为以功能效应为节点、以执行顺序为边的图. 具体来讲, 任意 2 个功能效应 t_1 与 t_2 之间根据时序算子的不同, 可以转化为如图 5 所示 3 种路径.

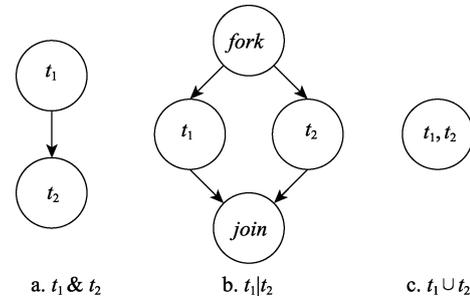


图 5 操作路径转化规则

由于在物理验证后已经记录了每个状态所对应的功能效应, 因此, 通过在状态图中查找与操作路径相匹配的子图, 便可以判断是否实现该操作. 当功能模型中的所有操作均得到满足, 则时序验证成功; 否则, 时序验证失败. 时序验证成功后, 需要将状态图中与操作相匹配的子图合并为一个复合状态节点并标记其所实现的操作, 作为后续模式验证的基础. 之后, 可进入下一步模式验证.

4.3 模式验证

模式验证的目的是判断系统行为模型中受事件或条件所驱动的模式选择是否符合系统功能模型中的规约. 系统行为模型的底层形式化基础是 FSM, 而功能模型的规约基于 CTL. 因此, 可以借助已有的模型检测器 NuSMV 来自动实现二者之间的模型验证. NuSMV 是一款应用十分广泛的开源模型检测器, 它提供一种文本化的语言来对有限状态系统以及基于 CTL 或线性时序逻辑(linear temporal logic, LTL)的系统规约进行描述. 使用该语言所描述的模块及其规约可以输入模型检测器, 从而自动验证系统是否满足规约所描述的特征.

本节首先对基于 NuSMV 的复杂机电系统模式验证过程进行说明, 之后, 就其中的关键步骤进行详细说明.

4.3.1 基于 NuSMV 的系统模式验证

整个验证过程如图 6 所示. 该过程的输入为基于 SysML 的图形化行为模型与功能模型. 该过程需要在物理及时序验证通过后才可开展, 因此, 输入的行为模型是经过操作合并后的 STM.

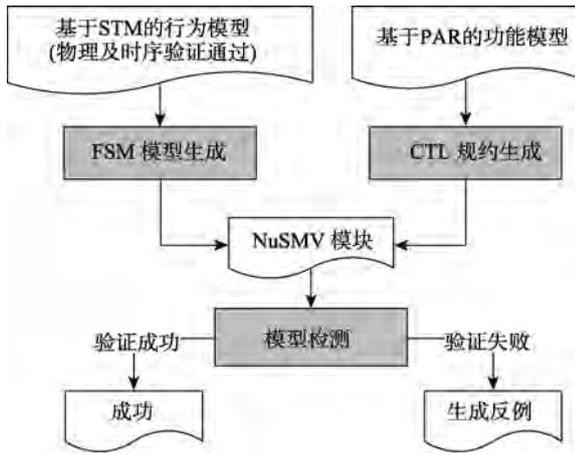


图 6 基于 NuSMV 的模型验证过程

该过程的核心步骤是从基于 SysML 的图形化模型中提取符合 NuSMV 语法的 NuSMV 模块. 它包括从基于 STM 的行为模型中提取 FSM 文本化描述, 以及从基于 PAR 的功能模型中提取 CTL 规约 2 个步骤. 这 2 个步骤将在后续章节中进行详细说明.

提取后的 FSM 模型与 CTL 规约合并为一个 NuSMV 模块, 该模块可以通过 NuSMV 模型检测器自动验证. 若验证失败, 检测器会返回相应反例以供用户参考.

