基于智能物件的实时企业复杂事件处理机制*

臧传真 范玉顺
(清华大学国家 CIMS 工程技术研究中心 北京 100084)

摘要：实时企业能够主动地提供及时、准确、详细的信息，这些信息除了业务系统产生的大量数据、智能物件(Smart Items)数据流以外，更主要的是隐藏在其背后的信息。目前缺少功能强大的机制和方法来获取和展现这些隐藏在背后的信息，而复杂事件处理(CEP)可提供有效的途径和方法。提出基于 Smart Items 的实时企业体系结构，给出事件基本概念、时间模型、层次模型的清晰定义；现有的事件聚合方法基本都是在事件类型级别上进行的，提出的事件实例操作符能够在实例级别上进行事件聚合，同时定义基本的事件类型操作符；提出语义空间的方法为事件聚合提供环境信息，并给出事件实例的使用策略、消耗策略以及老化策略来维护事件实例，特别是给出的事件实例分区方法能够提高事件聚合的效率。使用 Colored Petri 网检测复合事件具有很好的并行性和效率，给出相应的检测算法。在某制造企业的实际应用中证明该方法具有很好的效果。

关键词：智能物件 无线射频识别 实时企业 复杂事件处理 Petri 网
中图分类号：TP39

0 前言

现有的制造企业信息系统几乎涵盖了企业所有的业务范围，产生大量的数据。在这些数据背后隐藏了大量有价值的信息，需要有相应的机制来发现和展现。比如一个信用卡在北京发生了一笔交易，几分钟以后，该信用卡又在纽约发生了一笔交易，系统推测：信用卡很可能被盗用了，并及时通知用户。

随着无线射频识别(Radio frequency identification，RFID)[1]技术在企业中的试点应用，势必产生大量的数据流。这些数据为企业信息系统提供了新的数据源，包含企业真实的库存水平、生产线运作情况等。同样，这些数据流背后，特别是与企业已有信息系统数据的结合，也蕴涵着大量用户感兴趣的信息。

如今，市场环境瞬息万变，企业竞争越来越激烈。企业要想取得竞争优势，这些蕴涵在企业业务数据、RFID 数据流、以及对这些数据的业务操作背后的、用户感兴趣的信息是至关重要的。因为通过这些信息，系统或者用户可以知道企业发生了哪些变化，现有的企业运作存在哪些风险和机会等。但是，目前缺少有效的途径和方法来获取和展现这些隐藏的信息。由于这些信息是以用户或者系统的“感兴趣”为出发点的，把系统中有意义(用户或者系统感兴趣)的变化叫做“事件”，采用事件处理机制[2]来获得这些信息。

提出基于 Smart Items 的实时企业复杂事件处理机制，将企业的业务操作数据、Smart Items 流数据结合起来，从这些数据中抽象出系统的原始事件(用户感兴趣的底层信息)，根据复杂事件处理机制将这些原始事件聚合成复合事件(用户感兴趣的高层信息)，以便用户或者系统快速、准确地决策。复合事件检测采用染色 Petri 网的方法，具有较高的检测效率。经过实际应用，该方法能够有效地获得用户感兴趣的信息，在一定程度上提高企业的决策效率和准确性。

1 基于 Smart Items 的实时企业体系结构

Smart Items[3]是指那些能够提供关于自身或者与其相关联的对象的数据，并且能够将这些数据进行通信的物体。而 RFID 和无线传感器网络[4]能够很好地实现这种思想，称为 Smart Items 技术。

实时企业是指企业能够主动、及时、准确且详细地获取关于库存、生产、市场等的相关信息，以便企业能够及时地做出正确的决策，实现企业的最大效益。

提出基于 Smart Items 的实时企业体系结构，如图 1 所示。企业的 ERP 等系统会产生大量的方法事件，事件处理代理(EPA)获得这些事件，并且进行特定的预处理，然后将其发送至事件发布/订阅总线。

同样，EPA 会从 Smart Items 产生的大量流数据中获得特定事件，经过一定的预处理(过滤、增补和

* 国家 863 计划资助项目(2005AA411912)。20060418 收到初稿，20060921 收到修改稿

改错等)，发送至事件发布/订阅总线。

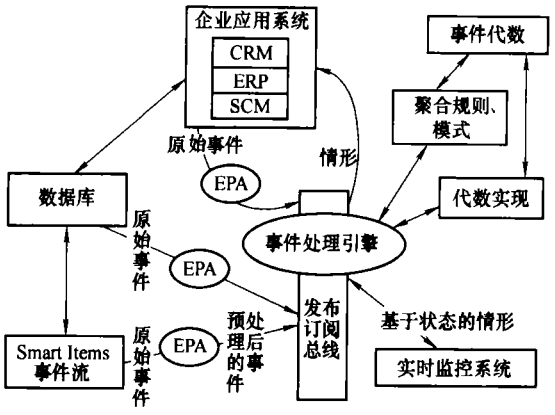

图 1　基于 Smart Items 的实时企业体系结构

企业信息系统中，Smart Items 数据都需要相应的数据库来保存他们的数据。数据库管理系统也会产生大量的事件，比如插入、删除和修改等。这些事件由 EPA 获取，预处理以后发送至事件发布/订阅总线。

事件发布/订阅总线负责收集系统中的原始事件，将原始事件聚合成复合事件，然后将复合事件发送给订阅的用户。用户可以向总线订阅感兴趣的事件。其中最重要的部分就是事件处理引擎，它内部实现了事件代数的各个操作符，根据各种规则和模式检测不同的复合事件。这种复杂事件处理机制是整个体系结构中的核心。

该体系结构具有以下优势。

(1) 事件驱动从面向特定的领域变为主要的编程模式。

(2) 使用发布/订阅机制，能够很好地构建松散耦合的应用系统，提高系统的可适应性。

(3) 从被动的响应到主动的驱动，主动地驱动系统中的行为，主动地为系统其他组件和用户提供信息。

(4) 监控大量的事件源，检测出用户感兴趣的“复杂事件”，即“复杂事件处理机制(Complex event processing，CEP)[2, 5-6]”。该处理机制即为这里关注的对象。

