软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2019,30(7):1966−1979 [doi: 10.13328/j.cnki.jos.005753]

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

面向实时数据的 CPS 一体化建模方法*

罗晨霞[1], 王 瑞[1], 关 永[2], 李晓娟[1], 施智平[1], Xiaoyu SONG[3]

[1](轻型工业机器人与安全验证北京市重点实验室(首都师范大学 信息工程学院),北京 100048)

[2](电子系统可靠性与数理交叉学科国家国际科技合作示范型基地(首都师范大学),北京 100048)

[3](Portland State University, Portland 97207, OR 97207, USA)

通讯作者: 王瑞, E-mail: rwang04@cnu.edu.cn

摘 要: 信息物理系统(cyber-physical system,简称 CPS)是一个在环境感知的基础上整合了物理和计算元素的系统,它可以智能地响应真实世界的动态变化,具有重要而广阔的应用前景.然而,CPS 工作在复杂的物理环境中,周围的物理变化会对CPS的行为产生影响.因此,确保CPS在复杂环境中的安全性和可靠性至关重要.提出了一种面向实时数据的一体化建模方法,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过程中去,从而保证CPS在不确定环境中的安全性和可靠性.该方法首先为环境建立数学模型.然后,设计合并规则将相同系统参数下仅有一个环境影响因子的数学模型合并为相同系统参数下有一个或多个环境影响因子的数学模型.之后,定义转换规则,将数学模型转换为伪代码表示的环境模型.最后,根据组合规则将环境模型组合到运行时监视模型中执行验证.该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 中的安全属性在复杂的物理环境中仍然得以满足.将该方法应用到移动机器人避障实验中,对影响电池容量的温度和湿度进行数学建模,然后将环境模型组合到监视模型中去,最终实现在执行任务前可以根据不同的物理环境准确地给出续航时间安全提醒.

关键词: 运行时验证;实时性;CPS;环境建模;安全性

中图法分类号: TP311

中文引用格式: 罗晨霞,王瑞,关永,李晓娟,施智平,Xiaoyu Song.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

英文引用格式: 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). http://www.jos.org.cn/1000-9825/5753.htm

Integrated Modeling Method of CPS for Real-time Data

LUO Chen-Xia[1], WANG Rui[1], GUAN Yong[2], LI Xiao-Juan[1], SHI Zhi-Ping[1], Xiaoyu SONG[3]

[1](Beijing Key Laboratory of Light Industrial Robot and Safety Verification (College of Information Engineering, Capital Normal University), Beijing 100048, China)

[2](National International Science and Technology Cooperation Demonstration Base of Interdisciplinary of Electronic System Reliability and Mathematics (Capital Normal University), Beijing 100048, China)

[3](Portland State University, Portland 97207, OR 97207, USA)

Abstract: Cyber-physical systems (CPS) is a system that integrates physical and computational elements based on context-awareness. It can intelligently respond to dynamic changes in the real world and has important and broad application prospects. However, CPS works in

* 基金项目: 国家自然科学基金(61877040, 61702348, 61602325); 国家重点研发计划(2017YFB1303000)

Foundation item: National Natural Science Foundation of China (61877040, 61702348, 61602325); National Key R&D Plan of China (2017YFB1303000)

本文由“软件形式化验证”专题特约编辑贺飞副教授、张立军研究员推荐.

收稿时间: 2018-07-15; 修改时间: 2018-09-28; 采用时间: 2018-12-13; jos 在线出版时间: 2019-03-28

CNKI 网络优先出版: 2019-03-29 09:14:19, http://kns.cnki.net/kcms/detail/11.2560.TP.20190329.0914.005.html

a complex physical environment, and changes around it can affect the behavior of CPS. Therefore, ensuring the safety and reliability of CPS in complex environments is critical. This study proposes an integrated modeling method for real-time data. By defining a series of rules, the domain environment model is combined into the runtime verification process to ensure the safety and reliability of CPS in an uncertain environment. First, the method builds a mathematical model for the environment. Then, design the merge rules to merge the mathematical models with only one environmental factor under the same system parameter into a mathematical model with one or more environmental factors under the same system parameter. Next, the transformation rules are defined to convert the mathematical model into an environment model represented by pseudocode. Finally, the environment model is combined into the runtime monitoring model to perform verification according to the combination rules. The method makes the verification process more complete and accurate. When the environment changes, it ensures that the safety properties in the CPS are still satisfied by dynamically adjusting the parameter range. Finally, the method is applied to the mobile robot obstacle avoidance experiment, to model the temperature and humidity physical environment and then, to combine it into the monitoring model, eventually, the life time safety reminder is accurately given in different environments.

Key words: runtime verification; real-time; CPS; environment modeling; security

CPS 是一个在环境感知的基础上整合了物理和计算元素的系统,可以智能地响应真实世界场景的动态变化.它通过计算进程和物理进程相互影响的反馈循环实现深度融合和实时交互增加或扩展新的功能,以安全、可靠、高效和实时的方式监测或是控制一个物理实体[1].从 2006 年 2 月《美国竞争力计划》将 CPS 列为重要的研究项目后,信息物理系统取得了很好的研究和发展[2].直至现在,CPS 已经涵盖了小到智能家庭网络,大到工业控制系统和智能交通系统,渗透到了航空、汽车、能源、医疗卫生和物流等多个领域.随着 CPS 的广泛应用,它的安全性问题也越来越得到关注.其中,由于 CPS 在真实的物理环境中工作,各种环境因素对于 CPS 的影响是实际存在且不容忽视的,它们可能会导致 CPS 从传感器中获得的数据与实际环境不符,从而使得 CPS 做出错误的或是危险的行为[3−5].因此,在对 CPS 的安全性和可靠性研究中考虑物理环境的影响是非常有必要的.

保证系统安全性和可靠性的传统手段有静态分析、测试和模型检测等.静态分析是一种离线分析程序的技术,旨在根据程序结构来确定程序的属性.通常这样的静态分析只能检测一组有限的通用错误,如数组绑定操作潜在的死锁问题,而且常常会产生误报[6].测试是对选定的输入数据集运行系统,通过比较所产出的输出与预期值来判断系统是否会有错误,但依然存在测试用例不完备的问题,通常不能保证系统绝对没有错误[7].与测试不同,模型检测是针对某类性质来检查系统是否合乎规约,它的基本思想是将一个过程或系统抽象成一个有穷状态模型,然后加以分析验证.由于对系统空间的穷举搜索,在并发系统中,其状态的数据往往随着并发分量的增加呈指数增长,因此,模型检测会存在空间爆炸问题.与以上这些传统质量保障技术相比,运行时验证方法是一种轻量级的形式化方法,并且具有高度的可扩展性[8−13],它对运行的软件系统进行实时监控,可以实时地验证系统的执行路径是否满足属性规范.由于运行时验证方法是针对程序或者程序产生的踪迹进行分析的,并且系统在运行时的执行路径是有穷的,因此不存在空间爆炸问题.形式化验证方法作为传统质量保障方法的补充,已被应用到了多个领域,如:无线传感器应用[14]、车辆总线系统[15]、C 程序[16]和医疗系统[17]等.同样地,在 CPS 领域也取得了一定的成果.例如,针对 CPS 中计算和物理系统由于复杂交互出现的问题,利用形式化验证方法检测和验证了可能导致复杂交互失败的条件.提出了改进的形式化方法来验证 CPS 中组件的复杂性问题,提高了 CPS 的可靠性[18].也有学者研究自动转换方法和规范逻辑语义以降低验证复杂性和运行开销[19].对于特殊情况,如关于 CPS 是分布式系统并且在动态物理环境下存在离散控制问题,有关学者提出了针对性的形式化验证方法和框架[20].

