维普资讯 http://www.cqvip.com

逻辑程序与数据库研究*)

Research on Logic Programming and Databases

邹 阳[1] 宁 卓[1] 张明义[2] 许道云[3]

(贵州大学计算机科学系 贵阳 550025)[1]

(贵州科学院 贵阳 550001)[2] (南京大学数学系 南京 210093)[3]

Abstract The confluence between logic programming and databases is a general trend in computer science and technology. Logic programming offers a great power for expressing queries and constraints and strong inference strategies, while databases technology guarantee efficiency and reliability, the combination of logic programming and databases leads to more powerful database systems, which also enjoy perfect optimization properties. First of all, this paper introduces the typical model for integration of logic programming and databases: CPR system, which consists of four separate subsystems: Prolog engine, Prolog interface, database interface and database engine. This system can be loosely or tightly coupled respectively. Optimization strategies for Prolog/database interface are also given. Then we present the syntax, the semantics and evaluation models for Datalog programs and discuss two prominent classes of optimization approaches: evaluation algorithm and rewriting strategy in detail. Consequently, we address the introduction of database functions to logic programming and extensions of its syntactical structure by augmenting operators. The resulting language can provide much more powerful expression ability and inference mechanism.

Keywords Logic programming, Databases, CPR system, Syntax, Semantics Evaluation algorithm, Rewriting strategy

1 引言

逻辑程序和数据库之间关系的研究始于1977年12月在法国图卢兹召开的"逻辑和数据库"会议。而在此之前,逻辑程序和数据库技术是并行发展的。两者之间的成功汇流得益于日本的"第五代计划",其目标是所谓的"下一代计算机"的开发和研制,在该计划中Prolog被选为编程语言,而关系数据库被用于数据表示。最为流行的逻辑程序设计语言Prolog,作为一般定理证明技术的实现,具有高效和可编程的特点。相应地,关系数据模型作为复杂的层次与网状模型的简化与改进,能提供面向集合的、非过程性的数据操作。"第五代计划"极大地刺激了逻辑程序与数据库之间接口的研究。1985年Ullman发表了题为"Implementation of logical query languages for databases"的论文,奠定了逻辑程序和数据库之间结合的理论基础。

逻辑程序和数据库的结合产生了一个新的研究方向,逻辑程序与数据库。逻辑程序与数据库的研究,其主要目的是构造一个功能更强、效率更高的数据库系统,以拓展计算机理论的研究,并满足新的应用需求。文章的后续部分将首先分析逻辑程序和数据库结合的缘由,并给出两者结合的典范:CPR系统的组成部分和组织结构。然后讨论逻辑程序与数据库系统的优化问题,以CPR系统为例分别介绍了系统各组成部分的优化。文章的最后探讨了逻辑程序语言的扩充,以寻求功能更为强大的数据库系统,并展望了逻辑程序和数据库方向的研究前景。

2 逻辑程序和数据库的结合

2.1 逻辑程序和数据库结合的原因

逻辑程序和数据库的结合,缘于它们之间的许多共同特性:数据库、查询与约束,以及这些共同特性之间的互补性。相比数据库系统的数据定义与操作语言,逻辑程序提供了更为强大的查询与约束的表达能力。而且,由于查询和约束的形式化方法类似,求解也采用相同的推理机制,因而可以进行基于数据库的更为复杂的推理。但另一方面,逻辑程序不具备管理大型、共享、持续和可靠数据的能力。因此,在逻辑程序和数据库的交叉点上,可以构造一类新的系统,它利用逻辑程序来形式化查询和约束,而外存数据库的高效性和可靠性由数据库技术来保证。

逻辑程序和数据库的结合,究其本质,是将逻辑程序的表达能力、推理机制应用于数据库,实现数据库功能的扩充和效率的提高。

2.2 CPR系统

与代数系统等其他系统相比,逻辑程序Prolog是具有较强表达能力的查询语言,而关系数据库系统作为数据库的典型代表是毋庸置疑的。因而CPR(Coupling Prolog System to Relational Databases)系统理所当然地成为逻辑程序和数据库结合的典范。