2　复杂事件处理机制

2.1　基本概念

事件是指系统中有意义的变化[2]，比如：对象状态的改变、数据库中数据的插入和硬件故障等。

原始事件、基本事件：是指系统最小的、原子性的发生，需要系统对该发生做出反应。原子性是指基本事件要么完全发生，要么根本就不发生[6]。基本事件的发生是瞬时的。

基于数据流的事件：Smart Items 技术会产生大量的数据。数据流查询与事件处理机制存在着很多相似性[6]，可以使用事件机制来处理数据流的查询。为此，首先要从数据流中抽取事件，有以下两种情况。

(1) 数据流中的每一个数据都看作是一个事件，都是用户感兴趣的信息，比如 RFID 标识的冰箱生产线，冰箱进入发泡间、冰箱离开发泡间和冰箱发泡检验合格等。

(2) 数据流中数据满足一定的条件才看作是事件，比如：发泡温度超过 100 ℃。

复合事件、复杂事件：将基本事件、复合事件聚合在一起，形成新的有意义的事件类型。复合事件的发生有一个发生的时间区间。

情形：它扩展了 Composite event 的概念、表达能力、灵活性和可用性，更面向用户[7]。

事件类型：描述了属于同一类别事件的元数据，在抽象层次上描述了相似事件实例集的公共特征；比如包含哪些属性，这些属性的数据类型是什么等。

定义 1：事件类型，用大写的字母表示，比如 E。$E=(i,a,t_0,t_1)$，i 用来唯一标识事件类型，即：$\forall E_i,E_j,i\neq j:E_i.i\neq E_j.i$；$a$ 表示事件的属性，是有限的属性集合，即 $a=\{a_1,a_2,\cdots,a_n\},n\geqslant 0$。$t_0,t_1$ 表示事件起始时间和结束时间。对于基本事件来说 $t_0=t_1$(详见“时间模型”一节论述)。

事件实例：是指某一个事件类型的单个事件发生。用小写字母表示，比如 e。

定义 2：事件实例操作符，记为 instance(E)，表示事件类型 E 的实例集合，$e\in \text{instance}(E)$。

2.2　事件模型

不同的事件之间存在着以下关系。

(1) 时间关系：事件之间的时序关系利用时间模型来描述：时序关系对于事件流来讲尤为重要，是事件流内在的、固有的特征。

(2) 层次关系：事件之间存在的逻辑层次关系使用层次模型来描述。

(3) 因果关系：事件之间因果关系，这里没有涉及因果关系，是下一步的研究内容。

2.2.1　时间模型

事件时间是复杂事件处理的一个重要概念，它分为检测时间和发生时间。前者是指事件被系统检测到的时间；后者是指事件发生的时间，有两种表达事件发生时间的方式：点时间和区间时间，其区别[8]如图 2 所示。事件 e_1 发生的初始时间是 t_1，结

束时间是t_2，如果采用点时间，事件e_1的发生时间是t_2。实际上，事件e_1在时间t_1已经开始发生，其真实的发生时间是$[t_1,t_2]$，虽然实际系统中大部分事件的发生都是瞬时的[7]，即$t_1=t_2$。

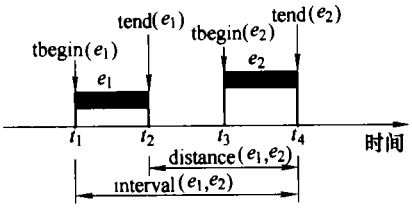

图2 事件的不同时间形式

使用点时间会引起一些问题[8]，这里采用事件的发生时间来表示事件时间。原始事件的时间是点时间，复合事件由于涉及多个成分事件(原子事件、复合事件)，采用区间时间。

对于区间时间有以下定义[5]。

定义3：时间距离操作符，记为distance(e_1,e_2)，表示两个事件实例结束时间的间隔，如图2所示。

定义4：时间区间操作符，记为interval(e_1,e_2)，表示第一个事件实例的开始事件与第二个事件实例的结束时间之间的间隔；如果该操作符只有一个操作数，如interval(e_1)，表示该事件实例的开始时间和结束时间的间隔，如图2所示。

定义5：开始时间操作符，记为tbegin(e_1)，表示事件实例的开始时间。

定义6：结束时间操作符，记为tend(e_1)，表示事件实例的结束时间。

2.2.2 层次模型

层次关系基本上可以分为两类，即泛化和关联，如图3所示。

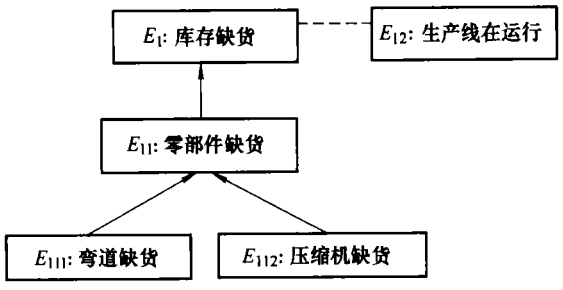

图3 事件的层次模型

事件的层次模型和事件聚合之间的联系与区别如下所述。

(1) 层次模型可以用事件聚合关系(事件代数)来表达。比如$E_{11}=\mathrm{dis}(E_{111},E_{112})$。

(2) 用层次模型描述的事件具有相同的语义范围，而事件的聚合关系描述的复合事件与成分事件可以属于一个语义范围，也可以不属于，而且大部分情况下不属于。比如，$E_1,E_{11},E_{111},E_{112}$描述的都是“库存缺货语义”；而有订单和库存聚合而成的复合事件“补充库存”，不属于订单的语义范围。

(3) 事件的层次模型描述那些事件之间显式的语义关系，而事件聚合描述的是事件之间的隐式语义关系。

定义7：事件泛化，记为generalization(E)，指事件类型E的泛化事件类型的集合，比如：generalization(E_{11})=$\{E_1\}$。

定义8：条件事件泛化，记为conditional generalization(E)=$\{E_1 \mid E_1 \in \mathrm{generalization}(E) \wedge \mathrm{condition}\}$，指事件类型$E$满足一定条件的泛化事件类型的集合。

定义9：事件具体化，记为specification(E)，指那些泛化事件类型为E的事件类型的集合。比如specification(E_{11})=$\{E_{111},E_{112}\}$。

