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.

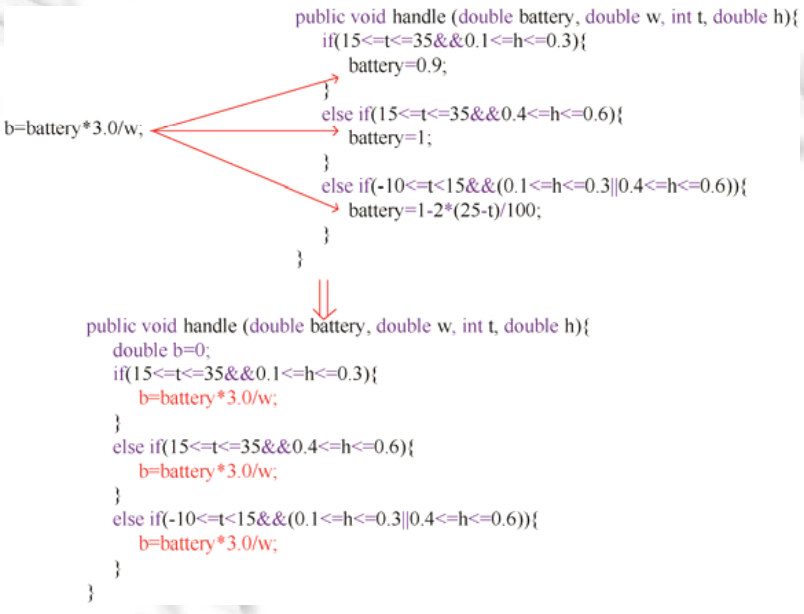

Fig.7 The code block replace if statement

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.

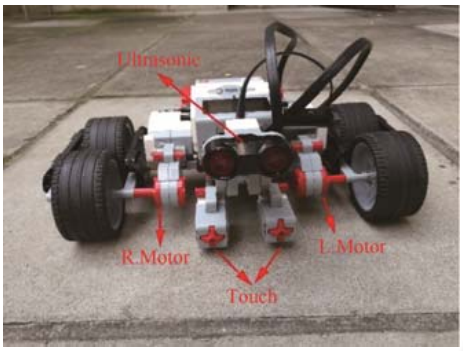

Fig.10 Lego EV3 robot

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.