软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2020,31(6):1587−1599 [doi: 10.13328/j.cnki.jos.005993]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于 SHML 的 CPS 行为建模及仿真*

杜德慧[1,2,3], 管春琳[1,2,3], 王 耀[1,2,3], 郭 童[1,2,3]

[1](华东师范大学 软件工程学院,上海 200062)

[2](上海市高可信重点实验室(华东师范大学),上海 200062)

[3](教育部可信软件国际合作联合实验室(华东师范大学),上海 200062)

通讯作者: 杜德慧, E-mail: dhdu@sei.ecnu.edu.cn

摘 要: 信息物理融合系统(cyber-physical systems,简称 CPS)是深度融合了计算进程和物理进程的统一体,是集计算、通信与控制于一体的下一代智能系统,具有广阔的应用前景.CPS 的行为具有混成性、随机性等特征,建模及仿真 CPS 的动态行为对于开发高质量的 CPS 系统至关重要.但是目前缺乏面向 CPS 的领域建模方法及建模 CPS 的领域建模语言,也迫切需要支持仿真 CPS 领域模型的仿真工具.针对以上问题,提出一种面向 CPS 领域的随机混成建模语言(stochastic hybrid modeling language,简称 SHML)以支持建模 CPS 系统的行为.首先,根据 CPS 的领域特征定义了 SHML 的元模型作为其抽象语法,并定义了 SHML 的具体语法和操作语义;其次,基于 GEMOC 框架实现了 SHML 的可视化建模工具.此外,集成 GEMOC 的序列化执行引擎和 Scilab 的连续行为仿真引擎,实现仿真 CPS 的混成行为.提出了一种面向 CPS 领域的建模及仿真方法,设计并实现了一个集成的面向 CPS 行为的建模与仿真平台,为 CPS 的建模及仿真提供了一种有效的方法及工具支撑.

关键词: 信息物理融合系统;领域建模语言;元建模;仿真;GEMOC

中图法分类号: TP311

中文引用格式: 杜德慧,管春琳,王耀,郭童.基于 SHML 的 CPS 行为建模及仿真.软件学报,2020,31(6):1587−1599. http://www.jos.org.cn/1000-9825/5993.htm

英文引用格式: Du DH, Guan CL, Wang Y, Guo T. Towards modeling and simulating behavior of CPS based on SHML. Ruan Jian Xue Bao/Journal of Software, 2020,31(6):1587−1599 (in Chinese). http://www.jos.org.cn/1000-9825/5993.htm

Towards Modeling and Simulating Behavior of CPS Based on SHML

DU De-Hui[1,2,3], GUAN Chun-Lin[1,2,3], WANG Yao[1,2,3], Guo Tong[1,2,3]

[1](Software Engineering Institute, East China Normal University, Shanghai 200062, China)

[2](Shanghai Key Laboratory of Trustworthy Computing (East China Normal University), Shanghai 200062, China)

[3](MOE International Joint Laboratory of Trustworthy Software (East China Normal University), Shanghai 200062, China)

Abstract: Cyber-Physical systems (CPS) is a unified entity that deeply integrates computing processes and physical processes. It is a next-generation intelligent system integrating computing, communication, and control. It is widely used in various applications. The dynamic behavior of CPS is always hybrid and stochastic. Modeling and simulating the behaviors of CPS is crucial for developing high-quality CPS. However, it is still lack of domain modeling approach for CPS supporting the construction of CPS domain models and simulation techniques to simulate CPS domain models. To address these issues, this study proposes a domain-specific modeling language for CPS named SHML to support modeling the behaviors of CPS. Firstly, the metamodel of SHML is defined as the abstract grammar

* 基金项目: 国家自然科学基金(61972153); 国家重点研发计划(2018YFE0101000)

Foundation item: National Natural Science Foundation of China (61972153); National Key Research and Development Program of China (2018YFE0101000)

本文由“信息物理系统软件设计自动化”专题特约编辑卜磊教授、陈铭松教授、朱祺教授、刘超教授推荐.

收稿时间: 2019-08-21; 修改时间: 2019-10-23; 采用时间: 2020-01-13; jos 在线出版时间: 2020-04-18

according to the domain knowledge of CPS. Moreover, the concrete syntax and the operational semantics are also given. Secondly, based on the GEMOC studio framework, the graphical modeling tool of SHML is implemented. In addition GEMOC sequential execution engine and Scilab engine that can simulate continuous behavior are integrated, which supports the simulating hybrid behavior of CPS. The proposed wrok provides a domain modeling and simulation approach for CPS, which provides an effective approach and tool to support the modeling and simulation for CPS.

Key words: CPS; domain modeling language; meta-modeling; simulation; GEMOC

信息物理融合系统(cyber physical system,简称 CPS)[1]是一类综合计算、网络和物理环境的复杂异构系统,深度融合了计算进程和物理进程的统一体,是集计算、通信与控制于一体的下一代智能系统,在健康医疗、智能交通、航空电子、智慧城市等领域有着广泛的应用前景和发展空间.CPS 具有两个显著特征:1) 随机性:CPS 运行在开放的环境中,不确定的运行环境(例如网络延迟、物理设备、信号干扰)以及用户行为造成 CPS 的行为具有随机性;2) 混成性:CPS 是异构的混成系统,融合了连续的物理过程和离散的系统行为.CPS 的新特征对传统的软件开发方法提出了新的问题及挑战.本文重点讨论如何以模型驱动的方式开发 CPS,构建 CPS 的领域模型,并仿真、分析系统的随机、混成行为.

以领域模型(domain model)为基础的模型驱动工程(model-driven engineering,简称 MDE)[2]已经成功应用于多种领域的软件开发.复杂、异构的 CPS 系统如航空航天、智能交通、智能电网、智慧城市等的开发需要来自不同领域的专家对系统的不同方面进行设计与分析.与传统软件开发方法相比,MDE 更关注于如何使用抽象建模技术降低系统复杂性,提高软件开发的质量和效率.领域特定建模语言(domain-specific modeling language,简称 DSML)为构建领域模型奠定了基础,被广泛应用于构造特定领域的系统模型[3].领域模型抽取了该领域中的关键概念、知识,并建立了这些概念之间的关系、约束,它更接近人们的认知,更加方便专家和系统设计人员之间进行沟通、交流.因此在 MDE 社区中,越来越多的开发人员选择创建面向特定领域的建模语言,帮助其构建特定的领域模型[3−5].与标准的统一建模语言 UML[6]不同,领域建模语言 DSML 能够借助特定领域的知识构建系统的领域模型,支持以模型驱动的方式开发特定的应用系统,提高软件开发的质量和效率.然而,目前面向 CPS 领域的建模语言仍然不够成熟,缺乏面向 CPS 的领域建模语言.另一方面,CPS 的混成行为特征对传统的仿真方法提出了挑战.传统仿真方法通常是选用领域特定的解决方案和工具来满足特定的仿真需求,然而在 CPS 系统中,针对计算实体所描述的离散行为,需要采用基于事件驱动的离散事件系统对其进行仿真建模;而其物理实体所刻画的基于连续时间域上的物理量变化,需要采用基于连续时间的仿真建模.在工程中,分别采用特定的仿真工具对离散事件系统或连续系统建模、仿真,如 Simulink[7].CPS 仿真的困难之处在于如何有效融合异构仿真工具,实现混成行为的协同仿真.

本文的主要贡献包括以下 3 个方面.

(1) 根据 CPS 的领域特征,提出一种面向 CPS 行为的随机混成建模语言(stochastic hybrid modeling language,简称 SHML).根据 CPS 行为特征,定义了 SHML 的抽象语法、具体语法及操作语义;
(2) 基于 GEMOC 平台实现了该领域建模语言,支持用户基于 GEMOC 创建 SHML 模型;
(3) 集成 GEMOC 的顺序执行引擎和 Scilab 的连续行为仿真引擎,支持仿真 SHML 模型,实现仿真 CPS 的混成行为,帮助设计人员分析模型的执行轨迹.

