3.1 AL_3的性质检验

(1) 运行时轨迹规约

根据定义2,使用Ambient Calculus对于运行时状态进行规约后,将形成具有时间顺序的轨迹,轨迹中的每个部分是由一个时刻的系统运行状态(包括各个实体、它们之间的关系、符合的性质)转换而成的一个 ambient 树,由于一个 ambient 表达式可能无法转换为统一的根节点的 ambient 树(即没有统一的最外层 ambient,如 *room*1[]|*room*2[]),我们统一增加一个最外层 ambient 作为树的根节点以确保 ambient 树的形成,这些根节点构成了Π_0集合.

图 1 给出了运行轨迹的实例,这个例子描述了在用户 Bob 在一个房间里生活的片段,Bob 首先在客厅里阅读,之后进入浴室开始洗澡.在这样轨迹中,根节点集合为Π_0={*s*0,*s*1,*s*2}.

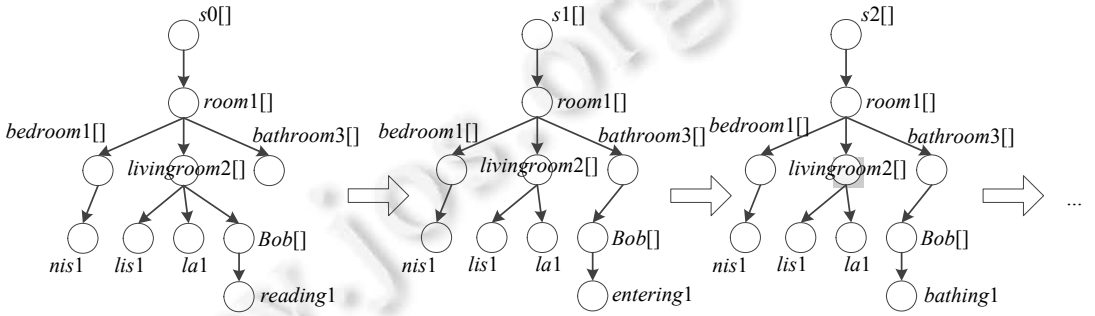


Fig.1 An example of runtime traces depicted by Ambient Calculus

图 1 Ambient Calculus 描述的运行轨迹示例

基于这样的运行轨迹规约,进程的空间关系(*LR* 集合)可通过 ambient 树节点的父子关系完成构造.时间顺序关系(*TR* 集合)可通过不同状态中的相同进程名进行构造,在图 1 的例子中,尽管 Bob 进程发生了移动,但是状态 2 的 Bob 进程仍然是状态 1 中 Bob 进程的后续进程.

各个进程满足的原子公式集合 *L* 基于应用需求和运行状态进行设置.

(2) 性质的预处理

为方便起见,我们提供了使用由基本算子定义的其他算子(如∧,→,∃,□)描述性质的能力;同时,为处理便捷,我们规定:描述公式时,需将连续的合取项($\Phi_1\wedge\Phi_2\wedge\ldots\wedge\Phi_n$)以及连续的析取项($\Phi_1\vee\Phi_2\vee\ldots\vee\Phi_n$)置入括号中.

在执行检验之前,需要对非基本算子进行预处理,转化为基本算子.

算法 1. 性质预处理 preprocess.

Input:公式,Φ;

Output:转换后的公式,Φ'.

1: Φ':=Φ;

2: Φ'中所有$\exists x.\Phi_1$转为$\neg(\forall x.\neg\Phi_1)$;

3: Φ'中所有$\Box\Phi_1$转为$\neg(\Diamond\neg\Phi_1)$;

4: Φ'中所有$\Phi_1\rightarrow\Phi_2$转为$\neg\Phi_1\vee\Phi_2$

5:将Φ'各个部分(包括◇,∀,∇等符号内的子公式)中的所有合取式$\Phi_1\wedge\Phi_2\wedge\ldots\wedge\Phi_n$转为

$$\neg(\neg\Phi_1\vee\neg\Phi_2\vee\ldots\vee\neg\Phi_n);$$

6: **return** Φ';

3) 性质检验算法

我们在文献[21]给出的算法基础上设计了基于 AL_3 公式的性质检验算法,其目的是判断进程 *P* 是否满足公式Φ($P\models\Phi$).需要注意的是:由于我们引入了三值逻辑,算法的返回值不再是布尔型(Boolean)结果,而是一个由{*T*,?,*F*}组成的枚举类型 AL3Result.在算法实现中,为处理方便,我们分别设 *T*=1,?=0,*F*=−1,这样,AL3Result 的结

果是可以进行数值上的大小比较的.