关于考虑物理环境对 CPS 的影响,国内外已有相关研究.为了使自主机器人在远程感知的开发研究中应对环境的变化并允许探索全新的领域,Tabak 等人介绍了一种算法,通过在环境中不同位置收集数据来生成或更新环境的 3-D 体积模型[21].为了解决 CPS 中由于多维异构性导致的环境信息描述不统一的问题和环境信息融合交互困难的问题,于洋等人基于元建模设施和建模技术,提出一种面向 CPS 环境信息的建模方法.该方法利用层次化的建模思想和领域化的建模方法,定义了环境信息的元元模型及不同领域中环境信息的元模型,实现了环境信息的元信息统一和不同领域中展现形式的多样性[22].在医疗领域,允许工程师明确而精确地指定有关

CPS 设计的物理环境假设,并通过算法将物理模型与系统模型相结合,利用 UPPAL 进行医疗器械的形式化验证[23].

本文针对物理环境的影响提出一种面向实时数据的一体化建模方法,通过定义一系列规则,将环境模型集成到运行时监视模型中.该方法的原理是通过动态调整监视模型中的参数范围,使得 CPS 中的安全属性在复杂的物理环境中仍然得到满足.具体步骤是,首先建立环境的数学模型.然后,依据合并规则将数学模型进行整合,整合的最终结果是,相同的系统参数有且仅有一个相同的环境模型.接着,定义了转换规则,用伪代码作为中间转换语言描述环境模型.最后,根据组合规则将环境模型代码组合到运行时监视模型中进行验证.最后,本文搭建了 EV3 实验平台,以移动机器人避障为例,对电量续航进行安全性提示,通过分析和建立环境模型,然后将此模型组合到监视模型中,以使监视模型更加完整,在不同物理环境中对续航时间的提示更加准确.

本文第 1 节介绍运行时验证方法的执行过程和 JavaMOP 基础知识,为本文提到的运行时监视模型和环境建模方法的研究提供理论基础.第 2 节详细介绍环境建模方法,定义了合并、转换和组合规则,并举例说明.第 3 节对第 2 节中提出的方法进行实验评估,将影响电池容量的环境模型整合到监视模型中进行实验评估.第 4 节总结本文工作.

1 预备知识

本节首先介绍了运行时验证方法、原理和本文用到的运行时验证工具框架,为本文后续的环境建模方法提供理论基础.

1.1 运行时验证

运行时验证是一种在线的、实时的形式化验证方法,它允许测试者使用数学的形式来指定系统并监测这些属性规约在系统运行时是否是一直满足的,当系统行为违反属性规约时,激发处理程序给出提醒或是执行用户自定义程序引导系统到正确的行为.它避免了模型检测方法的空间爆炸问题和传统测试的测试用例不完备问题,是一种实时的、全自动的、技术规模更小的质量保障方法.

运行时验证方法的原理是构造实时时序逻辑的时间自动机.如果系统的执行路径表示为 $v=v_1v_2\ldots v_i\ldots$,其中,v_i 是一个时间状态,那么,运行时验证方法则是将时间自动机扩展为监视器来验证该执行路径是否满足属性规约.如图 1 所示,对一个系统进行运行时验证,首先,根据该系统的属性规约生成监视器.然后,将目标系统置于监控下,监控器下的事件记录器负责窃听目标程序的数据和执行状态,将目标程序的每一次类的加载、调用和函数的调用、执行等操作进行非侵入式的收集,这些操作将会随着系统的执行实时地、不断地在存储器中更新.状态检查器根据需要验证的属性规范匹配系统状态,确定是否违背或满足属性.最后,通过执行监控程序给出验证结果,当违背属性时,它会触发处理程序,向目标程序发送自定义的反馈信息,以便目标程序可以意识到不安全的操作及时修正或是向用户发出警示提醒.

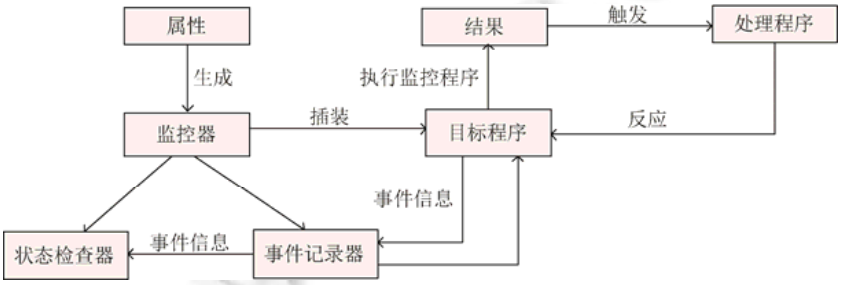

Fig.1 Runtime verification process

图 1 运行时验证过程图

运行时验证根据监控器监控对象的不同有两种应用.一种是实时监测正在执行的行为序列,验证当前行为

是否满足属性规约,违背时给出提示,这种运行时验证称为在线验证.另一种则监测历史执行序列,离线分析存储执行路径,被应用在离线的自动评估测试中,它被称为离线验证[24].对于 CPS 系统来说,要求尽早地监测违背安全属性的行为,从而可以积极地做出反应,所以本文考虑的是在线验证方式.

1.2 JavaMOP 语义

面向监控的编程(monitor-oriented programming,简称 MOP)是一个软件开发工具支持和分析框架,它支持面向方面编程的编织模式,支持多种形式化逻辑语言来描述属性规约,是一个自动化程序较高的运行时验证框架[11].MOP 工具会自动地根据属性规约合成监视器,并将它们集成在应用程序中.

MOP 框架具有以下特点.

(1) 将形式化规约和监视代码集成为一个系统,允许开发测试人员使用数学形式来指定系统,并证明这些属性.

(2) 它是一种轻量级的形式化方法,仅考虑一个模型的计算,而不是整个状态空间模型的检查.

(3) 具有逻辑语言的可扩展性(可以在程序中的任何地方添加逻辑语句,可以指定过去或未来的状态).其中,时序逻辑语言不仅限于纯布尔命题的序列,还可以是具有时间特性的序列,即可以约束随时间变化而变化的数据值的命题序列.