总之,我们的工作提出了一种模型驱动式 CPS 建模及仿真方法,设计、实现了一种面向 CPS 的领域建模语言,能够支持建模 CPS 的随机、混成行为.此外,集成离散与连续行为的仿真器,实现 CPS 混成行为的仿真,为设计人员提供了一个集成的、面向 CPS 的建模与仿真平台,能够有效支持 CPS 的建模与仿真分析.

本文第 1 节介绍模型驱动、领域建模方法,并介绍 GEMOC 及支持连续行为仿真的 Scilab 工具,为 CPS 领域建模、仿真提供基础.第 2 节介绍面向 CPS 领域的随机混成建模语言 SHML,重点讨论其语法及语义模型,并介绍如何基于 GEMOC 技术框架实现 SHML 建模语言.第 3 节介绍 SHML 的仿真方法,提出将 Scilab 的 ODE 求解器与 GEMOC 的执行引擎集成在一起,实现 CPS 混成行为的仿真.第 4 节进行详细案例分析,结合温控系统的案例,展现领域建模语言 SHML 的建模能力,并基于本文所提出的仿真方法对温控系统的 SHML 模型进行仿

真.最后总结全文,并对未来值得关注的研究方向进行初步探讨.

1 预备知识

1.1 模型驱动开发及领域建模语言

对象管理组织(object management group,简称 OMG)[8]提出模型驱动架构(model-driven architecture,简称 MDA),其目标是将软件的开发行为提升到更高的抽象层次——模型层.以模型驱动的方式开发软件,整个开发过程中产生的核心制品是模型,从需求分析、设计实现到最终的代码生成,每个阶段的模型制品可借助模型转换工具,实现不同抽象层次的模型之间的转换,最终将系统的模型映射生成 C++或者 Java 代码等.模型驱动软件开发方法已经被广泛应用于大规模、复杂软件系统的开发.与其他软件开发方法相比,模型驱动开发方法更加关注领域知识,构造领域模型(domain models).其优势在于:使用更接近于人的理解和认知的可视化模型,有利于设计人员将注意力集中于分析业务逻辑,而不用过早地考虑与平台相关的底层实现细节.领域建模语言[9]与通用的建模语言相比更聚焦于某一特定问题,更加面向领域问题本身,能够帮助领域专家或系统开发人员使用更恰当、与领域知识相关的语言及工具来建模复杂的领域模型.

领域建模语言的语法用于描述怎样刻画模型,并且规定了模型元素之间的约束关系.语义的作用是描述模型到底具有怎样的含义,是模型执行、仿真的基础.语法又分为抽象语法(abstract syntax)和具体语法(concrete syntax),抽象语法通常由元模型(metamodel)来描述,刻画了该领域的关键建模元素(领域概念)及建模元素之间的关系.此外,元模型的定义需要遵循对象管理组织 OMG 提出的元建模标准 MOF(meta-object facility)[10],因此,在设计领域建模语言的元模型时需要根据 MOF 的规定,定义建模元素之间的关系.Eclipse 的建模框架(eclipse modeling framework,简称 EMF)提供了元模型的基础设施,受到了工业界和学术界的广泛认可.EMF 提供的元语言 Ecore 提供了构建 DSML 元模型的基本建模元素,我们可以使用 Ecore 灵活地定义面向 CPS 的领域建模语言的元模型.

1.2 GEMOC建模框架

目前,领域建模工具平台的典型代表性是 GEMOC 建模框架[11,12],由法国 INRIA 开发的基于 Eclipse 的开源建模框架,旨在为开发、集成、使用基于 EMF 的 DSML 提供技术支持.该框架提供了通用的接口,可用于集成与特定 DSML 相关联的执行引擎[12].GEMOC 框架中集成了各种建模技术及工具(包括 EMF,Ecore,Xtext,Sirius),并且提供了一个执行引擎以支持模型执行、调试、模拟等功能.GEMOC 框架由两个平台组成——语言平台(language workbench)和建模平台(modeling workbench):语言平台为语言设计人员设计 DSML 或组合各种 DSML 提供支撑,而建模平台为系统的建模人员提供模型的创建、执行与协同功能.同时,通过使用基于 Ecore 元模型实现领域建模语言的操作语义,使得领域建模语言具有可执行性.目前,GEMOC 的执行引擎只能支持仿真具有离散语义的模型,而无法支持仿真连续行为.因此,GEMOC 的执行引擎无法直接实现混成行为的仿真,需要对其执行引擎进行扩展,使其支持混成行为的仿真.幸运的是,GEMOC 框架提供了通用的接口用于集成特定领域建模语言的执行引擎,可以直接集成支持仿真连续行为的执行引擎.

Scilab 是一款专注于数值计算的开源软件,其历史可以追溯到上世纪 80 年代由 INRIA 开发的计算辅助系统 Blaise[13,14].Scilab 的主要功能包括数值计算、数据可视化、算法设计、开发及应用部署,其涉及的主要应用领域包括数学、统计、信号处理及控制系统领域.在计算结果可视化方面,Scilab 能通过各种 *plot* 函数绘制图形,主要包括二维图形和三维图形,而且能够根据用户设定的参数生成满足特定需求的图形,具有方便、灵活的特点.Scilab 的仿真器能够有效支持连续行为的仿真,因此可以将其集成到 GEMOC 执行引擎中.

2 面向 CPS 的领域建模语言 SHML

本节重点介绍一种面向 CPS 的领域建模语言 SHML,用于建模 CPS 的混成行为及随机行为.首先给出 SHML 语言的元模型用于定义 SHML 的抽象语法,用于描述建模 CPS 所必须的建模元素并规约建模元素之间

的约束关系;其次定义SHML的操作语义规则,为模型的仿真执行提供语义支撑;最后,基于GEMOC框架实现可视化的领域建模语言SHML,并为模型设计者提供一个面向CPS的领域建模、仿真平台.

2.1 SHML的语法设计

为了建模混成、随机的CPS行为,我们根据CPS的领域特征抽象出CPS领域建模语言中所必须包含的建模元素,并在此基础上定义了SHML的元模型,用以表示其抽象语法.具体地,CPS的行为具有混成性,即包括离散与连续行为,因此,面向CPS的领域建模语言应该能够有效地支持建模系统的混成行为.此外,随机性也是CPS的另一重要特性.通常,CPS运行在开放的环境中,会受到外界环境中不确定性因素(如环境干扰、物理噪声、信号的延迟)的影响;同时,人与CPS系统的交互频繁,而人的行为难以预测,具有不确定性.这些不确定因素导致CPS的系统行为具有随机性,因此需要使用特定的建模元素建模系统的混成行为及随机行为.