算法 2. 性质检验算法 check.

Input:进程,P;公式,Φ;

Output:AL3Result 类型的结果,*result*.

```
1:    switch (公式Φ呈现的形式):
2:    case (T)          : return T;
3:    case (a∈AP)       : if (a∈L(P)) return T; else return F;   //公式Φ是 P 满足的原子公式
4:    case (¬Φ')        : return –check(P,Φ'));
5:    case (Φ1∨Φ2)      : return max{check(P,Φ1),check(P,Φ2)};
6:    case (n[Φ'])      : if (P.name=n)
7:                            if (Φ'是 k 个公式组成的Φ1|Φ2|…|Φk 的形式)
8:                                if (存在 P1…PK,满足(P,Pi)∈LR,且 check(Pi,Φi)皆为 T)
9:                                    return T;
10:                               else if (存在 P1...PK,满足(P,Pi)∈LR,且 check(Pi,Φi)皆为 T 或?)
11:                                   return ?;
12:                               else return F;
13:                           else return max{check(Pi,Φ')|(P,Pi)∈LR}
14:                       else return F;
15:   case (Φ'@n)       : if (存在进程 P',使得(P',P)∈LR AND P'.name=n)
16:                           return check(P',Φ');
17:                       else return F;
18:   case (∀x.Φ')      : return min{check(P,Φ'{x←fn})|fn∈Λ};
19:   case (∇Φ')        : maxTemp:=check(P,Φ');
20:                       for each P'满足(P,P')∈LR do begin
21:                           maxTemp:=max{maxTemp,check(P',∇Φ')};
22:                       return maxTemp;
23:   case(◇Φ')         : if(check(P,Φ')=T) return T;
24:                       for each P'满足(P,P')∈TR do begin
25:                           if (check(P',Φ')=T) return T;
26:                       return ?;
27: end switch
```

该算法按照 AL_3 的语义递归地处理每个算子,最终返回三值逻辑表示的 $P\models\Phi$性质的满足性.在实际使用时,验证一条性质往往需要调用多次,如验证第 2.1 节的性质(1)这类表示为$\forall loc\in Location.loc[P]\models\Phi$的性质时,需要用轨迹中所有类型为 Location 的各个进程分别进行验证.

从算法复杂度的角度看,与标准 Ambient Logic 检验算法[21]一样,本算法可能需要从一组 $P_1|P_2|\ldots|P_m$ 的进程中选出 k 个以检验公式$\Phi_1|\Phi_2|\ldots|\Phi_k$(第 8 行、第 10 行),这样的排列问题理论上的最坏时间复杂度是阶乘级的,但是在实现过程中,我们进行了一些优化,主要包括:

- 记忆化搜索:在搜索过程中,如果已经检验过 P_i 是否满足公式Φ_j,则将$[P_i\models\Phi_j]$值计入数组,以避免重复计算的时间开销;
- 剪枝:当出现$[P_i\models\Phi_j]=F$ 时,相关的排列可以不再继续搜索;如果$[P_i\models\Phi_j]$=?且当前已经找到结果为?的排列,同样可以不再继续搜索;如果找到结果为 T 的排列,则直接返回;
- 随机化:随机选择下一个搜索的进程,以避免特殊格式进程、性质导致的搜索开销过大情况.

通过以上优化,结合实际应用规模,我们的验证算法时间开销基本可以接受(参见第 4.2 节实验).

3.2 普适计算应用监控器

我们通过普适计算应用监控器执行运行时验证、监控普适计算的运行状态是否满足给定的性质.根据应用需求的不同,这一监控过程可能采取应用定时推送状态(push)的方式,也可能由监控器主动获取(pull).监控器的结构如图 2 所示,分为以下几个部分.