4.3.2 FSM 模型生成

基于 NuSMV 语言, 一个有限状态系统的描述包括变量定义 VAR 以及变量赋值 ASSIGN 这 2 部分. 使用这种语法, 可以采用如下方式对一个 FSM 进行描述: 在其 VAR 部分定义系统的状态、事件以及条件参数; 在其 ASSIGN 部分描述状态之间的转换. 基于这一方法, 用于生成 FSM 模块的模板定义如下:

```

MODULE main
VAR
state: {s0, s1, ..., sn};
even : {e0, e1, ..., em};
par1: {v0, v1, ..., vl};
par2: {...};
...
parx: {...};
ASSIGN
init(state) := s0;
next(state) := case
state = si & (event = ek | parp = vq): sj;
...
TRUE: {s0, s1, ..., sn};
esac;
该模板中, 变量 state 用于记录系统状态, event
    
```

表示系统中的事件, $\{par_1, par_2, \dots, par_x\}$ 定义系统中转换上条件判断所使用的变量及其可能取值. init 语句表示系统的初始状态为 s_0 . next 语句中的每一个 case 表示系统中的每一个转换, 其中 s_i 和 s_j 表示转换的源状态和目标状态, e_k 和 par_p 表示触发该转换的事件或条件.

通过从 SysML 行为模型中提取相应信息并对上述模板进行实例化, 可以生成相应的基于 NuSMV 的文本化描述.

4.3.3 CTL 规约生成

在 NuSMV 中, 采用 SPEC 关键字描述每一条 CTL 规约. 在 SysML 的 PAR 中, 一条 CTL 规约对应一个版型为 «CTLSpec» 的约束模块的实例. 由于每一个约束模块通过其参数与命题实例相连, 而命题实例又通过其参数对操作实例进行引用, 因此, CTL 规约提取的关键是依据参数关联关系逐层开展参数名替换. 该算法过程包括如下步骤:

Step1. 对每一个 «CTLSpec» 实例, 提取其约束表达式 $c = f(x_1, x_2, \dots, x_n)$. 将约束表达式中的参数名替换为与该参数相绑定的命题实例名, 生成新的约束表达式 $c' = f(p_1, p_2, \dots, p_n)$.

Step2. 对上述步骤中的每一个命题 p_i , 提取其约束表达式 $c_i = f_i(x_1, x_2, \dots, x_m)$. 将约束表达式中的参数名替换为与该参数相绑定的相应实例名, 生成新的约束表达式 $c'_i = f_i(y_1, y_2, \dots, y_m)$. 对于状态命题, y_j 为操作实例名; 对于事件命题, y_j 为事件名; 对于条件命题, y_j 为变量名.

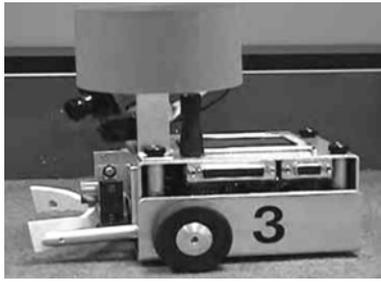
Step3. 将替换后的命题约束表达式带入 CTL 规约的约束表达式中, 生成最终的 CTL 规约 $c = f(f_1(y_1, \dots, y_m), f_2(y_1, \dots, y_m), \dots, f_n(y_1, \dots, y_m))$.

以图 2 为例, 从该 PAR 提取的符合 NuSMV 语法的 CTL 规约为 SPEC AG(state = init & m = 1 → EF state = op₁).

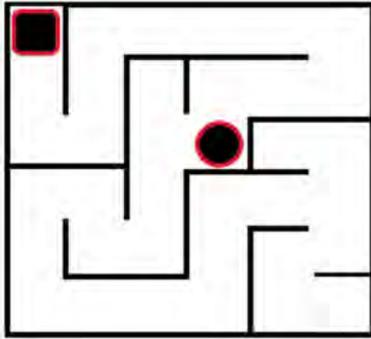
5 实例

本文以移动机器人系统^[22]为例来说明基于 SysML 的复杂机电系统设计的形式化验证过程. 该移动机器人的概念如图 7a 所示; 该机器人的任务是探索并寻找放置在如图 7b 所示的迷宫中的一个罐子, 图中方形所在格子代表机器人的起始位置, 圆形所在格子代表罐子所在位置. 为实现该任务, 机器人需要首先探索其周围的障碍物, 并判断障碍物类型(墙壁或罐子); 根据障碍物类型及所在

