文章编号:1001-2486(2006)02-0059-05

基于标注事件图的空间数据库主动规则终止性分析*

熊　伟,廖　巍,景　宁,陈宏盛
(国防科技大学 电子科学与工程学院,湖南 长沙　410073)

摘　要:主动规则的终止性分析强化了规则的设计高效性。给出主动规则元模型的概念作为主动规则终止性分析和比较的基础;对精化触发图进行扩展,提出标注事件图分析模型。结合实例给出终止性分析算法。与各种方法的对比说明标注事件图是可应用于终止性分析的通用抽象模型。

关键词:标注事件图;空间数据库;主动规则;终止性分析
中图分类号:TP392　　文献标识码:A

Termination Analysis for Active Rules of Spatial Database Based on Labeled Events Graph

XIONG Wei, LIAO Wei, JING Ning, CHEN Hong-sheng
(College of Electronic Science and Engineering, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: Termination analysis for active rules improves the design of rule set. Active rule meta-model is proposed as a basis for termination analysis and comparison. The refined triggering graph was extended to construct labeled events graph(LEG). Termination analysis algorithm of LEG was presented, whose efficiency was illustrated by an example. It can be easily seen that LEG is a general abstract model for termination analysis in comparison with other methods.

Key words: labeled events graph; spatial database; active rule; termination analysis

为实现 DBMS 数据完整性和一致性的自动维护以及满足实时信息处理的需要,引入了形如"事件—条件—动作"(ECA, event-condition-action)的主动规则,当事件 E 发生且条件 C 成立时自动执行动作 A。随着空间信息的广泛应用,除完整性约束外,对于基于复杂业务逻辑的空间主动规则的需求也呈不断上升趋势。然而主动规则带来了与行为控制有关的问题,如终止性问题是指给定一个数据库的初始状态和一个初始触发事件,该触发事件引起一组规则集的可能是无止境的级联触。为此,需要提供一些分析方法使主动数据库能更加有效地运行。

在商业数据库中采取主动规则的运行时终止性分析方法比较简单,比如固定规则级联深度或者规则执行时间来控制规则终止,这无疑会回滚掉许多可以终止的规则事务。本文主要讨论编译时分析方法。Ray、Baralis 等采用转化的方式将 ECA 规则转化为其它问题领域来解决终止性问题[1-2],但没有考虑触发条件的影响。Zimmer 等将 Petri 网理论引入终止性分析中[3];其大部分静态分析方法研究呈现一个主线,即以 Aiken 等人定义的触发图[4]概念为基础进行扩展和改进。Baralis 等通过引入激活图改善了触发图的条件分析能力。左万利等引入了惰化图,提出关联图方法增强了对激活条件的考虑[5]。Karadimce 等提出基于精化触发图的算法[6]。基于精化触发图,Lee 提出了路径删除技术和 k 圈解圈技术[7],Debray 等提出一种基于约束的方法计算圈的最大迭代次数[8]。Couchot 也提出图分割方法、基于复合事件的分析算法和基于触发规则和路径的优先级分析方法[9],进一步改进了精化触发图。上述方法未考虑基于空间关系的事件以及基于空间查询的条件测试,缺乏将规则条件和事件结合起来考虑的思路。

* 收稿日期:2005-04-21
基金项目:国家 863 高技术资助项目(2003AA5110)
作者简介:熊伟(1976—),男,博士生。

1 标注事件图

假设空间数据库系统基于对象—关系数据模型,本文构造标注事件图及其相关定义如下。

定义1 主动规则元模型由四元组 < Event, Condition, Action, Priority > 组成,其中 Event 为规则的事件集合,根据不同的分析方法,可以在原子事件基础上扩展多种复合事件。Condition 为规则的条件集合,用 Condition _ Set(R)表示规则 R 的条件集合。Action 为规则的动作集合,Condition _ Set(R.A)表示规则 R 动作中更新操作后得到的更新规则条件集合。Priority 为规则集的优先级集合。规则 R 的前置条件和后置条件定义为:$\mathrm{Precondition}(R) = \bigwedge(c \in \mathrm{Condition\_Set}(R))(c)$;$\mathrm{Postcondition}(R) = \bigwedge(c \in \mathrm{Condition\_Set}(R.A))(c)$。称 $\mathrm{Precondition}(R) \wedge \mathrm{Postcondition}(R)$ 为 R 的激活公式。