- 应用状态翻译器:该组件接收普适计算应用运行轨迹中的一系列状态,将其转为 Ambient Calculus 的形式系统,从实现上说,即是补充根节点、构造若干 ambient 树结构,树中的每个节点皆是一个进程(包括 ambient 这种特殊进程).具体的层次关系、转换方式和应用类型有关,我们的前期工作[19]有针对特定应用进行形式规约的例子;
- 进程转换系统构造器:对于构造完成的 ambient 树进行分析,建立本次监控对应的有限轨迹进程转换系统,包括空间关系 *LR* 集合、时间顺序关系 *TR* 集合等;
- 性质预处理器:使用算法 1 对于性质进行预处理,使得公式中只含有基本算子,将这样的性质保存、推送进性质验证器;
- 性质检验器:使用算法 2 对于普适计算的运行轨迹是否满足给定的性质进行检验,以三值逻辑的形式输出检验结果,对于一个轨迹而言,只要出现进程的结果为 F 则输出 F;否则,如果出现进程的结果为?则输出?,如果所有相关进程检验结果皆为 T 则输出 T;
- 监控结果反馈设施:接收到检验结果后,将相应结果反馈给计算系统.需要注意的是:对于监控结果(尤其是结果为?的情况下)的解释是和具体性质有关的,需要设计应用性质的人员实现相应的解释器.

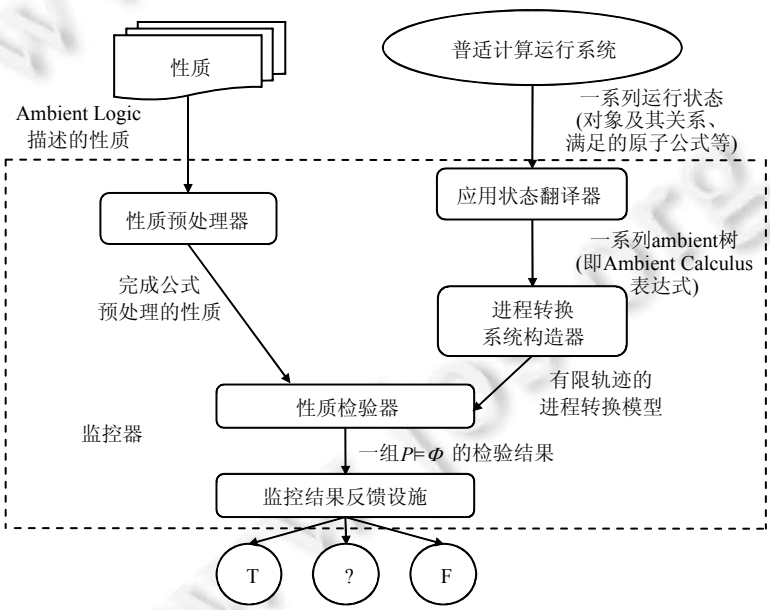


Fig.2 Structure of the monitor for pervasive computing applications

图 2 普适计算应用监控器的结构

4 评 估

本节首先通过一个案例讨论将 AL_3 方法应用于普适计算应用运行时验证的有效性,再通过一个性能评估实验讨论该方法的效率.

4.1 案例研究

老人看护系统是一类常见的普适计算应用,老人在普适计算环境中自主地进行日常生活,计算系统感知老人的身体状况、位置、动作等信息,也感知环境的温度、湿度、光线强度、噪音等信息,根据应用需求调整环境的状态(如开灯),或者给老人及其家人一些相关的提示.

(1) 应用运行状态规约

我们通过监控器中的应用状态翻译器将计算系统转为形式系统.采用 Ambient Calculus 作为形式工具对于应用的运行状态轨迹进行规约,并根据应用需求定义进程满足的原子性质,以图 1 所示的运行轨迹为例,其 3 个状态可分别规约为:

- $s0[room1[bedroom1[nis1]|livingroom2[lis1|la1|Bob[reading1]]|bathroom3[]]]$
  where $Bob[P]\vDash isElder, reading1\vDash hasActType(reading)$;
- $s1[room1[bedroom1[nis1]|livingroom2[lis1|la1]|bathroom3[Bob[entering1]]]]$
  where $Bob[P]\vDash isElder, entering1\vDash hasActType(entering)$;
- $s2[room1[bedroom1[nis1]|livingroom2[lis1|la1]|bathroom3[Bob[bathing1]]]]$
  where $Bob[P]\vDash isElder, bathing1\vDash hasActType(bathing)$.