方向, 机器人需要转向相应方向, 并移动一格或拾取罐子.



a. 移动机器人系统



b. 迷宫示意图

图 7 移动机器人系统及其任务说明

在验证过程中, 首先需要对目标系统的功能进行建模. 以该机器人的移动子系统为例, 其功能模型包含 3 个层次:

(1) 功能效应层包含 3 个功能效应. T_f 表示前进一格, t_l 表示向左旋转 90° , t_r 表示向右旋转 90° .

(2) 操作层可以定义 4 个操作. $op_1 = t_l \& t_f$ 表示向左移动一格, $op_2 = t_f$ 表示向前移动一格, $op_3 = t_r \& t_f$ 表示向右移动一格, $op_4 = t_l \& t_l \& t_f$ 表示向后移动一格.

(3) 规约层包含 4 个规约, 它们根据主控系统发来的移动方向参数 d 的取值选择相应的操作. 例如, 当 $d=1$ 时, 机器人需要向左移动一格. 该规约可以表示为 $AG(state = idle \& d=1 \rightarrow EF state = op_1)$.

在上述形式化功能模型的基础上, 以 PAR 为主要视图建立图形化的功能模型. 图 8 展示了上述规约的图形化表示.

功能建模完毕后, 建立目标系统的行为模型. 该模型以 SysML STM 为主要视图进行建模和展示. 例如, 移动子系统的行为模型如图 9 所示.

在功能与行为建模完成后, 可以开展层次化自动模型验证.

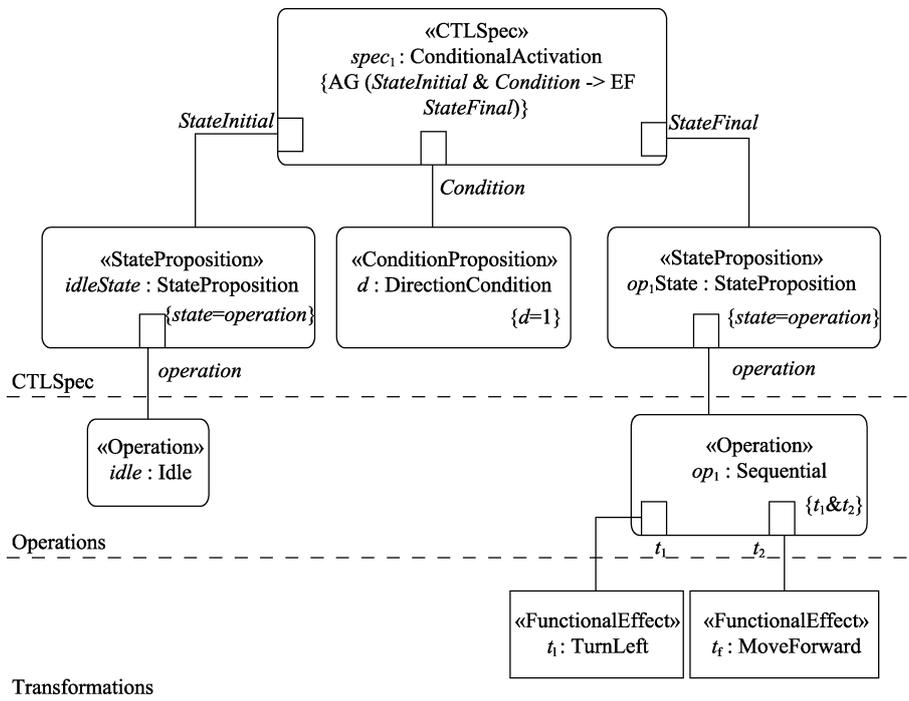


图 8 移动子系统功能模型示例

物理验证过程检验 3 个功能效应 t_f , t_l 和 t_r 是否得到满足, 并将实现这些功能效应的状态进行标记. 例如, 图 9 中的 t_{11} , t_{12} 和 t_{13} 用于实现功能效应 t_l .