根据MOF标准的规定,我们使用Ecore元语言定义的SHML元模型如图1所示,图中的各个类分别对应着CPS领域中与行为密切相关的抽象概念.其中,

- 类System表示一个SHML模型,其组成元素包括TSHS(timed stochastic hybrid statechart)、全局变量(global variable)、全局时钟(global clock)及全局事件(global event);
- 全局变量代表了CPS中物理部件所涉及到的物理变量;
- 时钟表示系统中涉及到的时间事件;
- 全局事件表示各组件之间的同步通信;
- 在元模型中,System可包含多个TSHS用于建模系统中多个并发组件的行为.一个TSHS由状态(state)、迁移(transition)、局部变量(local variable)、局部时钟(local clock)及常微分方程(ordinary differential equation,简称ODE)组成.通常,TSHS模型包含一个初始状态(initial state).当迁移使能时,系统将从源状态迁移到目标状态.在某个状态处,可以使用一个常微分方程ODE等式建模该状态处的连续物理变化过程,这种从属关系由关联关系(slaveode)表示;
- 常微分方程ODE由函数(function)、约束条件(condition)以及求解区间(interval)组成.函数由自变量(IndeVariable)、因变量(DeVariable)、函数右式(fright)组成.例如,一个常微分方程为 $y'=y^2-2\times\sin(x)+\cos(x)$,其自变量为 x,因变量为 y,而函数右式为 $y^2-2\times\sin(x)+\cos(x)$;
- 约束条件是函数求解的一些限制条件,通常是包含函数变量的等式或不等式;函数的求解区间定义了函数自变量的取值范围.在这里,我们定义了一个简单且常用的ODE形式,它包括函数、约束条件及求解区间,足够支持建模系统的连续行为.注意:为了在仿真阶段直接集成Scilab ODE的求解器,我们采用的ODE的形式与Scilab中使用的ODE类型是一致的,以支持使用Scilab对SHML模型中的连续行为进行仿真;
- 为了建模系统的随机行为,我们提出在SHML元模型中使用概率迁移表示迁移发生的概率,如图1所示.矩形表示迁移类(transition),是一个抽象类,有两个子类——普通迁移(ComTransition)及概率迁移(ProbTransition).与普通迁移不同,概率迁移有一个额外的概率属性(probability),它是一个介于0和1之间的实数,表示该迁移能被触发的概率值.需要注意的是:从同一源状态出发的概率迁移的概率之和为1.卫士条件(guard)是迁移类的组成部分,有3种具体的类型——时序条件、事件条件、变量条件.其中,时序条件与特定的时钟onclock关联,事件条件与特定的事件onevent关联,变量条件与特定的物理变量onvariable关联.

以上建模元素以及它们之间的关系形成了SHML元模型的主体部分.此外,为了建模复杂的系统行为,我们采用层次化的嵌套建模结构,将SHML模型的状态分为两层——父状态层及子状态层.这里使用状态类的自反关联关系表示状态的嵌套.SHML元模型是定义具体语法及操作语义的基础,是构建领域建模语言的最关键内容.基于元模型,我们可以定义相应的具体语法、操作语义.

领域建模语言的具体语法,即建模元素的具体表示有两种常见的形式:文本和图形.文本形式的模型常以ASCII或XML文本形式出现,而图形化具体语法常以框线图、树、表格的形式出现.针对SHML元模型中的各

个建模元素,我们设计了它们的具体图标如图 2 所示,给出了各个建模元素的具体表示形式,比如颜色、形状、大小、边框、标签等.

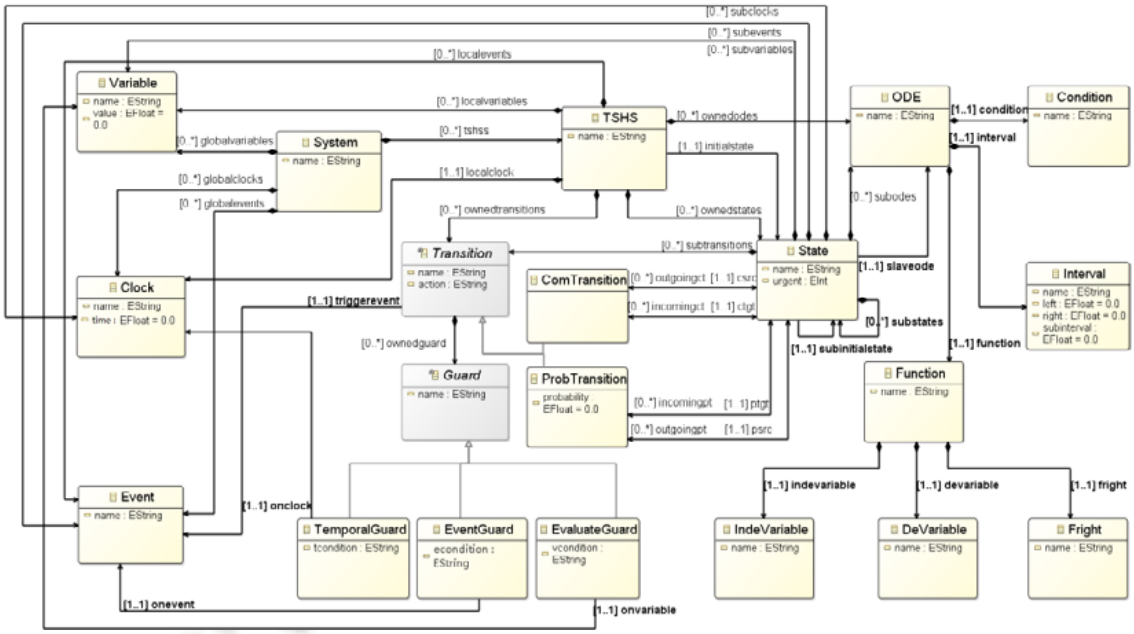

Fig.1 Metamodel of SHML for CPS

图 1 面向 CPS 的领域建模语言 SHML 的元模型

建模元素名称	具体图标表示	描述
State		使用蓝色圆角矩形表示 TSHS 的状态
ComTransition	→	使用带箭头实线表示普通迁移边
ProbTransition	⇢	使用带箭头虚线表示概率迁移边
slaveode	------	使用黑色虚线表示 State 与 ODE 的关联关系
ODE		使用折角矩形表示常微分方程
Function	function:	提示标签后跟函数名称
DeVariable	devariable:	提示标签后跟因变量名称
IndeVariable	indevariable:	提示标签够跟自变量名称
Fright	ydot=	提示标签后跟函数右式
Condition	condition:	提示标签后跟约束条件
Interval	interval	提示标签后跟求解区间
left	left:	提示标签后跟求解区间的左值
right	right:	提示标签后跟求解区间的右值
subinterval	subinterval:	提示标签后跟求解区间的子区间大小
Variable	Variable:	提示标签后跟物理变量
Clock	Clock:	提示标签后跟时钟
Event	e	带缩写 e 的图标后跟具体的 event
Guard	g	带缩写 g 的图标后跟具体的布尔表达式
action	a	带缩写 a 的图标后跟具体的 action
probability	P	带缩写 p 的图标后跟具体的概率值

Fig.2 Concret syntax of SHML

图 2 领域建模语言 SHML 的具体语法

除了定义建模元素的样式,具体语法还涉及定义建模元素的操作,例如创建实例、设置属性或者导航到另外一个模型图,这些操作为用户提供了便捷的建模方式.Sirius 技术提供一种具体语法的实现方法,用户可以使用画板创建实例模型,例如,可以在建模平台的画板上找到圆角矩形图标,然后把它拖到建模界面中,这样就创建了一个状态实例.此外,用户还可以为其设置属性和跳转操作,例如将它的名称设置为 State1,用户还可以双击 State1 导航到它的子图.总之,根据 SHML 的元模型,我们给出了该语言的具体语法表示形式,为用户提供了可视化的领域建模语言.此外,基于 Sirius 技术框架实现了 SHML 语言的具体语法表示,为用户提供了一个可视化的建模环境.