当然,此外还有一些进程满足其他的原子性质,例如,“*nis*1 是噪音传感器”、“*lis*1 是光线传感器”可描述如下:

$$nis1\vDash sensingFea(noiseIntensity), lis1\vDash sensingFea(lightIntensity).$$

进一步地,我们构造有限轨迹的进程转换系统,用进程 P 满足的原子性质构造 $L(P)$集合,通过 ambient 层次关系构造 LR 集合,通过不同时刻的进程名构造 TR 集合.

(2) 运行时验证

与普通的二值 Ambient Logic 一样,AL_3 的性质检验器也可用于针对一个特定时刻的运行状态,检验不含时态算子的性质.当然,其特点和优势在于可以针对运行时的有限运行轨迹,检验同时含有空间包含关系和时间顺序关系的时空性质.

以第 2.1 节的性质(1)为例,即“如果一位老人进入浴室洗澡,则应当监测到他离开浴室的动作(以免洗澡时发生事故)”,如果将该性质输入性质检验器,对于图 1 的 3 个状态构成的有限轨迹进行检验,返回结果为?(因为 Bob 尚未离开浴室).该结果将由和性质对应的解释器进行解释,例如出现“?”时间过长需要告知家人.

如果在图 1 的这 3 个状态之后出现了

$$s3[room1[bedroom1[nis1]|livingroom2[lis1|la1]|bathroom3[Bob[leaving1]]]]$$
$$\text{where } Bob[P]\vDash isElder,\ bathing1\vDash hasActType(leaving)$$

这一状态,则说明 Bob 已经离开了浴室,性质检验结果为 T.

对于 2.1 节的性质(2),即“老人的动作始终要被监测到”,如果将该性质输入性质检验器,对于图 1 的 3 个状态构成的有限轨迹进行检验,返回结果为“?”,该结果将由和性质对应的解释器进行解释,这是系统中有老人存在时的正常情况.如果系统中没有老人存在,则返回结果将是 T(蕴含式前设 *isElder* 不成立).如果老人的动作未被监测到或动作类型无法被识别(老人 ambient 中没有表示动作的进程),则返回结果为 F.

对于 2.1 节的性质(3),即“人员进入一个房间后一定要能监测到其位置(可能在嵌套的位置中)”,如果将该性质输入性质检验器,对于图 1 的三个状态构成的有限轨迹进行检验,返回结果为“?”,该结果将由和性质对应的解释器进行解释,这是系统中有人员进入某个位置后的正常情况;如果系统中没有出现“进入(*entering*)”动作,则返回结果将是 T;如果有人员进入之后尚未离开却无法监测到其位置,则返回结果为 F;如果监测到“离开(*leaving*)”动作,则由解释器停止该性质的检验.

4.2 性能实验

本节对于提出并实现的普适计算应用时空性质运行时验证方法的运行效率进行实验分析.

(1) 实验设计

我们设计了一种模拟的普适计算环境,包括具有空间包含关系的房间,随机设置的用户在这些房间中进行动作,房间中也配置有相应的模拟传感器、设备等,模拟产生数据,包括用户位置、用户动作、环境状态数据等,这些数据在系统中以 RDF 三元组存储.在本实验中,我们以每个运行状态产生的三元组数目体现应用规模,如一个含有 5 个房间、5 类传感器、4 个用户的家居场景(如老人看护)每个运行状态产生 50 个左右的三元组;一个含有 500 个房间、10 类传感器、300 个用户的办公楼场景每个运行状态产生 5 000 个左右的三元组.在运行时,监控器中的应用状态翻译器将当前运行状态翻译为 Ambient Calculus 表示的形式系统.

我们用 Ambient Logic 描述了待验证的运行时性质.在这些性质中,一半性质设定为较为简单,包含 1~2 个原子公式(如第 2.1 节的性质(2));另一半较为复杂,包含 3 个或 3 个以上原子公式(如第 2.1 节的性质(1)).实验中涉及的性质基本皆是类似于第 2.1 节的 3 个例子,针对普遍的用户、位置、设备等实体进行描述而不是针对一个预先给定的特定实体,对于这类性质进行验证的实验,可使得实验结果更具有一般性.