MOP 框架可以实时地监测系统的执行,当发现系统行为违背或满足属性规约时控制程序的执行.这种控制使得用户可以为当前危险行为提供处理程序,这些处理程序不仅可以向用户报告错误或抛出异常,还可以执行更加复杂的动作,例如重置状态或重启系统.一个规范定义的运行时监视器可以自动地组合到目标系统中,以便在违背规约时及时地更正系统行为,保证系统的安全性和可靠性.MOP 的实例包括 JavaMOP 和 BusMOP,由于 Java 语言具有跨平台性以及实验平台 EV3 的第三方插件 Lejos 支持 Java 语言,所以本文应用 JavaMOP 实例.完整语义如图 2 所示[11].

```
// BNF below is extended with {p} for zero or more and [p] for zero or one

⟨JavaMOP Specification⟩
        ::={⟨Modifier⟩} ⟨Id⟩ ["(" ⟨Java Parameters⟩ ")"] "{"
              ⟨Java Declarations⟩
              {⟨Event⟩}
              {⟨Property⟩
                 {"@" ⟨LOGIC State⟩ "{" ⟨Java Statements⟩ "}"}}
           "}"
⟨Modifier⟩          ::= "unsynchronized" | "decentralized" | "perthread" | "suffix"
⟨Event⟩             ::= ["creation"] "event" ⟨Id⟩ ⟨AspectJ advice⟩ "{" [ ⟨Java Statements⟩ ] "}"
⟨Property⟩          ::=⟨LOGIC Name⟩ ":" ⟨LOGIC Syntax⟩
⟨Java Declarations⟩ ::= ... ⟨!-- syntax of declarations in Java --⟩
⟨Java Parameters⟩   ::= ... ⟨!-- syntax of method parameter list in Java --⟩
⟨Java Statements⟩   ::= ... ⟨!-- slightly extended syntax of statements in Java --⟩
⟨ApsectJ Advice⟩    ::= ... ⟨!-- slightly extended syntax of advice in AspectJ --⟩
```

Fig.2 JavaMOP syntax

图 2 JavaMOP 语义

JavaMOP 支持的逻辑语言包括上下文无关语法(context free grammar,简称 CFG)、线性时序逻辑(linear temporal logic,简称 LTL)、有限自动机(finite state machine,简称 FSM)、过去时间的线性时序逻辑(past time linear temporal logic,简称 ptLTL)等.它提供图形用户界面(graphical user interface,简称 GUI)和命令行界面,用于编辑和处理规约.同时通过 JavaMOP,用户可以定义自己的属性描述语言,具有可插拔性和灵活性.

JavaMOP 主要包括 5 个部分:头部、变量声明、事件声明、属性的形式化描述和处理程序.

头部:头部包括修饰符、规范 ID 和参数列表 3 个部分.修饰符定义了监视模型中事件的访问方式和索引方式.规约 ID 是用户自定义名,即监控模型的名称,不可重复定义.

变量声明:变量声明和参数列表中的变量不同.参数列表中的变量是JavaMOP规范中的变量,通常情况下是监控模型中的系统变量.而变量声明是本地监视器变量,可以在事件和属性处理程序中修改.监视器变量的作用多种多样的,可以记录监视的状态,写日志等.其声明方式和使用方法遵循 Java 语法.

事件声明:其作用是定义事件.事件是指在目标程序执行过程中立即发生的一个动作,在监控程序中对应于被监控变量的设置与更新,或是被监控方法的调用与执行.JavaMOP 的事件声明遵循 AspectJ 语义[25],包含处理逻辑"advice"定义〈AspectJ advice〉和事件操作〈Java Statements〉,其中,事件操作部分可以通过 Java 编程修改被控程序或是监视器状态.事件编织到 Java 程序中后,将会记录事件执行的路径,用来检查是否符合属性规约.另外,事件的声明顺序是有意义的,如果多个事件同时发生,最终监视模型将按照事件的声明顺序执行.

属性定义:属性在 JavaMOP 中是可选的,即一个 MOP 规范中可能包含多条属性或没有属性.一条属性由一个属性描述逻辑名(〈LOGIC Name〉)和逻辑语义(〈LOGIC Syntax〉)构成,之间以冒号隔开.由于运行时验证方法监视的是系统的执行路径,因此,属性是一系列事件与逻辑的组合.MOP 中规定属性中所涉及到的所有事件必须是已定义的.JavaMOP 支持的属性描述语言包括 FSM、ERE、CFG、ptLTL、LTL 等.

处理器:处理器以@fail、@violation 或@validation 开头,分别代表自动机的不确定、违背或满足这 3 种状态.在〈Java Statements〉中用户可以自定义动作,在满足处理器触发条件时执行.

2 环境建模方法

为了解决 CPS 中由于环境影响导致的系统数据与实际环境不匹配的问题,本文结合运行时验证技术,提出了一种在 CPS 中应对环境变化的一体化建模方法.该方法定义了合并规则、转换规则和组合规则,将领域模型定义为数学模型,然后集成到监控模型中.最终实现将环境信息分享到监控模型中的目标,使得监控程序的执行更加精确,安全属性在不确定的环境中始终得以满足.

2.1 系统框架

一体化建模方法不改变运行时验证方法的原理和执行过程,该方法主要是在图 1 所示运行时验证过程的基础上,增加环境建模方法,对监视模型中的属性和参数产生影响,即对监视模型生成的前期数据产生影响.系统框架如图 3 所示.

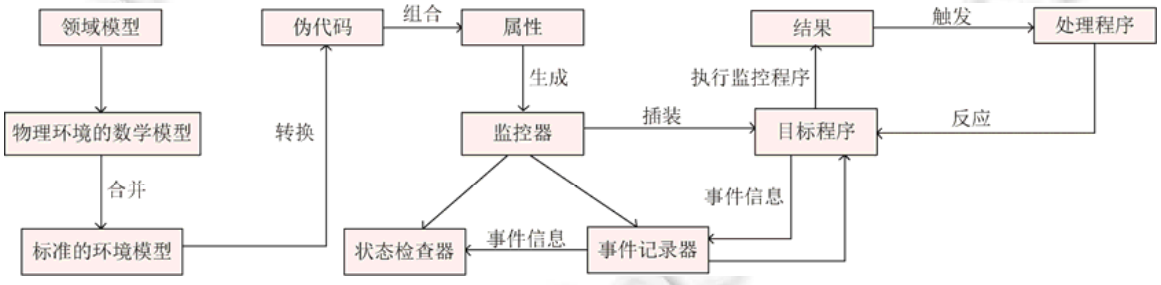

Fig.3 System framework

图 3 系统框架图

在 CPS 中,属性是由用户提出的需求或是需求文档中的内容形式化后的规约,包括 CPS 的功能性需求、非功能性需求和设计约束这 3 个主要部分.目标程序一般是指 CPS 控制器中的计算或控制程序,将监视器插装在控制器中,通过实时地监测控制程序来验证 CPS 的执行路径是否满足或违背安全属性规约.对 CPS 进行一体化建模,将环境影响集成到监控模型中的过程如图 3 所示.首先,由于数学具有抽象、精准的特点,将领域模型描述为数学模型.然后,根据合并规则标准化环境模型,主要是对作用于相同系统参数的不同环境模型进行整合,即将影响同一个系统参数的所有环境影响关系都集合在一起进行数学的划分,使最终的环境模型之间不产生交集,这样做可以避免发生验证逻辑错误.因为运行时验证工具众多,使用的语言多种多样,所以必须要有一个统

一的表达方式,使得不同语言的使用者可以清晰的阅读“中间模型”.而伪代码结构清晰、代码简单、可读性好,可以使得使用者很容易地以任何一种编程语言(Pascal、C、Java 等)无歧义地实现算法,所以最后,定义转换规则,将标准化模型转换为伪代码.最后,通过监视模型对应的编程语言,将环境模型的伪代码遵循模型组合规则集成到运行时监视模型中.