时序验证过程检验 4 个操作 op_1 , op_2 , op_3 和 op_4 是否得到满足, 并将相应状态进行合并和标记. 如图 9 中 t_{11} 和 m_1 合并为 op_1 , t_{12} , t_{13} 和 m_4 合并为 op_4 .

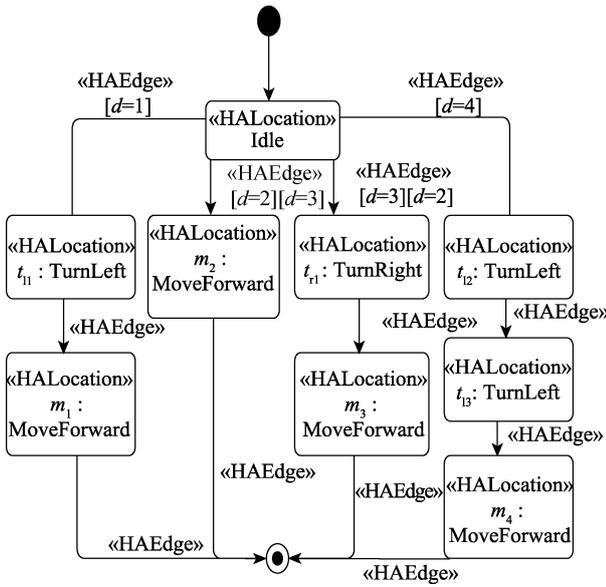


图 9 移动子系统行为模型示例

最后, 对行为模型和功能模型进行模式验证。基于这 2 个模型所生成的 NuSMV 脚本片段如下:

```

MODULE main
VAR
state: {Initial, Idle, op1, op2, op3, op4, Final};
d: {1, 2, 3, 4};
ASSIGN
init(state):= Initial;
next(state):= case
state = Initial: Idle;
state = Idle & d = 1: op1;
...
state = op1: Final;
...
TRUE: {Initial, Idle, op1, op2, op3, op4, Final};
esac;
SPEC AG(state = Idle & d = 1 → EF state = op1)
...
    
```

使用 NuSMV 模型验证工具执行该脚本, 可自动验证行为模型是否符合功能模型所描述的规约。其验证结果如图 10 所示。

除了上述验证通过的情况之外, 对于验证失败的情形, 系统也会根据出错情况返回相应的提示信息。这可以具体分为以下几种情况:

- (1) 物理层验证失败。例如, 某个功能效应无法在行为模型中找到实现它的状态。
- (2) 操作层验证失败。例如, 某个操作在行为模型中无法找到与其匹配的路径。
- (3) 规约层验证失败。例如, 行为模型不符合某条规约, 当图 9 中 $d=2$ 与 $d=3$ 写反时, 如图 11 所示, NuSMV 会判定验证失败并生成反例。

```

*** This is NuSMV 2.6.0 (compiled on Wed Oct 14 15:36:00 2015)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <Please report bugs to <nusmv-users@fbk.eu>>

*** Copyright (c) 2010-2014, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://minisat.se/MiniSat.html
*** Copyright (c) 2003-2006, Niklas Een, Niklas Sorensson
*** Copyright (c) 2007-2010, Niklas Sorensson

WARNING *** This version of NuSMV is linked to the zchaff SAT solver (see http://www.princeton.edu/~chaff/zchaff.html).
WARNING *** Zchaff is used in Bounded Model Checking when the system variable "sat_solver" is set to "zchaff".
WARNING *** Notice that zchaff is for non-commercial purposes only.
WARNING *** NO COMMERCIAL USE OF ZCHAFF IS ALLOWED WITHOUT WRITTEN PERMISSION FROM PRINCETON UNIVERSITY.
WARNING *** Please contact Sharad Malik (malik@ee.princeton.edu) for details.

-- specification AG <<state = Idle & d = 1> → EF state = op1 is true
-- specification AG <<state = Idle & d = 2> → EF state = op2 is true
-- specification AG <<state = Idle & d = 3> → EF state = op3 is true
-- specification AG <<state = Idle & d = 4> → EF state = op4 is true
    
```

