第5卷第9期
1994年9月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 5, No. 9
Sept. 1994

事件代数与主动知识库系统*

何新贵

(北京系统工程研究所,北京 100101)

摘要 本文提出了一种事件代数,它由一组基本事件和作用在事件上的运算组成.从而各种复杂事件都可用这种代数中的事件表达式表示.然后,本文在事件代数的基础上提出了主动知识库系统的概念,在这种系统中可以根据事件的发生主动触发执行各种功能,本文还讨论了主动知识库的用途、结构和实现技术等问题.

关键词 主动知识库,事件代数.

传统数据库在数据的存储与检索等方面已为各种用户提供了良好的服务.这种服务用决策支持系统的术语来说就是提供了"数据支持",当人们需要获得、改变、加入或删除数据库中的某些满足一定条件的数据时,用户可以通过相应的命令或操作来实现其目的.但是值得指出,传统数据库的所有这些功能都有一个公共的特性,就是数据库本身都是被动的.或更确切地说所有数据库管理系统都只能根据用户的命令被动服务.用户给什么命令,系统就做什么动作.它丝毫不会根据数据库的内部状态等情况主动做些什么,例如主动提出警告、主动切换检索方法和主动实现某些动态修改等等.近年来出现的主动数据库(Active Databases)概念打破了这条惯例,提出了应让数据库系统具有主动进行服务的功能.目前,这主要是通过将一些规则预先嵌入数据库系统的办法来实现的.系统中提供了一个自动"监视"模块,它主动地不时地检查着这些规则中包含的各种事件是否已经发生,一旦发现某事件发生时,就主动触发执行某个动作.显然,这样一来,数据库管理系统就可主动履行一些预先由用户设定的动作.可把诸如完整性约束、存取控制、例外处理、监督和警告、状态开关自动切换、检索策略的切换,乃至复杂的演绎推理功能等等以一种统一的机制得以实现;还可方便地实现实时数据库、合作数据库(Cooperative Databases)、动态数据库和演绎数据库等.这显然是一件极具吸引力的事情.从最近一些数据库国际会议上反映出[1],它与面向对象的数据库技术一起已成为数据库界研究的两个热点.知识库与数据库本来就有许多渊源关系,知识库技术过去曾从数据库技术的进展中获得过很多启示.我们认为近年来主动数据库这股热,也很值得研究知识处理与知识库的人思考,从中获得有益的启示.本文将首先提出一种"主动知识库"的概念,然后为了使"事件"的概念更加一般化,提出"事件代数"的概念,并指出其可能的各种应用.

* 本文1992-03-31收到,1992-07-09定稿

作者何新贵,56岁,研究员,主要研究领域为数据库,模糊知识处理的理论与技术,专家系统等.

本文通讯联系人:何新贵,北京 100101,北京系统工程研究所

1 主动知识库系统

回顾目前常用的一些知识表示方法,其实已有一些"主动处理"思想的萌芽.例如,在框架表示语言FRL[2]中的Demons,它是一段当某种事件发生时会自动触发执行的程序,所谓事件简单地就由槽的侧面名来指明.这些事件一般指该槽的值被修改(包括更新、插入或删除)的时刻、或用户认为需要激活某事件的时刻.这些设施虽简单,不甚完善,事件的种类都是系统实现时做死的,不允许用户根据需要自己来设置各种特定事件,但是它已能解决不少实际问题.例如槽值的完整性约束检查、例外处理以及一定的动态监视功能等.我们认为这种设施主要有两方面的缺陷.第1,这种用槽的侧面名来指明事件的方式局限性很大,不能提供用户一种自由地构造和设置自己所需事件的机制,更没有从简单事件构造更复杂事件的能力.第2,这种设施对其他知识表示方法缺乏一般性和统一性,它基本上是依附在一种知识表示方法上,且十分依赖于系统的具体实现.为了克服这些缺陷,实现一个真正的主动知识库系统,我们来提出一个一般的主动知识库的模型,并提出一个事件代数,以便大大扩充事件的表示能力.