定义2 设 Rule 为任意的规则集合,Event 为规则集的事件集合,Rule 的标注事件图(LEG)是一个有向图 $< V, E, L >$,其中 V 是节点集,E 是边集,L 是标注集,图中每个节点 $v_i \in V$ 对应于规则或事件,满足 $V \subseteq \mathrm{Rule} \cup \mathrm{Event}$。有向边 $< v_j, v_k > \in E$ 且 $v_j \in \mathrm{Event}$、$v_k \in \mathrm{Rule}$,当且仅当 v_j 是规则 v_k 的触发事件,这里 v_j 可以是原子事件或复合事件。有向边 $< v_j, v_k > \in E$ 且 $v_j \in \mathrm{Rule}$、$v_k \in \mathrm{Event}$,当且仅当规则 v_j 的动作的执行会引起事件 v_k 的发生,这里限定事件为原子事件。如果有向边 $< v_i, v_j >$ 包含一个规则节点 R,则 $l_{i,j} \in L$ 是对应于 $< v_i, v_j >$ 的属于集合 $\mathrm{Condition\_Set}(R) \cup \mathrm{Condition\_Set}(R.A)$ 中的条件,对于 $< v_j, v_k >$,如果 $v_j \in \mathrm{Rule}$,则 $l_{jk} = \mathrm{Precondition}(R)$ 如果 $v_k \in \mathrm{Rule}$,则 $l_{jk} = \mathrm{Postcondition}(R)$,如图2所示。

定义3 $N_1, N_2, \cdots, N_n$ 是 LEG 的 n 个节点,满足 N_{i+1} 到 N_i 有一条边,则 $N_1, N_2, \cdots, N_n$ 组成一条简单路径,用 $N_n \rightarrow \cdots \rightarrow N_2 \rightarrow N_1$ 表示。设 $N_n \rightarrow \cdots \rightarrow N_2 \rightarrow N_1 (n > 1)$ 是一条简单路径,则 $(C_k, \cdots, C_1, (N_n \rightarrow \cdots \rightarrow N_2 \rightarrow N_1))$ 构成复合路径,其中 $C_j (1 \leqslant j \leqslant k)$ 是最后一个节点为 N_n 的简单路径,并且满足 C_{j1} 与 C_{j2} 最后一条边不相同。如果根据空间复合事件的语义将复合算子分为 AND 和 OR 两种,则复合路径可以分解为

$$(C_1 \rightarrow (N_n \rightarrow \cdots \rightarrow N_2 \rightarrow N_1)) \wedge (C_2 \rightarrow (N_n \rightarrow \cdots \rightarrow N_2 \rightarrow N_1)) \wedge \cdots \wedge (C_k \rightarrow (N_n \rightarrow \cdots \rightarrow N_2 \rightarrow N_1))$$

$$(C_1 \rightarrow (N_n \rightarrow \cdots \rightarrow N_2 \rightarrow N_1)) \vee (C_2 \rightarrow (N_n \rightarrow \cdots \rightarrow N_2 \rightarrow N_1)) \vee \cdots \vee (C_k \rightarrow (N_n \rightarrow \cdots \rightarrow N_2 \rightarrow N_1))$$

定义4 设 G 是 LEG,G 的最多 N 次出现路径表示包含处理一条规则 R_0 所有简单路径的复合路径,并且参与复合的简单路径中,每条规则的出现次数小于 N,用 $\mathrm{MaxPath}(R_0, N, \mathrm{LEG})$ 表示。因此 N = 1 表示一个无圈的执行路径,N > 1 则表示执行路径构成圈。设 R_0 是一条规则,算法 MaxPath 通过沿着边的反方向执行深度搜索遍历计算 LEG 中处理 R_0 的最多 N 次出现路径。