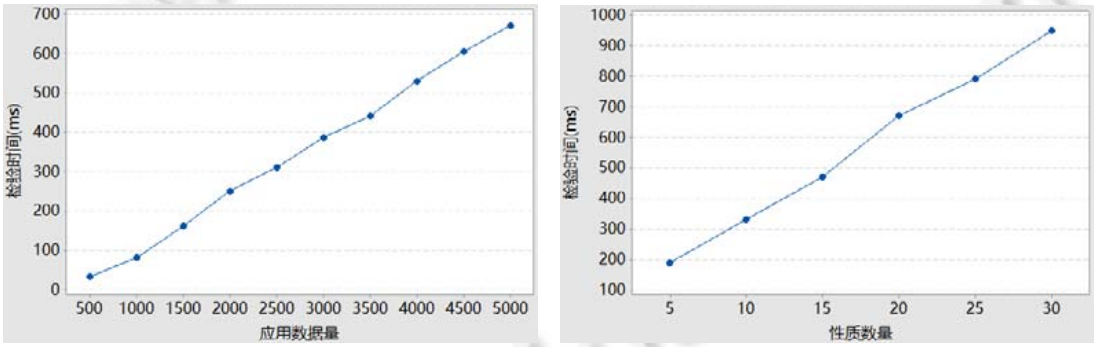
我们分别进行了以下两组实验.

- 实验 1:固定性质数量为 20 条(其中 10 条涉及时空性质),以不同规模的模拟数据量(即 RDF 三元组个数)进行实验,分别在每次运行状态数据量为 500 个~5000 个三元组的不同规模下记录运行时验证的时间开销,考察应用状态数据量对于验证时间的影响;
- 实验 2:固定每次运行状态数据规模为 5 000,以性质数量分别为 5~30(其中一半左右涉及时空性质)的情况进行实验,记录运行时验证的时间开销,考察性质数量对于验证时间的影响.

上述实验运行以 4G RAM,Intel 3.2GHz 双核 CPU 的计算机作为服务器运行,网络带宽等因素保持不变,忽略其影响.每组实验进行 20 次取平均值.

(2) 实验结果与讨论

图 3(a)和图 3(b)分别展示了两组实验的结果.



(a) 实验 1:应用数据量与时间开销的关系 (b) 实验 2:性质数量与时间开销的关系

Fig.3 Results of the performance evaluation

图 3 性能评估实验的结果

从这个结果中我们可以看出:随着计算系统数据规模的扩大、性质数量的增加,运行时验证的时间开销会相应地增加.尽管时间开销仍可能受到这两个因素之外的其他因素(如具体应用相关的性质复杂程度,本实验已尽可能地设置了多种类型的性质)影响,但是通过本实验可以看出:本文提出的运行时验证方法的时间开销是可以接受的,每次运行状态产生 5 000 个左右三元组的普适计算系统、30 条性质已可满足一个较大规模的智能环境的要求,在这样的情况下,运行时验证的时间开销在 1s 以内.

5 相关工作

普适计算领域的运行时验证技术已得到国内外研究者广泛关注,已有若干工作提出了针对医疗护理[25]、自治车辆[26]等典型普适计算应用,以及 CPS(cyber-physical system)[27]、云计算[28]等相关计算模式的运行时验

证方法.这些方法基本都是使用时态逻辑的各种变种作为逻辑基础进行验证,没有空间性质方面的考虑.

对于空间性质的验证而言,使用 Ambient Calculus 进行系统状态规约、使用 Ambient Logic 进行性质规约是一种常用的方法,已有 Ranganathan[17],Coronato[18]等研究者进行了这样的尝试.针对 Ambient Logic 在有限轨迹时间性质方面表达能力的不足,Coronato[18]以及我们的前期工作[19]分别给出了使用绝对时间定量地进行时间性质表达的做法,相比于这类尝试,采用定性的三值逻辑语义更加具有一般性.

引入三值逻辑是在轨迹有限或者状态不全情况下进行时态逻辑性质验证的常用手段.Brus 等人[29]将三值语义引入了局部克里普克结构(partial Kripke structure);Bauer 等人[20]提出了三值 LTL 语义,该语义被 Yu 等人[27]采用进行 CPS 的验证;Wei 等人[23]提出了三值 CTL 语义,并用以验证普适计算中的部分性质.这一系列工作对我们有所启发,但是这些工作仅有时间性质的验证手段,对于时空性质的验证,仍需要给出新的形式工具.