2.2 SHML的语义模型

SHML 的操作语义模型精确刻画了 SHML 模型的执行规则,为 SHML 实例模型的执行、仿真提供了语义支持,可以生成系统的执行迹,用于系统模型的验证、分析. SHML 模型可以使用一个五元组 $TSHS=(S,\rightarrow,L,s_0,O)$ 表示,其中,

- S 表示状态集合,其中:每个状态 $s(l,v,c,r)$,l 表示位置(location),v 表示变量,c 表示时钟,r 表示层次(r_0 表示父层,r_1 表示子层).此外,还有一些跟状态相关的符号表示,具体如下: S_{r_0} 表示父层 r_0 中状态的集合,S_{r_1} 表示子层 r_1 中状态的集合,$F(s)$表示状态 S 的父状态,c_s 表示复合状态 S 的子状态的集合,Is 表示复合状态 S 的初始子状态,$OT(s,t)$表示源状态 S 有一条出去的迁移边 t,$R(o,s)$表示状态 S 与一个 ODE o 相关联;
- $\rightarrow$:$State\times Labels\times State$ 表示源状态和目标状态之间的迁移.迁移有两种形式:普通迁移(CT)和概率迁移(PT).L 表示迁移的集合 $l=\{e,g,a,p\}$,其中,e 表示触发事件,g 表示卫士条件,a 表示动作,p 表示概率.此处,e 可以是某个外部触发事件,也可以是同步事件($e!$与 $e?$);
- s_0 表示 TSHS 的初始状态;
- o 表示常微分方程 ODE 的集合,集合中的每个 ODE 等式可表示为 $o=\{f,c,i\}$,其中,f 表示函数,c 表示约束条件,i 表示求解区间 $i=\{il,ir,step\}$,il 表示区间左值,ir 表示区间右值,$step$ 表示求解区间的长度.

语义上,SHML 模型是由一个或多个 TSHS 进程组成的,操作语义需要描述各个进程在当前状态下,当某个触发事件发生时,系统该做出何种反应.这里,重点讨论 4 种操作语义规则:调用常微分方程、同一层次的状态迁移、不同层的状态迁移以及迁移终止语义.

规则 1. 调用 ODE 等式:

$$\frac{s,\exists eT(e)\wedge(\nexists tOT(s,t)\wedge(v,c\vDash g))\wedge C_s==\varnothing\wedge\exists oR(s,o)\vdash v'=v+f(\Delta step)}{(l,v,c,r_0)\xrightarrow{e,g,a}(l,v',c',r_0)}\quad\text{(Rule CT1)}$$

$$\frac{s,\exists eT(e)\wedge(\nexists tOT(F(s),t)\wedge(v,c\vDash g))\wedge(\nexists tOT(s,t)\wedge(v,c\vDash g))\wedge\exists oR(s,o)\vdash v'=v+f(\Delta step)}{(l,v,c,r_1)\xrightarrow{e,g,a}(l,v',c',r_1)}\quad\text{(Rule CT2)}$$

$$\frac{s,\exists eT(e)\wedge(\nexists tOT(s,t)\wedge(v,c\vDash g))\wedge C_s==\varnothing\wedge\exists oR(s,o)\vdash v'=v+f(\Delta step)}{(l,v,c,r_0)\xrightarrow{e,g,a,p}(l,v',c',r_0)}\quad\text{(Rule PT1)}$$

$$\frac{s,\exists eT(e)\wedge(\nexists tOT(F(s),t)\wedge(v,c\vDash g))\wedge(\nexists tOT(s,t)\wedge(v,c\vDash g))\wedge\exists oR(s,o)\vdash v'=v+f(\Delta step)}{(l,v,c,r_1)\xrightarrow{e,g,a,p}(l,v',c',r_1)}\quad\text{(Rule PT2)}$$

规则 2. 同一层次的状态迁移:

$$\frac{s,\exists eT(e)\wedge(\exists tOT(s,t)\wedge(v,c\vDash g))\vdash s'\in S_{r_0}\wedge C_{s'}==\varnothing}{(l,v,c,r_0)\xrightarrow{e,g,a}(l',v',c',r_0)}\quad\text{(Rule CT3)}$$

$$\frac{s,\exists eT(e)\wedge(\exists tOT(s,t)\wedge(v,c\vDash g))\vdash s'\in S_{r_0}\wedge C_{s'}\neq\varnothing\ s',\exists e'T(e')\wedge(\exists tOT(s',t)\wedge(v',c'\vDash g'))\vdash s''\in S_{r_0}}{(l,v,c,r_0)\xrightarrow{e,g,a}(l'',v'',c'',r_0)}\quad\text{(Rule CT4)}$$

$$\frac{s,\exists eT(e)\wedge(\nexists tOT(F(s),t)\wedge(v,c\vDash g))\wedge(\exists tOT(s,t)\wedge(v,c\vDash g)\vdash s'\in S_{r_1}}{(l,v,c,r_1)\xrightarrow{e,g,a}(l',v',c',r_1)}\quad\text{(Rule CT5)}$$

$$\frac{s,\exists eT(e)\wedge(\exists tOT(s,t)\wedge(v,c\vDash g))\vdash s'\in S_{r_0}\wedge C_{s'}==\varnothing}{(l,v,c,r_0)\xrightarrow{e,g,a,p}(l',v',c',r_0)}\quad\text{(Rule PT3)}$$

$$\frac{s,\exists eT(e)\wedge(\exists tOT(s,t)\wedge(v,c\vDash g))\vdash s'\in S_{r_0}\wedge C_{s'}\neq\varnothing\ s',\exists e'T(e')\wedge(\exists tOT(s',t)\wedge(v',c'\vDash g'))\vdash s''\in S_{r_0}}{(l,v,c,r_0)\xrightarrow{e,g,a,p}(l'',v'',c'',r_0)}\quad\text{(Rule PT4)}$$

$$\frac{s,\exists eT(e)\wedge(\nexists tOT(F(s),t)\wedge(v,c\vDash g))\wedge(\exists tOT(s,t)\wedge(v,c\vDash g))\vdash s'\in S_{r_1}}{(l,v,c,r_1)\xrightarrow{e,g,a,p}(l',v',c',r_1)}\quad\text{(Rule PT5)}$$

规则 3. 不同层次的状态迁移:

$$\frac{s,\exists eT(e)\wedge(\exists tOT(s,t)\wedge(v,c\vDash g))\vdash s'\in S_{r_0}\wedge C_{s'}\neq\varnothing\wedge(\nexists tOT(s',t)\wedge(v',c'\vDash g')),s'=I_{s'}}{(l,v,c,r_0)\xrightarrow{e,g,a}(l',v',c',r_1)}\quad\text{(Rule CT6)}$$

$$\frac{s,\exists eT(e)\wedge(\exists tOT(F(s),t)\wedge(v,c\vDash g))\vdash s'\in S_{r_0}}{(l,v,c,r_1)\xrightarrow{e,g,a}(l',v',c',r_0)}\quad\text{(Rule CT7)}$$

$$\frac{s,\exists eT(e)\wedge(\exists tOT(s,t)\wedge(v,c\vDash g))\vdash s'\in S_{r_0}\wedge C_{s'}\neq\varnothing\wedge(\nexists tOT(s',t)\wedge(v',c'\vDash g')),s'=I_{s'}}{(l,v,c,r_0)\xrightarrow{e,g,a,p}(l',v',c',r_1)}\quad\text{(Rule PT6)}$$

$$\frac{s,\exists eT(e)\wedge(\exists tOT(F(s),t)\wedge(v,c\vDash g))\vdash s'\in S_{r_0}}{(l,v,c,r_1)\xrightarrow{e,g,a,p}(l',v',c',r_0)}\quad\text{(Rule PT7)}$$

规则 4. 终止执行:

$$\frac{s,\exists eT(e)\wedge(\nexists tOT(s,t)\wedge(v,c\vDash g))\wedge(C_s==\varnothing)\wedge\nexists oR(s,o)\vdash s'=s}{(l,v,c,r_0)\xrightarrow{e,g,a}(l,v,c,r_0)}\quad\text{(Rule CT8)}$$

$$\frac{s,\exists eT(e)\wedge(\nexists tOT(F(s),t)\wedge(v,c\vDash g))\wedge(\nexists tOT(s,t)\wedge(v,c\vDash g))\wedge\nexists oR(s,o)\vdash s'=s}{(l,v,c,r_1)\xrightarrow{e,g,a}(l,v,c,r_1)}\quad\text{(Rule CT9)}$$

$$\frac{s,\exists eT(e)\wedge(\nexists tOT(s,t)\wedge(v,c\vDash g))\wedge(C_s==\varnothing)\wedge\nexists oR(s,o)\vdash s'=s}{(l,v,c,r_0)\xrightarrow{e,g,a,p}(l,v,c,r_0)}\quad\text{(Rule PT8)}$$

接下来,针对每一类操作语义规则,给出详细的解释说明.