定理1 如果对于 LEG 中的简单路径 Path_1 最后一条规则 Rule_1 的激活公式,路径 Path_1 只出现有限次后使之为假,则记 $\mathrm{Termination}(\mathrm{Rule}_1, \mathrm{Path}_1, \mathrm{Graph}_1) = \mathrm{True}$。如果不能确定 Path_1 的出现次数,记 $\mathrm{Termination}(\mathrm{Rule}_1, \mathrm{Path}_1, \mathrm{Graph}_1) = \mathrm{False}$。对于复合路径有:

$$\mathrm{Termination}(\mathrm{Rule}_1; \mathrm{Path}_1 \vee \mathrm{Path}_2; G) = \mathrm{Termination}(\mathrm{Rule}_1; \mathrm{Path}_1; G) \wedge \mathrm{Termination}(\mathrm{Rule}_1; \mathrm{Path}_2; G)$$

$$\mathrm{Termination}(\mathrm{Rule}_1; \mathrm{Path}_1 \wedge \mathrm{Path}_2; G) = \mathrm{Termination}(\mathrm{Rule}_1; \mathrm{Path}_1; G) \vee \mathrm{Termination}(\mathrm{Rule}_1; \mathrm{Path}_2; G)$$

证明 为讨论方便,不区分 Path_1 和 Path_2。对于析取情况,则分支路径 Path_1、Path_2 均可以独立到达 Rule_1,因此如果 $\mathrm{Termination}(\mathrm{Rule}_1, \mathrm{Path}_1, \mathrm{Graph}_1) = \mathrm{True}$,则只有 $\mathrm{Termination}(\mathrm{Rule}_1, \mathrm{Path}_2, \mathrm{Graph}_1) = \mathrm{True}$ 时,对于复合路径 $(\mathrm{Path}_1 \vee \mathrm{Path}_2)$ 才有 $\mathrm{Termination}(\mathrm{Rule}_1, \mathrm{Path}_1 \vee \mathrm{Path}_2, \mathrm{Graph}_1) = \mathrm{True}$。对于合取情况,如果路径 Path_1 出现有限次使得规则 Rule_1 条件为假,那么 $\mathrm{Termination}(\mathrm{Rule}_1, \mathrm{Path}_1, \mathrm{Graph}_1) = \mathrm{True}$,而根据合取事件语义知,无论 Path_2 的 Termination 如何取值,规则 Rule_1 条件都为假,因此 $\mathrm{Termination}(\mathrm{Rule}_1, \mathrm{Path}_1 \wedge \mathrm{Path}_2, \mathrm{Graph}_1) = \mathrm{True}$,只有两条路径的 Termination 值都为 False 的时候合取路径的取值才是 False。证毕。

定义5 节点路径集包含一个规则实例执行前的所有规则路径。节点 N 的路径集 PathSet(N, LEG) 中的路径 Path 满足:(1)路径 Path 的最后一条节点是 N;(2)路径 Path 不包含同一个节点两次;(3)路径 Path 不属于满足属性(1)和(2)的路径(其自身除外)。路径的路径集对应于路径一次出现之前执行的路

径集合。设 $Path = N_n \rightarrow N_{n-1} \rightarrow \cdots \rightarrow N_1$,$PathSet(N_n, LEG) = \{Path_1, Path_2, \cdots, Path_p\}$,路径 Path 的路径集合为:$PathSet(Path, LEG) = \{(Path_1 \rightarrow N_{n-1} \rightarrow \cdots \rightarrow N_1), (Path_2 \rightarrow N_{n-1} \rightarrow \cdots \rightarrow N_1), \cdots, (Path_p \rightarrow N_{n-1} \rightarrow \cdots \rightarrow N_1)\}$,路径集的构造算法 PathSetBuilding 通过对 LEG 图的深度搜索遍历实现。

定义 6　路径的优先级等于路径上优先级最弱的规则。路径集的最小优先级等于路径集中优先级最弱的路径,路径集的最大优先级等于路径集中优先级最强的路径。用 $m\_p(Path\_Set(R;LEG))$、$m_p(Path_Set(Path;LEG))$和 $M\_p(Path\_Set(R;LEG))$、$M_p(Path_Set(Path;LEG))$分别表示路径集的最小优先级和最大优先级。

定理 2　设 Path 是 LEG 的路径,$R_1, R_2, \cdots, R_s$ 是 G 的 s 条规则,$\{R_1, R_2, \cdots, R_s\}$是 Path 的非稳定集,当且仅当满足 如果规则处理过程 P 中只出现 $R_1, R_2, \cdots, R_s$ 有限实例数,那么在 P 中只有有限次 Path 的出现。用 $DSet(Path, G)$表示标注事件图 G 中路径 Path 的非稳定集,如果 $PathSet(Path, G)$的最大优先级小于 $PathSet(R_i, G)$的最小优先级,或者 $PathSet(R_i, G)$的最大优先级小于 $PathSet(Path, G)$的最小优先级,那么 R_i 可以从非稳定集中删除。

