第 47 卷第 3 期
2015 年 6 月

南 京 航 空 航 天 大 学 学 报
Journal of Nanjing University of Aeronautics & Astronautics

Vol. 47 No. 3
Jun. 2015

DOI:10.16356/j.1005-2615.2015.03.015

多情境感知环境下 RFID 复合事件检测算法

李博涵[1] 李东静[1] 王学良[2] 刘亚丽[1] 张 潮[1] 夏 斌[1]

(1.南京航空航天大学计算机科学与技术学院,南京,210016;
2.黑龙江大学信息科学与技术学院,哈尔滨,150080)

摘要:无线射频识别(Radio frequency identification,RFID)技术的发展让感知世界变得更加便利。本文针对 RFID 数据特点以事件为中心进行数据建模,在多约束环境下利用各种规则和模式,有效地检测出特定的复杂事件。通过对 RFID 事件的分析和处理,利用抽象的底层 RFID 事件的检测对复杂事件进行有效监控和处理。根据 Cascadia 系统对 RFID 数据的事件分类与检测给出了利用事件解析图的事件检测方法。提出了在不同的约束执行环境中各操作的事件检测算法,结合具体应用实例对采用事件解析图的 RFID 复杂事件检测模型进行了解释说明。实验结果表明本文提出的 RFID 复杂事件检测算法能够较好地满足正确性要求,并提高了 RFID 系统整体的执行效率。

关键词:无线射频识别;复杂事件检测;事件解析;检测算法

中图分类号:TP393 **文献标志码**:A **文章编号**:1005-2615(2015)03-0412-09

Operation and Detection Algorithm of Composite Event of RFID in Multi-Context Aware Environment

Li Bohan[1], *Li Dongjing*[1], *Wang Xueliang*[2], *Liu Yali*[1], *Zhang Chao*[1], *Xia Bin*[1]

(1. College of Computer Science & Technology, Nanjing University of Aeronautics & Astronautics, Nanjing, 210016, China;
2. College of Information Science and Technology, Heilongjiang University, Harbin, 150080, China)

Abstract: With the development of radio frequency identification (RFID) technology, RFID data processing technique grows rapidly in demand in the multi-context aware environment. According to the characteristics of RFID data, the event-centric event model is built. Composite events are abstracted by analyzing and processing the raw RFID data. And then the abstract low events can be monitored by RFID event detection. With the base of event specifying and detecting of RFID data in Cascadia, the integrated event operators and the main idea of detection algorithms are given based on event parsed tree. The detailed event detection algorithms are presented in the different constraint executing circumstances. The theoretic analyses of the algorithms are also given accordingly. Furthermore, the RFID composite event detection model using event parse graph is explained by the applied examples. The experimental results indicate the efficiency and effectiveness of the algorithms in three varied constraint environments in RFID data management.

Key words: radio frequency identification (RFID); composite event detection; event parse; detection algorithm

基金项目:国家自然科学基金青年基金(41301407)资助项目;江苏省自然科学基金青年基金(BK20130819)资助项目;中央高校基本科研业务费(NZ2013306)资助项目;江苏省优势学科资助项目;研究生创新基金(Kfjj20151607)资助项目。

收稿日期:2015-02-12;**修订日期**:2015-04-23

通信作者:李博涵,男,副教授,E-mail:bhli@nuaa.edu.cn。

无线射频识别(Radio frequency identification,RFID)技术是一种通过无线射频方式进行非接触式双向无线通信的自动识别与获取技术,与传统的条形码相比,具有快速、实时、可重复使用、穿透性强、环境适应性强、数据容量大等诸多优点。RFID 技术在军事、通信、交通、医疗和服务业等领域都有着广泛应用。RFID 数据处理作为核心支撑技术,已不仅仅局限于技术范畴的应用,而且已形成了广泛的经济价值链体系。在学术界关于 RFID 数据处理技术的研究近年来呈现出激增态势,内容涵盖底层感知技术、相关应用领域、安全及政策等诸多方面问题[1]。例如普及的各种智能移动终端、车载导航等,都具备互通互联的功能,并且能够根据 GPS、北斗、Bluetooth、RFID 等技术实现位置共享。工业界分析师预测,无线、移动终端将超过台式电脑,移动计算技术应运而生。这种技术能够提供多样化的服务,满足更多用户和企业需求。对物理的感知可以与政治的发展相结合,例如,GPS 信号和美国的 E911 授权,在基础设施和基于手机定位技术方面技术持续发展。这些技术已经广泛应用于基于城市规划与室内地图,以及机场或 MALL 等众多应用领域。但是在 RFID 时空相关性数据的建模和处理方面的研究还相对较少,特别是针对 RFID 的数据建模还很不完善。这主要是由于 RFID 的数据处理部分比较复杂,应用软件的可重用性、可伸缩性都非常低。当前的发展趋势是为 RFID 应用提供基于的平台,在物理世界与逻辑世界间建立便捷的连接[2]。