(1)CPR系统的组成 CPR系统由Prolog引擎、Prolog接口、数据库接口和数据库引擎四个子系统组成。

Prolog引擎的功能是运用推理机制执行Prolog程序。

Prolog接口能识别数据库谓词(Database Predicate)并予

*)本文得到国家自然科学基金10161005、贵州省自然科学基金和贵州大学校基金资助。邹 阳 硕士,主要研究方向为人工智能。宁 卓 硕士,主要研究方向为网络安全。张明义 教授,博士生导师,主要研究方向为人工智能和非单调推理。许道云 博士研究生,主要研究方向为逻辑程序、非单调推理及其复杂性。

维普资讯 http://www.cqvip.com

以特别对待。就 Prolog 程序员的角度而言，Prolog 接口具有不同层次的透明性：全透明、半透明、不透明。当 Prolog 接口全透明时，数据库谓词能在没有用户支持的情况下被 Prolog 接口所识别；半透明时，用户必须给出数据库谓词的说明；而不透明时，Prolog 程序员被迫使用数据库引擎的查询语言。

数据库接口用于和数据库进行交互，并从中提取所需的元组。在多处理机环境中，可以同时运行数据库接口的多个实例，每一个实例对应于一个特定的查询。数据库接口的设计涉及以下两方面的选择：处理的查询类型、返回查询结果的方法。

数据库引擎是任何能处理查询与更新的数据库软件，它提供不同层次的数据操作语言：过程性语言、非过程性语言、具有处理特殊查询的能力增强型语言。

(2)CPR 系统的体系结构　CPR 系统主要有两种组织方法：松散耦合系统和紧密耦合系统。

①松散耦合系统。在松散耦合系统中，交互地进行独立于实际的 Prolog 推理过程，因而松散耦合系统又称作静态耦合系统。图 1 中描述了 LCPR (Loosely CPR)系统的一般情形。其中，耦合的过程发生在程序被载入，即程序文件被 Prolog 引擎访问时。一旦相关的数据库事实被装入 Prolog 存储器，Prolog 程序便独立于数据库而运行。

②紧密耦合系统。在紧密耦合系统中，Prolog 和数据库的交互由 Prolog 系统的推理行为驱动。由于耦合发生在目标执行过程中，因此紧密耦合又叫做动态耦合。图 2 给出了 TCPR(Tightly CPR)系统的框架。其中，耦合过程可以在下列情况之一出现时发生：①目标提交，即用户将一个新的目标提交给引擎；②规则执行，即规则被 Prolog 引擎激活；③数据库谓词匹配。

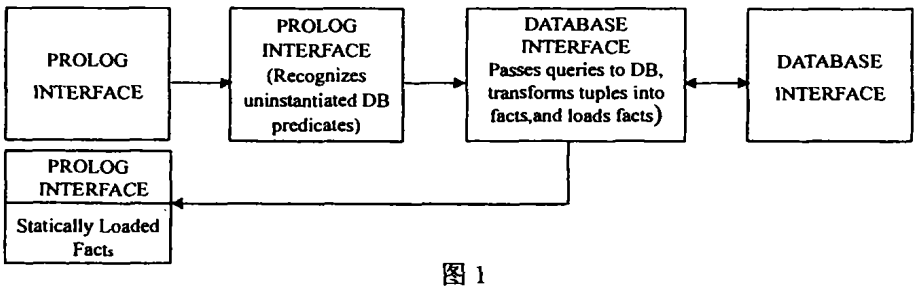

图 1

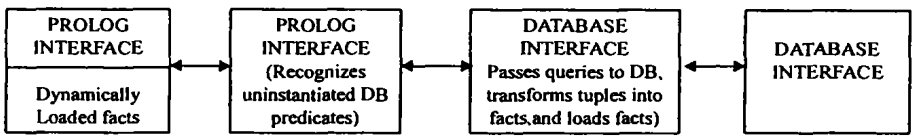

图 2

在介绍了逻辑程序与数据库系统的组成和结构之后，为了进一步寻求系统的高效性，以下将探讨系统的优化问题。

3　系统的优化

系统的优化即 Prolog 引擎、Prolog 接口、数据库接口和数据库引擎四个子系统的优化。由于数据库技术相对成熟，在此不作进一步讨论。以下首先讨论 Prolog/数据库接口的优化问题。