证明　首先考虑情况 1。首先证明 R_i 的两个规则实例间没有路径 Path 的出现。设 R_i 的两个规则实例间有路径 Path 的一次出现,那么意味着有一条规则 R'属于 Path 的路径集,它在属于 R_i 的路径集的规则 R''触发前判定和执行。但是 R'的优先级严格小于 R''的优先级。矛盾,假设不成立,因此 R_i 可以从非稳定集中删除,同理可以证明 2 的情况。基于该定理,如果某个圈中的一条路径的非稳定集是空集,那么只要能验证该路径的激活公式是不可满足的,就可以删除该路径,进一步归约该圈。

2　基于标注事件图的终止性分析

2.1　分析实例

以基于空间数据库的某个目标监视系统为例,我们首先用自然语言描述主动规则集。R_1:当扫描区域目标数更新,并且在区域东北方向目标位置更新时,如果目标位于扫描区域内,那么目标威胁级别设置为高。优先级为 2。R_2:当目标威胁级别更新,或者扫描区域更新时,如果目标不在扫描区域内,那么更新目标位置并且更新扫描区域目标数。优先级为 2。R_3:当目标威胁级别更新时,如果威胁级别为高并且目标不在扫描区域内,更新扫描区域目标数。优先级为 1。R_4:当目标位置更新,更新目标威胁级别。优先级为 1。上述规则集形式化描述如图 1。

Rule:R_1 Priority:2 Event:NE(Update(Z_1), Update(T_1.position)) Condition:Within(T_1.position, Z_1) Action:Update(T_1.level)	Rule:R_2 Priority:2 Event:OR(Update(Z_2), Update(T_2.level)) Condition:Not Within(T_2.position, Z_2) Action:Update(T_2.position), Update(Z_2)
Rule:R_3 Priority:1 Event:Update(T_3.level) Condition:T_3.level ="High" AND Not Within(T_3.position, Z_3) Action:Update(Z_3)	Rule:R_4 Priority:1 Event:Update(T_4.position) Condition:True Action:Update(Z_4.level)

图 1　规则集的形式化描述

Fig.1　Formal description of rule set

根据规则集的形式化描述,分别用事件 E_1, E_2, E_3 表示 Update(Z),Update(T.position),Update(T.level),得到标注事件图如图 2 所示。