目前多数 RFID 平台主要关注 RFID 底层数据过滤与搜集,如 Weblogic RFID Edge Server 和 Java System RFID Software 等。RFID 平台是 ALE(Application level events)[3]标准的一种实现途径。由 EPC(Electronic product code)Global 组织提出的且作为 RFID 标准的 ALE 也是关注于底层数据,且不会附加 EPC 数据的语义[4]。ALE 层面的数据可以抽象出多种类型的 RFID 事件。早期 RFID 开发中数据直接传送给应用程序,应用程序负责将源数据解释为逻辑业务数据。这种以数据为中心的方法采用传统数据库技术对 RFID 数据进行建模,并将数据存储于 DBMS 中。该方法对 DBMS 依赖性强,开发事件处理机制需要扩展 DBMS,适应性较低。通过以上分析可以发现 RFID 数据处理技术更多的是关注于高层的信息处理,包括了 EPC 数据转化等定制功能。但是由于从原始 EPC 数据到 EPC 业务信息转化包含大量隐含语义,因此对底层原子事件的检测就显得尤为重要。现有的研究对复杂事件在多种约束环境下的检测主要集中在数据的检测,而对含有丰富语义的时空数据的事件建模能力较弱。首先参照 Cascadia 系统处理的 RFID 数据的事件模型,提出 3 种基本约束环境和归纳出 3 类 6 种事件操作,进而给出事件解析树的定义和利用事件解析树和事件解析图进行事件检测的实例。基于事件解析图给出了在不同约束环境中包含各种事件操作的复杂事件检测算法。

1　相关工作

RFID 事件处理的主要目的是进行事件检测,即按照给定的规则和模式,检测出指定的事件。下面分别介绍两种解决方案:以数据为中心的方法和以事件为中心的方法。形式化的事件检测模型是事件检测正确性验证的一个重要基础,也是实现 RFID 技术的理论保证。

以数据为中心的方法是 RFID 处理系统中最早采用的方法。采用传统的数据库技术对 RFID 数据进行建模,并将数据保存在数据库中,在 DBMS 基础上支持事件检测,具有代表性的系统有 Simens 公司的 RFID 系统[5]。Simens 的 RFID 系统提出了面向时态模式的 ER 模型,不仅能描述 RFID 数据的特征,还能够表达业务逻辑,并支持基于分片的存储。基于规则的框架提供自动的 RFID 数据过滤、转换和聚合,生成高层次的语义数据。在 DBMS 基础上利用规则可进行复杂事件检测。

以事件为中心的方法是一种更高级别的 RFID 事件处理技术,通过事件建模直接对复杂事件进行高效率的处理。主动数据库中事件检测技术和数据流系统技术都与这种技术有着一定的相关性。其中主动事件建模和主动复杂事件处理方法可为以事件为中心的 RFID 事件处理技术提供研究依据。但由于主动数据库的处理技术没有考虑数据流的特性和语义的复杂性,并且主动数据库主要支持静态的事件分析,因而不能直接应用于 RFID 数据处理[6-7]。另一方面,RFID 数据处理主要考虑同一事件流上不同事件之间的相关性和限制,不同于 Publish/Subscribe 系统中的简单事件过滤和数据流处理中的基于关系代数的处理[8-9],因此需要重新设计能够在内存中完成的高效、实时和增量的事件处理机制。Cascadia 系统给出了 RFID 的 3 种数据模型,分别是位置模型、实体模

型和事件模型[10]。该系统从RFID原始数据中提取和管理高层事件,特别针对不确定和模糊的原始数据采用概率事件建模方法对高层事件进行了定义、检测和管理,其关键步骤在于对事件的检测。然而,由于底层事件本身存在时间或空间特性,其高层事件的执行顺序可能与底层事件的顺序也相关,因此复杂事件的可检测性并不能得到保证。最终也就造成了复杂事件的定义和检测之间存在了一道天然鸿沟[11]。若需要高效地对高层事件进行定义,检测等工作,首先就需要对底层事件进行定义和检测。原子事件定义并不复杂,但是当RFID数据具有时空性、动态性和关联性等特征时[12],其底层事件检测效果的优劣将直接决定高层复杂事件的定义、检测和管理,因此对RFID数据处理最具挑战性的工作就是在物理世界和标签之间建立一座桥梁[13]。文献[14]提出了一种在应用层进行近似查询和计算的方法。但是,底层RFID复杂事件检测算法需要完备的逻辑表达来确保其高层复杂事件的正确性和一致性,进而实现类似KNN等个性化查询以及多情境感知功能[15]。

2 RFID事件处理模型与算法

RFID底层复杂事件的检测是支持高层复杂事件的检测和管理的重要支撑环节。Cascadia的事件模型(Event model)是其系统的核心,是连接API和RFID基础结构的桥梁。实际上,Cascadia系统处理的RFID数据包含了时空数据,根据其位置模型和实体模型的建模,事件模型可以抽象出3类6种事件操作原语。(1) with and without:代表实体间的邻接关系;(2) inside and outside:代表了实体间的包含关系;(3) near and far:代表了实体间距离和范围的关系,具有模糊语义。

分析以上典型事件模型的查询语义可以发现其中主要考虑事件的位置、空间关系,这些操作都定义了事件发生时单点事件的空间抽象位置。而实际应用中实体标签本身可以具有移动特征,由此存在着大量含有“AND”、“OR”、“同时”等语义的查询。此外,现有的研究没有重点考虑对于复杂事件更进一步复合操作,特别是例如判断两个复杂事件是否重合或包含这样的复杂语义。而且对于前面提出的RFID复杂事件检测的特点,很少有模型能够做到全面地检测。

2.1 事件解析图

RFID事件检测的语法规则本身并不复杂,和主动数据库中ECA规则比较类似。该规则定义了事件执行的整个行为,而且每当检测系统发现事件,便开始检测是否存在关联的条件[1]。但是RFID复杂事件的语义规则却比较丰富。不同于Petri网模型对复杂事件检测的方法[16-17],利用事件解析树检测复杂事件可以合并复杂事件的公共子表达式,这样可以实现多事件共享公共事件节点[18]。