图 10 NuSMV 模型验证结果

```

-- specification AG <<state = Idle & d = 1> → EF state = op1 is true
-- specification AG <<state = Idle & d = 2> → EF state = op2 is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
state = Initial
d = 1
-> State: 1.2 <-
state = Idle
d = 2
-- specification AG <<state = Idle & d = 3> → EF state = op3 is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 2.1 <-
state = Initial
d = 1
-> State: 2.2 <-
state = Idle
d = 3
-- specification AG <<state = Idle & d = 4> → EF state = op4 is true
    
```

图 11 NuSMV 反例生成

6 结 语

本文针对复杂机电系统物理与软件融合的特征, 提出了基于 SysML 的系统功能与行为建模方法, 并实现了功能与行为模型的自动验证。主要工作如下:

- (1) 将 CTL 与基于流的功能表示相结合, 并以其为形式化基础对 SysML PAR 相关模型元素进行扩展, 形成了物理与软件融合的功能建模方法;
- (2) 以 HA 为形式化基础对 SysML STM 相关模型元素进行扩展, 提出了物理与软件融合的行为建模方法;
- (3) 根据复杂机电系统中物理与软件子系统不同的动态特征, 借助 NuSMV 模型检测器, 以层次化的方式实现了系统模型自动验证过程;
- (4) 以移动机器人为例, 对系统模型自动验证过程进行了说明。

本文工作的优势在于:

- (1) 提出了物理与软件相融合的功能与行为

建模方法,使得设计人员在对复杂机电系统进行设计和建模时,可以将二者协同考虑,从而有效地提高设计的质量和效率。

(2) 采用层次化的模型验证手段,从多个维度对复杂机电系统的动态特征进行全面验证,提高了验证的覆盖率和置信度。

然而,本文方法目前仅适用于规模较小、较为独立的物理与软件融合机电系统,对于大规模复杂系统的建模与验证,则仍需做进一步研究。具体来讲,系统功能与行为模型元素相对有限,对于复杂场景的描述能力有所欠缺;该方法只能对单个系统、子系统或构件进行验证,而无法验证这些对象之间的交互行为。

因此,本文未来工作主要包括以下 2 个方面:

(1) 引入复合结构、复杂时序关系等元素对系统功能与行为建模方法进行扩展,使其能够支持更为复杂的时序和逻辑特征的描述及验证。

(2) 选取合适的形式化基础对 SysML 序列图进行扩展,以对系统中对象之间的交互行为进行描述和验证。

参考文献(References):