定义10：事件关联，记为association(E)，指与事件类型E关联的事件类型集合，比如association(E_1)=$\{E_{12}\}$。

使用上述的操作符定义了事件类型之间层次关系，在复合事件的定义中可以指定子事件类型的实例是否可以作为父事件类型的实例，详见第2.5节描述。

2.3 语义空间

事件都是在特定的环境中发生的，这些环境信息包括：时间、地点、状态和角色等。环境信息能够为应用程序提供更多的功能，并且提高系统的可适应性。有两种方法可以表达这些环境信息，即事件参数和上下文。这里提出语义空间的方法来表达事件的环境信息，能够更全面地描述事件的环境信息。

“语义空间”不是一个点的概念，而是事件环境信息多维空间中的一个子空间，该空间界定某种具有相对独立语义的相关信息。比如“工作班次”语义空间表示某一个工作班次期间的相关事件信息，如图4所示。

```
<SemanticSpace name="工作班次">
    <initiator event="换班", location="生产车间",
        role="工人", state="生产">
        <condition name="工作班组＝班组1" />
        <correlate name="Add" />
    </initiator >
    <terminator event="换班", location="生产车间",
        role="工人", state="生产">
        <type name="discard" />
    </terminator >
</SemanticSpace>
```

图4 语义空间实例

其中“换班”是如下的事件：换班(生产线号、换班时间、离岗班组和上岗班组)，表示冰箱生产线不同的班组交接班的信息。

定义 11：语义空间，记为 SemanticSpace (name, initiator, terminator)，name 是语义空间的名字，initiator 是语义空间的初始化信息，而 terminator 是终止信息。

定义 12：initiator(event, location, role, state, condition, correlate)，event 表示语义空间的初始化事件，location 表示初始位置，role 表示初始角色，state 表示初始状态，condition 表示满足该条件语义空间才开始，correlate 表示如果已经有一个语义空间在运行，现在系统满足 initiator 的所有条件。该如何处理，有两个选择，“Add”表示新开一个同样的语义空间，“ignore”表示忽略这些条件[7]。

定义 13：terminator (event, location, role, state, condition, type)，其中 event, location, role, state, condition 分别表示语义空间结束的事件、位置、角色和相应的条件，type 表示语义空间结束时，和其关联的、没有报告的复合事件该如何处理，有两个选择，“discard”、“terminate”，前者表示放弃复合事件的检测，后者表示完成复合事件的检测后再结束语义空间。

2.4 事件实例的使用和消耗策略

事件代数都是按照事件类型来定义的。一个事件类型可以有大量的事件实例，事件实例的使用策略指定选择哪个事件实例进行事件聚合。同时，事件实例参与了复合事件的计算以后，是继续保留还是被删除掉，这是事件实例的消耗策略。

这里使用如下的使用策略，同时结合复合事件 $E=\mathrm{con}(E_1,E_2)$ 来说明，其事件实例为 $\left(e_1^1,e_1^2,e_1^3\right)$，$\left(e_2^1\right)$，$e_i^j$ 表示事件类型 i 的第 j 个事件实例。

(1) First：选择事件实例中的第一个进行事件聚合，比如上例，$e=\left(e_1^1,e_2^1\right)$。

(2) Last：选择事件实例中的最后一个进行事件聚合，比如上例，$e=\left(e_1^3,e_2^1\right)$。

(3) Each：成分事件类型的每一个事件实例都参与事件聚合，组成不同的复合事件实例，比如：$e=\left(e_1^1,e_2^1\right)$，$e=\left(e_1^2,e_2^1\right)$，$e=\left(e_1^3,e_2^1\right)$。

(4) All：成分事件类型的所有事件实例都参与事件聚合，$e=\left(\left(e_1^1,e_1^2,e_1^3\right),e_2^1\right)$。

(5) Specific：通过具体的事件实例函数，根据指定的参数选择满足条件的事件实例参与事件聚合，比如 $e=\left(\mathrm{operator}(p_1,p_2,\cdots,p_n),e_2^1\right)$，事件实例操作符见第 2.5.4 节所述。使用该使用策略的时候，要同时指定使用的事件实例函数，以及实际参数。

事件实例消耗策略如下。

(1) Delete：事件实例参与复合事件聚合后立即删除。

(2) Reserve：事件实例参与复合事件聚合后仍然保留。

(3) Conditional reserve：事件实例参与复合事件聚合后，满足一定的条件就保留下来，不满足则删除。

通常，由于系统资源的限制，对于每个事件类型系统都有一个默认的容量函数，指定最多可以存放的事件实例的个数，一旦事件实例达到这个数量，就必须根据一定的策略删除一些事件实例，以便能够存储新的事件实例。这就需要“老化”策略(Aging strategy)，将事件实例“变老”，必要的时候删除最“老”的实例。这些老化策略如下。

(1) 时序老化：按照时间顺序，即最早发生的事件是最“老”的事件，也最先被删除。

(2) 语义老化：根据具体的语义来老化事件，比如：新的冰箱订单要求交货的日期越晚，事件就越老。使用该策略时，要同时指定相应的老化语义。

2.5 复合事件操作符

复合事件表达式由事件操作符(Operator)和事件操作数(Operand)组成。

定义 14：复合事件表达式，记为 expression= operand operator operand。操作数一般是指事件类型；操作符将事件类型组合在一起，它的定义需要如下几部分内容，如图 5 所示。

```
<operator name="conjunction">
    <mode>Immediate</mode>
    <length>50</length>
    <use>First</use>
    <consume>Delete</consume>
    <specification>true</specification>
</operator>
```

图 5　操作符的定义

(1) 使用该操作符的事件表达式在“语义空间”中的检测模式，这里采用类似文献[7]的检测模式，即：Immediate、Delayed、Deferred。Immediate 是指如果一个事件实例产生，就立即检查复合事件是不是能够产生，如果产生就立即向系统报告复合事件；Delayed 是指如果一个事件实例产生，就立即检查复合事件是不是能够产生，如果产生，在语义空间结束的时候向系统报告复合事件；Deferred 是

指如果一个事件实例产生，在语义空间结束的时候判断复合事件是否产生，如果产生同时报告复合事件。

(2) 该操作符所连接的 Operand 的事件实例的个数。超过了这个数量，就要采用相应的策略来选择保留下来的事件实例。在操作符定义时，可以指定一个默认值。在复合事件的定义过程中，可以指定新的值来覆盖默认值。