定义1 事件解析树是由叶节点、父节点(分支节点)以及边构成。其中,叶节点代表原子事件,而父节点对应于复杂事件。除根节点外,每个节点都有一对边,分为入边和出边。事件操作沿着边由叶节点向根节点传播。

如图1(a)所示为事件解析树与叶节点结构的示例;图1(b)为由事件解析树构成的事件解析图。

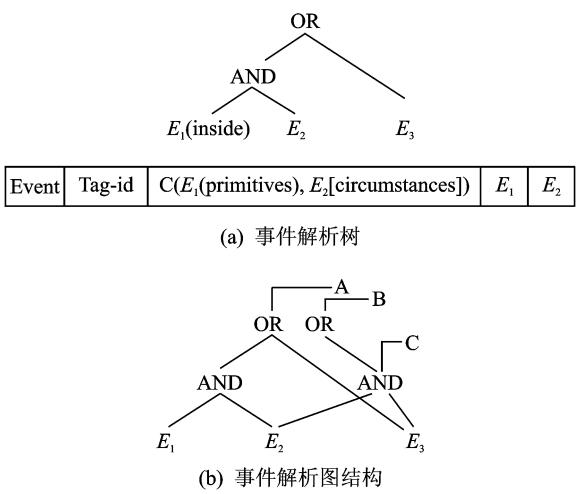

图1 复杂事件检测示例

Fig. 1 Example of composite event

构建事件解析树遵从以下规则。假定在事件检测过程中RFID规则集为$R=\{r_1,r_2,\cdots,r_n\}$,其构建过程如下:

(1) 规则1:为每一条规则的事件构建一棵子树。即通过对每一个$r_i\in R$分别定义规则,借此构建全局事件解析树。其中全局解析树T_i的叶子节点代表原子事件,中间节点代表复杂事件。入边和出边连接着原子事件和复杂事件或者复杂事件和复杂事件。T_i的根结点代表了规则r_i所定义的复杂事件部分。

(2) 规则2:由于遵从自底向上原则,传播间隔限制条件。对于每一个事件解析树T_i,$E_i\in T_i$,E_i向其所有子节点传播间隔限制。复杂事件的存在时间间隔总是长于其组成事件,同样上层节点也必然在子节点构建之后。因此,在检测算法中,若含有时间顺序(例如Conjunction和Sequence)约束

限制操作符,则组成事件按时间先后顺序进行,否则,只需累积其时间周期。

(3) 规则 3:在形成多个事件解析树后,对于每一个形成的 T_i,将共有的子树部分结合形成新的事件解析树。这样可以避免对相同组成的分支事件进行多次的检测以提高时空利用率。

根据不同约束环境事件,事件操作原语在时间上有 3 种基本的约束环境中执行:①Conjunction:将同时发生的或近似同时发生的事件检测为一个复杂事件。②Duration:在时间段上延续一个事件的过程,具有事件累积效应。③Sequence:事件在发生时间的先后顺序,不具有事件累积效应。Cascadia 系统可以通过概率事件抽取器对不确定性 RFID 事件进行检测,但是其事件在不同约束环境下的操作并没有给出语义分析和复杂事件的检测算法。故通过对 Cascadia 系统的 3 种基本模型的改进,利用事件解析树来表示 RFID 复杂事件,进而基于事件解析图实现了 RFID 复杂事件检测的算法。

Conjunction 约束环境属于非周期操作,监测事件和终止事件是不同事件,而对于 Duration 和 Sequence 操作,两者则是同一事件。根据语义的不同,3 种约束环境细分为以下操作:

(1) Conjunction 约束环境

① AND 操作:事件 E_1,E_2 的合取,记为 $E_1E_2\Delta$。当 E_1 发生且 E_2 发生时发生,不论事件以何种顺序发生,$E_1E_2\Delta$ 总要发生。E_1,E_2 既可以做初始事件,也可以做终止事件。其复杂事件可以形式化地表示为:$E_1E_2\Delta=\exists t,t'(t_1\leqslant t\leqslant t_2\wedge t_1\leqslant t'\leqslant t_2\wedge(((E_1[t_1,t]\wedge E_2[t',t_2])\vee(E_1[t',t_2]\wedge E_2[t_1,t])))$。

② OR 操作:事件 E_1,E_2 的析取,复杂事件记为 $E_1E_2\nabla$。当 E_1 发生或者 E_2 发生时,$E_1E_2\nabla$ 发生。同样地,E_1 和 E_2 既可以做初始事件,也可以做终止事件。其复杂事件可以形式化地表示为 $E_1E_2\nabla=E_1[t_1,t_2]\vee E_2[t_1,t_2]$。

③ NOT 操作:假设有事件 E_1,E_2,E_3,在由 E_1 的终止时刻,E_3 的初始时刻组成的时间闭区间内,监测到事件 E_2 没有发生时 NOT 事件发生。就可以表示为 $(\neg(E_2)[E_1,E_3],[t_1,t_2])=(E_1,t_1)\wedge(E_3,t_2)\wedge\neg(E_2,[t_1,t_2])$。

需要说明的是,对于 Conjunction 的语义扩展已经不同于一般的主动规则所代表的合取语义,因此可对大量事件实时读取的 RFID 数据进行更加全面地事件检测。

(2) Sequence 约束环境

Sequence 约束环境除包含以上 3 种操作外,还有 2 种特殊操作:

① 非累积非周期的事件操作表示为 A,表示两个事件 E_1,E_2 的顺序发生。当 E_2 发生时,E_1 已经发生,Sequence 事件发生。这隐含着 E_1 的终止时间值 t 小于 E_2 的初始的时间值 t'。E_1 是顺序事件的初始事件,E_2 是终止事件。例如,(E_1,E_2,E_3,A)表示复杂事件发生,仅当 E_2 每次发生时间都落在由 E_1 的终止时刻和 E_3 的初始时刻构成的时间区间内。其复杂事件可以形式化地表示为 $E_1E_2A=\exists t,t'(t_1\leqslant t<t'\leqslant t_2\wedge(E_1,[t_1,t]\wedge(E2,[t',t_2])))$。

② 非累积周期性事件操作表示为 P,表示周期性发生的事件。其复杂事件可形式化地表示为 $(E_1,T,E_3,P)=\exists t'<t\,\exists i\in Z^+(t=t'+T_i\wedge(E_1,t')\wedge\neg As(E_3,[t'+1,t]))$。其复杂事件可以表示为 (E_2,T,E_1E_3,P)。T 为周期表达式。周期性事件在 E_1 的初始时刻与 E_3 的终止时刻构成的时间闭区间内,每隔周期 T 发生一次。E_1 是初始事件,E_2 是监测事件,E_3 是终止事件。

(3) Duration 约束环境

同样地,Duration 除了包含 Conjunction 的 3 种操作外,也有 2 种特殊操作:

① 累积非周期事件操作可表示为 A^*,非周期操作符可以表达在一个封闭的时间段内的一个非周期事件的发生。累积非周期事件可表示为 E 可以表示为 (E_1,E_2,E_3,A^*)。在由 E_1,E_3 构成的闭区间内累积所有 E_2 的发生,直到事件 E_3 发生,则事件 E 发生。A^* 操作符的形式化定义如下:$(E_1,E_2,E_3,A^*)=(E_2,A^*)\wedge\exists t<t_1(E_1\wedge\neg As(E_3,[t+1,t_2]))$。

② 累积周期事件操作可表示为 P^*,累积周期事件可以表示为:$E_1E_3P^*$。与 P 不同的是,仅当 E_3 发生,P^* 只发生一次。其复杂事件可形式化地表示为 $(E_1,T,E_3,P^*)=\exists t'<t\,\exists i\in Z^+(t=t'+n_i\wedge(E_1,t')\wedge\neg As(E_3,[t'+1,t]))$。

由以上操作语义可看出 Duration 与 Sequence 显著的区别是事件累积效应。而 Conjunction 约束环境不具有周期和顺序特性,故也不具有累积作用。

定义 2　复杂事件的第 1 个事件作为左叶节点事件,复杂事件的第 2 个事件作为右叶节点事件。为便于在后续事件检测算法中描述,分别将 AND,OR 操作的两个事件称为左部事件(LPE)和右部事件(RPE)。而对于 Not,A,P,A^*,P^* 操作的中间的叶子事件称为中间事件(MPE)。

复杂事件的检测是一个识别复杂事件发生的

过程,并在此过程中收集和记录事件的各个参数。因为在事件解析图模型中复杂事件的检测算法是基于事件解析树的,每个复杂事件用一个事件解析树来表示,事件解析树的叶节点是原子事件,事件解析树的中间节点是事件操作符;对于有着公共子表达式的事件解析树进行合并,构成一个事件解析图。在用户进行规则定义后,系统为每个事件表达式建立事件解析图。建立事件解析图时,不仅要读取规则定义中的事件表达式,还要读取用户的事件表达式。事件解析树经过公共子表达式合并可以实现事件共享节点,避免同一事件多次检测,减少数据计算并节省存储空间。

2.2 RFID 复杂事件检测算法

通过以上分析可知事件解析图表达能力强、描述规范,能够很好解决时序参数处理问题,而且可以弥补 Petri 网公共事件子表达式重复存储和检测的缺陷[11]。基于利用事件解析图模型来检测事件的优点,本文提出 3 种约束环境下的多种可匹配操作的复杂事件检测算法。算法基本思想是:当原子事件发生时,首先读取代表本事件的叶节点,然后叶节点沿着边将操作传给父节点,并存入相应的事件列表。在此进行复杂事件的检测,若有复杂事件发生,将事件实体 tag 存入事件列表。在分支节点设置一个标记,用来标记此复杂事件是否为系统定义的事件,标记为空,说明该复杂事件只是系统定义事件的成员事件(构成分支节点),否则通知系统复杂事件的发生。我们利用事件解析图模型分别给出 3 种约束环境下的多种可匹配操作的复杂事件检测算法。

(1) Conjunction 环境下的复杂事件检测

首先给出 Conjunction 环境下的事件检测算法的基本思想和性质描述,然后分别给出其参数传递算法和复杂事件检测算法。算法均采用自底向上的由叶节点到根节点。如果一个原子事件的发生引发复杂事件,则会在相应的复杂事件的节点做标记,并向系统汇报复杂事件的发生;否则没有复杂事件发生,不会在节点做标记。算法 1 对 Conjunction 环境下的 3 种操作下节点对 LPE,RPE 处理,或者 LPE,RPE,MPE 的处理都充分考虑,每个事件的发生均可执行事件检测。