3.1　Prolog/数据库接口的优化问题

Prolog/数据库接口的优化目标是：对查询进行最大程度的限制，而且仅在需要时才执行。在 CPR 系统中，合取基(Base Conjunction)是 Prolog 和数据库之间最为典型的交互内容。合取基指的是一个由数据库谓词和算术比较谓词组成的序列，它可以是一条 Prolog 规则体中的一部分，也可以由多条 Prolog 规则合成得到。Prolog 中的每一个合取基对应于数据库中的一个连接查询，也称合取查询。Prolog/数据库接口的性能可以从以下两个方面来考察：一是就 Prolog 而言，接口所需的延时最小时能获得较好的性能。换句话说，一个好的接口应能屏蔽从外存数据库和从常驻内存的 Prolog 数据库中提取数据性能之间的差别；二是从数据库的角度而言，接口产生的查询的执行时间和开销最小时具有较好的性能。

从 Prolog/数据库接口的角度而言，其优化目标是：对查询进行最大程度的限制，而且仅在需要时才执行。当 Prolog 接口产生最大的合取基时，查询将被最大程度地约束。在此，我们探讨避免查询重复执行的机制。常用的优化策略如下：

(1)数据缓冲　在常驻内存的 Prolog 数据库中存储所有为合取基所检索的元组。匹配下一个合取基时，首先在主存中检索，然后激活数据库接口在外存数据库中进行。用规则描述如下：

```
match(BC):-BC.
match(BC):-call_database_interface(BC).
```

(2)数据与查询缓冲　是对数据缓冲的一种改进，它不仅缓冲结果元组，而且缓冲已经提交给数据库的查询。谓词 queried(bc(..))指明合取基 bc(..)已经传送给了数据库接口。算法思想可表述如下：首先在内存中进行元组匹配，直到失败；一旦失败，测试当前合取基是否已被数据库接口处理过，如果是，则和数据库的交互已无必要；否则，激活数据库接口，载入检索到的相关元组，并在内存中断言相应信息。

其它的策略还有包孕的使用、查询缓冲和数据库接口中的并行和预取策略等。包孕的使用避免了重复访问数据库，而只需在已执行过的查询中找出包孕当前查询的合取基即可。查询缓冲指的是从数据库检索的数据被存入外部文件，需要时访问该文件，而不必重新访问数据库。从 Prolog 的角度看，并行和预取减少了延时，但同时增加了数据库的负荷。

许多 CPR 系统及原型已经被开发和研制，如 PRO-SQL，EDUCE，ESTEAM，BERMUDA，CGW，QUINTUS 等。尽管存在优化算法，但 CPR 系统没有在真正意义上提供一个处理逻辑程序的特殊机制，特别是对递归的处理。为了弥补这一缺陷，以下我们将进一步探讨逻辑程序的优化。

3.2　逻辑程序的优化

为了克服 Prolog 面向元组、顺序敏感性、过程性、包含函数符号以及使用特殊谓词控制程序执行等缺陷，我们推出新的逻辑程序语言 Datalog。Datalog 具有面向集合、非过程性、无函数符号、允许复杂查询等优良特性。下面将以 Datalog 为

维普资讯 http://www.cqvip.com

例探讨逻辑程序的优化。在此之前，有必要分析一下 Datalog 程序的语法、语义以及求值模型。

(1)Datalog 程序的语法　一个 Datalog 程序 P 是一个有限的 Horn 子句集，对所有子句 C∈P，要么 C∈EDB，要么 C 是一条满足以下条件的规则：

①出现在规则头中的谓词为内涵谓词；

②所有出现在规则头中的谓词都在规则体中出现。

条件①保证了外延数据库在程序执行过程中被改动，条件②则指出从 Datalog 程序中只能演绎出有限的事实。

(2)Datalog 程序的语义　Datalog 程序以外延数据库为输入，输出一个内涵的基事实集，其中的每一基事实都是 P∪EDB 的逻辑推论。因而，Datalog 程序的语义是一个从外延数据库幂集到内涵数据库幂集的映射 M_P，满足：

$$\forall V \in EDB, M_P(V)=\{G \mid G \in IHB \wedge P \cup V \models G\}$$

(3)Datalog 的证明理论及求值模型

①Datalog 的证明理论。在此，我们给出一个证明理论的框架，它能从给定的 Datalog 子句集演绎出它的所有基事实推论。首先定义事实推导规则：给定形如 $l_0:-l_1,\cdots,l_n$ 的规则 R 和基事实的一个序列 $F_1,\cdots,F_n$，如果存在这样的替换 θ，且 $l_i\theta=F_i(1\leqslant i\leqslant n)$，那么，从规则 R 和基事实序列 $F_1,\cdots,F_n$ 已不可以推导出 $l_0\theta$。该规则称作基本产生式规则(Elementary Production)。可以证明，EP 规则是可靠的和完全的。