6 总结与展望

本文针对普适计算应用的时空性质运行时验证问题进行研究:首先,将三值语义引入 Ambient Logic 提出了 AL_3 方法;在此基础上,进一步地设计实现了支持基于 AL_3 进行运行时验证的性质检验算法和运行时监控器.在未来工作中,我们考虑进行以下方向的推进:首先,考虑改进验证算法,如采用滑动窗口等方法减少空间开销;其次,考虑进一步地在更真实的场景中进行应用的实践,并面向真实应用的需求,对于监控器的设计、性质的定义方式进行改进.

References:

[1] Weiser M. The computer for the 21st century. Scientific American, 1991,261(30):94−104. [doi: 10.1145/329124.329126]

[2] Satyanarayanan M. Pervasive computing: Vision and challenges. IEEE Personal Communications, 2001,8(4):10−17. [doi: 10.1109/98.943998]

[3] Dey AK, Abowd GD, Salber D. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction, 2001,16(2):97−166. [doi: 10.1207/S15327051HCI16234_02]

[4] Román M, Hess C, Cerqueira R, Ranganathan A, Campbell RH, Nahrstedte K. A middleware infrastructure for active spaces. IEEE Pervasive Computing, 2002,1(4):74−83. [doi: 10.1109/MPRV.2002.1158281]

[5] Kulkarni D, Ahmed T, Tripathi A. A generative programming framework for context-aware CSCW applications. ACM Trans. on Software Engineering and Methodology (TOSEM), 2012,21(2):11. [doi: 10.1145/2089116.2089121]

[6] Gu T, Pung HK, Zhang DQ. Toward an OSGi-based infrastructure for context-aware applications. IEEE Pervasive Computing, 2004,3(4):66−74. [doi: 10.1109/MPRV.2004.19]

[7] Arcelus A, Jones MH, Goubran R, Knoefel F. Integration of smart home technologies in a health monitoring system for the elderly. In: Proc. of the 21st IEEE Int'l Conf. on Advanced Information Networking and Applications (AINA) Workshops. 2007. 820−825. [doi: 10.1109/AINAW.2007.209]

[8] Julien C, Roman GC. Egospaces: Facilitating rapid development of context-aware mobile applications. IEEE Trans. on Software Engineering (TSE), 2006,32(5):281−298. [doi: 10.1109/TSE.2006.47]

[9] Artho C, Barringer H, Goldberg A, Havelund K, Khurshid S, Lowry M, Pasareanu C, Rosu G, Sen K, Visser W. Combining test case generation and runtime verification. Theoretical Computer Science (TCS), 2005,336(2):209−234. [doi: 10.1016/j.tcs.2004.11.007]

[10] Zhang X, Dong W, Qi ZC. Conflicts detection in runtime verification based on AOP. Ruan Jian Xue Bao/Journal of Software, 2011, 22(6):1224−1235 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4016.htm [doi: 10.3724/SP.J.1001.2011.04016]

[11] Bakhouya M, Campbell R, Coronato A, De Pietro G, Ranganathan A. Introduction to special section on formal methods in pervasive computing. ACM Trans. on Autonomous and Adaptive Systems (TAAS), 2012,7(1):6. [doi: 10.1145/2168260.2168266]

[12] Ranganathan A, Campbell RH. An infrastructure for context-awareness based on first order logic. Personal and Ubiquitous Computing (PUC), 2003,7(6):353−364. [doi: 10.1007/s00779-003-0251-x]

[13] Pnueli A. The temporal logic of programs. In: Proc. of 18th Annual Symp. on Foundations of Computer Science. IEEE, 1977. 46−57. [doi: 10.1109/SFCS.1977.32]

[14] Emerson EA, Halpern JY. Decision procedures and expressiveness in the temporal logic of branching time. In: Proc. of the 14th Annual ACM Symp. on Theory of Computing. ACM Press, 1982. 169−180. [doi: 10.1145/800070.802190]

[15] Cardelli L, Gordon AD. Mobile ambients. In: Proc. of the Int'l Conf. on Foundations of Software Science and Computation Structure. Springer Berlin Heidelberg, 1998. 140−155. [doi: 10.1007/BFb0053547]