算法 1 Con_Oper_Detec (node, parameter_list)

```
Input: node;
Output: parameter_list; /* 有关参数实体 tag 单元列表 */
begin
case branch_node: /* 根据规则 1、2 和对应事件操作进行 */
AND:
  if node 是 LPE then
    if E2 的列表≠Ø then
      for E2 中的每一个 e2 满足(t_s(e2)≤t_s(e1)∧(t_e(e2)≤t_e(e1))do
        将(<e2,e1>,[t_s(e2),t_e(e1)])存到本节点;
        将 e1 的新加实体 tag 添加到 E1 的列表;
  call Detect_Compo (node, parameter_list);
    if node 是 RPE then
      if E1 的列表≠Ø then
        for E1 中的每一个 e1 满足(t_s(e1)≤t_s(e2)∧(t_e(e1)≤t_e(e2)) do
          将(<e1,e2>,[t_s(e1),t_e(e2)])存到本节点;
          将 e2 的新加实体 tag 添加到 E2 的列表;
    call Detect_Compo (node, parameter_list);
OR:
      for 任何事件发生 do 将发生的事件实体 tag 存到本节点 do
      call Detect_Compo (node, parameter_list);
NOT:
      if node 是 LPE then
        将 e1 的新加实体 tag 添加到 E1 的列表;
      if node 是 MPE then
        if E1 的列表≠Ø 且(t_e(E1 中的最早终止时刻)<t_s(e2)) then
          将 e2 的新加实体 tag 添加到 E2 的列表;
        if node 是 RPE then
          if E1 的列表≠Ø then
            for E1 列表中每一个 e1 满足(t_s(e2)>t_e(e1)) do
              for E2 列表中的每个 e2 满足(t_e(e2)<t_e(e1)∧t_s(e3)<t_s(e2)) do
                将(<e1,e2,e3>,[t_e(e1),t_s(e3)])存到本节点;
                  call Detect_Compo (node, parameter_list);
end.
```

算法 2 Detect_Compo (node, parameter_list)

```
Input: occur event;
Output: composite event;
begin
    if 检查本节点标记,若是系统定义的事件 then
       通知系统原子或复杂事件发生;
       /* 在此输出复杂事件 */
       if 节点 node 的父节点 node_parent≠Ø then
          由规则 2 将子节点 node 及其参数复制到节点的父节点 node_praent 中;
          call Con_Oper_Detec (node_praent, parameter_list);
```

end.

算法说明：Conjunction 环境下的复杂事件检测是一个递归调用的过程，包含两个子算法：算法 Con_Oper_Detec 和算法 Detect_Compo，其中的 case 语句，for 循环语句和 if 判断语句都是可终止的。当一个原子事件发生时，激活的叶节点首先判断是不是系统所需的事件，若是向系统发信号。然后检验是否存在父节点；如果不存在，则算法终止；如果存在父节点，则调用算法 Con_Oper_Detec，判断是何种操作，并将检测到的复杂事件存到本节点的实体 tag 列表中，调用算法 Detect_Compo。整个过程自底向上判断，直到不满足或者没有父节点时终止算法。算法中设整个事件解析图的节点数为 n，自底向上检测复杂事件全部是由二元操作符构成，其检测过程的时间复杂性为 $O(\log_2 n)$，而对于其他全部由二元操作符构成复杂事件检测过程的时间复杂性为 $O(\log_2 n)$。

(2) Sequence 环境下的复杂事件检测

Sequence 约束环境对组成事件的时间先后要求强于 Conjunction。根据所包含的操作，Sequence 环境下的事件检测算法的基本思想如下。如果一个原子事件的发生引发复杂事件，则会在相应的复杂事件的节点做标记，并向系统汇报复杂事件的发生。算法对 Sequence 环境下的五种操作下节点在 LPE，RPE，或者 LPE，RPE，MPE 分别进行了考虑，对每个事件的发生均可执行事件检测。

算法 3　Seq_Oper_Detec (node, parameter_list)

```
Input: node;
Output: parameter_list;
begin
case branch_node:/*根据规则 1,2 和对应事件操作进行*/
AND:
    if node 是 LPE then
      if E2 的列表≠∅ 且((E2 head)≤t_s (e1) ∧ (t_e (E2 head)≤t_e (e1)) then
          将 (<E2 head, e1>, [t_s (E2 head), t_e (e1)]) 存到本节点;
          将 e1 从 E2 的表头列表中删除;
          call Seq_Detect_Compo (node, parameter_list)
          else 将 e1 的新加实体 tag 添加入到 E1 的列表;
    if node 是 RPE then
      if E1 的列表≠∅ 且(t_s(E1 head)≤t_s(e2) ∧ (t_e (E1 head)≤t_e (e2)) then
          将 (<E1 head,e2>, [t_s (E1 head), t_e (e2)])存到本节点;
          将 e2,E1 的表头节点从列表中删除;
          call Seq_Detect_Compo (node, parameter_list);
          else 将 e2 的新加实体 tag 添加到 E2 的列表;
OR:
    for 任何事件发生 do
        将发生的事件实体 tag 存到本节点;
        call Seq_Detect_Compo (node, parameter_list);
NOT:
    if node 是 LPE then
        将 e1 的新加实体 tag 添加到 E1 的列表;
    if node 是 MPE then
        if E2 的列表≠∅ 且 (t_e (e1) <t_s (e2)) then
          将 e2 的新加实体 tag 添加到 E2 的列表;
    if node 是 RPE then
      if E1 的列表≠∅ 且 (t_s (e2)>t_e (E1 head)) then
        if E2 的列表≠∅ then
          for E2 列表中的每个 e2 满足 (t_e (e2)<t_e (e1) ∧ t_s (e3) <t_s (e2)) do
              将(<E1 head,,e2,e3>, [t_e(E1 head),t_s (e3)])存到本节点;
              清空 E1,E2 的列表;
              call Seq_Detect_Compo(node, parameter_list);
        else
          将 (<E1 head, e2,e3>,[t_e (E1 head), t_s (e3)])存到本节点;
          将 E1 的表头节点从列表中删除;
          call Seq_Detect_Compo(node, parameter_ list);
A:
    if node 是 LPE then
      获取 node 的 eid;
      创建存储单元，存入 eid 和对应 E2 的时间表达式;
      将 e1 的新加实体 tag 添加到 E1 的列表;
    if node 是 MPE then
      获取 node 的 eid;
      if E1 的列表≠∅ 且 e1 的 eid 与 e2 的一致 then
        创建存储单元，存入 eid 和对应 E2 时间表达式;
        将 (<e1,e2>,[t_s (e2), t_e (e2)]) 存到本节点;
        call Seq_Detect_Compo (node, parameter_list);
    if node 是 RPE then
        if E1 列表≠∅ then
          if (t_e (e1) <t_s (e3)) then
            删除 E1 的表头节点;
P:
    if node 是 LPE then
      获取 node 的 eid;
      创建存储单元，存入 eid 和对应 E2 的时间表达式;
      将 e1 的新加实体 tag 添加到 E1 的列表;
    if node 是 MPE then
      获取 node 的 eid;
      if E1 的列表≠∅ 且 e1 的 eid 与 e2 的一致 then
        创建存储单元，存入 eid 和对应 E2 的时间表达
```