(3) Operand 事件实例的使用策略和消耗策略：在操作符定义时，可以指定一个默认值。在复合事件的定义过程中，可以指定新的值来覆盖默认值。

(4) 指定事件层次结构中的子事件类型的事件实例是否归属为父事件类型的实例。这里定义的策略，覆盖了事件类型层次结构中定义的事件类型之间实例的关系。

下面定义系统中使用的复合事件操作符，包括其语法和语义。至于每个操作符上述的四部分内容就不再赘述。

2.5.1 功能操作符

定义 15：投影(Projection)，记为 $E_2=\pi_x(E_1)$，其中 x 是属性集合，将事件类型 E_1 的属性按照 x 的属性集合投影，得到新的事件类型 E_2，即 $E_2.a=x\subset E_1.a$。

2.5.2 基本操作符

定义 16：与 (Conjunction)，使用“con”表示，记为 $E=\mathrm{con}(E_1,E_2,\cdots,E_n)$。指所有成分事件都发生，复合事件才发生。

定义 17：或 (Disjunction)，使用符号“dis”表示，记为 $E=\mathrm{dis}(E_1,E_2,\cdots,E_n)$。只要有一个成分事件发生，复合事件就发生。

定义 18：否定(Negation)，使用符号“NOT”表示，记为 NOT(E)，表示事件类型 E 没有发生。

定义 19：迭代(Iteration)，记为 iteration (E_1, n,E_2)，表示在 E_2 发生以前，E_1 发生了 n 次。

2.5.3 时序操作符

定义 20：顺序(Sequence)，使用符号“seq”表示，记为 $E=\mathrm{seq}(E_1,E_2,\cdots,E_n)$，指事件之间按照时间顺序发生。

定义 21：条件顺序(Conditional sequence)，使用“seq_θ”表示，记为 $E=\mathrm{seq}_\theta(E_1,E_2)$，$\theta$ 为条件。事件 E_1 与 E_2 存在时间上的先后关系，同时满足条件 θ。

2.5.4 事件实例操作符

现有的大部分的事件检测机制都是基于事件类型的[5, 9]。实际上，不同的事件实例具有不同的语义。

定义 22：事件实例操作符，记为 EventInstance ($E,p_1,p_2,\cdots,p_n$)。根据给定的参数，从事件类型的实例集合中找到指定事件实例。其中 E 是事件类型，$p_1,p_2,\cdots,p_n$ 是参数。这个操作符只是逻辑形式的定义，在具体实现的时候，需要根据不同的参数提供不同的操作符实现。

还有另外一种形式，即根据事件索引号确定事件实例。记为 EventInstance(E, i)，i 是事件实例在事件类型 E 的实例集合中的顺序号。

定义 23：索引号操作符，在指定的时间内(如果 t 指定的话)，最后一个发生的事件实例的索引号。记为 $i=\mathrm{Index}(E,[t])$。

定义 24：相对时间操作符，$T_r(E,t_1,i,[t])$，指定时间 t_1 之后第 i 个事件实例的发生时间，如果是复合事件，就是指开始时间；如果指定 t，并且在 $[t_1,t]$ 内没有 i 个事件实例发生，结果为 ∞。

定义 25：属性操作符，记为 AttrValue(E,e,a)，求出事件类型 E 的实例 e 的属性 a 的值。

定义 26：相对属性操作符，记为 $\mathrm{AttrValue}_r$ (E,t,i,a) =AttrValue (E, EventInstance (Index (E, t) + i), a)，求出时间 t 之后第 i 个事件实例的属性 a 的值。

2.6 事件实例的分类与分区

事件流中的事件实例由分类器(Classifier)进行分类，分别归类于相应的事件类型。对于每个复合事件表达式，可以根据其成分事件的属性，将成分事件的事件实例划分为不同的分区，以便高效地进行事件聚合。

分区的划分有两种方法，即整体分区和局部分区。整体分区是指将与复合事件关联的整个语义空间分为不同的子空间，每一子空间对应一个相对独立的语义范围；局部分区是指将同一个子空间的事件实例分成不同的组，每个组内的事件实例具有相同的特定属性。如图 6 所示。

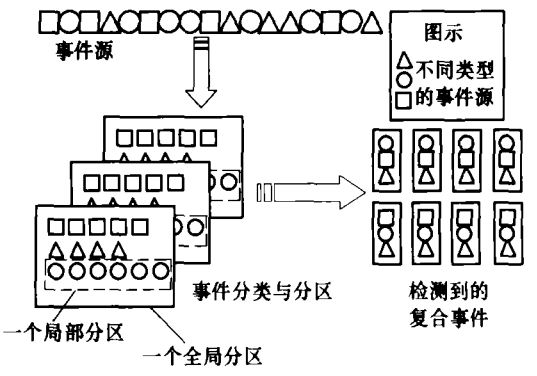

图 6 事件实例的分类与分区

使用键(Key)来定义分区[7]，键是事件类型的某一属性，用来将事件实例按照特定的语义进行分组。

对于一个复合事件表达式来说，有如下定义。

定义 27：全局分区使用全局键值来定义，global key={a |a∈situation.operand.a and a∈situation. semanticspace.initiator.a and a∈situation. semanticspace. terminator.a}。也就是说，全局键是复合事件操作数、语义空间初始事件、语义空间结束事件的公共属性。

定义 28：局部分区，使用局部键来定义，local key={a |a∈situation.operand.a}，也就是说，局部键是复合事件操作数的公共属性。

2.7　具体实例

下面给出一个具体实例来说明上述的各个概念。

有如下事件：生产期间出现质量不合格=dis(“发泡不合格”，“性能检验不合格”，“弯道检验不合格”)，其详细定义的 XML 文件如图 7 所示。

```
<Situation name="生产期间出现质量不合格">
    <SemanticSpace>"工作班次"</SemanticSpace>
    //语义空间
    <GlobalKey>"生产线号"</GlobalKey>
    //全局分区
    <LocalKey>"冰箱类型号"</LocalKey>
    //局部分区
    <operator name="disjunction">
        <length>100</length>
        //事件实例的缓存个数
        <use>First</use>
        //事件实例的使用策略
        <consume>Delete</consume>
        //事件实例的消耗策略
        <specialization>false</specialization>
        //指定子类型的事件实例是否归属于父事件类型
    </operator>
    <operand name="发泡不合格">
        <specialization>true</specialization>
        //指定其子类型的事件实例是否归属于该事件类型
    </operand>
    <operand name="性能检验不合格">
        <length>200</length>
    </operand>
    <operand name="弯道检验不合格">
    </operand>
</Situation>
```