一个主动知识库系统(AKBS)定义为一个传统知识库系统(KBS)之外加一个事件驱动的规则库,简称事件库(EB),及其相应的事件监视器(EM),即

$$AKBS=KBS+EB+EM$$

其中事件库由系统和用户定义的各种事件驱动的规则组成.事件驱动的规则具有如下一般形式:

WHEN〈事件〉

IF〈条件1〉THEN〈动作1〉;

……

IF〈条件n〉THEN〈动作n〉; (n≥1)

其中〈事件i〉是事件代数(见下节)中的任一事件表达式,〈条件i〉(i=1,2,…,n)是某种模糊逻辑[3,4]中的任意的一个合法的逻辑表达式,〈动作i〉(i=1,2,…,n)既可以是系统预先定义的一些标准动作,也可是用户定义的一个动作,或是用某种语言编写的一个过程.上述事件驱动规则的语义是:一旦〈事件〉发生时,计算机就主动触发执行其后的IF—THEN规则.即如果〈条件1〉为真,则执行其后的〈动作1〉,并且接着逐个检查下一个IF—THEN规则,直至执行完为止.注意,这区别于一般程序设计语言中的CASE语句,在那里只执行第一个使〈条件〉为真的〈动作〉.在这里,我们也可把上述事件驱动规则的语义定义为,一旦〈事件〉发生时,就主动触发执行其后的规则集合(规则库),按产生式系统执行的模式[2](即匹配—解决冲突—执行循环)来解释执行相应规则集合.

由上可见,一个知识库中的知识被分成了两部分,一部分称为"被动知识",即传统知识库中的知识,它们是供知识引擎(或推理机)在解题过程中使用的;另一部分称"主动知识",它是由上述事件驱动规则构成的一个集合.这些主动知识受系统中一个"事件监视器"的监视控制,该事件监视器主动地时刻监视着事件库,一旦发觉某事件发生时,就立即触发执行其后的规则(或规则集合),从而引发一些所需动作的执行.这样,用户可以通过设置(或编制)各种不同的事件驱动规则,以一种统一的机制实现许多知识处理功能,例如,对知识库的

动态监视,知识库的完整性和一致性检查、例外情况处理、推理示踪、知识库分块处理、元知识或深层知识的主动切换、实现某些实时功能、多知识库合作解题、分布知识库系统中的同步与通讯、乃至推理或搜索策略的自动切换和推理中"黑板"内容的动态切换等等,应用将是十分广泛的.

实现主动知识库系统的关键在于实现一种有效的事件监视器,这往往需要软硬件结合起来解决问题,特别当事件是一些中断性质的事件时,必须有相应硬件的支持.一般是系统预先定义一些用户可用的事件、条件和动作等,以便用户根据需要用它来编写各种事件驱动规则.下面我们来给出一些可用于系统预先定义的基本事件.

2　常用基本事件

事件按发生的持续时间可分为瞬时事件与区间事件.每个事件都有一个事件名标识,并有开始(发生)时间、终止(发生)时间和发生期等属性.下面我们按应用的场合来分别叙述各种基本事件.

2.1　与时间相关的基本事件

(1)事件 time=t_0,其中 t_0 表示一个绝对时间,可为一个常数,也可由用户或程序来动态设置;time 的语义可由用户说明(或定义),在未作说明时表示当前时间,即事件监视器检查该事件时的时间.该事件是一个瞬时事件,它在时刻 t_0 发生.

(2)事件 time$<t_0$,其中 t_0 与 time 的意义同上.该事件是一个区间事件,它在 time 小于 t_0 时(即在 t_0 之前)都发生.

(3)事件 time$\geqslant t_0$,其中 t_0 与 time 的意义同上.该事件是一个区间事件,它在 time 大于等于 t_0 时(即在 t_0 之后,包括 t_0)都发生.

(4)事件 time$\in[t_0,t_1]$,其中 time 的意义同上,t_0 与 t_1 都是绝对时间.该事件是一个区间事件,它在 t_0 到 t_1 之间都发生.

这类事件由于都与时间有关,可用它们来描述实时知识库中的各种时间限制条件,在实时监控和时间同步等应用时使用.

2.2　与知识库状态相关的基本事件

(1)当知识库的一个指定的部分内容 X 发生改变(包括插入、删除和更新等)时发生的事件 Change(X).