(1) 规则 1:调用 ODE 等式

规则 CT1 表示 TSHS 的当前状态 $s(l,v,c,r_0)$,当某个触发事件发生时(可以是某个判定式为真也可以是发生了同步事件),状态 s 处没有迁移可以满足约束条件$(v,c\vDash g)$使其被触发,同时,状态 s 子状态集合为空,此时,s 也不能跳转到它的子层初始状态.但是,状态 s 与一个 OED o 关联,这意味着 o 能改变 v 的值,使 v 变成 $v'=v+f(\Delta step)$,其中,f 是 o 的函数名称,$\Delta step$ 是 ODE o 的子区间长度.Scilab ODE 求解器会根据用户设定的子区间长度,每$\Delta step$ 步长更新变量值 v',而一旦新的变量值满足状态 s 处的约束条件,状态 s 就会迁移到下一个状态 $s'(l,v',c',r_0)$.规则 CT2 表示 TSHS 当前的状态 s 处在某个状态的子层 r_0 中,当某个触发事件发生时,状态 s 及其父状态 $F(s)$都没有可以被触发的迁移,但是状态 s 一个 OED o 关联,变量 v 随着 ODE o 变化,使得 v'在某一时刻满足约束条件,状态 s 便可以迁移到下一个状态 s'.规则 PT1,PT2 和规则 CT1,CT2 的描述是类似的,不同之处在于使能迁移边上带有概率 p,表明在相同条件下,当前状态处迁移使能的概率是 p.

(2) 规则 2:SHML 中同一层次的状态迁移

规则 CT3 表示若当前状态 s 在父层中,当某个触发事件 e 发生时,它能触发使能迁移,实现状态迁移到同层中的另一个非嵌套状态 s'.规则 CT4 表示若当前状态 s 处在父层中,当事件 e 触发某个使能迁移,状态 s 能够迁移到下一层的父状态 s',这时首先会判断在父层中 s'是否存在一条出边可以立即被触发,若存在这样的使能迁移,那么 s'会跳转到下一个在父层中的状态 s'.规则 CT5 表示:当前状态 s 处在某个父状态的子层中,它的父状态 $F(s)$.没有出边可以被触发,但是其子状态处,存在某条出边可以被触发,这时迁移会跳转到同一子层的下一个状态 s'.规则 PT3~PT5 与规则 CT3~CT5 的描述是类似的,只是当前状态 S 的使能迁移具有概率值 p,表示迁移发生的概率.

(3) 规则 3:SHML 中不同层次的状态迁移

规则 CT6 表示当前状态 s 处,当某个触发事件发生时,状态 s 能迁移到其子层 r_0 中的一个状态 s'.这里,状态

s'不能直接跳转到 r_0 中的另一个状态.s'设置为它的子层的初始状态 Is'.规则 CT7 表示当前状态 S 处在 r_1,这里首先对其父状态 $F(s)$中检查,判断其有没有迁移可以被触发.若 $F(s)$中某一条迁移边被触发,那么状态 S 会跳转 r_0 中的某一个状态,实现从父状态层迁移到子状态层.规则 PT6,PT7 与规则 CT6,CT7 的描述是类似的,只在相同条件下,当前状态 S 处存在多条使能边可以被触发,迁移被触发的概率值为 p.

(4) 规则 4:终止执行

CT8 表示如果父层中的状态 s 处,当某个触发事件发生时,没有使能的迁移,也没有一个关联的 ODE 等式.这意味着当前状态 s 是一个终止状态.规则 CT9 表示如果当前状态处在子层 r_1 中,当某个触发事件发生时,s 及它的父状态 $F(s)$都没有使能迁移,状态处也没有 ODE 等式,则当前状态 s 是处于子层的终止状态.规则 PT8,PT9 的描述与 CT8,CT9 的描述类似,表示当前状态 s 处存在多条使能的迁移,而迁移使能的概率为 p.

以上操作语义规则为模型的仿真执行奠定了语义基础.我们将根据这些语义规则,在 GEMOC 执行引擎中实现 SHML 的语义模型,以支持 SHML 模型的仿真执行.

2.3 基于GEMOC实现SHML语言

GEMOC 建模框架为领域建模语言定义及执行提供了一系列的技术支撑,为设计、实现面向 CPS 领域的随机混成建模语言 SHML 提供了一种有效的途径.我们提出基于 GEMOC 实现 SHML,以图形化的方式建模 CPS 的动态行为,包括混成行为、随机行为,为 CPS 的领域建模提供工具支撑.本节重点讨论如何基于 GEMOC 框架实现 SHML 语言,包括抽象语法、具体语法及执行语义.

基于 GEMOC 实现 SHML 的技术框架如图 3 所示.

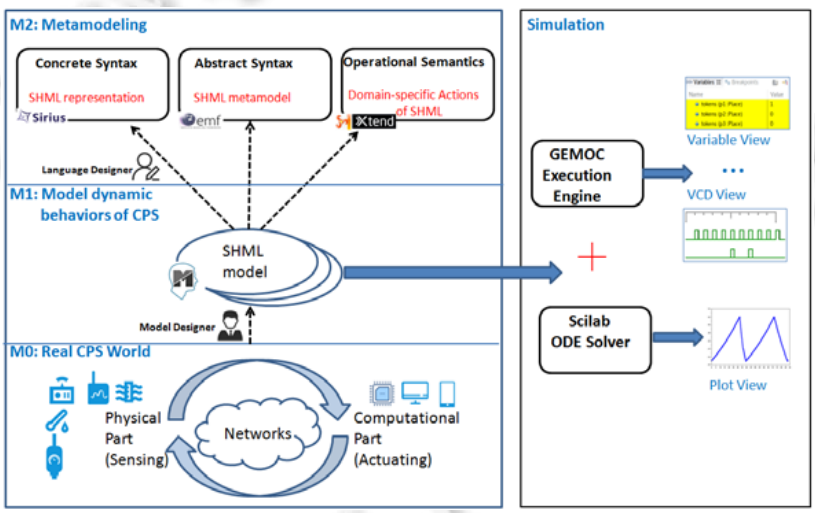

Fig.3 Technical framework for SHML based on GEMOC

图 3 基于 GEMOC 实现 SHML 的技术框架

图中左半部分是基于元建模方法的 CPS 建模层次框架,其中,SHML 语言定义处于最高抽象层次,包括抽象语法和具体语法.首先,根据领域特定建模语言 SHML 的抽象语法即元模型,使用 EMF 框架提供的 Ecore 元语言,实现 SHML 的元模型定义.在 SHML 元模型中,详细描述了建模 CPS 行为模型所需要的领域概念及这些概念间的关系.其次,根据 SHML 的抽象语法,使用 Sirius 实现其图形化的语法表示,并使用 Xtext 实现其文本化的表示.图形化的模型为用户提供了可视化的建模方式,更加便于用户理解、交流,而采用文本语法的模型通常占用相对较少的存储空间,易于计算机对其进行查询、定位.基于 GEMOC 框架,我们实现了 SHML 语言的图形及文本表示形式,可以灵活建模 CPS 模型,为以模型驱动的方式开发 CPS 系统提供了建模支持.建模语言的操作语义为模型的模拟、仿真提供了丰富的语义信息.通常,操作语义的定义会涉及两个重要的对象:运行时数据和执行函数.运行时数据反映的是当前模型的信息,例如状态、变量等;而执行函数则是修改当前模型信息的具体操作,例

如执行状态迁移、调用 ODE 等式.本质上,这些是与第 2.2 节中定义的操作语义规则是对应的.基于 GEMOC 框架,我们使用 Kermeta3[15]和 Xtend 语言,根据 SHML 的操作语义模型对 SHML 元模型进行语义编织,最终实现 SHML 的操作语义.