图 7　复合事件定义的具体实例

3　复杂事件检测机制与算法

复合事件检测基本上有如下几种方法，即：基于 Petri Net 的方法[10]、基于自动机的方法[11]以及基于树的方法[12]。它们各有利弊，这里采用基于 Colored Petri Net 的方法来检测复合事件，因为 Petri Net 有如下优势。

(1) 非常适合建模并行系统[13]，因而可以很方便地处理复合事件检测时多个成分事件同时激活的情况。

(2) 使用 Colored Petri Net，利用 Token 类型可以很方便地建模事件参数，从而更多地在系统传递信息。

3.1　Petri 网模型

该部分定义了用来进行复合事件检测的 Colored Petri 网模型。

定义 29：Colored Petri 网定义为 CPN=(N, M_0)，N 表示系统的静态的结构；M_0 表示系统的初始标识。

定义 30：Colored Petri 网的静态结构 N=(P, P_i, P_o, P_s, T, Arc, TType, TF, GF, PF, UType, CType, AF_i, AF_o)。

(1) P 是库所的集合。

(2) P_i 是输入库所的集合。

(3) P_o 是输出库所的集合。

(4) P_s 是语义空间库所的集合。

(5) Arc 是连接库所和变迁的弧的集合。

(6) TType 是 Token 类型的集合。

(7) TF 是 Token 类型函数。TF: =P→TType，用来将库所映射到特定的 Token 类型上。

(8) GF 是守卫函数(Guard function)，GF: =T→Boolean expression，用来将变迁映射到特定的布尔表达式，以控制复合事件的激活。

(9) PF 是分区函数，根据全局分区键和局部分区键，将一个库所映射为几个隔间，也就是将库所中的 Token 分类。

(10) UTYPE 是事件实例的使用策略的集合，即 UTYPE={First, Last, Each, All, Specific}。

(11) CTYPE 是事件实例的消耗策略，即 CTYPE={Delete, Reserve, Conditional reserve}。

(12) AF_i 是输入弧函数，AF_i:=ARC→(UTYPE, CTYPE, Variable)，用来将输入弧映射为输入弧表达式，包含三部分内容，即事件实例的使用策略、事件实例的消耗策略、表示事件实例参数信息的变量。

(13) AF_o 是输出弧函数，主要内容是根据输入库所的 Token 信息转化为输出库所的 Token 信息。

根据上述 Petri 网模型的描述，库所用来建模事件类型，作为事件实例的容器。它可以包含原始事件，也可以包含复合事件，每一个事件类型对应一个库所。输入库所用来建模复合事件的成分事件类型；而输出库所用来建模复合事件。语义空间也用库所来建模，语义空间库所作为一个输出库所，其

输入库所是语义空间的开始事件对应的库所。另外，还有辅助库所，用来表达事件之间特殊的语义。

变迁用来建模复合事件的激活，一旦变迁发生，就相当于复合事件已经被检测到。变迁有两种状态，即“使能”和“激活”，“使能”是指变迁的所有输入库所都有相应的 Token 存在，具备了复合事件检测的基本条件。但是使能状态的复合事件并没有真正被系统检测到，因为它还需要判断 Guard 条件是否满足，如果条件不满足复合事件不能发生，条件满足，复合事件被“激活”。

对于任何一个变迁 T，定义其输入库所 ${}^{\bullet}T$ 和输出库所 $T^{\bullet}$。

定义 31：${}^{\bullet}T=\{p\mid(p,t)\in \mathrm{Arc}\}$。

定义 32：$T^{\bullet}=\{p\mid(t,p)\in \mathrm{Arc}\}$。

对于任意一个库所 P，定义前向变迁 ${}^{\bullet}P$ 和后向变迁 $P^{\bullet}$。

定义 33：$P^{\bullet}=\{t\mid(p,t)\in \mathrm{Arc}\}$。

定义 34：${}^{\bullet}P=\{t\mid(t,p)\in \mathrm{Arc}\}$。

Token 用来表示事件实例，在模型中传递事件的信息。其数量用来表示事件实例的个数。不同的 Token 类型表示不同的要在系统中传递的参数类型。每一个库所都有一个 Token 类型与之关联，表示事件实例的信息。对于原始事件的库所，其 Token 的类型就是事件的类型；对于复合事件，其 Token 类型有如下几种类型[10]：T_1=record(event_i, t_1, t_2, list of record(comp_event_i, parameters))，这种 Token 类型适合于复合事件的参数是各个成分事件的参数的结合，比如 con、seq 等。T_2=record (event_i, t_1, t_2, list of parameters)，用来表示复合事件的成分事件是同类型的事件，比如迭代操作符等。

CPN 的动态行为如下所述。在 CPN 模型中，每一个事件类型对应一个库所，当一个事件类型的实例被检测到以后，其相应的库所就增加一个 Token。用 $M_t(p)$表示 t 时刻，系统中每个库所的 Token 的个数，称作模型的标识，为系统的初始标识，表示系统初始每个库所中 Token 的数量。

当系统检测到一个新的事件实例，相应的库所被标识，系统查找该库所的后向变迁 $P^{\bullet}$，对于其中的每一个变迁，看其输入库所是否都被标识，如果是，并且有满足事件实例使用策略条件的事件实例，则该变迁处于“使能”状态，同时如果变迁的守卫条件满足的话，变迁激活。此时，系统根据输入弧表达式定义的事件实例消耗策略，从输入库所中转移相应的 Token，同时，根据输出弧表达式中定义的函数，将输入的 Token 转化为输入库所的 Token。此时系统中库所的 Token 数量发生变化，系统标识变为 M_1。如此继续下去，得到了系统标识的序列：$M_0,M_1,\cdots,M_n$，从而记录了系统复合事件检测的过程。

3.2 使用 Petri 网建模复合事件

使用上述的 CPN 可以为复合事件构建 Petri 网模型，下面以 $E=\mathrm{con}(E_1,E_2)$为例说明其中涉及到的问题。最简单的模型如图 8 所示。库所 E_1 下面的数字 10 表示该库所的容量，其输出弧的标识 x 变量表示事件实例参数信息。

