26—29,80 演绎数据库 数据库 逻辑程序设计 ⑦

计算机科学1993Vol.20 No.3

TP311.13

具备归纳和演绎能力的数据库系统Duce-92的设计与实现*)

李爱中
（吉林大学计算机科学系　长春130023）

何宇夫
（哈尔滨焊接研究所　哈尔滨150080）

摘　要

In order to improve the abilities of deductive database systems, the application of machine discovery to deductive database is studied. The inductive and deductive database system Duce-92 is designed and implemented, which uses machine discovery as a means of handling retrieval failure and automating rule generation.

一、引言

演绎数据库是近几年数据库研究的热点。根据Grant和Minker的分类[1]，演绎数据库的研究可以分为三个阶段：第一阶段(1957—1968)，开始了演绎数据库的研究工作，开始应用J.A. Robinson提出的归结原理于演绎数据库中的演绎的实现[2]。第二阶段(1969—1978)，逻辑程序设计和Prolog在演绎数据库中的应用；演绎数据库语义和否定在定义数据库中的研究。第三阶段（1979—现在），Reiter[3]用证明论观点重新解释了传统的模型论的数据库方式，并给出了形式理论；关于递归的有效实现的研究[4]；关于否定的进一步研究和析取的研究。

演绎数据库集数据库和逻辑程序设计于一体。传统的关系数据库已在应用领域取得了极大的成功，但关系数据库本身缺少推理能力。演绎数据库在继承了关系数据库的所有优点的前提下，利用逻辑程序设计中的递归规则，增加了推理功能。演绎数据库扩展了关系数据库使逻辑演绎成为可行并丰富了数据库语义。

演绎数据库可以定义为一个三元组 ‹C, P, I›，其中：C是非逻辑符号（常元和谓词）的集合，P是公理的有穷集，I是对数据库完整性限制的语句集。在一个确定的数据库（definite database）中，P仅包含下列形式的规则：$B \leftarrow A_1, \cdots, A_n$，$n \geq 0$，其中B，$A_1, \cdots, A_n$为原子。外延数据库仅包含$B \leftarrow$形式的规则并且B中只含常元。外延数据库表示了事实，即关系数据库中的数据。内涵数据库包含了P中的另外一些规则，即$n \neq 0$或含变元。如果内涵数据库中不含递归规则，则演绎数据库和关系数据库相等价；但若内涵数据库含有递归规则，则内涵数据库构成了数据库的演绎成分。故此递归成为演绎数据库与关系数据库的分界线。递归扩展了数据库语义。从上述演绎数据库定义，我们可以给出图1的演绎数据库的结构：

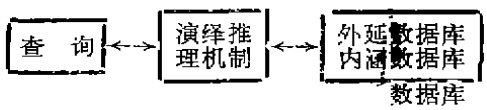


图1　演绎数据库的结构

李爱中　博士，现在吉林大学计算机软件博士后站工作。主要研究方向为机器发现的计算理论及其应用。何宇夫　硕士，主要研究演绎数据库和微机应用。　*)863基金和自然科学基金资助课题。

上述定义的演绎数据库的结构中外延数据库和内涵数据库中的数据库均由用户来完成的，这样我们在使用时会遇到两个问题：一是查询失败，演绎数据库只能给出演绎的结果作为查询的回答，这就意味着用户在使用演绎数据库一开始就要定义完整的数据库，故此说演绎数据库是封闭的。二是规则的用户定义，可能是不完全的或不正确的，数据也可能存在误差，从而导致了演绎的困难，其原因在于缺乏规则的自动生成。

为了解决上述问题，作者试图以基于归纳的机器发现为手段，进行归纳和演绎的综合性研究并应用到数据库之中。

机器发现是近几年来人工智能领域内研究的热点[5]。机器发现作为一种从数据库中发现规律的自动化手段，其开创性工作始于H. A. Simon教授领导的工作及其著名的BACON系统[5,6]。其最著名的应用是为美国航天局从大量数据中发现了新的规律并得到了应用上的实证。由于上述工作基于传统的数学分析，难于形式化并且发现能力有限，在本文中主要以作者在博士学位论文中提出的基于递归函数理论的机器发现方法为基础[0]，并做了很大的改进，引入了统计信息作为启发式信息[7]。