(2)当知识库的一个指定的部分内容 X 被使用(包括用于查询、搜索、推理、读出和显示等)时发生的事件 Use(X).

(3)当知识库的一个指定的部分内容 X 处于某种特定状态(包括取特定的值、取值于或超出某特定的范围)时,或与另一指定部分的内容 Y 之间具有某种特定关系时发生的各种事件.

(4)当知识库的一致性或完整性等被破坏时发生的事件.

(5)当知识库的容量大于某个限制时,或其它状态出现异常时发生的各种事件.

这类事件可用于知识库状态的监视,实现一致性或完整性等的自动检查、控制元知识或深层知识的切换以及知识库的分块处理等等.

2.3 与知识处理语言或推理机等执行机构有关的基本事件

(1)在执行知识处理语言的某种(个)语句时(或执行完后)发生的各种事件.

(2)在推理机或其它执行机构完成了一步(或若干步)推理(或搜索)之后发生的各种事件.

(3)当知识处理(推理或搜索等)进入一种"死循环"时引发的事件.

这类事件可用来实现推理(或搜索等)的跟踪、自动审计、日志自动建立、例外处理、出错监控以及对推理机动态行为的监视等等.

2.4 与信号灯有关的基本事件

为了实现同一用户各模块之间或不同用户之间互相传递信息,可在系统中设立一些"信号灯".这里不妨设信号灯 S 可取任一整数为值.

(1)事件 S=i,其中 i 表示一个整数值,它可以是一个常数,也可由用户动态地设定.该事件表示信号灯 S 正好取 i 为值时发生的事件.

(2)事件 S≤i,其中 i 意义同上.该事件表示信号灯 S 的值小于等于 i 的值时发生的事件.

(3)事件 S>i,其中 i 的意义同上.该事件表示信号灯 S 的值大于 i 的值时发生的事件.

显然,可用这类事件实现多知识库(或各部分)间的通讯与同步,从而实现合作解题.我们不妨称这种能合作解题的知识库系统为"合作知识库系统".此外,表示外来事件(如键盘操作或外接设备送来信号等)发生的各种信号也可采用信号灯设施.

2.5 与公共变量(或全局变量)有关的基本事件

除信号灯之外,一个系统还可设置若干公共变量(或全局变量)$X_1,X_2,\cdots,X_n$,供各个用户存放各种需要互相交换的信息用.这样,包含这些公共变量的任一合法的逻辑表达式 $B(X_1,X_2,\cdots,X_n)$都可用来构成事件.

(1)当 $B(X_1,X_2,\cdots,X_n)$真时发生的事件.

(2)当 $B(X_1,X_2,\cdots,X_n)$假时发生的事件.

借助这个设施,用户可以自由地通过程序或键盘对公共变量赋值等办法来构造其需要的事件,以便激活一些动作的执行.此外,它也可起信号灯的作用.特别,当模块(或进程)间、用户间、或与外界间的通讯需要互相传递参数值时,更是有用的.例如,它们在互相之间需要按数据驱动原则相互驱动执行时,就可利用这个设施.这对实现实时过程的监控以及诸如合作知识库或分布知识库等都是十分有用的.

2.6 与机器的各种中断有关的事件

(1)键盘或鼠标等外部设备上的一个操作完成时引发的各种中断事件.

(2)当计算机的外接设备(如雷达或通信线等)接收到外部信号时发生的事件.

(3)当计算机的各种运算或操作中产生的各种硬中断(例如运算溢出等)发生时的事件.

(4)当计算机中可设置的各种陷井中断发生时的事件.

利用这类中断事件可以灵活地实现对知识库中各种处理对象的实时控制、监视和处理等功能.此外,在实现各种知识处理模块间的通讯与同步时也十分有用.

以上我们已经给出了不少基本事件,可以分别应用在各种不同的场合.但是,有时用户往往需要更复杂的事件,它们是这些基本事件的某种组合,因此,我们还有必要来给出一种

事件代数，以便用户根据需要从这些基本事件构造各种内容丰富的复合事件.