式；

将 e_2 的新加实体 tag 添加到 E_2 的列表；

if node 是 RPE then

if E_1 的列表≠Ø 且 (t_e (e_1) <t_s (e_3)) then

获取 e_1 的 eid；

将 E_2 中与 e_1 的 eid 一致的添加到临时列表 tempE_2；

删除 E_1 列表表头；

if tempE_2 非空 then

将(<eq_1, tempE_2, e_3>, [t_s (tempE_2 最早初始时刻), t_e (tempE_2 最后终止时刻)]) 存到本节点；

删除 tempE_2；

call Seq_Detect_Compo (node, parameter_list);

end.

算法 4 Seq_Detect_Compo (node, parameter_list)

Input: occur event;

Output: composite event;

begin

if 检查本节点标记,若是系统定义的复杂事件 then

通知系统复杂事件发生; / * 输出复杂事件 * /

if 节点 node 的父节点 node_parent≠Ø then

将节点 node 及其参数复制到节点的父节点 node_praent 中；

call Seq_Oper_Detec (node_praent, parameter_list);

end.

算法说明：算法 Seq_Oper_Detec 和 Seq_Detect_ Compo 中，case 语句，for 循环语句和 if 判断语句都是可终止的，其过程是一个递归调用的过程。当一个原子事件发生时，激活的叶子节点首先判断是不是系统所需的事件，若是向系统发信号。然后检验是否存在父节点，如果不存在，则算法终止；如果存在父节点，则调用算法 Seq_Oper_Detec，判断是哪种复杂操作符，并将监测到的复杂事件存到本节点的实体 tag 列表中，调用算法 Seq_Detect_ Compo。整个过程自底向上判断，直到不满足或者没有父节点时终止算法。设整个事件解析图的节点数为 n，自底向上监测复杂事件全部是由三元操作符构成，其监测过程的时间复杂性为 $O(\log_3 n)$，而对于其他全部由二元操作符构成复杂事件监测过程的时间复杂性为 $O(\log_2 n)$。由于 $O(\log_3 n)$ 小于 $O(\log_2 n)$，所以整个算法的时间复杂性为 $O(\log_2 n)$。

(3) Duration 环境下的复杂事件检测

类似于 Sequence 约束环境下的执行过程。算法是由叶节点到根结点的过程。如果一个原子事件的发生引发复杂事件，则会在相应的复杂事件的节点做标记，并向系统汇报复杂事件的发生；否则没有复杂事件发生，不会在节点做标记。算法的基本思想和分析与算法 3，4 相似，故此省略。

3 RFID 复杂事件检测实例和实验

基于 Petri 网模型的复杂事件检测仅匹配按照时间顺序到达的原子事件。传统的基于树或图的事件检测在过滤过程中也没有考虑到匹配基本事件的时空关系。例如，复杂事件中的左部事件没有发生，那么右部事件可能也会被过滤掉，不仅造成系统开销的浪费也会对系统的响应时间造成一定影响，因此已有方法在实际应用中的局限性是不容忽视的。下面针对应用实例解释本文所采用的事件解析图的 RFID 事件检测方法，并与采用 Petri 网模型方法进行对比说明。

物流实时位置跟踪需要对货品装卸操作进行记录。为了表达“一批货品被装载到某个货盘”这一业务逻辑，可以定义复杂事件 E(Item(n) primitives Pallet)，其中 primitives 是 Cascadia 系统中 RFID 事件处理模型的操作原语，Item 和 Pallet 是两个 EPC 事件类型，分别表示 RFID 读写器发现货物 tag 和货盘的 tag，n 代表货物的参数。

为了验证算法的实际作用从执行效率和正确性两方面说明检测算法的有效性。实验环境仿真环境配置如下：算法采用 C++实现，CPU Pentium D 3.0 GHz，内存 2 GB，Windows XP sp3，硬盘 1TB SATA。由于尚不具备真实的物流环境，实验在仿真环境下模拟完成。