在一个监视模型中,事件在监视程序执行过程中通过捕获被监控的操作来获取参数,其中,部分参数会参与到事件操作的计算中,这类参数被称为系统参数.系统参数的值可能会受到环境的影响而发生变化,在监视模型不知晓这种变化关系而执行运行时监视时,会出现无法正确识别 CPS 不符合实际环境的危险行为,从而不会发出安全提醒的情况.该建模方法在标准化环境模型后,将转化的伪代码通过特定的语言写入事件操作中,将环境影响映射到系统参数的值域选择,从而使得参数可以随着环境的变化而发生改变,达到环境自适应的目的.这一过程不影响运行时验证过程,只对监视模型中事件的操作部分进行执行前的修改.

2.2 环境建模的规则描述

2.2.1 物理环境的数学模型定义

物理环境对 CPS 行为产生影响的原因是系统中的某些参数会因为物理因素的改变而发生变化,具体体现为系统的动态输入发生偏差,从而影响控制系统的逻辑判断,故而波及到 CPS 的行为.要避免这种影响,首先需要确定系统参数和物理环境的关系.这里先定义相关的符号表示,系统参数的集合 $S=\{s_1,\ldots,s_i,\ldots,s_n\}$,环境参数的集合 $E=\{e_1,\ldots,e_j,\ldots,e_m\}$,其中,集合中的每一个元素都用一个二元组(*name*,*type*)表示.下面是对物理环境的相关定义.

定义 1(环境数学模型). 用一个三元组 $M(s,E',R)$来描述环境数学模型.其中,$s\in S$,是一个系统参数.$E'\in E$,是一组会对 s 产生影响的环境因素的集合.R 是 s 和 E'之间的一组数学关系的集合$\{r_1,\ldots,r_k,\ldots,r_l\}$.

定义 2(关系模型). 关系模型 $r\in R$ 由一个三元组(F,VC,L)定义.其中,F 是函数的声明部分,包括函数名、函数的参数和返回值类型.F 指明了系统参数 s 在整个系统代码中出现的位置.VC 是一个二元组$(value(s),C(E'))$,其中,$C(E')$是物理条件,所以 VC 表示在条件 $C(E')$下的系统参数 s 的值为 $value(s)$.L 是在此条件下该环境参数对系统参数的影响级别,影响级别是一系列的整数,从 1 开始,数值越大代表影响程度越小.此影响级别由领域模型定义.

例 1:假设 CPS 的电池初始容量为 1Ah,也就是说,电池每小时提供 1A.温度对电池容量存在影响[22].

S_1:当温度 t 在[15°C,35°C]时,电池容量不变.

S_2:当温度 t 在[–10°C,15°C)时,$C=1–2\times(25–T)\div100$.

以上述两种场景为例建立数学模型.这个例子涉及到了两个物理变量,温度 $T=(t,real)$和 $Battery=(C,real)$.上述两种场景列出了这两个环境变量的两种关系,假设建立的环境数学模型为 M_0,关系模型为 R_0,则此例所对应的数学模型为 $M_0(Battery,T,R_0)$.为了便于撰写,这里为复杂的式子定义了别名.

c_1:$15\leqslant t\leqslant35$,

c_2:$-10\leqslant t<15$,

v_2:$C=1–2\times(25–T)\div100$.

假设温度对电池的影响级别为 1,系统变量 C 存在于函数 *handle* 中,根据 S_1 和 S_2 这两种情况,关系模型 R_0 建立为$\{r_{01},r_{02}\}$,具体如下.

r_{01}:$(handle,(1,c_1),1)$,

r_{02}:$(handle,(v_2,c_2),1)$.

另外,电池电容还受到环境湿度 H 的影响[22],具体关系如下.

S_3:当环境湿度 H 在[10%RH,30%RH]时,电池容量下降 10%.例如,若原电池容量 $C=1$,则当电池处在该环境湿度后,$C=0.9$.

S_4:当温度 t 在[40%RH,60%RH]时,电池容量不变.

同理,定义别名:

c_3:0.1≤h≤0.3,

c_4:0.4≤h≤0.6.

假设该环境关系建立数学模型为 M_1,关系模型为 R_1,影响级别为 2,则此例所对应的数学模型为 M_1(*Battery*,H,R_1).关系模型 R_1 为{r_{11},r_{12}}.

r_{11}:(*handle*,(0.9,c_3),2),

r_{12}:(*handle*,(1,c_4),2).

在一个完整的数学模型中,*VC*应具有完备性,也就是说,不论环境处于怎样的状态,都能在*VC*集合中找到一个二元组(*value*(s),$C(E')$)与之匹配,同时,$C(E')$之间互斥不相交.这样,在运行时验证时,在特定场景下的筛选结果是唯一的,因此程序执行路径是唯一确定的、无歧义的.

2.2.2 环境模型的合并规则

一个系统参数可能会受到多个物理因素的影响,同样地,一个物理因素也可能会影响到多个系统参数.它们之间的关系是复杂的,在领域知识中,这些数学关系表达是简单的,但在运行时验证中也是很难操作的,因为无法有效地避免同一个系统参数的不同影响因素之间的关系是完备且互斥的.因此,需要通过制定规则来规范化和标准化这些独立的数学模型,将多个相同系统参数单个物理影响因素的数学模型合并为一个系统参数多个物理影响因素的数学模型.此规则为后续将数学模型转换为伪代码提供逻辑基础.

假设有两个环境模型 $M_2(s_2,E_2',R_2)$和 $M_3(s_3,E_3',R_3)$,如果 s_2=s_3,则这两个对同一个系统参数建立的数学模型可以进行合并,合并结果为 $M(s,E',R)$,其中,s=$s_2(s_3)$,具体的合并规则定义如下.

前提条件:系统参数相同是判断两个模型是否可以合并的前提条件.

模型计算规则:集合 $M\{m_1,m_2,\ldots,m_m\}$和 $N\{n_1,n_2,\ldots,n_n\}$做组合操作会得到 $f(m,n)$个结果,其中,每两个元素取“&&”,得到的结果之间取“||”,即{m_1&&n_1||m_1&&n_2||…||m_m&&n_n}.

$$f(m,n)=\frac{n!}{m!(n-m)!} \tag{1}$$

模型合并规则(1). 模型合并规则(1)也叫作模型内标准化规则,是一种针对关系模型的合并规则.具体为关系模型集合内的元素之间作“||”操作.其中,*VC* 执行“||”操作,F 和 L 不变.如 M_2 中的关系模型 R_2={r_{21},r_{22},…,r_{2n}},执行此规则变为新的 R_2={r_{21}||r_{22}||…||r_{2n}}={F_2,(VC_{21}||VC_{22}||…||VC_{2n}),L_2},M_3 同理.