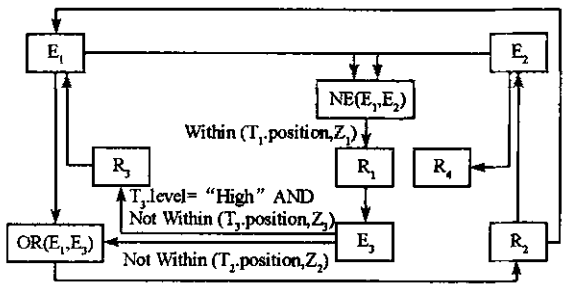

图2 规则集的 LEG
Fig.2 LEG of rule set

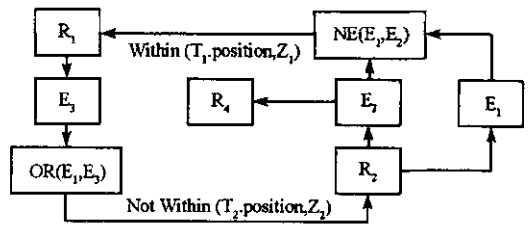

图3 经过算法归约后的 LEG
Fig.3 LEG after reducing by algorithm

2.2 终止性分析算法

终止性分析算法 TerminationLEG **分为三部分,第一部分删除规则处理中有限次触发的规则和有限次抛出的事件;第二部分删除规则处理路径中条件被惰化的规则;第三部分根据优先级进一步删除。算法正确性通过定理** 1 **和定理** 2 **得到保证。经过该算法归约后,如果** LEG **为空,那么规则集终止;如果** LEG **不为空,那么规则集有可能呈现非终止性。**

```
Algorithm TerminationLEG(G)
While nodes of G are removed {
  While nodes of G are removed {
   G←G-no-incoming-edge node;
   G←G-one-incoming-edge conjunction nodes;}
  While (G not Null) AND (condition of R is not deactivated) {
   Path←MaxPath(R,N,G);
   Evaluate Termination(R,Path,G);
   If Termination(R,Path,G) = True {G←G-R;}}
  For Each rule R of G {
   For Each path path of pathSet(R,G) {
    For Each set DSet(Path,G) {
     For rule Ri of DSet(Path,G) {
      If(Mp(PathSet(Path,G)) < mp(PathSet(Ri,G)))
        OR(Mp(PathSet(Ri,G)) < mp(PathSet(Path,G)))
         {DSet(Path,G)←DSet(Path,G)-Ri;}}}
   If there is a DSet(Path,G) such as DSet(Path,G) = Null {
    PathSet(R,G)←PathSet(R,G)-Path;}}
  If PathSet(R,G) = NULL {G←G-R;}}
}
```

```
MaxPath(R0,N,LEG) = PathBuilding((R0))
Algorithm PathBuilding(Pathin) {
N0←Pathin.first
For each node Ni with an edge to N0 (i = 1 to p)
    If Ni appears less than N times in Pathin {
        Pathi←PathBuilding(Ni←Pathin)}}
    If N0 is a conjunction semantic event {
        Pathout←(Path1 ∧ Path2 ∧ … ∧ Pathp)}
    Else {Pathout←(Path1 ∨ Path2 ∨ … ∨ Pathp)}
}

Algorithm PathSetBuilding(Pathin)
N←Pathin. first;
For each node Ni with an edge to N (i = 1 to p) {
    If Ni is not in Pathin {
        PathSetBuilding(Ni→Pathin);
    Else PathSet←Pathin + PathSet;}
}
```

由于 2.1 **节中** LEG **每个节点入度均不为** 0,**算法第一部分无法归约规则集。于是进入算法第二部分。对于规则** R_3,Max _ Path(R_3, 1, G) = ((OR(E_1, E_3)→R_2→E_1→NE(E_1, E_2)→R_1→E_3→R_3) ∧ ((OR(E_1, E_3)→R_2→E_2→NE(E_1,E_2)→R_1→E_3→R_3) ∨ (R_4→E_1→OR(E_1, E_3)→R_2→E_2→NE(E_1,E_2)→R_1→E_3→R_3)),**因为规则集中没有动作影响路径**(OR(E_1, E_3)→R_2→E_1→NE(E_1, E_2)→R_1→E_3→R_3)**上激活公式,其非稳定集为空集,对于激活公式:**Within(T_1. position, Z_1) ∧ T_3. level = "High" ∧ Not Within(T_3. position, Z_3) ∧ $Z_1 = Z_3$ ∧ $T_1 = T_3$,**显然不可满足。因此** Termination(R_3, Max _ Path(R_3, 1, G), G) = True OR (False AND False) = True,**所以** R_3 **及其相关边可以从规则集中删除。这样图** 2 **中还有圈无法归约。如图** 3 **所示,继续算法第三部分。对于规则** R_2,Max _ Path(R_2, 1, G) = ((E_2→NE(E_1, E_2)→R_1→E_3→OR(E_1, E_2)→R_2) ∧ (E_1→NE(E_1, E_2)→R_1→E_3→OR(E_1, E_2)→R_2)),**该路径的非稳定集为** {R_4},**从而有:**Path _ Set(Max _ Path(R_2, 1, G), G) = {(E_2→NE(E_1, E_2)→R_1→E_3→OR(E_1, E_2)→R_2), (E_1→NE(E_1, E_2)→R_1→E_3→OR(E_1, E_2)→R_2)},Path _ Set(R_4, G) = {(E_2→NE(E_1, E_2)→R_1→E_3→OR(E_1, E_2)→R_2→E_2→R_4)},**确定路径集的优先级有:**m _ p(Path _ Set(Max _ Path(R_2, 1, G), G)) = 2, M _ p(Path _ Set(R_4, G))