二、Duce-92的结构设计

为了使Duce-90同时具备归纳和演绎能力，我们设计了如图2的结构。

在Duce-92中，首先对数据库进行了动态和静态的划分，其目的是为了提高推理效率。静态数据库又分为外延数据库和内涵数据库；外延数据库存贮事实，内涵数据库存贮规则。静态数据库的内容表明了整个系统的全部知识。动态数据库仅存贮和当前查询相关的知识，是静态数据库内容的一部分。归纳机制对外延数据库进行归纳形成规则并存贮于内涵数据库之中。联想机制根据当前询问和静态知识库中知识间的联想关系，自动地从静态知识库中提取相关的知识并存贮于动态知识库之中。推理机制则根据当前动态知识库的内容对查询进行演绎给出回答。

联想关系是定义在谓词空间上的二元关系A，谓词空间 $P=\{p/$存在$p(x_1, \cdots, x_n)$ $\in$外延数据库或存在$B_0 \leftarrow B_1, \cdots, B_m \in$内涵数据库，$B_i=p_i(x_1^i, \cdots, x_{n_i}^i)$，$p=B_j$，$0 \leqslant j \leqslant m$，$0 \leqslant i \leqslant m\}$，$A=\{(p, p)/p \in P\} \cup \{(p, q)/$存在$B_0 \leftarrow B_1, \cdots, B_m \in$内涵数据库，$B_i=p_i(x_1^i, \cdots, x_{m_i}^i)$，$1 \leqslant i \leqslant m$，$p=p_0, q=p_j$，$1 \leqslant j \leqslant m\} \cup \{(p, q)/$存在$r$，$(p, r) \in A$，$(r, q) \in A\}$。

询问p的联想$=\{q/q=p'(x_1, \cdots, x_n)$且$(p, p') \in A$或$q=B_0 \leftarrow B_1, \cdots, B_m$，$B_i=p_i'(x_1^i, \cdots, x_{m_j}^i)$，$(p, p_0') \in A\}$。动态数据库$=$询问p的联想。

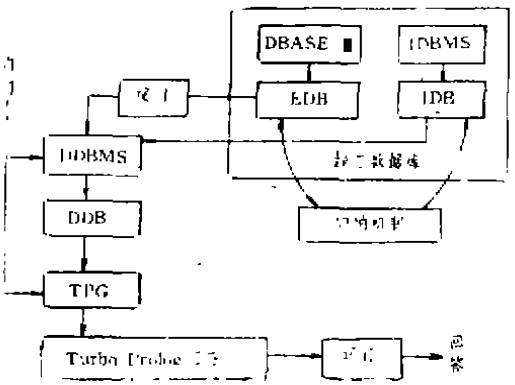


图3 Duce-92的实现方案

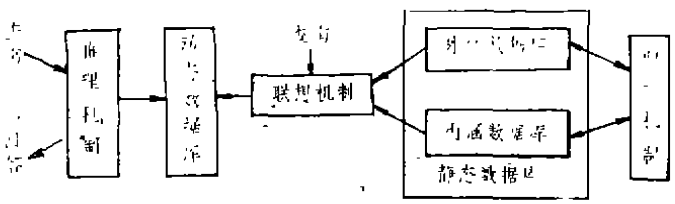


图2 Duce-92的结构

三、Duce-92的实现

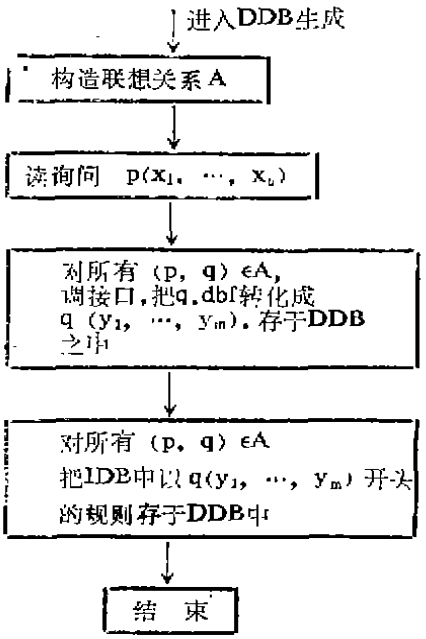
Duce-92是在IBM/PC机上应用Turbo-Prolog实现的。为了实现上的方便，我们设计了如图3的实现方案。