模型合并规则(2). 如果 E_2'=E_3',F_2!=F_3,即系统参数 s 在系统代码中的函数位置不同,那么,环境模型则相互独立,无需合并.为了规范化书写,M可以写成(s,{E_2',E_3'},R_2∪R_3),也可以保持不变.例如:在 M_2 模型中,R_2=(F_2,VC_2,L_2),在 M_3 模型中,R_3=(F_3,VC_3,L_3),F_2!=F_3,规范化书写后 M 的 R={R_2∪R_3}={(F_2,VC_2,L_2)∪(F_3,VC_3,L_3)}.

模型合并规则(3). 模型合并规则(3)也叫作模型间标准化规则.如果 E_2'!=E_3'且 F_2=F_3,即系统参数 s 的物理影响因素不同,但在系统代码中的函数位置相同.合并规则为 E={E_2',E_3'},关系模型 R 中 F=$F_2(F_3)$,*VC* 执行模型计算规则,L 取 L_2 和 L_3 中的最小值,即 L=Min{L_2,L_3}.在组合条件下,系统参数的值 *value*(s)取影响级别最大的值,即 *Worst*{*value*$_2$(s),*value*$_3$(s)}.

这里不存在 E_2'=E_3'且 F_2=F_3 的情况,因为在一个环境模型 $M(s,E',R)$中,对于一个系统参数 s 和特定的 E',R是具有完备性的.当环境模型不符合合并规则的前提条件时,执行模型合并规则(1),对自身进行规范化操作.模型合并规则(2)和规则(3)是模型之间的合并规则,其中,系统参数是否在同一个函数中是决定环境模型之间是否可以合并的关键.多于两个模型进行规范化合并时,两两执行合并规则,直到不满足合并的前提条件.当存在多种物理因素影响多个系统参数的情况时,依据模型合并的前提条件,分别将具有相同系统参数的数学模型遵循规则合并.

例 2:假设 $M_2(s_2,E_2',R_2)$中的 E_2'和 $M_3(s_3,E_3',R_3)$中的 E_3'不相等且 F_2=F_3,R_2={r_{21},r_{22}},R_3={r_{31},r_{32}},其中,VC_2={VC_{21},VC_{22}},VC_3={VC_{31},VC_{32}}.关系模型的每一个元素的展开式如下.

R_{21}:(F,VC_{21},L_2)=(F,(*value*$_{21}$(s),C_{21}),L_2).

R_{22}:(F,VC_{22},L_2)=(F,(*value*$_{22}$(s),C_{22}),L_2).

R_{31}:$(F,VC_{31},L_3)=(F,(value_{31}(s),C_{31}),L_3)$.

R_{32}:$(F,VC_{32},L_3)=(F,(value_{32}(s),C_{32}),L_3)$.

现在对 M_2 和 M_3 两个模型进行合并,由本小节开始时的假设可知 $s_2=s_3$,因此符合前提条件,可以进行合并.由 $E_2'!=E_3'$且 $F_2=F_3$,根据模型合并规则(3),合并后的模型 M 中 $E'=\{E_2',E_3'\}$,关系模型 R 中 $F=F_2(F_3)$,VC 遵循组合规则“其中每两个元素取“&&”,得到的结果之间取“||””,最终的关系模型为 $R=((r_{21}\&\&r_{31})||(r_{21}\&\&r_{32})||(r_{22}\&\&r_{31})||(r_{22}\&\&r_{32}))$. 以 $(r_{21}\&\&r_{31})$ 为例 ,$(r_{21}\&\&r_{31})=(F,(Worst\{value_{21}(s),value_{22}(s)\},C_{21}(E_2')\&\&C_{31}(E_3')),\min\{L_2,L_3\})$.

在例1中的模型 $M_0(Battery,T,R_0)$和 $M_1(Battery,H,R_1)$中,由于系统参数相同,都为 *Battery*,满足模型组合的前提条件,即 M_0 和 M_1 可以进行组合.由于 $T!=H$,即环境影响因素不同,分别是温度和湿度,所以符合模型合并规则(3).对 M_0 和 M_1 执行组合规则后形成的新模 M 为 $M=(Battery,\{T,H\},Worst\{1,2\})=(Battery,\{T,H\},1)$,$R=(handle,(r_{01}\&\&r_{11})||(r_{01}\&\&r_{12})||(r_{02}\&\&r_{11})||(r_{02}\&\&r_{12}),1)$,展开式如图4所示.

$$
\begin{aligned}
&R\ \ (handle,(r_{01}\&\&r_{11})\,\|\,(r_{01}\&\&r_{12})\,\|\,(r_{02}\&\&r_{11})\,\|\,(r_{02}\&\&r_{12}),1)\\
&=(handle,((1,15<=t<=35)\&\&(0.9,0.1<=h<=0.3))\,\|\\
&((1,15<=t<=35)\&\&(1,0.4<=h<=0.6))\,\|\\
&((1-2\times(25-t)/100,\ -10<=t<15)\&\&(0.9,\ 0.1<=h<=0.3))\,\|\\
&((1-2\times(25-t)/100,\ -10<=t<15)\&\&(1,0.4<=h<=0.6)),1)\\
&=(handle,(0.9,\ 15<=t<=35\ \&\&0.1<=h<=0.3)\,\|\\
&(1,\ 15<=t<=35\ \&\&0.4<=h<=0.6)\,\|\\
&(1-2\times(25-t)/100,\ -10<=t<15\ \&\&0.1<=h<=0.3)\,\|\\
&(1-2\times(25-t)/100,\ -10<=t<15\ \&\&0.4<=h<=0.6),1)
\end{aligned}
$$

Fig.4 R_0's expansion formula

图4 R_0'的展开式

2.2.3 环境模型与伪代码的转换规则

对环境建立数学模型的最终目的是要将此模型组合到运行时监视模型中,使得在线监视过程加入了环境变化的影响,让安全属性能够在不确定的物理环境中一直满足.在组合之前,为了相关领域人员和测试开发人员都能够简单、明了地表达和实现数学模型,本文提出将伪代码作为中间转化代码.伪代码是一种算法描述语言,其结构清晰且可读性好,并且类似自然语言.用它来描述数学模型,可以使使用者以任何一种编程语言容易地、无歧义地实现算法.对于任意一个 $M(s,E',R)$,关系模型为 $R(F,(value(s),C(E')),L)$的数学模型,转换规则分为以下几步.

(1) 根据关系模型中的 F 生成函数.

(2) 变量声明:在函数中,如果 s 不是函数的本地参数,生成一个 s 变量并初始化.

(3) 根据$(value(s),C(E'))$生成条件判断语句,其中,$C(E')$为条件判断,$value(s)$为在当前条件下 s 的值.其中,特殊符号“||”在程序中代表“else if”.

(4) “U”符号出现,则重新生成一个新的函数,执行(1)~(3)步.环境模型转换为伪代码的关键是$(value(s),C(E')$对的逻辑转换,将环境对系统参数的关系转换为程序逻辑语言,便于之后集成到监控模型中.第(2)步中的变量声明是伪代码中的语法需要,在监视模型中,s 由模型中的事件捕获得到.

例3:在例2中,对模型 $M_0(Battery,T,R_0)$和 $M_1(Battery,H,R_1)$进行了标准化,得到数学模型 $M(Battery,\{T,H\},1)$,其中,关系模型 $R=(handle,(r_{01}\&\&r_{11})||(r_{01}\&\&r_{12})||(r_{02}\&\&r_{11})||(r_{02}\&\&r_{12}),1)$.接下来,通过执行以下步骤,将 M 转换为伪代码.