从 EP 规则以及合一算法，不难构造一个以 Datalog 子句的有限集 S 为输入，输出 S 的所有事实推论的算法，不妨称其为 INFER(S)。此外，从不动点理论出发，通过构造相应的变换 T，可以给出最小不动点算法 LFT(T)。不难证明，以上两种求值方法是完全等价的。

②自顶向下的求值策略。自顶向下的求值方法又称反向链接法。该方法特别适用于包含目标的 Datalog 程序，其基本思想是先构造一棵查找树，再将它转化为证明树，即得一个目标实例的证明。而且，查找树的深度可以不超过某一特定值。此方法是可靠的和完全的。体现反向链接思想的求值算法很多，但区别不外乎以下几个方面：导出树、数据结构、树遍历算法、终止条件、子树因子化。

③归结方法。最早用于定理机器证明领域，适用于一般子句，而不仅仅是 Horn 子句。常用于逻辑程序中的归结方法是 SLD-归结(Linear Resolution with Selection Function for Definite Clauses)，其可靠性和完全性不言而喻。

(4)Datalog 程序的优化　若以搜索策略、算法目标以及形式化方法为准则，可将 Datalog 程序的优化方法大致分为四类：自底向上的求值、自顶向下的求值、逻辑重写方法、代数重写方法。以下逐一分析：

①自底向上的求值

· 代数质朴求值(Algebraic Naive Evaluation)，也称为 Jacobi 算法，计算代数方程系统的不动点。若有关系方程系统 $\sum$：

$$R_i=E_i(R_1,\cdots,R_n)\quad (i=1,2,\cdots,n)$$

算法首先将空集赋给所有关系变量 R_i，然后重复计算 $R_i=E_i(R_1,\cdots,R_n)$，$i=1,2,\cdots,n$。直到在连续的两次重复中，R_i 的值无一发生变化，即 R_i 到达某个不动点。形式化描述如下：

```
FOR i=1 TO n DO Ri:=Φ;
REPEAT
 Cond:=true;
 FOR i=1 TO n DO Si:=Ri;
 FOR i=1 TO n DO
  BEGIN Ri=Ei[R1,…,Rn];
    IF Si≠Ri THEN cond:=false;
  END;
UNTIL cond;
```

不难证明，以上算法的输出正是变换 $T\sum(X_1,\cdots,X_n)=(E_1(X_1,\cdots,X_n),\cdots,E_n(X_1,\cdots,X_n))$ 的最小不动点。

· Gauss-Seidel 算法。在 Jacobi 算法的第 k 次循环中，为了计算关系变量的最新值 $R_i^{(k)}$，可在同一个循环中用已计算出的 $R_j^{(k)}$ 去替换原来的 $R_j^{(k-1)}$，$j=1,\cdots,i-1$。修改后的算法称作 Gauss-Seidel 算法，具有更好的性能。此外，还可以在上述两个算法中引入并行机制以提高效率，但其仍存在明显缺陷：某些元组被重复计算、与目标无关的元组被计算。

· 通用半质朴算法(Semi-Naive Evaluation)。半质朴求值能在循环中排除冗余，且无需使用任何与目标结构有关的信息。但它只适用于线性的代数方程系统。

线性代数方程系统比较常见，也有较为高效的求值算法，但情况并不总是这样。非线性规则的一个处理方法是将其转化为一组规则，其中每条规则都线性于原始规则中的变量。如规则 $C:p:-q_1,\cdots,q_h,p^{(1)},\cdots,p^{(m)}.$，可将其转化为以下一组规则：

$$p:-q_1,\cdots,q_h,p,r_1.$$
$$r_1:-p,r_2.$$
$$\cdots\cdots$$
$$r_{m-1}:-p.$$

由于 p 与 r_i 之间的互递归关系，以上规则都是非线性的。因而在非线性系统中，对 Gauss-Seidel 或 Jacobi 算法的修改仅限于：对任一关系变量，只在定义它的方程中用其方差替换该变量本身。相比于半质朴算法而言，该算法的效率较低，但其性能比 Jacobi 算法稍好。