图中右半部分是 SHML 的模型执行引擎,根据本文提出的方法,我们集成了 GEMOC 的执行引擎与 Scilab 的执行引擎,实现仿真 CPS 系统的混成行为,并根据 SHML 的语义模型实现模型语义执行规则.总之,SHML 的模型执行引擎,为仿真 CPS 的混成行为提供了工具支撑.

3 SHML 模型的仿真

仿真通过模型的模拟执行来帮助我们进一步分析系统的行为,发现模型中的错误之处.CPS 的行为仿真是分析 CPS 模型正确性的有效技术,其难点在于如何仿真系统的混成行为.现有的仿真工具主要针对某个特定领域开发的,擅长仿真系统离散行为,而缺乏对系统混成行为的支持.目前,GEMOC 执行引擎也只能支持具有离散语义模型的仿真,无法直接实现仿真 SHML 模型的混成行为,需要对其进行扩展,使其不仅能支持离散行为的仿真,而且能支持连续行为的仿真.因此,本文提出将 Scilab 插件集成到 GEMOC 中,实现在 GEMOC 建模平台下调用 Scilab 求解器,进而实现仿真系统的连续行为.该方法将 GEMOC 的序列化执行引擎和 Scilab ODE 求解器结合,实现了仿真系统的混成行为,解决了仿真 CPS 混成行为的难题.

GEMOC 提供了一个通用的执行引擎,负责将 DSML 与 GEMOC 中提供的插件集成起来,通过一系列操作(例如模型初始化、开始执行某条语义规则)以达到控制模型、修改模型的效果.同时,该框架还集成了一些以插件形式存在的运行时服务,例如逐步调试、图形动画模拟、执行路径跟踪服务等.此外,GEMOC 提供了一个通用接口,用以集成面向某些特定领域的执行引擎,而这些不同的执行引擎往往与定义了离散操作语义的 DSML 相关联.最终,以插件形式存在的通用运行时服务和执行引擎一起实现模型的仿真、模拟.这些技术特性,为集成 Scilab 的求解器提供了技术支持.

在集成 Scilab 的过程中,需要解决的主要问题是如何在 Eclipse 插件模式下调用 Scilab.Scilab 提供的相关 jar 包并不是 Eclipse 标准插件,无法将其直接放入 GEMOC 的插件库,即无法实现在建模平台中直接调用 Scilab.因此,我们仔细分析了 Scilab 的调用方式及相关 jar 包之间的依赖关系,设计、实现了 3 个与 Scilab 调用相关的新 jar 包:Javasci,Jvm,Types,以标准的 Eclipse 插件的形式化集成到现有的 GEMOC 插件库.这 3 个 jar 包主要负责 Scilab 调用、数据处理,帮助用户直接调用 Scilab ODE 求解器.图 4 是集成了 Scilab 插件之后的仿真平台的 GUI,工具界面包括实例模型调试、变量框观测、默认信息输出框, 在模型执行时这些信息不断动态变化,帮助我们分析理解模型的行为.图 4 的右下角“Scilab 输出框”是新集成的 Scilab 模拟输出界面,以图形化的形式展示物理变量的连续变化过程,具有直观、易于理解的特点.

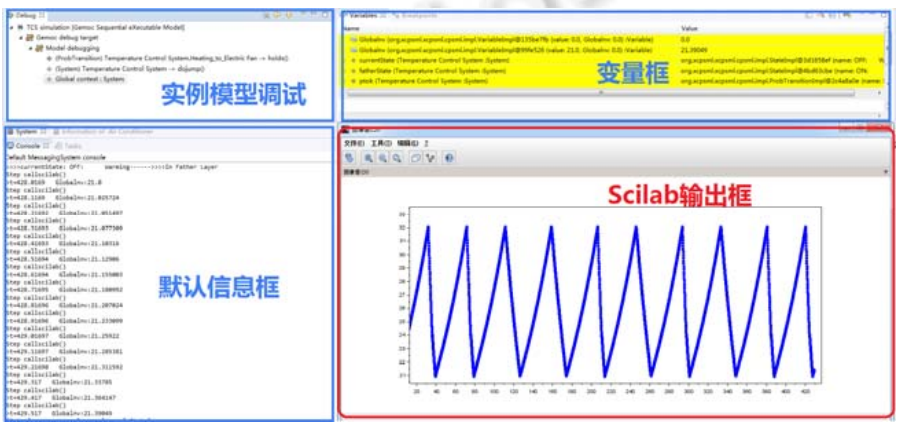

Fig.4 Simulation platform for SHML models

图 4 SHML 的仿真平台

总之,基于 GEMOC 的仿真执行引擎,并集成 Scilab 为 CPS 系统模型的仿真执行提供了有效的方式,所生成的仿真执行结果能帮助用户进行系统行为的分析.接下来,我们将通过实际的案例展示本文所提方法的可用性.

4 案例分析

温控系统是一个经典的 CPS 系统,空调、风扇等物理设备控制温度的变化,同时,温度控制器实现开关的切换,温度的变化也会反过来影响控制器的决定,二者处在一个反馈回路之中.在炎热的夏日,某房间(初始温度为 21°C)有一个控制器,连接了一台电扇和一台空调,外界高温会逐渐影响房间温度使其不断升高.人体感到舒适的温度区间为 21°C~32°C.当房间温度大于 32°C 时,控制器选择一种制冷设备进行制冷,选择电扇的概率为 0.4,选择空调的概率为 0.6.此外,假定空调具有强档制冷和弱档制冷,不同档位下的制冷功率不同.假设空调首先会进行强档降温,当温度低于 24°C 时,空调会跳转到弱档继续降温.当温度降到 21°C 以下时,控制器会关闭制冷设备.

图 5 是基于 GEMOC 建模平台构建的温控系统的 SHML 模型,主要建模了温度控制器的动态行为,包括随机行为和混成行为.温度控制器的 SHML 模型包含嵌套状态,在父层中有 3 个状态:OFFWarming 表示房间处于升温状态,ONElectricFan 表示电扇降温状态,ONAirConditioner 表示空调降温状态.其中,OFFWarming 状态使用加粗框表示,表明它是父层的初始状态,该状态下制冷设备是关闭的,此状态处的温度上升过程由 Warming_ODE 表示.从 OFF Warming 出发有两条概率迁移,分别指向 ONElectricFan 状态和 ON AirConditioner 状态,一旦检测到房间温度超过了 32°C,系统状态将由当前状态 OFF Warming 迁移到 ONElectricFan 或者 ON AirConditioner,两条虚线迁移边上都带有概率值标记,表示相同条件下该迁移边使能的概率.