(1) 依据 R 中的 F 生成 *handle* 函数.

(2) 声明 *battery* 变量,并赋值为0.在使用编程语言实现时,如果该变量已经存在于函数的参数列表中,则步骤2可省略.

(3) 根据关系模型中的 VC 关系建立 if 逻辑部分.

最终 *M* 伪代码如 Pseudocode 1 所示.将数学模型转化为伪代码后,测试开发人员则负责将伪代码实现为具体编程语言的程序并整合到运行时监视模型中,最后利用运行时验证工具对模型中描述的属性进行安全性验证.在本文中,运行时验证工具为 JavaMOP,所以 *M* 模型的伪代码用 Java 实现后组合到监视模型中,如图 5 中为 *M* 的 Java 实现代码.

Pseudocode 1:Transformation result of model *M*.

```
Input: The input parameters of handle.
 1:function HANDLE
 2:     battery ← 0
 3:     if 15<=t<=35&&0.1<=h<=0.3 then
 4:          battery=1;
 5:     else if 15<=t<=35&&0.4<=h<=0.6 then
 6:          battery=1;
 7:     else if -10<=t<=15&&0.1<=h<=0.3 then
 8:          battery=1-2*(25-t)/100;
 9:     else if -10<=t<=15&&0.4<=h<=0.6 then
10:          battery=1-2*(25-t)/100;
11:     end if
12:end function
```

```
public void handle (double battery, double w, int t, double h){
    if(15<=t<=35&&0.1<=h<=0.3){
        battery=0.9;
    }
    else if(15<=t<=35&&0.4<=h<=0.6){
        battery=1;
    }
    else if(-10<=t<15&&(0.1<=h<=0.3||0.4<=h<=0.6)){
        battery=1-2*(25-t)/100;
    }
}
```

Fig.5 The Java implementation of *M*

图 5 *M* 模型的 Java 实现

2.2.4 环境模型和运行时监视模型的组合规则

组合规则的目的是要将伪代码实现为具体编程语言的程序后组合到监视模型的事件操作部分,具体组合规则分为以下几个步骤.

(1) 遍历所有运行时监视模型的事件定义,选择事件切入点的函数和模型代码中的函数 *F* 相同的事件,且在事件操作中含有 *s* 参数参与计算.

(2) 选定代码块:其中,代码块的开始是第 1 条 *s* 在赋值等号右边的语句,代码块的结束时是最后一条 *s* 在赋值等号右边的语句.

(3) 将代码块作为整体替换模型代码中 if 条件控制的执行语句.如果存在一个代码块对应于多个模型代码的情况,即多个模型代码中的系统参数出现在同一个表达式中,此时将所有模型代码中的 if 条件组合操作后的新模型代码替换掉原代码块.

(4) 赋值号右边的 *s* 被置换为模型代码中相同判断条件执行语句中的 *s* 的新值.

(5) 将最终形成的函数代码插入到当前事件中替换代码块.

连接点是切面插入应用程序的地方,包含函数的调用和执行、构造函数的调用和执行、变量的获取与设置等.而切入点的作用则是过滤这些连接点,匹配符合条件的连接点从而激活事件.它是连接点的子集合,在组合规则中,代码块的划分是关键,将修改目标锁定在系统参数参与计算的语句上,以便于添加环境条件.

例 4:例 3 中生成 *M* 的伪代码后,使用 Java 语言编写函数实现伪代码中的算法逻辑,按照组合规则将其插入到运行时监视模型中,步骤如下.

(1) 遍历所有运行时监视模型的事件定义,并选择函数与 *handle* 相同的事件.如图 6 所示,在 *electricity* 验证模型的 *start*、*init* 和 *sufficientElectricity* 这 3 个事件中,只有 *init* 事件满足要求,且系统参数 *battery* 参与了危险

距离 *b* 变量的赋值计算.

(2) 选定第 1 条 *battery* 在赋值等号右边的语句作为代码块的开始,最后一条 *battery* 在赋值等号右边的语句作为代码块的结束,划定代码块区域.如图 6 中红框所示,代码块包含 *b*=*battery*×3.0÷*w* 一条语句.

```
electricity(double battery, double w, int t, double h) {
double b=0;

//事件定义
event start after():
call(ClientOnPC.new(..)){}

event init before(double battery, double w, int t, double h) :
call(* *.handle(double,double,int,double))
&&args(battery, w, t, h) {
   b=battery*3.0/w;  ⟶ battery第一次出现并参与计算的地方
}
event sufficientElectricity after(double battery, double w, int t, double h) :
call(void handle(double,double,int,double))
&&if(b>0.17) {}

//属性描述
ptltl :[*](sufficientElectricity S start)

//用户自定义程序
@violation {
System.out.println("在当前功率下，电池续航不足十分钟！");
_RESET;
}
}
```

Fig.6 The runtime monitoring model

图 6 运行时监视模型

(3) 将代码块作为整体替换模型代码中 if 条件控制的执行语句,如图 7 所示.

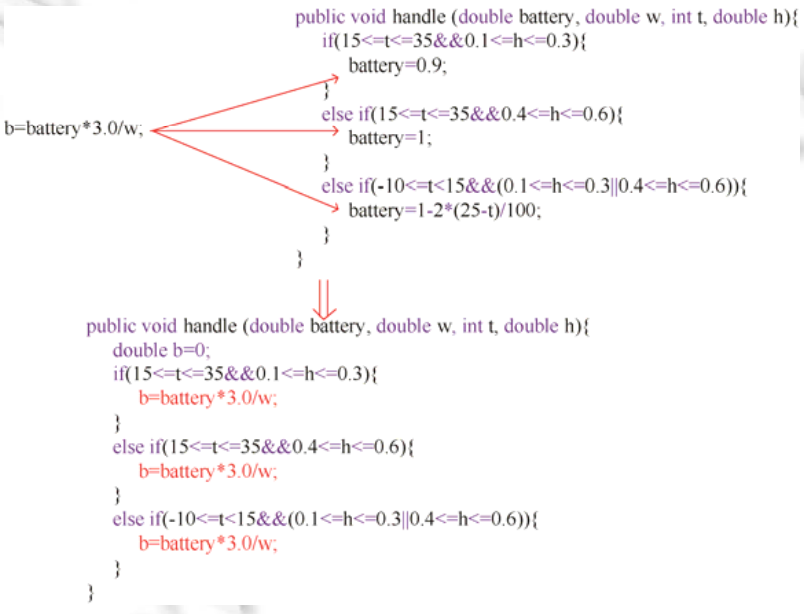

Fig.7 The code block replace if statement

图 7 代码块替换 if 语句

将赋值号右边的 *battery* 置换为模型代码中相同判断条件执行语句中的 *battery* 的新值,如图 8 所示.最后,

所获得的结果如图 9 所示.

```
public void handle (double battery, double w, int t, double h){
    double b=0;
    if(15<=t<=35&&0.1<=h<=0.3){
        b=0.9*3.0/w;
    }
    else if(15<=t<=35&&0.4<=h<=0.6){
        b=1*3.0/w;
    }
    else if(-10<=t<15&&(0.1<=h<=0.3||0.4<=h<=0.6)){
        b=(1-2*(25-t)/100)*3.0/w;
    }
}
```