[16] Cardelli L, Gordon AD. Ambient logic. In: Proc. of the Mathematical Structures in Computer Science. 2003.
[17] Ranganathan A, Campbell RH. Provably correct pervasive computing environments. In: Proc. of the 6th IEEE Int'l Conf. on Pervasive Computing and Communications (PerCom). 2008. 160−169. [doi: 10.1109/PERCOM.2008.116]
[18] Coronato A, De Pietro G. Formal specification of wireless and pervasive healthcare applications. ACM Trans. on Embedded Computing Systems (TECS), 2010,10(1):12. [doi: 10.1145/1814539.1814551]
[19] Li XS, Tao XP, Lü J, Song W. Specification and runtime verification for activity-oriented context-aware applications. Ruan Jian Xue Bao/Journal of Software, 2017,28(5):1167−1182 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5215.htm [doi: 10.13328/j.cnki.jos.005215]
[20] Bauer A, Leucker M, Schallhart C. Runtime verification for LTL and TLTL. ACM Trans. on Software Engineering and Methodology (TOSEM), 2011,20(4):14. [doi: 10.1145/2000799.2000800]
[21] Charatonik W, Zilio SD, Gordon AD, Mukhopadhyay S, Talbot J. Model checking mobile ambients. Theoretical Computer Science, 2003,308(1-3):277−331. [doi: 10.1016/S0304-3975(02)00832-0]
[22] Li XS, Tao XP, Lü J. Programming method and formalization for activity-oriented context-aware applications. In: Proc. of the 12th Int'l Conf. on Ubiquitous Intelligence and Computing (UIC). IEEE, 2015. 174−181. [doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.48]
[23] Wei H, Huang Y, Cao J, Ma XX, Lü J. Formal specification and runtime detection of temporal properties for asynchronous context. In: Proc. of the 10th Int'l Conf. on Pervasive Computing and Communications (PerCom). IEEE, 2012. 30−38. [doi: 10.1109/PerCom.2012.6199846]
[24] Kleene S. Introduction to Metamathematics. Wolters-Noordhoff, 1971.
[25] Jiang Y, Liu H, Kong H, Wang R, Hosseini M, Sun J, 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. ACM Press, 2016. 112−121.
[26] Kane A, Chowdhury O, Datta A, Koopman P. A case study on runtime monitoring of an autonomous research vehicle (ARV) system. In: Proc. of the 6th Int'l Conf. on Runtime Verification. LNCS 9333. Springer Int'l Publishing, 2015. 102−117. [doi: 10.1007/978-3-319-23820-3_7]
[27] Yu K, Chen Z, Dong W. A predictive runtime verification framework for cyber-physical systems. In: Proc. of the 8th IEEE Int'l Conf. on Software Security and Reliability-Companion. IEEE, 2014. 223−227. [doi: 10.1109/SERE-C.2014.43]
[28] Zhou J, Chen Z, Wang J, Zheng Z, Dong W. A runtime verification based trace-oriented monitoring framework for cloud systems. In: Proc. of IEEE Int'l Symp. on Software Reliability Engineering Workshops. IEEE, 2014. 152−155. [doi: 10.1109/ISSREW.2014.84]
[29] Bruns G, Godefroid P. Model checking partial state spaces with 3-valued temporal logics. In: Proc. of the Int'l Conf. on Computer Aided Verification. Berlin, Heidelberg: Springer-Verlag, 1999. 274−287. [doi: 10.1007/3-540-48683-6_25]

附中文参考文献:

[10] 张献,董威,齐治昌.基于 AOP 的运行时验证中的冲突检测.软件学报,2011,22(6):1224−1235. http://www.jos.org.cn/1000-9825/4016.htm [doi: 10.3724/SP.J.1001.2011.04016]
[19] 李暄松,陶先平,吕建,宋巍.面向动作的上下文感知应用的规约与运行时验证.软件学报,2017,28(5):1167−1182. http://www.jos.org.cn/1000-9825/5215.htm [doi: 10.13328/j.cnki.jos.005215]

李暄松(1985－),男,山东博兴人,博士,讲师,CCF 专业会员,主要研究领域为软件工程与方法学,形式化方法,普适计算技术.

宋巍(1981－),男,博士,副教授,CCF 高级会员,主要研究领域为软件工程与方法学,形式化方法,服务计算.

陶先平(1970－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件中间件技术,网构软件方法学,普适计算技术.