$=1$,则 $M\_p(Path\_Set(R_4, G)) < m\_p(Path\_Set(Max\_Path(R_2, 1, G), G))$,因此 R_4 可以从 $Max_Path(R_2, 1, G)$的非稳定集中删除,这样非稳定集是空集。考虑激活公式:$Within(T_1.position, Z_1) \wedge Not\ Within(T_2.position, Z_2) \wedge Z_1 = Z_2 \wedge T_1 = T_2$,显然,是不可满足的,从而可以把该路径从 LEG 中删除,这样得到规则集最终归约为空集,因此可以知道该规则集终止。

2.3　与相关方法比较

应用激活图、关联图方法来分析[5]。构造触发图和激活图如图 4 所示,显然这两个图都无法进一步归约得到空集,因此无法保证规则集终止。

应用解圈方法进行分析[7],根据图 4 分析圈($R_1 \rightarrow R_2 \rightarrow R_1 \rightarrow R_2 \rightarrow R_1$),由 Lee 给出的算法,该路径的激活公式只能包含不被规则动作更新的属性。但规则 R_1 和 R_2 包含被规则 R_3 和 R_4 所更新的属性,而 R_3 和 R_4 又无法从触发图中删除,因此解圈方法无法保证终止性。

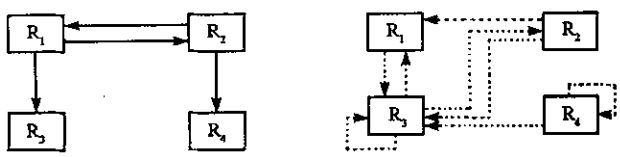

图 4　触发图和激活图

Fig.4　The trigger graph and activate graph

应用抽象解释方法来分析,该方法是通过抽象数据库状态来仿真规则的处理,需要列出规则处理前所有的初始触发规则。但是在此例中,数据库事务可以在规则处理前触发每条规则任意次数目的出现,如圈($R_1 \rightarrow R_2 \rightarrow R_1 \rightarrow R_2 \rightarrow R_1$)中 R_1 的三次出现和 R_2 的两次出现,因此无法准确列出初始触发规则,该方法也无法保证规则集的终止。

采用上述方法均无法得到终止性的结论。与 Couchot 的方法相比,我们在终止性分析中考虑了基于空间关系的复合事件。Baralis 等的方法通过动态分析来确定 k 圈的 k 值,但是分析代价很高。而标注图方法可以引入 Debray 的方法来确定 k 值,同时通过在标注中加入空间关系约束,延伸到了空间数据库中的空间属性条件查询,因此 LEG 是有效通用的规则圈终止条件分析策略。

3　结 论

基于主动规则元模型对各种方法进行对比,标注事件图方法中的事件模型是通用的事件模型,标注是对相关方法中提到的激活公式、触发公式的抽象表示,并且优先级也纳入分析中,因此是一个通用的抽象分析模型。结合空间数据库的特点建立一个完备的主动规则分析模型是很艰难的工作,本文在总结现有工作的基础上提出了一个探讨的思路。下一步将继续研究引入更多主动数据库特征的分析方法,如动作的执行模式,分布式环境下的规则执行情况等。

参 考 文 献:

[1] Ray I. Detecting Termination of Active Database Rules Using Symbolic Model Checking[A]. ADBIS '01[C], 2001.

[2] Baralis E, Widom J. An Algebraic Approach to Static Analysis of Active Database Rules[J]. ACM Trans. Database System, 2000,25(3):269 – 332.

[3] Zimmer D, Unland R, Meckenstock A. Using Petri Net for Rule Termination Analysis[A]. DART '96[C], 1996.

[4] Aiken A, Widom J, Hellerstein J M. Behavior of Database Production Rules: Termination, Confluence, and Observable Determinism[A]. ACM – SIGMOD[C],1992.

[5] 左万利,刘居红,刘淑芬,关联图与主动规则集的终止性分析[J].软件学报,2001,12(2):276 – 282.

[6] Karadimce A P, Urban S D. Refined Triggering Graphs: A Logic-based Approach to Termination Analysis in an Active Object-oriented Database [A]. ICDE[C], 1996.

[7] Lee S Y, Ling T W, Unrolling Cycle to Decide Trigger Termination[A]. VLDB '99[C], 1999.

[8] Debray S, Hickey T. Constraint-based Termination Analysis for Cyclic Active Database Rules[A]. DOOD '00[C], 2000.

[9] Couchot A. Improving Termination Analysis of Active Rules with Priorities[A]. DEXA '03[C],2003:846 – 855.