Fig.8 *Battery* modify new value

图 8 *Battery* 变量修改新值

```
electricity(double battery, double w, int t, double h) {
double b=0;

//事件定义
event start after():
call(ClientOnPC.new(..)){}

event init before(double battery, double w, int t, double h) :
call(* *.handle(double,double,int,double))
&&args(battery, w, t, h) {
    b=battery*3.0/w;
    if(15<=t<=35&&0.1<=h<=0.3){
        b= 0.9*3.0/w;
    }
    else if(15<=t<=35&&0.4<=h<=0.6){
        b= 1*3.0/w;
    }
    else if(-10<=t<15&&(0.1<=h<=0.3||0.4<=h<=0.6)){
        b= (1-2×(25-t)/100) *3.0/w;
    }
}
event sufficientElectricity after(double battery, double w, int t, double h) :
call(void handle(double,double,int,double))
&&if(b>0.17) {}

//属性描述
ptltl :[*](sufficientElectricity S start)

//用户自定义程序
@violation {
System.out.println("在当前功率下，电池续航不足十分钟！ ");
_RESET;
}
}
```

Fig.9 The runtime monitoring model after combining

图 9 合并后的运行时监视模型

3 实验评估

本文的实验平台选用了如图 10 所示的 EV3 机器人,它是乐高公司开发的第三代 MINDSTORMS 机器人,于 2013 年下半年上市.它不仅能够加载多种传感器,如超声波传感器、触碰传感器、陀螺仪等,而且还可以使用 leJOS 第三方库支持 Java 语言编程.实验中,传感器的采样频率设为 1s,电池电压为 3.0v,容量为 1Ah.

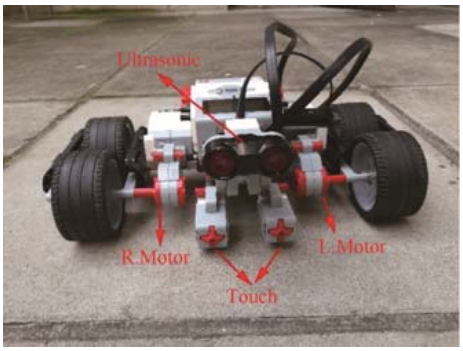

Fig.10 Lego EV3 robot

图 10 乐高 EV3 机器人

实验设计为移动机器人匀速行驶执行避障任务[26,27],每一次执行时间至少需要 10min.机器人将使用超声波传感器和触碰传感器来感知周围环境避开障碍物.为了使移动机器人成功地执行避障任务,设定一条安全属性为“当电池续航不足 10 分钟时给出提示”.分析影响电池容量的环境因素,如例 1~例 4 所示,将影响电池电容的温度和湿度建立环境模型并组合到 *electricity* 监视模型中去.已知存在关系式:容量(Ah)×电压(V)=功率(W)×时间(h),在满足电量条件下执行了 6 000 多次实验,实验结果见表 1.

Table 1 Experimental results

表 1 实验结果

环境条件	理论提示功率(W)	实际提示功率(W)
10°C, 20%RH	18	5
–10°C, 40%RH	18	5
0°C, 40%RH	18	9
0°C, 60%RH	18	9
10°C, 20%RH	18	13
10°C, 60%RH	18	13
15°C, 20%RH	18	16
15°C, 40%RH	18	18
20°C, 20%RH	18	16
20°C, 50%RH	18	18
25°C, 30%RH	18	17
25°C, 60%RH	18	18

当不考虑环境影响因素时,理论上当移动机器人的功率为 18W 时,会提示“在当前功率下电池续航不足十分钟!”.但当环境中温度和湿度发生变化时会出现续航时间比提示时间略长或缩短的情况.由表 1 可以看出,当温度在 15°C~35°C 时,电池电容与湿度有关.当温度在–10°C~15°C 时,电池电容只与温度有关,与湿度无关.将环境模型组合到运行时监视模型中,可以随着环境中温度和湿度的改变而调整电容值,从而调整安全提示的功率临界值.

实验结果表明,监视模型集成环境模型后,监视程序的执行将会更加精确,可以有效地保证在复杂工作环境下机器人系统的安全性和可靠性.该建模方法不仅可以用于移动机器人的安全电量监测,也适用于采用运行时验证方法保障系统的安全性和可靠性的 CPS,不限制逻辑描述语言和操作系统类型.

4 总 结

本文提出一种面向实时数据的 CPS 一体化建模方法,对环境进行建模,然后对环境模型进行标准化,转化为伪代码后组合到监视模型中进行运行时验证,目的是使得监视模型更加完整、准确,在环境发生变化时,通过动态调整模型中的参数范围,使得 CPS 中的安全属性在复杂的物理环境中仍然得以满足.该方法定义了模型合并

规则、模型和伪代码之间的转换规则、环境模型和监视模型的组合规则,并通过温度和湿度对电池容量的影响举例加以说明.最后在移动机器人平台上,通过使用 JavaMOP 验证安全属性的实验证明在机器人系统中组合物理模型后可以使得监视模型的监视更加准确.

References:

[1] Luo JH, Xiao ZH, Zhong CP. Analysis of the development trend of information physics systems. Telecommunications Science, 2012,28(2):127−132 (in Chinese with English abstract). [doi: 10.3969/j.issn.1000-0801.2012.02.025]

[2] Li L. Research on security technology of cyber-physical systems (CPS). Automated Expo, 2017,2016(7):58–61 (in Chinese with English abstract). [doi: 10.3969/j.issn.1003-0492.2016.07.031]

[3] Tsiou C, Efthymiatos G, Katostaras T. Noise in the operating rooms of Greek hospitals. The Journal of the Acoustical Society of America, 2008,123(2):757–765. [doi: 10.1121/1.2821972]

[4] Siu KC, Suh IH, Mukherjee M, Oleynikov D, Stergiou N. The effect of music on robot-assisted laparoscopic surgical performance. Surgical Innovation, 2010,17(4):306–311. [doi: 10.1177/1553350610381087]

[5] Barai S, Sau B. Path following mobile robot using passive RFID tags in indoor environment. Int'l Journal on Recent and Innovation Trends in Computing and Communication, 2015,3(7):3652–3655.

[6] D'Amorim M, Havelund K. Event-based runtime verification of Java programs. ACM SIGSOFT Software Engineering Notes, 2005,30(4):1–7. [doi: 10.1145/1082983.1083249]

[7] Lin HM, Zhang WH. Model checking: Theory, method and application. Chinese Journal of Electronics, 2002,30(12a):1907–1912 (in Chinese with English abstract). [doi: 10.3321/j.issn:0372-2112.2002.z1.002]

[8] Luo CX, Wang R, Jiang Y, Yang K, Guan Y, Li XJ, Shi ZP. Runtime verification of robots collision avoidance case study. In: Proc. of the 42nd IEEE Annual Computer Software and Applications Conf. (COMPSAC). Tokyo: IEEE Computer Society, 2018. 204–212. [doi: 10.1109/COMPSAC.2018.00033]

[9] Zhang S, He F. Research progress of runtime verification technology. Computer Science, 2014,41(s2):359–363 (in Chinese with English abstract).