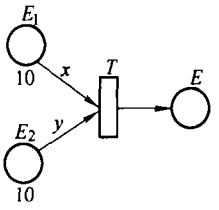

图 8 con 操作符模型

如果 $E=\mathrm{con}(E_1,E_2)$是在某个特定的语义空间中发生的，那么其纯 Petri 网模型如图 9 所示。其中加入了语义空间的库所，以及其开始事件 Start 1、Start 2 和结束事件的库所 End。为了转化为纯 Petri 网，加入了辅助库所 H。

如果 $E=\mathrm{con}(E_1,E_2)$在某个特定的语义空间中发生，同时定义了复合事件检测模式，模型如图 10 所示。如果检测模式是“Immediate”，那么成分事件一旦发生，复合事件就检测到，并且向系统报告；

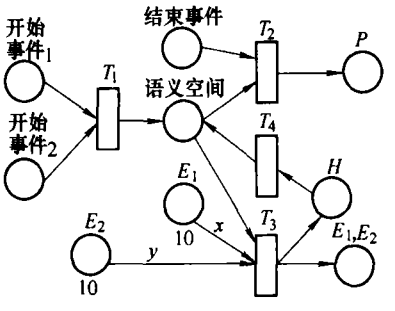

图 9 具有语义空间的 con 操作符模型

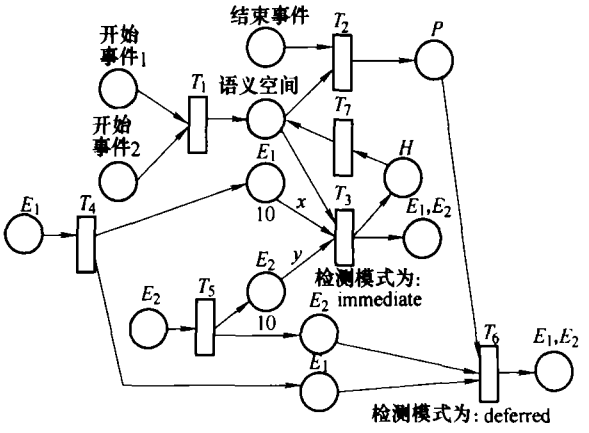

图 10 带有语义空间和检测模式的 con 操作符模型

如果检测模式是“Deferred”，那么在语义空间结束的时候，才检测复合事件是否满足发生的条件。

如果 $E = \mathrm{con}(E_1, E_2)$定义了分区，模型如图 11 所示。语义空间库所被全局键分为不同的全局分区，而成分事件的库所被局部键分为不同局部分区。

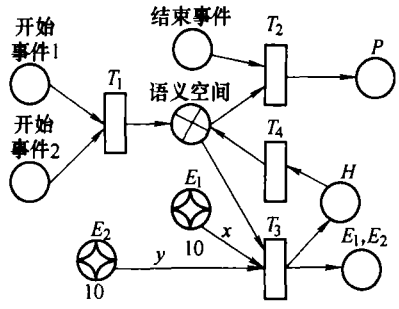

图 11　带有库所隔间的 con 模型

总之，使用 CPN 来建模复合事件，主要涉及如下几个方面。

(1) 操作数(事件类型)事件实例的缓存个数可以用库所的容量来表达。

(2) 事件实例的使用策略和消耗策略在输入弧函数中表达。

(3) Semantic space 作为复合事件的一个附加库所，因为只有在 Semantic space 开始的时候，与其关联的复合事件的检测才有效。

(4) 检测模式，Immediate 模式要求复合事件在语义空间内立即检测；Deferred 模式要求语义空间结束后才检测，如图 10 所示。

(5) 如果复合事件指定了分区，那么语义空间附加库所可以根据全局分区键分为不同的隔间，用来表示不同的虚拟的语义空间。同样，Operand 库所可以根据局部分区键来分为不同的隔间，存放不同类别的事件实例，如图 11 所示。

3.2.1　复合事件检测算法

为了实现复合事件的检测算法，首先定义如下数据结构。

定义 35：Petri 网的静态结构可以用矩阵 N 表示，$N = \mathrm{matrix}[m,n]$，m 表示库所的个数，n 表示变迁的个数。规定，$N(i,j) < 0$ 时，$P_i \in {}^{\bullet}T_j$；$N(i,j) > 0$ 时，$P_i \in T_j^{\bullet}$。

定义 36：库所矢量，Petri Net 的动态结构可以用库所矢量(Place vector) PV 来表示。PV=Array(1..m)表示每个库所中的 Token 的数量，也就是事件实例的个数。

定义 37：标志矢量，flag(m)，m 是模型中库所的数量。用来表示模型库所的状态，2 表示语义空间库所，1 表示分区库所，0 表示正常的库所。

定义 38：分区库所数组(Partition place vector)，$\mathrm{PPV} = \mathrm{Array}(m,g,l)$，$g$ 表示一个事件表达式全局分区的数量，l 表示一个事件表达式每个全局分区中局部分区的数量。该数组只对分区库所有效。库所矢量和分区库所数组满足如下关系 $\mathrm{PV}[m] = \sum_{i=1}^{g}\sum_{j=1}^{l}\mathrm{PPV}[m,i,j]$。

根据图 9 所示的例子，库所在库所矢量中按照如下的顺序排列(Start1, Start2, End, SemanticSpace, $E_1, E_2, H, P, (E_1, E_2)$)。其矩阵 N 如下

$$N = \begin{pmatrix} -1 & & & \\ -1 & & & \\ & -1 & & \\ 1 & -1 & -1 & 1 \\ & & -1 & \\ & & -1 & \\ & & 1 & -1 \\ & 1 & & \\ & & 1 & \end{pmatrix}$$

用库所矢量表示的模型动态结构如下

$$\mathrm{PV}_0 = (1,1,0,0,0,0,0,0,0)$$
$$\mathrm{PV}_1 = (0,0,0,1,0,0,0,0,0)$$
$$\mathrm{PV}_2 = (0,0,0,1,1,1,0,0,0)$$
$$\mathrm{PV}_3 = (0,0,0,0,0,0,1,0,1)$$
$$\mathrm{PV}_4 = (0,0,0,1,0,0,0,0,1)$$
$$\mathrm{PV}_5 = (0,0,1,1,0,0,0,0,1)$$
$$\mathrm{PV}_6 = (0,0,0,0,0,0,0,1,1)$$