3　事件代数

设一个事件 e 的发生期 D(e)为时间轴上有限个区间的集合，称这些区间的最左一个区间的左端点 B(e)为事件 e 的开始(发生)时间，称这些区间的最右一个区间的右端点 E(e)为事件 e 的终止(发生)时间. 为了构造更复杂的称为事件表达式的复合事件，今来定义下列各种事件运算.

3.1　同时发生运算 ∧

这是一个二元运算. 设 e 与 e′是两个事件，则 e∧e′=e′∧e 表示事件 e 与 e′同时发生的复合事件，其发生期为 e 与 e′同时发生的时间，即 D(e∧e′)=D(e′∧e)=D(e)∩D(e′)，其中∩为集合中的“交”运算.

3.2　选择发生运算 |

这是一个二元运算. 设 e 与 e′是两个事件，则 e|e′=e′|e 表示在其发生期中 e 与 e′有一个且仅有一个发生的复合事件，即择一发生的事件. D(e|e′)=D(e′|e)=D(e)|D(e′)，其中最后一个|表示集合中的“排除或”运算.

3.3　合并发生运算 ∨

这是一个二元运算. 设 e 与 e′是两个事件，则 e∨e′=e′∨e 表示 e 与 e′只要有一个发生就发生的复合事件. D(e∨e′)=D(e′∨e)=D(e)∪D(e′)，其中∪表示集合中的“并”运算.

3.4　相继发生运算 ·

这是一个二元运算. 设 e 与 e′是两个事件，则 e·e′表示事件结束后马上发生 e′事件，使 e 与 e′连在一起的复合事件，B(e·e′)=B(e)，E(e·e′)=E(e′)，E(e)=B(e′).

3.5　之前发生运算 <

这是一个一元运算. 设 e 是一个事件，则<e 表示当 e 开始发生时就终止的一个事件，其发生期 D(<e)=(−∞,B(e)).

3.6　之后发生运算 >

这是一个一元运算. 设 e 是一个事件，则>e 表示 e 终止时就开始发生的一个事件，其发生期 D(>e)=[E(e),+∞).

3.7　不发生运算 「

这是一个一元运算. 设 e 是一个事件，则「e 表示事件 e 不发生那个事件，其发生期 D(「e)=「D(e)，其中最后一个「表示集合中的求补运算.

现在，我们可把事件代数定义如下：

定义 1. 设 E={e_0,e_1,…,e_n}是一个原子事件的集合，其中 e_0 为恒不发生事件，e_1 为恒发生事件，其它为各种基本事件；设 O={∧,∨,|,·,<,>,「}是按上述定义在事件上的运算的集合；从 E 开始，反复运用 O 中的运算进行复合，称如此施行代数运算所能构成的全部事件的集合为一个事件空间. 称 A={E,O}为一个事件代数.

定理 1. 在这种事件代数中，具有如下性质：

设 e,e′,e″,是事件空间中的任意事件，则：

1. $e \wedge e_0 = e_0 \wedge e = e_0$,
2. $e \vee e_0 = e_0 \vee e = e$,
3. $e \wedge e_1 = e_1 \wedge e = e$,
4. $e \vee e_1 = e_1 \vee e = e_1$,
5. $e|e_0 = e_0|e = e$,
6. $e \wedge e' = e' \wedge e$,
7. $e \vee e' = e' \vee e$,
8. $e|e' = e'|e$,
9. $e \wedge (e' \wedge e'') = (e \wedge e') \wedge e''$,
10. $e \vee (e' \vee e'') = (e \vee e') \vee e''$,
11. $e|(e'|e'') = (e|e')|e''$,
12. $e \cdot (e' \cdot e'') = (e \cdot e') \cdot e''$,
13. $e \wedge (e' \vee e'') = (e \wedge e') \vee (e \wedge e'')$,
14. $e \vee (e' \wedge e'') = (e \vee e') \wedge (e \vee e'')$,
15. $\ulcorner(\ulcorner e) = e$,
16. $\ulcorner(e \wedge e') = (\ulcorner e) \vee (\ulcorner e')$,
17. $\ulcorner(e \vee e') = (\ulcorner e) \wedge (\ulcorner e')$.