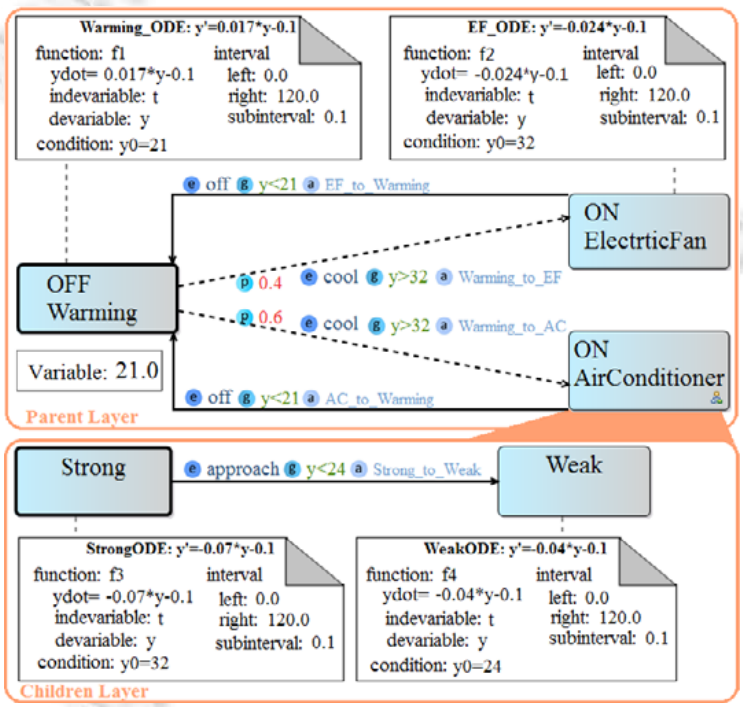

Fig.5 SHML models for the temperature controller

图 5 温度控制器的 SHML 模型

类似地,电扇制冷状态处,温度下降的动态变化过程由 EF_ODE 表示.空调制冷状态较复杂,可使用嵌套状态表示.在子层中,初始状态为 Strong 状态,它会在温度低于 24°C 时跳转到 Weak 状态,这两个状态也都关联了各自的 ODE 刻画温度下降的过程.基于我们实现的 GEMOC 建模平台构建的温度控制器的 SHML 模型,以可视化的

方式建模了系统行为导致的复杂状态变迁.进一步地,我们可以基于本文设计、实现的仿真器对温度控制器的 SHML 模型进行模拟、仿真,可用于分析系统行为的正确性.

图 6 是温度控制器 SHML 模型的仿真结果.菱形的线段表示当控制器处于 OFFWarming 状态时,温度上升的过程;三角形的线段表示空调强制冷状态下的温度下降过程;带实心圆的线段表示空调弱制冷状态下的温度下降过程;线段表示电扇制冷状态下的温度下降过程.如图所示:房间的初始温为 21°C,房间温度不断上升,超过临界值时,控制器打开空调降温,在空调强制冷状态下,房间温度开始下降,在某一时刻,当房间温度为 23.849°C,已经低于 24°C 了,这时,空调切换到弱制冷状态.与三角形线段相比,实心圆线段稍微平缓一些,说明空调弱制冷状态的降温速率相对较小.在空调弱制冷状态下,房间温度继续降低,当房间温度低于 21°C 时,状态迁移到 OFFWarming 状态.总之,在该案例中,温度的变化遵循 ODE 等式规约的物理变化过程,同时,控制器根据温度进行开关的切换选择.温控系统是典型的混成系统,包括离散的控制行为与连续的温度变化,借助我们实现的混成行为的仿真引擎,实现了仿真 SHML 建模的混成行为,为 CPS 混成行为的验证、分析提供了一种有效的途径.

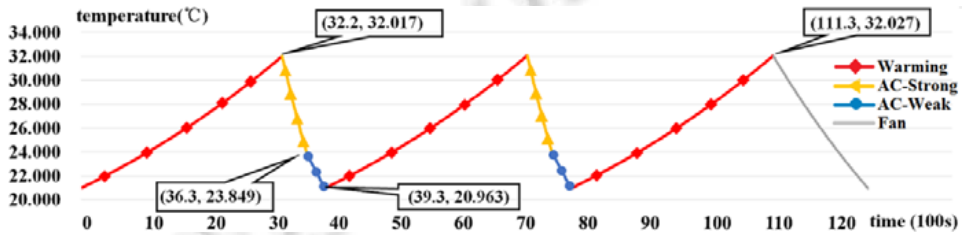

Fig.6 Simulation result for the temperature

图 6 温度变化的仿真结果

5 相关工作分析与比较

由于 CPS 融合了物理和信息世界,其系统行为包括离散的控制行为和连续的物理变化.建模、仿真 CPS 系统行为一直面临着诸多困难和挑战.CPS 的建模与验证分析也逐步引起工业界和学术界的广泛关注,取得了初步的研究成果.Derler 等人在文献[16]中指出,CPS 具有异构性、并发性及对时间的敏感性等特性,导致 CPS 建模方法具有挑战性,并以车辆燃油子系统为例,说明如何建模 CPS 的混成行为.该项工作的侧重点是提出基于混成自动机建模 CPS 的混成行为.

由 UC Berkeley 开发的 Ptolemy II 是一款开源的面向 CPS 的建模和仿真工具[17],可用于建模、仿真实时嵌入式系统,其执行语义依赖于角色模型(actor model)中的指示器(director).Ptolemy II 能够建模异构系统,但是模型的执行受限于工具本身提供的计算模型(model of computation,简称 MoC),当模型语义不能找到合适计算模型来表示时,系统设计人员只能通过修改工具的核心部分,以满足实际系统设计的需求.此外,Matlab/Simulink 支持对动态系统进行建模、仿真和分析,被广泛应用于工业级的案例.它为用户提供了用方框图建模的图形化接口,与传统的软件包用微分方程或差分方程建模相比,有更直观、方便、灵活的优点.Matlab/Simulink 的最新版本支持建模 CPS 的计算模型,此外,可以使用 Simscape 建模物理模型,例如,使用图形化的状态机建模智能应急响应系统、智能制造机器人等领域,并可以对模型进行仿真、分析.通过仿真,系统设计人员可以更好地观测系统变量、理解系统行为、修复系统模型的错误.

基于标准建模语言例如 MARTE/SysML[18,19]建模信息物理融合系统的相关工作也取得了初步进展.文献[20]提出将 OMG 的建模语言 MARTE 和 SysML 结合起来建模 CPS.Mallet F 在文献[21]中,从异构性、平台感知、资源受限、非功能属性等角度总结了 CPS 的特性,给出了一些使用 MARTE 和 SysML 建模 CPS 的示例,并提出使用时钟约束规范语言(clock constraint specification language,简称 CCSL)描述逻辑时钟之间的约束关系.此外,仿真工具 Timesquare[22]能支持 CCSL 模型的仿真、模拟.在文献[23]中,Gemoz 提出一种基于 MARTE 和 SysML 的多视图建模方法,并结合能量消耗的案例展示了如何基于标准建模语言建模 CPS.

罗晨霞等人提出了面向实时数据的 CPS 一体化建模方法[24],通过定义一系列的规则,将领域环境模型组合

到运行时验证的过程中去,从而保证 CPS 在不确定环境中的安全性和可靠性.此方法使得监视模型更加完整准确,当环境发生变化时,通过动态调整参数范围,使得 CPS 的安全属性在复杂的物理环境中仍然满足.此外,我们前期的工作针对 CPS 的混成、随机行为[25]提出了概率时钟约束语言 pCCSL 用于规约 CPS 的随机行为,并结合 MARTE/SysML 建模 CPS 的混成、随机行为,该方法能够有效建模 CPS 的混成、随机行为[26,27].

综上所述,CPS 建模与仿真的研究工作已经取得了初步的进展,能够针对 CPS 的不同特性提出建模、仿真方法,相关的建模及仿真工具的设计、开发也引起了人们的广泛关注.与已有研究工作不同,我们从领域建模语言设计的角度出发,基于模型驱动开发方法的技术框架,针对 CPS 的随机、混成行为,设计、实现了一种面向 CPS 的领域建模语言 SHML,旨在为 CPS 的领域建模提供一种可视化的建模语言.SHML 的主要特点是能够以可视化的方式建模 CPS 系统的复杂行为,包括混成行为、随机行为.此外,在工具实现方面,我们设计、实现了基于 GEMOC 框架的建模及仿真工具,为 SHML 的建模及仿真提供工具支撑,为 CPS 的自动化设计及实现,提供了一种有效的途径.