EDB（外延数据库）使用广泛使用的DBASE-Ⅲ作为管理系统，IDBMS是使用Turbo-Prolog自行完成的一个管理系统，用来管理规则。约定若p.dbf是一个DBASE-Ⅲ文件，则在IDB（内涵数据库）中出现以其数据相关的规则时，以$p(x_1, \cdots, x_n)$作为谓词。IDB中存贮形为$B_0:-B_1, \cdots, B_m$的规则，$B_0:-B_1, \cdots, B_m$和$B_0 \leftarrow B_1, \cdots, B_m$等价。接口完成把p.dbf转化成Turbo-Prolog事实，形式为$p(x_1, \cdots, x_n)$，n为域数。DDBMS负责DDB（动态数据库）的生成。TPG是一个程序生成器，根据询问和DDB自动生成一个Turbo-Prolog程序，该程序的运行结果即是询问的回答。归纳机制则利用p.dbf生成新的.dbf文件和规则。约定：所有谓词在DBASE-Ⅲ中有结构说明。

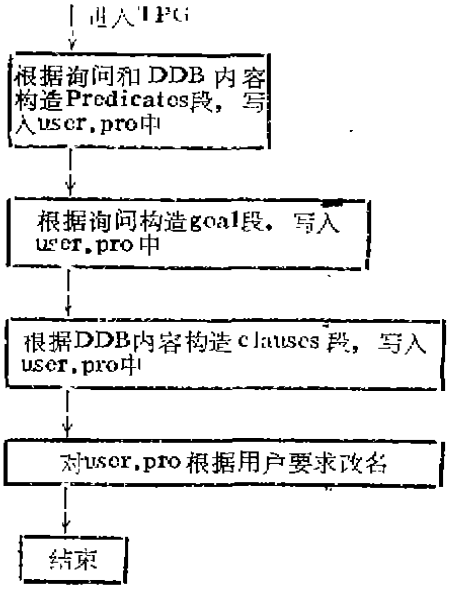
四、主要算法

下面给出几个Duce-92的主要算法的框图（算法的细节参见文[6, 7]）。

4.1 DDBMS的DDB生成算法



4.2 TPG算法



4.3 归纳机制的机器发现算法

本算法对f.dbf的数据进行归纳，生成函数$f(x_1, \cdots, x_n)$的定义，若是事实则重新生成f.dbf并写入，若是规则，写入IDB。本算法分两步骤。第一步是对f.dbf的结构的识别，找出字段名$x_1, \cdots, x_n$，文件名是用户给出的。第二步是函数或谓词的机器发现及相应的写入。其中函数的机器发现是核心。约定函数$f(x_1, \cdots, x_n)$表示成谓词$f(x_1, \cdots, x_n, z)$，$z=f(x_1, \cdots, x_n)$的形式，谓词和函数同名。基于此约定，我们给出了能发现递归函数或递归规则的启发式发现算法HR，HR是一个归纳算法[7]。

机器发现算法HR

输入：变元组$(x_1, \cdots, x_n, y)$及其数据$\{(x_{11}, \cdots, x_{n1}, y_1), \cdots, (x_{1m}, \cdots, x_{nm}, y_m)\}$和限定系数R，$0 \leqslant R \leqslant 1$。

输出：$y=f(x_1, \cdots, x_n)$的定义。

第一步：计算$(x_1, \cdots, x_n, y)$的数据的启发式信息r（r的定义见附录），若$r \geqslant R$，则发现$y=a_0+a_1x_1+\cdots+a_nx_n$，其中$a_i(0 \leqslant i \leqslant n)$由最小二乘法计算得出，结束；若$r<R$，则进入第二步；