使用上面的数据结构，复合事件具体的检测步骤如图 12 所示。步骤 1～3 进行必要的初始化工作；4～5 找出事件类型的序号，并在相应的库所向量加 1；6～41 循环查找其他类型的事件实例是否已经发生，是否可以进行变迁触发，触发以后相应库所的 Token 数量发生变化。第 10 行判断一个变迁的是否所有输入库所都有事件实例；11～12 行判断每个库所的分区中是否有事件实例；第 22 行判断每个分区的事件实例是否满足变迁的使能条件；如果变迁处于使能状态，同时其有输出库所，守卫条件得到满足，变迁就处于激发状态(22～31 行)；变迁触发后，库所的 Token 进行转移，维护相应的库所矢量和分区库所数组(32～35 行)。

3.2.2　基于 Petri Net 的复合事件检测器

如上所述，系统或者用户指定的每一个复合事件都与一个 Petri Net 模型对应，这些不同的模型之间有些元素是重复的，比如两个复合事件可能共用

```
(1)flag[m]初始化
(2)PV[m]初始化
(3)PPV[m,g,l]初始化
(4)mp:=find(E)
(5)PV[mp]:=PV[mp]+1
(6)FOR j:=1 TO n DO
(7) enabling:=FALSE
(8) firing:=FALSE
(9) IF N[mp,j] < 0 THEN
(10)  IF N[ ,j] + PV > 0 THEN
(11)    FOR g1:=1 To g DO
(12)      FOR l1:=1 TO l DO
(13)        FOR k:=1 TO m DO
(14)          IF flag[k]==0 THEN
(15)              Temp[k]:=PV[k]
(16)          ELSE IF flag[k]==1 THEN
(17)              Temp[k]:=PPV[k,g1,l1]
(18)          ELSE IF flag[k]==2 THEN
(19)              Temp[k]:=1
(20)          END IF
(21)        END FOR
(22)        IF N[ ,j]+Temp>0 THEN
(23)          Enabling:=TRUE
(24)          FOR k:=1 To m DO
(25)              IF N[k,j] > 0 THEN
(26)                mp:=k
(27)                  IF guard condition==TRUE THEN
(28)                    Firing:=TRUE;
(29)                  END IF
(30)              END IF
(31)          END FOR
(32)          IF firing==TRUE THEN
(33)            PV:=PV+N[ ,j]
(34)            PPV[ ,g1,l1]:= PPV[ ,g1,l1]+ N[ ,j]
(35)          END IF
(36)        END IF
(37)      END FOR
(38)    END FOR
(39)  END IF
(40)  END IF
(41)  END FOR
```

图 12 复合事件检测算法

一个成分事件，或者一个复合事件的结果可能是另外一个复合事件的成分事件。因此，可以将系统中所有的复合事件的 Petri 网模型合并为一个大的模型。每当系统或者用户指定一个新的复合事件时，与其对应的 Petri 网模型就加入到已有的模型中去。事件检测器的主要功能就是实现这样一个大的模型。

将不同的复合事件的 Petri 网模型合并在一起，需要有相应的原则和方法来保证合并的模型能够包含所有子模型的语义，同时尽可能简单。这是 Petri 网模型的化简和验证问题[10]。

4 应用案例

所述的复杂事件处理机制在某著名冰箱生产企业得到初步应用。该企业中存在以下隐患。

(1) 从原料来讲，产品的原材料和部件可能存在质量问题，可是往往因为没有信息证明而难以追究供应商和责任人。

(2) 从操作员来讲，操作人员在忙乱之中可能上了错误型号的零部件(为了避免生产线闲置，通常采用混合生产，一条生产线穿插生产多种型号的冰箱)。

(3) 从进度控制来讲，某种零部件可能已经缺货，生产线已经因为缺料而停工，而管理人员由于对生产现场的失控，根本无法从 ERP 系统及时获知这种信息。

(4) 从操作工人的管理来讲，复杂的班次和劳动使得很难对他们进行绝对公平的奖罚。

为了解决上述问题，同时为了满足 Wal-mart 等大型超市的订货需求，企业决定实施 RFID 技术。

通过实施 RFID 技术，采用复杂事件处理机制，能够较好地解决上述问题。除此之外，还能够主动为企业提供更多及时、准确、详细的信息。

(1) 目前，仅在冰箱重要零部件上贴有电子标签，通过其中的 EPC(Electronic product code)码可以很方便地找出原材料的供货信息。比如某一类型的压缩机发生质量问题，可以使用 RFID 的阅读器读取压缩机的 Tag 信息，得到零部件信息事件，即 PART-READING，主要有如下一些属性：零部件名称、类型、生产厂商、生产日期和进货日期等。利用这里定义的复合事件操作符可以快速地确定零部件的来源，比如 AttrValue(PART-READING，“8E5-YEK691I60K52DN”，生产厂商)得到标签为“8E5YEK691I60K52DN”的零部件的生产厂商。

(2) 零部件与冰箱的自动匹配。当冰箱在生产线上运转到安装压缩机的工位时，Reader 自动识别该冰箱的类型，同时读取该冰箱的上下文信息(所需零部件信息)，得到 R 事件。当工人拿起一个压缩机准备安装时，Reader 读取该压缩机的类型，得到 C 事件。使用条件顺序操作符 $E=\mathrm{seq}_{R.\mathrm{compressor}=C.\mathrm{type}}(R, C)$ 可以判断压缩机与冰箱类型是否匹配，即 R.compressor=C.type。如果 E = NULL，表示当前工位工人安装的零部件不是该冰箱所需要的，应该发出警报，提醒工人安装正确的零部件，即 R.compressor。如果 E 不为空值，则表明该压缩机与冰箱类型匹配。

(3) 自动补充库存。在一个语义空间(工作日)结束的时候，不同部门的事件如下所示。调度部门：S 事件，这周要生产 2 台医院特需冰箱；生产部门：P 事件，10 台医院特需冰箱正在生产，2 台宾馆特需冰箱正在生产，还没有安装压缩机；销售部门：M 事件，60 台小冰箱订单，40 台航天冰箱订单；

客户服务：C 事件，8 台冰箱压缩机坏了；库存：I 事件，库存压缩机数量为 95 台。复合事件 E_1=seq (con(S, P, M, C), I)首先统计不同部门需要的压缩机数量，然后与库存比较，得到需要补充的库存为 AttrValue_r (E_1, now, 1, num) =27 台。其中复合事件 con(S, P, M, C)的检测模式为“Deferred”，now 表示当前时间。