有了这个事件代数以后,主动知识库的事件驱动规则中的事件的表达能力就相当丰富了.

4 基于主动知识库的专家系统的功能

特别值得一提的是,用这种主动知识库系统来实现专家系统,将使专家系统除有在某领域内解题的功能之外,还具有各种十分有用的辅加功能.诸如:各种实时检测和控制功能;知识库状态的动态监视,包括一致性或完整性检查等功能;根据不同情况对元知识或深层知识进行动态切换的功能;知识库的分块调度功能,实现"黑板"内容的自动更换等等;例外情况处理和错误监测、警报和处理功能;推理过程示踪功能;多知识库合作解题时,各种知识库和相应推理机的自动切换功能;分布式专家系统中各子专家系统间的自动通讯和同步功能;推理或搜索策略的自动选择和切换功能;广泛的中断处理功能,便于与"外界"(包括其它程序系统、外部设备和终端用户等)的交互作用;对专家系统运行情况的各种自动统计功能;乃至做一些简单的学习过程的模拟等等.总之,这种专家系统的辅加功能将是十分丰富的,具有很大实用价值.

5 基于主动知识库的专家系统的结构与实现

一个包含主动知识库的专家系统称为一个"主动专家系统".它与一般专家系统的区别在于除了它有一个被动的知识库之外,多了一个主动的事件规则库,此外在系统中加了一个事件监视器,用以主动地检测各种事件的发生,从而自动地触发所需动作的执行.主动专家系统的一般结构可用图1表述.

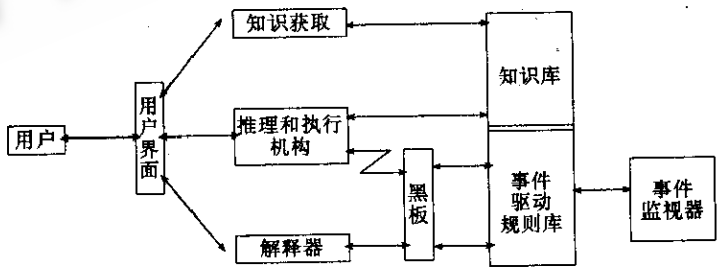

图1

其中用户界面、知识获取、推理和执行机构、解释器、黑板和知识库的组织和处理等模块的实现原则上不存在困难，在此只需提一下事件监视器的实现途径即可. 其实，在单处理器系统中，事件监视器，不妨用在操作系统控制下的一个优先级高的独立进程来实现，以保证它总能经常地被运行，起到主动监视各种事件发生的作用；在多处理器系统中，它可以独立分配一个处理器来完成.

事件监视器的另一种实现途径是，在传统专家系统中一当执行到所有可能发生事件的地方(或在预先设定的检查点上)都产生一个软中断，使机器被迫切换到事件监视器来工作，以便核实当时发生的事件是否已被用户置在事件库中了，若然，就触发执行其后跟的规则或规则组. 否则就返回继续执行.

以上两种实现途径各有利弊，一般说，后一种途径对单机上实现较简单的系统可能是颇具吸引力的.

参考文献

1 Dittrich K R, Dayal U. Active database system. Proceedings of the 17th international conference on Very Large Data Bases, Sept. 1991 Barcelona, Spain, 1991:209－210.

2 何新贵. 知识处理与专家系统. 北京：国防工业出版社. 1990.

3 He Xingui. Weighted fuzzy logic and its applications. Proc. of the 12th Annual International Conference on Computer Software and Applications, Chicago, Illinois, 1988:485－489.

4 He Xingui. Fuzzy Computational logic and neural networks, advancement of fuzzy theory and systems. International Academic Publishers, 1990, **DI4**:1－8.

EVENT ALGEBRA AND ACTIVE KNOWLEDGE BASES

He Xingui

(*Beijing Institute of System Engineering, Beijing* 100101)

Abstract　An event algebra is presented in the paper, which is composed of a set of basic events and some event operations performed on events. Thus, the sophisticated events relevant to many aspects can be expressed by various event expressions in the algebra. Then, a concept of "Active Knowledge Base" based on the algebra is introduced, and its applications, structure and implementation issues are discussed.

Key words　Active knowledge base, event algebra.