第二步：对$(x_1, \cdots, x_n, y)$的数据进行分解，形成新函数$A(x_2, \cdots, x_n)$，$B(x_1, \cdots, x_n, f(x_1, \cdots, x_n))$的数据，其中$A(x_2, \cdots, x_n)=f(0, x_2, \cdots, x_n)$，$B(x_1, \cdots, x_n, f(x_1, \cdots, x_n))=f(x_1+1, x_2, \cdots, x_n)$；计算A和B的数据的启发式信息$r_A$和$r_B$以及$r_f$；若$r_f<\min(r_A, r_B)$，则进入发现函数A和B的过程，即重新进入HR两次；发现结果为$f(x_1, \cdots, x_n)$的定义：

$$f(0, x_2, \cdots, x_n)=A(x_2, \cdots, x_n)$$

$$f(x_1+1, x_2, \cdots, x_n)=B(x_1, \cdots, x_n, f(x_1, \cdots, x_n))$$

若$r_f\geq\min(r_A, r_B)$，则进入第三步；

第三步：对每个$g_i\in T$，T是一个特定的一元函数集，如$\{1/x, \sqrt{x}\}$等；形成$(x_1, \cdots, x_n, g_i(y))$的数据，并计算出$r_i$，取$r_j=\max\{r_i/i=1, \cdots, |T|\}$，进入新函数$A(x_1, \cdots, x_n)=g_j(y))$的发现过程，结果为$f(x_1, \cdots x_n)=g_j^{-1}(A(x_1, \cdots, x_n))$。

HR算法的第一步是奠基，发现线性函数；第二步是发现采用原始递归式定义的函数；第三步是发现采用合成定义的函数，但对合成做了一些限制，即把一般的合成$h(g_1(x_1, \cdots, x_n), \cdots, g_m(x_1, \cdots, x_n))$限制到$m=1$，并且所有h构成集合T。进一步假定HR算法第二步中变元x_1的值域为$\{0, 1, \cdots, N\}$的一个非负整数集，否则需做一步舍入工作或变换来实现上述要求。

五、例子

表1 f.dbf

x	y
0	0
1	1
2	4
3	9
4	16

表2 f.dbf

x	y
0	0

为了完整地体现Ducc-92的功能，我们设计了下列数据库，其中EDB如表1，IDB=ϕ。显然，在查询f(5)时，将面临失败。但是此时若使用HR算法（R=1），可发现：

$f(0)=0$

$f(x+1)=2x+f(x)+1$

这时EDB变为表2，IDB变为$\{f(x+1)=2x+f(x)+1\}$。并且可以回答查询$f(5)=25$。

上述例子简要地说明了机器发现在演绎数据库中的应用：（1）查询失败的进一步处理，（2）规则的自动生成。但在考虑规则的自动生成时，有很多问题有待于研究，目前我们在Ducc-92中只实现了关于要归纳的f.dbf当IDB中无关于f的规则的情形。对于更复杂的情形有待于进一步考虑和实现。

六、结论

作为基于归纳的机器发现和演绎数据库的结合而设计和实现的数据库系统，在查询失败的进一步处理和规则自动生成方面做了初步的尝试。Ducc-92具有演绎数据库系统的全部优点并具有一定归纳能力。但还有许多工作有待进一步研究，如归纳结果的进一步动态修正等。（附录：启发式信息的定义见P.80）。

参考文献

[1] Grant, J. & Minker, J., The Impact of Logic Programming on Databases, Comunications of the ACM, March, 1992, Vol.35, No.3

[2] Green, C.C. et al., The use of theorem-proving techniques to question-answering systems, In Proc. of the 23rd National Conference of the ACM, 1968,

[3] Reiter, R., Towards a Logical Reconstruction of Relational Database Theory, In On Conceptual Modeling, M.L. Brodie, et al. (eds.) Springer-Verlag, 1984

[4] Ullman, J.D., Principles of Database and Knowledge-Base Sytems, Vol.1 & Vol.2, Computer Science Press, 1988

[5] Langley, P., Simon, H.A., et al., Scientific Discovery, An Account of the Creative Process, Cambridge, MA: MIT Press, 1987

[6] 李爱中，模型发现和智能决策支持系统工具研究，哈尔滨工业大学博士学位论文，1991

[7] 李爱中，刘叙华，统计信息、递归与机器发现，计算机科学，Vol.20 No1 1993

他们反对的理由不外乎是下面这些。

没有必要。他们无非认为，计算机科学今天的数学教育已经够用了，没有必要去自找麻烦。针对这个观点，我觉得引用一下Dij[illegible]的一句话是有用的。"关键的问题在于，大学应当为社会提供[illegible]力的领袖呢，还是仅仅按照社会的要求来培训人才？如果是前者，那自然大学就应当根据发展了的情况，去调整、更新和充实自己的教学内容。大学甚至应该去开创人类新的知识，即先于客观的需要而开拓、创造人类新的精神文明。如果是这样，那些没有必要的理由就站不住脚了。"