[10] Wang R, Wei YX, Song HB, Jiang Y, Guan Y, Song XY, Li XJ. From off-line towards real-time verification for robot systems. IEEE Trans. on Industrial Informatics, 2018,14(4):1712–1721. [doi: 10.1109/TII.2017.2788901]

[11] Meredith PO, Jin D, Griffith D, Chen F, Rosu G. An overview of the MOP runtime verification framework. Int'l Journal on Software Tools for Technology Transfer, 2012,14(3):249–289. [doi: 10.1007/s10009-011-0198-6]

[12] Jiang Y, Zhang HH, Li ZH, Deng YD, Song XY, Gu M, Sun JG. Design and optimization of multiclocked embedded systems using formal techniques. In: Proc. of the Joint Meeting on Foundations of Software Engineering. ACM, 2013. 1270–1278. [doi: 10.1145/2491411.2494575]

[13] Jiang Y, Liu H, Song HB, Kong H, Wang R, Guan Y, Sha L. Safety assured formal model driven design of the multifunction vehicle bus controller. IEEE Trans. on Intelligent Transportation Systems, 2016,(99):1–14. [doi: 10.1109/TITS.2017.2778077]

[14] Jiang Y, Song HB, Wang R, Gu M, Sun JG, Sha L. Data-centered runtime verification of wireless medical cyber-physical system. IEEE Trans. on Industrial Informatics, 2016,1(1):1–9. [doi: 10.1109/tii.2016.2573762]

[15] Zhang S, He F, Gu M. VeRV: A temporal and data-concerned verification framework for the vehicle bus systems. In: Proc. of the Computer Communications. IEEE, 2015. 1167–1175. [doi: 10.1109/INFOCOM.2015.7218491]

[16] Havelund K. Runtime verification of C programs. In: Proc. of the 20th IFIP TC 6/wg 6.1 Int'l Conf. on Testing of Software and Communicating Systems: The 8th Int'l Workshop. Berlin: Springer-Verlag, 2008. 7–22. [doi: 10.1007/978-3-540-68524-1_3]

[17] Jiang Y, Liu H, Kong H, Wang R, Hosseini M, Sun JG, Sha L. Use runtime verification to improve the quality of medical care practice. In: Proc. of the 38th IEEE/ACM Int'l Conf. on Software Engineering Companion (ICSE-C). IEEE Computer Society, 2016. 112–121. [doi: 10.1145/2889160.2889233]

[18] Gawanmeh A, Alwadi A, Parvin S. Formal verification of control strategies for a cyber physical system. In: Proc. of the 37th IEEE Int'l Conf. on Distributed Computing Systems Workshops. IEEE, 2017. 91–96. [doi: 10.1109/ICDCSW.2017.59]

[19] Bersani MM, Garcia-valls M. The cost of formal verification in adaptive CPS. An example of a virtualized server node. In: Proc. of the IEEE Int'l Symp. on High Assurance Systems Engineering. IEEE, 2016. 39–46. [doi: 10.1109/HASE.2016.46]

[20] Ishigooka T, Saissi H, Piper T, Winter S, Suri N. Practical use of formal verification for safety critical cyber-physical systems: A case study. In: Proc. of the IEEE Int'l Conf. on Cyber-physical Systems, Networks, and Applications (CPSNA). IEEE, 2014. 7–12. [doi: 10.1109/CPSNA.2014.20]

[21] Tabak Y, Jain R. Building an environment model using depth information. Computer, 1989,22(6):85–90. [doi: 10.1109/2.30724]

[22] Yang Y, Jin Y, Zhang J. Modeling method on environmental information in CPS. Journal of Jilin University (Science Edition), 2015,53(2):280–284 (in Chinese with English abstract). [doi: 10.13413/j.cnki.jdxblxb.2015.02.24]

[23] Fu ZC, Guo CH, Ren SP, Jiang Y, Sha L. Modeling and integrating human interaction assumptions in medical cyber-physical system design. In: Proc. of the 30th IEEE Int'l Symp. on Computer-Based Medical Systems (CBMS). IEEE, 2017. 1619–1622. [doi: 10.1109/CBMS.2017.50]

[24] Wang Z. Research on runtime verification of real-time systems [M.S. Thesis]. Wuhan: Huazhong Normal University, 2014 (in Chinese with English abstract).

[25] Kiczales G, Hilsdale E, Hugunin J, Kersten M. An overview of AspectJ. In: Proc. of the European Conf. on Object-oriented Programming. Berlin, Heidelberg: Springer-Verlag, 2001. 327–353. [doi: 10.1007/3-540-45337-7_18]

[26] Xin Y. Research on detection, prediction and collision avoidance methods of unobstructed vehicles [Ph.D. Thesis]. Hefei: University of Science and Technology of China, 2014 (in Chinese with English abstract).

[27] Wang R, Wang M, Guan Y, Li XJ. Modeling and analysis of the obstacle-avoidance strategies for a mobile robot in a dynamic environment. In: Mathematical Problems in Engineering. 2015. 1–11. [doi: 10.1155/2015/837259]

附中文参考文献:

[1] 罗俊海,肖志辉,仲昌平.信息物理系统的发展趋势分析.电信科学,2012,28(2):127–132. [doi: 10.3969/j.issn.1000-0801.2012.02.025]

[2] 李琳.信息物理系统(CPS)安全技术研究.自动化博览,2017,2016(7):58–61. [doi: 10.3969/j.issn.1003-0492.2016.07.031]

[7] 林惠民,张文辉.模型检测:理论、方法与应用.电子学报,2002,30(12a):1907–1912. [doi: 10.3321/j.issn:0372-2112.2002.z1.002]

[9] 张硕,贺飞.运行时验证技术的研究进展.计算机科学,2014,41(s2):359–363.

[22] 于洋,金英,张晶.CPS 中环境信息的建模方法.吉林大学学报(理学版),2015,53(2):280–284. [doi: 10.13413/j.cnki.jdxblxb.2015.02.24].

[24] 王珍.实时系统的运行时验证研究[硕士学位论文].武汉:华中师范大学,2014.

[26] 辛煜.无人驾驶车辆运动障碍物检测、预测和避撞方法研究[博士学位论文].合肥:中国科学技术大学,2014.

罗晨霞(1993－),女,河北张家口人,硕士,主要研究领域为形式化方法.

李晓娟(1968－),女,博士,教授,CCF 专业会员,主要研究领域为系统形式建模与验证,机器人系统软件安全,计算机网络协议分析.

王瑞(1981－),女,博士,副教授,CCF 专业会员,主要研究领域为形式化方法.

施智平(1974－),男,博士,教授,CCF 高级会员,主要研究领域为形式化,人工智能.

关永(1966－),男,博士,教授,博士生导师,CCF 专业会员,主要研究领域为形式化验证,系统可靠性,嵌入式系统.

Xiaoyu Song(1963－),男,博士,教授,博士生导师,主要研究领域为形式化方法.