6 总结与展望

信息物理融合系统具有异构性、混成性、随机性等特点,建模、仿真 CPS 的行为对于开发高质量的 CPS 系统具有重要意义.领域建模方法为以模型驱动的方式开发 CPS 系统提供了一种有效的途径,为 CPS 的自动化设计提供了可能.本文提出一种面向 CPS 的领域建模语言 SHML,能够支持建模 CPS 的混成行为、随机行为.根据领域语言的特征,我们定义了 SHML 的语法及语义模型,并基于 GEMOC 框架实现了 SHML 语言的建模环境.设计人员能够基于 GEMOC 的建模平台,使用 SHML 构建 CPS 的领域模型.此外,为了分析模型的动态执行过程,生成系统的执行迹,本文提出基于 GEMOC 的序列化执行框架,集成 Scilab ODE 求解器,实现仿真 CPS 的混成行为.本文提出的面向 CPS 的领域建模及仿真方法具有一定的通用性和可扩展性,设计、实现的工具平台为设计人员提供了一个集成的面向 CPS 的建模与仿真环境,为 CPS 的自动化设计提供了技术支撑.

最后,结合经典的 CPS 案例——温控系统,进一步展示了 SHML 的建模能力,并给出了温度变化的模拟结果.实际应用表明,我们提出的领域建模语言及建模、仿真工具能够有效建模、仿真 CPS 的动态行为.在未来的工作中,将继续改进领域建模语言 SHML,使其能够支持建模 CPS 的运行环境模型、系统的架构模型等,并设计、开发协同仿真算法,使得模拟引擎能够支持更多种类的计算模型的模拟与仿真.

References:

[1] Lee EA. System Design, Modeling, and Simulation using Ptolemy II. Beijing: China Machine Press, 2017. (in Chinese).

[2] Schmidt DC. Guest editor's introduction: Model-driven engineering. Computer, 2006,39(2):25−31.

[3] Bézivin J. Models everywhere. In: Proc. of the TOOLS USA 2001: Software Technologies for the Age of the Internet, 39th Int'l Conference & Exhibition. 2001. 348−349.

[4] Jumagaliyev A, Whittle J, Elkhatib Y. Using DSML for handling multi-tenant evolution in cloud applications. In: Proc. of the 2017 IEEE Int'l Conf. on Cloud Computing Technology and Science (CloudCom). IEEE, 2017. 272−279.

[5] Laurenzi E, Hinkelmann K, Reimer U, Van Der Merwe A, Sibold P, Endl R. DSML4PTM: A domain-specific modelling language for patient transferal management. In: Proc. of the 1st Int'l Workshop on Advanced Enterprise Modelling (AEM 2017), Conjunction with the 19th Int'l Conf. on Enterprise Information Systems (ICEIS 2017). 2017.

[6] OMG. OMG Unified Modeling Language (OMG UML) Infrastructure. formal/2010-05-03.

[7] Reedy J, Lunzman S. Simulation: Model-based design accelerates development of mechanical locomotive controls. In: Proc. of the SAE Off-Highway Engineering. 2011.

[8] OMG. MDA Guide Version 1.0.1. formal/2010-05-01.

[9] Kelly S, Tolvanen JP. Domain-Specific Modeling: Enabling Full Code Generation. ISBN 978-0-470-03666-2. New Jersey: John Wiley & Sons, 2008.

[10] OMG. OMG Meta-Object Facility (MOF) Core Specification. Version 2.4.2/2014-02-17.

[11] Combemale B, BaraisO, Wortmann A. Language engineering with the GEMOC studio. In: Proc. of the 2017 IEEE Int'l Conf. on Software Architecture Workshops (ICSAW). 2017. 189−191.

[12] Bousse E, Degueule T, Vojtisek D, Mayerhofer T, Deantoni J, Combemale B. Execution framework of the GEMOC studio (tool demo). In: Proc. of the 2016 ACM SIGPLAN Int'l Conf. on Software Language Engineering. Amsterdam, 2016. 84−89.

[13] Wu B, Bogaerts AJ. SCILab—A simulation environment for the scalable coherent interface. In: Proc. of the 3rd Int'l Workshop on Modeling, Analysis, and Simulation on Computer and Telecommunication Systems (MASCOTS'95). 1995. 242–247.

[14] Huang D, Wang F, Li ZW. Scilab Basic Course of Free Software for Scientific Computing. Beijing: Tsinghua University Press, 2006. (in Chinese).

[15] Jézéquel J, Combemale B, Barais O, Monperrus M, Fouquet F. Mashup of metalanguages and its implementation in the kermeta language workbench. Software & Systems Modeling Volume, 2015. 905–920.

[16] Derler P, Lee EA, Sangiovanni-Vincentelli AL. Modeling Cyber-Physical systems. Proc. of the IEEE, 2012,100(1):13–28.

[17] Buck JT, Ha S, Lee EA, Messerschmitt DG. Ptolemy: A framework for simulating and prototyping heterogenous systems. Int'l Journal in Computer Simulation, 2001. 527–543.

[18] OMG. OMG System Modeling Language Specification. formal/2017-05-01.

[19] OMG. UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems. formal/2011-06-02.

[20] Espinoza H, Cancila D, Selic B, Gérard S. Challenges in combining SysML and MARTE for model-based design of embedded systems. In: Proc. of the 5th European Conf. on Model DrivenArchitecture—Foundations and Applications (ECMDAFA 2009). 2009. 98–113.

[21] Mallet F. MARTE/CCSL for modeling cyber-physical systems. In: Proc. of the Formal Modeling and Verification of Cyber-Physical Systems, 1st Int'l Summer School on Methods and Tools for the Design of Digital Systems. 2015. 26–49.

[22] Deantoni J, Mallet F. TimeSquare: Treat your models with logical time. In: Proc. of the 50th Int'l Conf. on Objects, Models, Components, Patterns (TOOLS 2012). 2012. 34–41.

[23] Gomez C, Deantoni J, Mallet F. Power consumption analysis using Multiview modeling. In: Proc. of the Int'l Workshop on Power and Timing Modeling. 2013.

[24] Luo CX, Wang R, Guan Y, Li XJ, Shi ZP, Song XY. Integrated modeling method of CPS for real-time data. Ruan Jian Xue Bao/Journal of Software, 2019,30(7):1966–1979 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5753.htm [doi: 10.13328/j.cnki.jos.005753]

[25] Du DH, Guo T, Wang Y. SHML: Stochastic hybrid modeling language for CPS behavior. In: Proc. of the 26th Asia-Pacific Software Engineering Conf. (APSEC). IEEE, 2019. 220–227.

[26] Du DH, Huang P, Jiang KQ, Mallet F. pCSSL: A stochastic extension to MARTE/CCSL for modeling uncertainty in cyber physical systems. Science of Computer Programming, 2018,166:71–88.

[27] Du DH, Huang P, Jiang KQ, Mallet F, Yang MR. MARTE/pCCSL: Modeling and refining stochastic behaviors of CPSs with probabilistic logical clocks. In: Proc. of the Int'l Workshop on Formal Aspects of Component Software. 2016. 111–133.

附中文参考文献:

[1] Lee EA.信息物理融合系统(CPS)设计、建模与仿真——基于 Ptolemy II 平台.北京:机械工业出版社,2017.1–3.

[14] 黄铎,王风,李志伟.科学计算自由软件 SCILAB 基础教程.北京:清华大学出版社,2006.

[24] 罗晨霞,王瑞,关永,李晓娟,施智平,Song XY.面向实时数据的CPS一体化建模方法.软件学报,2019,30(7):1966–1979. http://www.jos.org.cn/1000-9825/5753.htm [doi: 10.13328/j.cnki.jos.005753]

杜德慧(1979－),女,河南信阳人,博士,教授,CCF 专业会员,主要研究领域为可信软件,信息物理融合系统建模与验证.

王耀(1995－),男,学士,CCF 学生会员,主要研究领域为可信软件,信息物理融合系统.

管春琳(1994－),女,硕士,主要研究领域为可信软件,CPS 建模、仿真、验证.

郭童(1996－),女,学士,CCF 学生会员,主要研究领域为可信软件,信息物理融合系统.