- [1] Tomizuka M. Mechatronics: from the 20th to 21st century[J]. Control Engineering Practice, 2002, 10(8): 877-886
- [2] Bricogne M, Le Duigou J, Eynard B. Design processes of mechatronic systems[M] //Mechatronic Futures. Heidelberg: Springer, 2016: 75-89
- [3] Alvarez Cabrera A A, Foeken M J, Tekin O A, et al. Towards automation of control software: a review of challenges in mechatronic design[J]. Mechatronics, 2010, 20(8): 876-886
- [4] Fisher J. Model-based systems engineering: a new paradigm[J]. Insight, 1998, 1(3): 3-4
- [5] Bassi L, Secchi C, Bonfé M, et al. A SysML-based methodology for manufacturing machinery modeling and design[J]. IEEE/ASME Transactions on Mechatronics, 2011, 16(6): 1049-1062
- [6] Thramboulidis K. The 3+1 SysML view-model in model integrated mechatronics[J]. Journal of Software Engineering and Applications, 2010, 3(2): 109-118
- [7] Liu Yusheng, Yuan Wenqiang, Fan Hongri, et al. Research on information integration framework of SysML based model-driven design of complex products[J]. China Mechanical Engineering, 2012, 23(12): 1438-1445(in Chinese)
(刘玉生, 袁文强, 樊红日, 等. 基于 SysML 的模型驱动复杂产品设计的信息集成框架研究[J]. 中国机械工程, 2012, 23(12): 1438-1445)
- [8] Foster H. Prologue: the 2012 Wilson research group functional verification study[OL]. [2019-06-19]. <https://blogs.mentor.com/verificationhorizons/blog/2013/04/23/prologue-the-2012-wilson-research-group-functional-verification-study/>
- [9] Cao Y, Liu Y S, Fan H R, et al. SysML-based uniform behavior modeling and automated mapping of design and simulation model for complex mechatronics[J]. Computer-Aided Design, 2013, 45(3): 764-776
- [10] Zhou Shuhua, Cao Yue, Zhang Zheng, et al. System design and simulation integration for complex mechatronic products based on SysML and Modelica[J]. Journal of Computer-Aided Design & Computer Graphics, 2018, 30(4): 728-738(in Chinese)
(周书华, 曹悦, 张政, 等. 基于 SysML 和 Modelica 的复杂机电产品系统设计与仿真集成[J]. 计算机辅助设计与图形学学报, 2018, 30(4): 728-738)
- [11] Mehrpouyan H, Giannakopoulou D, Brat G, et al. Complex engineered systems design verification based on assume-guarantee reasoning[J]. Systems Engineering, 2016, 19(6): 461-476
- [12] Feldmann S, Herzig S J I, Kernschmidt K, et al. Towards effective management of inconsistencies in model-based engineering of automated production systems[J]. IFAC-PapersOnLine, 2015, 48(3): 916-923
- [13] Chen R R, Liu Y S, Zhao J J, et al. Model verification for system design of complex mechatronic products[J]. Systems Engineering, 2019, 22(2): 156-171
- [14] Ke Wenjun, Chen Jing, Jiang Shan. System simulation and verification method based on Petri net model[J]. Systems Engineering and Electronics, 2017, 39(4): 924-930(in Chinese)
(柯文俊, 陈静, 江山. 基于 Petri 网模型的系统仿真验证方法[J]. 系统工程与电子技术, 2017, 39(4): 924-930)
- [15] Zhang Chen, Duan Zhenhua, Tian Cong, et al. Modeling, verification and test of interactive behaviors in distributed software systems[J]. Journal of Computer Research and Development, 2015, 52(7): 1604-1619(in Chinese)
(张琛, 段振华, 田聪, 等. 分布式软件系统交互行为建模、验证与测试[J]. 计算机研究与发展, 2015, 52(7): 1604-1619)
- [16] Chapurlat V. UPSL-SE: a model verification framework for systems engineering[J]. Computers in Industry, 2013, 64(5): 581-597
- [17] Pahl G, Beitz W, Feldhusen J, et al. Engineering design-a systematic approach[M]. 3rd ed. Heidelberg: Springer, 2007
- [18] Hirtz J, Stone R B, McAdams D A, et al. A functional basis for engineering design: reconciling and evolving previous efforts[J]. Research in Engineering Design, 2002, 13(2): 65-82
- [19] Chen Y, Liu Z L, Xie Y B. A knowledge-based framework for creative conceptual design of multi-disciplinary systems[J]. Computer-Aided Design, 2012, 44(2): 146-153
- [20] Yuan L, Liu Y S, Sun Z F, et al. A hybrid approach for the automation of functional decomposition in conceptual design[J]. Journal of Engineering Design, 2016, 27(4-6): 333-360
- [21] Alur R, Courcoubetis C, Halbwachs N, et al. The algorithmic analysis of hybrid systems[J]. Theoretical Computer Science, 1995, 138(1): 3-34
- [22] Bräunl T. Embedded robotics-mobile robot design and applications with embedded systems[M]. 2nd ed. Heidelberg: Springer, 2006