参考 EPCISO18000-6 标准，标签 tag 数量不超过 1 000，根据不同实验要求在此范围内选择。效率结果采用效率加速比(Speed-up)概念，即引入事件检测算法的执行情况与无事件检测算法的执行情况的比值，并以此为评估参数值。此参数值不大于 1 代表效率较系统原有方法更高，越小代表效率越高。为便于测试和更加准确的评价算法效率，数据库更新等操作执行代价未予统计在内。

例 1 在 Conjunction 约束环境下，当检测到复杂事件 E(Item near Pallet)，则该事件可能表示即将开始装货或刚刚卸货开始。若需进一步判断则考虑是否进入装卸货状态则可直接通过检测复杂事件 E(Item near Pallet)E(Item inside Pallet) Δ 或¬E(Item outside Pallet)来判断。而在 Petri 网模型中，由于 Token 代表的实体 tag 在准备装货、装货直至卸货涉及到多次状态变迁，因此需要检测所有的复杂事件。而利用 RFID 事件在 Conjunction 约束环境下仅需要检测一次复杂事件操作即可。

例 2　在 Sequence 约束环境下，实体货物 tag 按时间顺序和空间距离关系进行检测。当检测到复杂事件 E(Item near warehouse)E(Item inside Pallet) E(Item far warehouse)Δ，则该事件表示一个货物从进入仓库，然后被装卸货，直至离开仓库的过程。而在 Petri 网模型中，Token 需经历至少四次的状态的变迁，建模开销较大。假设多台货车重复进行装卸这一任务。设以上过程用事件 E_1 来表示，由于事件解析图可以从已检测复杂事件通过新的事件操作来继续检测新的复杂事件，那么 E_1P 则表示了装卸货物这一事件的完整周期性执行过程。

例 3　在 Sequence 约束环境下，RFID 复杂事件检测记录了整个过程，但是却不具有累积效应。货品装卸数量无法判断。在 Duration 约束环境下，当检测到复杂事件 E(Item outside Pallet) E(Item inside Pallet)Δ 时，设 E_2 表示装载过程，那么 $E_2\mathrm{P}^*(n)$则表示了完整的装载事件，其中 n 为装载数量。当检测到复杂事件 E(Item with Pallet) E(Item without Pallet)∇时，表示货物在装载过程中可能出现了遗漏，对于这种非周期性偶然性事件的记录，设 E_3 表示遗漏过程，那么 $E_3\mathrm{A}^*(n)$ 则表示了完整的遗漏事件，其中 n 为遗漏数量。而在 Petri 网模型中，复杂事件的增量检测通过 Petri 网中标记的位置来描述的。周期或累积事件需要通过 Token 计算跃迁守护函数。如果状态成立，则引发跃迁并标记位置结点，只有当序列中的最后一个位置结点被标记后，确定复杂事件发生。因此其逻辑关系复杂，会造成检测的效率不高。

如图 2 所示，在非累积情况的 3 种约束环境下的加速比均小于 1，这表明引入基于事件解析树的事件检测算法机制对于系统最终完成复杂事件检测的效率是明显的。在 3 种约束环境下，Conjunction 约束环境执行效果相对效率更低，这是由于在同时发生多个事件时，算法本身对底层 RFID 数据处理是所考虑的模式决定的。为了避免多读，漏读，采用 Sequence 和 Duration 两种模式是更好的选择。在这两种环境下，执行效率基本相当。

如图 3 所示，在累积情况下，采用 Conjunction 和 Duration 两种约束环境执行效率也是较为理想的。检测算法设计之初充分考虑了累积情况，因此在 Duration 约束环境下，执行效率在多种模式下是最优的。

如图 4 所示为非累积的 3 种约束环境下的检测复杂事件的正确率结果。为了验证算法的作用，实验采用了提高近一个数量级的标签事件数量。由结果可知引入 RFID 事件检测算法对于系统最终完成复杂事件检测的正确性保证作用是显著的，均高于 95.1%。

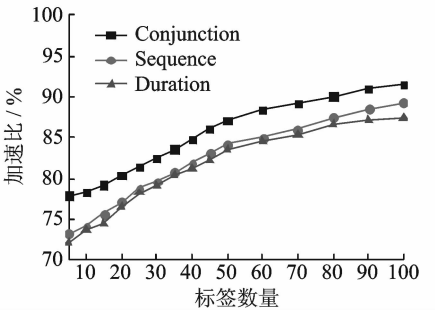

图 2　非累积的 3 种约束环境下的执行效率对比结果

Fig. 2　Efficiency in three non-duration constrained situations

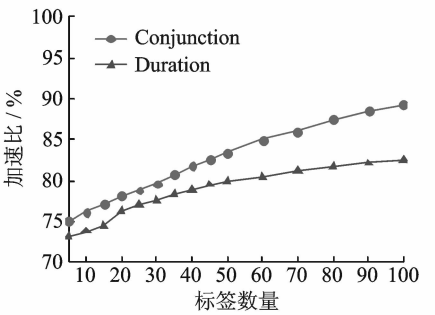

图 3　累积的两种约束环境下的执行效率对比结果

Fig. 3　Efficiency in two duration constrained situations

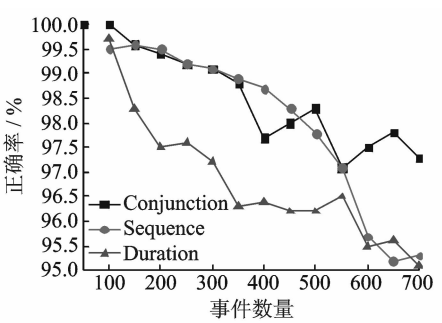

图 4　非累积的 3 种约束环境下的正确率

Fig. 4　Efficiency in three non-duration constrained situations

累积的 Conjunction 和 Duration 约束环境下正确率和非累积情况类似，其效率也是以事件的空间的代价来换取的，故此省略。

根据以上阐述可知，基于事件解析图的 RFID 复杂事件检测方法通过在检测复杂事件的同时有选择地处理基本事件，对有价值的复杂事件可以快速重复生成和利用，确保逻辑正确性、一致性和高效性。可应用于降低 RFID 复杂事件检测系统的