· Henschen 与 Naqvi 算法。适用于附带目标且目标中包含受限参数的 Datalog 程序。

②自顶向下的求值。在此，我们以查询-子查询(Query-Subquery)算法为例介绍自顶向下的求值策略。查询-子查询算法是一个面向集合且完全通用的求值策略，它包含两个集合：

· 回答元组集 P：由主目标及中间子目标的回答元组组成。

· 子查询集 Q：包含当前待处理的所有子目标。

程序执行的每一步，其状态均由以上两个集合来标识。在计算任一目标时，算法既产生新的元组，又形成新的子查询。该算法有两种实现策略：迭代的查询-子查询和递归的查询-子查询，两者分别优先回答元组的产生和优先子查询的产生。

③逻辑重写方法。在逻辑程序的某些求值方法中，存在以下问题：在程序的执行过程中，会产生许多与最终结果无关的中间事实，而且，这些事实在最终结果的演绎过程中不起任何作用。如果该程序运行于大型数据库之上，其效率不难想象。而逻辑重写方法能充分利用目标中的参数约束，将程序进行转换。一般情况下，经过转换的程序相对源程序而言具有较高的效率。我们讨论以下逻辑重写方法：

· Magic 集，其算法基于侧向信息传递(Sideways Information Passing)的思想，对目标谓词附带特定修饰的 Datalog 程序进行优化。为了模拟自顶向下的约束传递策略，Magic 集算法采用在初始规则的右边增加子目标，同时添加定义这些子目标的规则到源程序中的方法将约束引入 Datalog 程序，这些附加的规则迫使程序变量去满足非源程序中的谓词，即 Magic 谓词。因而，在自底向上的计算过程中，变量只取相关的值而不是所有可能的值，从而提高求解程序的效率。算法框架可大致描述如下：

a)定义附加的规则；

维普资讯 http://www.cqvip.com

b)将附加规则的子目标加入源程序的相关规则中；

c)加入补充规则以保证修改后程序的正确性。

其中，Magic 集即 Magic 规则定义的 Magic 谓词所对应的事实集。该算法是完全通用的。

· 计算方法(Couting Method)，是一个基于目标约束信息的重写方法，分两步：

a)对初始程序应用 Magic 集方法；

b)通过添加标识与目标常量距离的信息为 Magic 集的元素补充相关信息，以便更好地排除不相关事实，快速地计算出最终结果。

该算法适用范围较窄，但在其应用领域中非常有效。

此外，还有静态过滤方法及采用重写的半质朴求值等策略。

④ 代数系统的重写。常见的代数系统的重写方法有：

· 确定公共子表达式，以减少重复计算；

· 查询子集化，决定从给定目标可达的子程序，从而排除不可达规则；

· 强连通分支，将代数系统分解为可独立求解的子系统；

· 变量规约：采用变量标识算法，将查询变量中被约束参数列的序号通过规则尽可能地传播给程序中的变量和常量。然后将相应于常量的标识转换成选择操作。由于规约后的方程中常量被约束，在求解时，只需从数据库中提取相关的元组，从而提高执行效率。在完成系统的变量规约以后，如果还存在没有被标识的变量，则规约失败；

· 常量规约。对变量规约失败所产生的系统，可以进一步对其应用常量规约方法。常量规约的基本思想是：在系统求解之前，利用常量与目标中约束的关系以及常量在方程中参与的代数运算，构造关于常量的代数方程，并通过方程的求解，缩减常量的大小，从而简化系统的求解。

对以上诸多优化算法的性能进行较为精确的比较不太现实，因为这些算法本身比较复杂，都有一定的适用范围，而且算法的执行受很多因素影响。但算法思想之间存在一定的共性。如侧向信息传递的思想为 Magic 集、QsQ 以及大多数自顶向下算法所共享，Magic 集和变量规约之间有许多相似之处等。归纳起来，算法优化的基本原则在于：

· 尽量减少求值过程中不必要的数据的产生；

· 使转换后系统的变量和常量尽可能地被约束。

4 逻辑程序的扩充

以上我们所讨论的逻辑程序优化只基于纯 Datalog，其语法局限于一阶逻辑的有限子集：Horn 子句集。纯 Datalog 的表达能力和推理机制存在一定的局限性，对应的逻辑程序与数据库系统的应用范围也较窄。因而，寻求具有更强表达能力的逻辑程序语言是该领域的一个重要研究课题。