(4) 准确的工作定时：每个工位的工人都有一个标签，Reader 会自动识别他是否在工位上。同时，每个工人对每一台冰箱的工位操作都被记录下来，从而可以使用复合事件 $\text{seq}_{W.\text{pid}=Q.\text{pid}}(W, Q)$很容易地确定工人绩效。其中，事件 W 为工人的工作时间，Q 为工人的工作质量，比如安装压缩机合格、不合格等；条件 W.pid=Q.pid 将工人的工作时间与工作内容匹配起来。

总的来讲，部署 RFID 以后，企业的信息系统 MES、ERP 等有了新的数据来源，能够准确、及时地获得企业的库存情况、工人的操作情况，从而在一定程度上实现了计算环境和物理环境的集成。

通过复杂事件处理机制，系统能够从不同的数据源，比如：数据库、RFID 数据流等，分析出不同的事件，通过相应的复合事件将这些原始事件聚合成用户感兴趣的高层信息。并且这些信息的获取是及时的，为实现真正意义上的实时企业提供了一定的基础。

在实际应用的过程中，也存在一些问题。比如：复合事件检测的遗漏或者错误，主要的原因是现有的 RFID Reader 不能保证 100%的正确识别率，有时会出现数据的遗漏，甚至是错误读取，而与硬件配套的软件系统在处理这些数据时，功能显得比较单薄。

5　结论

实时企业是各个企业不断追求的目标，其主要思想就是主动为企业的决策者、管理者以及员工提供及时、全面、准确和详细的信息。

现有的企业信息系统几乎涵盖了企业所有的业务范围，全面支持企业的业务运作，保留企业业务运作的数据。它们大部分的功能是管理这些业务数据、被动地接受用户的查询，没有有效的机制来找出隐含在这些数据背后的信息

Smart Items 技术的使用能够为企业信息系统提供新的数据源，这些数据是大量的流数据，其本身能够提供企业真实的、准确的信息。另外，这些数据与企业的其他业务数据结合会提供更多地企业运作信息，更进一步地实现企业计算环境和物理环境的集成。

为了能够获取隐藏于企业业务数据、Smart Items 数据流背后的信息，复杂事件处理是一个有效的方法。讨论了事件处理机制，给出了其中的基本概念，与其他元素的联系和区别，论述了事件模型和事件代数，并使用 Colored Petri 网建模复杂事件，提供相应的算法来检测复合事件。实践证明该方法是有效的，能够在一定程度上获取更多的企业信息，更好地支持实时企业的思想。

参 考 文 献

[1] NATH B, REYNOLDS F, WANT R. RFID technology and applications[J]. IEEE Pervasive Computing, 2006, 5(1): 22-24.

[2] DAVID L. The power of events: an introduction to complex event processing in distributed enterprise systems[M]. Boston: Addison Wesley, 2002.

[3] CHRISTOF B, TAO L, STEPHAN H, et al. Integrating smart items with business processes an experience report[C]//Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS), 2005: 227-235.

[4] AKYILDIZ I F, SU W, SANKARASUBRAMANIAM Y. Wireless sensor networks: a survey[J]. Computer Networks, 2002, 38(4): 393-422.

[5] WANG F, LIU S, LIU P, et al. Bridging physical and virtual worlds: complex event processing for RFID data streams [C]//The 10th International Conference on Extending Database Technology (EDBT), Munich, Germany, 2006: 588-607.

[6] SHARIQ R. Complex event processing beyond active databases: streams and uncertainties[R]. UCB/EECS-2005-26. California: Department of EECS of University of California at Berkeley, 2005.

[7] ASAF A, OPHER E. Amit-the situation manager[J]. VLDB, 2004, 13(2): 177-203.

[8] ANTONY G, JUAN C A. Two approaches to event definition[C]//Database and Expert Systems Applications (DEXA), Aix-en-Provence, France, 2002: 547-556.

[9] ALOYSIUS K M, PRABHUDEV K, LIU G, et al. Specifying timing constraints and composite events: an application in the design of electronic brokerages[J]. IEEE Transactions on Software Engineering, 2004, 30(12): 841-858.

[10] STELLA G, KLAUS R D. Detecting composite events in active database systems using Petri nets[C]//Proceedings

of the Fourth International Workshop on Research Issues in Data Engineering, Houston,1994: 2-9.

[11] PETER R, PIETZUCH, BRIAN S, et al. Composite event detection as a generic middleware extension[J]. IEEE Network, 2004, 18(1): 44-55.

[12] MASOUD M S, MORRIS S. GEM: a generalized event monitoring language for distributed systems[J]. Distributed Systems Engineering, 1997, 4(2): 96-108.

[13] MURATA T. Petri nets: properties, analysis and applications[J]. Proceedings of the IEEE, 1989, 77(4): 541-580.

COMPLEX EVENT PROCESSING OF REAL TIME ENTERPRISES BASED ON SMART ITEMS

ZANG Chuanzhen FAN Yushun

(National Engineering Research Center for CIMS, Tsinghua University, Beijing 100084)

Abstract: Real time enterprise is able to provide timely, accurate and detailed information deliberately. The information is about the large volume of data from business systems and Smart Items (radio frequency identification, wireless sensor networks), and most importantly, is about the information behind all the data. There are few powerful methods to retrieve and present the behind information, and complex event processing is effective to do it. The architecture of real time enterprise based on Smart Items is firstly put forth. The clear definition of fundamental concepts of event, event model and hierarchical model are given. All the available event composition methods are almost based on event type, the event instance operators here make it possible to figure out complex event in instance level. The requisite event type operators are also defined. Semantic space is put forth to provide more event context information, and the strategies of event instance using, consuming and aging are given to process all the event instances. Especially, the partition methods of event instances improve the efficiency of event composition significantly. To detect composite event, colored Petri net serves as a good candidate in terms of its properties of concurrency and efficiency, and its mechanism and algorithm are discussed. Application of the mechanism shows that it is effective to provide more accurate and concrete information.

Key words: Smart Items Radio frequency identification Real time enterprise Complex event processing Petri net

作者简介：臧传真，男，1977年出生，博士研究生。主要研究方向为复杂事件处理、工作流。

E-mail: zangcz03@mails.thu.edu.cn