响应延迟。

4 结束语

机场的物流管理需要多情境感知环境下高效的复杂事件处理能力。本文针对多约束环境下RFID数据处理的特点对其进行以事件为中心的数据建模。本文在RFID事件的分析和处理过程中对抽象的底层RFID事件的检测实现复杂事件进行了有效监控和处理。给出了事件检测中的形式化事件操作定义,基于事件解析树和事件解析图提出了在不同的约束环境中各事件操作的事件检测的各种算法,并对事件检测算法给出理论分析和实验验证。但是本文所采用事件建模方法是基于确定事件的,RFID技术实际应用中由于存在漏读、多读、脏读数据等问他会产生其他不确定性事件,未来还需进一步对不确定事件的事件检测模型进行深入研究,利用真实环境对有源和无源RFID标签进行复合事件的处理。借助NFC实现基于多情境感知的支付功能。

参考文献:

[1] Ngai E W T, Moon K K L, Riggins F J, et al. RFID research: An academic literature review (1995—2005) and future research directions[J]. International Journal of Production Economics, 2008, 112(2): 510-520.

[2] Chawathe S S, Krishnamurthy V, Ramachandran S, et al. Managing RFID data[C]// Proceedings of the 30th International Conference on Very large Data-bases. Verlag: VLDB Endowment, 2004: 1189-1195.

[3] EPC global. The application level events (ALE) specification, version 1. 1[EB/OL]. http://www.epcglobalinc.org/standards/ale/ale_1_1-standard-core-20080227.pdf, 2008.

[4] Armenio F, Barthel H, Burstein L, et al. The EPC global architecture framework[J]. Oleg Ryaboy Sanjay Sarma Johannes Schmidt Kk Suen Ken Traub & John Williams, 2007, 55(4):488.

[5] Wang F, Liu P. Temporal management of RFID data[C]// Proceedings of the 31st International Conference on Very Large Data Bases. Verlag: VLDB Endowment, 2005: 1128-1139.

[6] Chakravarthy S, Krishnaprasad V, Anwar E, et al. Composite events for active databases: Semantics, contexts and detection[C]// VLDB. Verlag: VLDB Endowment, 1994: 606-617.

[7] 郝忠孝. 主动数据库系统理论基础[M]. 北京:科学出版社, 2009.
Hao Zhongxiao. Theoretical basis of active database system[M]. Beijing: Science Press, 2009.

[8] Fabret F, Jacobsen H A, Llirbat F, et al. Filtering algorithms and implementation for very fast publish/subscribe[J]. ACM, 2001, 30(2): 115-126.

[9] Carney D, Cetintemel U, Cherniack M, et al. Monitoring streams: A new class of data management applications[C]// Proceedings of the 28th International Conference on Very Large Data Bases. Verlag: VLDB Endowment, 2002: 215-226.

[10] Welbourne E, Khoussainova N, Letchner J, et al. Cascadia: A system for specifying, detecting, and managing RFID events[C]// Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services. Texas: ACM, 2008: 281-294.

[11] Wongpatikaseree K, Kim J, Makino Y, et al. Architecture for organizing context-aware data in smart home for activity recognition system[M]// Berlin Heidelberg: Springer Berlin Heidelberg, 2013.

[12] 谷峪, 于戈, 张天成. RFID复杂事件处理技术[J]. 计算机科学与探索, 2007, 1(3): 225-267.
Gu Yu, Yu Ge, Zhang Tiancheng. RFID complex event processing techniques[J]. Journal of Frontiers of Computer Science and Technology, 2007, 1(3): 225-267.

[13] Wang F, Liu S, Liu P. Complex RFID event processing[J]. VLDB Journal, 2009, 18(4): 913-931.

[14] Hao Zhongxiao, Li Bohan. Approximate query and calculation of reverse kNN based on voronoi cell[J]. Transactions of Nanjing University of Aeronautics & Astronautics, 2009, 26(2): 154-161.

[15] Lowe R, Mandl P, Weber M. Supporting generic context-aware applications for mobile devices[C]// PERCOM Workshops. San Diego, California: IEEE, 2013: 97-102.

[16] Riahi I, Moussa F. Formal modeling for pervasive design of human-computer interfaces[C]// UBICOMM 2014, The Eighth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies. Seattle, WA, USA: IARIA, 2014: 35-43.

[17] Rizvi S, Jeffery S R, Krishnamurthy S, et al. Events on the edge[C]// Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data. Texas: ACM, 2005: 885-887.

[18] Lowe R, Mandl P, Weber M. Context directory: A context-aware service for mobile context-aware computing applications by the example of Google Android[C]// PERCOM Workshops. Lugano, Switzerland: IEEE, 2012: 76-81.