就目前的研究而言，对纯 Datalog 存在以下扩充：

· 内置谓词，用于在逻辑程序的规则体中表述比较关系。这些谓词虽不在程序中定义，但其语义被系统理解，并自动予以执行。

· 否定。将否定植入 Datalog 将导致语义及计算方面的问题。使用于纯 Datalog 的闭世界假设理论，在允许否定出现在规则体中时会导致不协调性。

· 复杂对象。纯 Datalog 程序中所涉及的谓词，其对应的关系的属性值是不可再分的。而在其它的应用领域，如计算机辅助设计、办公自动化、知识表示等，需要对复杂结构进行存储与操作。如在 Datalog 中增加函数符号及集合构造器等。

· 数据更新。Datalog 最初被用作查询语言。可考虑将其增加构件以处理 EDB 关系的更新操作。

· 集合函数与非决定性。在 Datalog 中引入用于分块和选择的谓词。

· 不确定性。事实与规则的不精确推理。

析取演绎数据库(Disjunctive Deductive Databases)是目前逻辑程序与数据库研究的热点，它由析取逻辑程序和数据库技术结合而成。纯 Datalog 的语法局限于 Horn 子句，而析取逻辑程序事实上是 Horn 子句的扩充，它具有更强的推理机制和表达能力，因而，析取演绎数据库能适应更为广泛的应用。当然，析取逻辑规则的引入带来了逻辑程序语法、语义上的变化以及相应的优化算法的修改。这些将另文阐述。

结束语 逻辑程序与数据库已经成为人工智能的研究热点，今后的研究将主要分布在：①以逻辑程序的研究为背景，不断丰富逻辑程序表达能力与推力机制，以构造更强大的演绎数据库系统；②寻求更合理的逻辑程序与数据库系统模型，及相应的接口的优化策略；③寻求新的逻辑程序优化算法；④将优化算法移植到增强的逻辑程序语言。

参 考 文 献

1 Gelder A V, Ross K A, Schlipf J S. The Well-Founded Semantics for General Logic Programs. Journal of the ACM, 1991, 38(3): 620～650. Preliminary abstract appeared in Seventh ACM Symposium on Principles of Database Systems, March 1988. 221～230

2 Baral C, Lobo J, Minker J. Generalized Disjunctive Well-Founded Semantics for Logic Programs: Declarative Semantics. In: Proc. of the Fifth International Symposium on Methodologies for Intelligent Systems. ACM SIGMOD-SIGACT-SIGART, North Holland, 1989

3 Baral C, Gelfond M. http://www.cs.ttu.edu./～mgelfond/papers/suvey.ps. Logic Programming and Knowledge Representation. Journal of Logic Programming, 1994, 19(20): 73～148

4 Seipel D. Non-Monotonic Reasoning Based on Minimal Models and its Efficient Implementation. In: Proc. GI-Jahrestagung/IFIP World Computer Congress 1994, Springer, 1994. 53～60

5 Seipel D, Güntzer U. Mixed Fixpoint Theory for Disjunctive Deductive Databases. In: Proc. Eleventh Workshop on Logic Programming 1995(WLP'95), GMD-Studien Nr. 270, 1995. 227～236

6 Serpel D. Aggregation in Disjunctive Deductive Databases. In: Proc. Intl. Conf. on Applications of Prolog (INAP'99)

7 Jarre F, Lepenis R, Seipel D. Using Linear and Semi-Definite Programming for Computing Models of Disjunctive Deductive Databases. In: Proc. 13th Workshop on Logic Programming 1998 (WLP'98)

8 Brewka G. Well-Founded Semantics for Extended Logic Programs with Dynamic Preferences. Journal of Artificial Intelligence Research, 1996, 4: 19～36

9 Fernandez J A, Gryz J, Minker J. Disjunctive Deductive Databases: Semantics, Updates and Architecture. In: Proc. of the 4th Bar-Ilan Symposium on Foundations of AI, AAAI Press, 1996. 256～274

10 Ross K A. The Semantics of Deductive Databases. Aug. 1991

11 Gelfond M, Lifschitz V. Classic Negation in Logic Programming and Disjunctive Databases. New Generation Computing, 1991, 9: 365～385

12 Ceri S, Gottlob G, Tanca L. Logic Programming and Databases. Springer, 1990

13 Przymusinski T C. Stable Semantics for Disjunctive Programs. New Generation Computing, 1991, 9: 401～424

14 Zhang Mingyi. A New Research into Default Logic. Information and Computation, 1996, 129(2)