办不到。这种理由似乎也赞成作些改进、调整或充实。但他们认为要按我们上边的观点那样进行全面的充实和加强，则太多了，在有限的大学教育期间办不到。确实，如果要把我们提到的所有数学分枝，把各个分枝的所有内容都安排到大学四年期间内，无疑会大大加重学生的负担。这样当然办不到。

没有区别，就没有政策。我们提出全面加强计算机科学中的数学教育，只能是根据大学四年中学生的实际负担所能承受的程度，来考虑我们的安排。完全作为独立的一门一门课程来把需要的各个数学分枝加到计算机科学的课程表中去，显然负担太重。因此我建议新加一门课程，叫做连续数学，它可以像离散数学那样，集各种连续性的数学于一体，内容可以取自复变函数、实变函数、泛函分析、概率论、排队论、规划论等。这作为对计算机科学数学教育加强的具体措施之一。然后在研究生教育中，则可根据所攻读的方向，再进一步加大有关数学内容的份量。这样，就可以解决增加份量过大，超过学生承受能力的问题。

太数学化了。这又是一种似是而非的反对意见。我们无意使计算机科学又回到数学去，也无意和不可能去把今天或明天的学生培养成既是数学家又是计算机科学家，或者既是后者又是前者；我们的目标却是，使计算机系的学生具有良好的数学功底。如果这就是所谓的太数学化，那我们认为绝对需要。为了实现这样一个目标，我们不仅要在计算机系里开设上边提到的课程，还应该把数学和逻辑贯穿于一些计算机的课程中。如果说Dijkstra的一些观点有些偏颇之处，他对于在程序设计课程中就加强对学生的逻辑思维的训练这一观点是很有见地和正确的。要向学生们强调，程序员的任务不仅是把程序写出来，而且应尽力形式地证明，所写程序满足形式的功能说明。一旦他们接受了这一训练，他们就为能力的飞跃打下良好的基础。这也就是我们探索计算机科学的数学教育所要追求的目标。

（参考文献略）

（接第29页）

附录　启发式信息的定义

启发式信息的定义是基于统计学中线性相关系数的概念的，作者做了一点改动。

1°。单变元情况下启发式信息r定义如下：

$$r=\frac{\left|\sum_{i=1}^{m}(x_i-\bar{x})(y_i-\bar{y})\right|}{\sqrt{\sum_{i=1}^{m}(x_i-\bar{x})^2\sum(y_i-\bar{y})^2}}$$

其中：

$$\bar{x}=\frac{1}{m}\sum_{i=1}^{m}x_i \qquad \bar{y}=\frac{1}{m}\sum_{i=1}^{m}y_i$$

2°。多变元情况下启发式信息r定义如下：

$$r=\sqrt{\sum_{i=1}^{n}l_{i0}b_i}\Big/\sqrt{l_{00}}$$

其中：

$$l_{ij}=\sum_{k=1}^{m}(x_{ik}-\bar{x}_i)(x_{jk}-\bar{x}_j),(i,\ j=1,\cdots,n)$$

$$l_{i0}=\sum_{k=1}^{m}(x_{ik}-\bar{x}_i)(y_k-\bar{y}),(i=1,\cdots n)$$

$$l_{00}=\sum_{k=1}^{m}(y_k-\bar{y})^2$$

$$\bar{y}=\frac{1}{m}\sum_{k=1}^{m}y_k$$

$$\bar{x}_i=\frac{1}{m}\sum_{k=1}^{m}x_{ik}$$

$$L=\begin{vmatrix}l_{11}\cdots l_{1n}\\ \cdots\cdots\\ l_{n1}\cdots l_{nn}\end{vmatrix} \qquad L^{-1}=\begin{vmatrix}c_{11}\cdots c_{1n}\\ \cdots\cdots\\ c_{n1}\cdots c_{nn}\end{vmatrix}$$

$$B=L^{-1}L_0 \qquad B=(b_1,\cdots,b_n)^*$$

$$L_0=(l_{10},\cdots,l_{n0})^*。$$