计算机科学 1997 Vol. 24 №. 6

逻辑程序中的否定问题与非单调逻辑*)

Negation in Logic Programs and Nonmonotonic Logic

陈 荣 孙吉贵

(吉林大学计算机科学系 长春 130023)

摘 要 There are many results on the research of the declarative semantics of logic programs with negation as failure in recent years. In this paper, we compare these important semantics of normal logic programs, show their close relations with three nonmonotonic reasoning formalisms. We also present some new results on the semantics of extended logic programs and disjunctive logic programs, and introduce an unified formalism of various semantics based on autoepisdemic logic.

关键词 Logic programming, Negation as failure, Strong negation, Nonmonotonic reasoning

1 引言

常识推理的一个共同特性是在不完全知识下的推理，非单调推理提供了不完全知识下的推理方法，它能在有限知识的基础上得出一些结论，而当知识进一步丰富时这些结论是可撤销的。最著名的非单调理论有 McCarthy 的限制方法、Reiter 的缺省理论、Moore 的自认知逻辑(autoepisdemic logic)，其中限制理论是基于最小化模型的，它最小化的是部分谓词的外延，缺省逻辑与自认知逻辑是不动点非单调逻辑，缺省推理是关于事物或然性的推理，而自认知推理是关于"知道"和"相信"的推理，也就是关于自我知识和信念的推理。

逻辑程序的一个突出特点是通过失败即否定规则(Negation as Failure)可以很自然地支持非单调推理。尽管逻辑程序的语法有限，但是单非调推理中的许多概念在其中都有一个自然的对应物，二者有着密切的联系：一方面，作为推理机制和知识表示方法的逻辑程序可以有效支持非单调理论的子集；另一方面，考虑到实现效率和支持非单调推理要引入失败即否定，为了增强知识表示能力要引入强否定、析取式的头部等，而进一步扩展逻辑程序，但是，找到一个能抓住该程序主观含义的说明语义是一个困难而且重要的问题，它的解决需要得到非单调理论的支持。注意到这个关系，我们有充分的理由对否定采用非单调的语义，这是基于下述考虑的：

1)在逻辑程序中，尤其是在演绎数据库中只存

[5] A. Homaifar, IEEE Tran. Fuzzy Systems, 3(2), 1995, 129-139
[6] H. Ishigami, Fuzzy Sets and Systems, 71(3), 1995, 257-264
[7] C. L. Karr, ICGA' 91, 450-457
[8] J. Kim, IEEE Trans. Fuzzy Syatems, 4(3), 1996, 213-226
[9] J. Kinzel, IEEE-ICEC' 94, 28-33
[10] M. A. Lee, IEEE-FUZZ' 93, 612-617
[11] 李洪兴，模糊系统与数学，9(5)，1995
[12] J. Liska, IEEE-FUZZ' 94, 1377-1382
[13] Li RenHou, Fuzzy Sets and Syatems, 83(1), 1996, 1-10
[14] L. Magdalena, IEEE-FUZZ' 95, 1111-1117
[15] Z. Michalewicz, genetic algorithms + data structrues = evolution programs, Springer-Verlag, 1994
[16] D. Park, IEEE Trans. System Man and Cybernet, 24(1), 1994
[17] Y. S. Tang, Fuzzy Sets and Systems, 83(3), 1996, 301-310
[18] P. Thrift, ICGA' 91, 509-513
[19] A. Varse, IEEE Trans. Systems Man and Cybernet., 23(5), 1993
[20] 王立新，自适应模糊系统与控制——设计与稳定性分析，国防工业出版社，1995

*)国家自然科学基金和国家教委博士点基金资助课题。

储明显的正文字信息可以大大减少空间开销。

2)具有非单调语义的逻辑程序本身就是一个可实现的非单调系统。

3)完全知识通常是可望而不可及的,对否定采用经典的解释会导致全部的一阶理论证明而增加复杂度。

对于常规逻辑程序,出现了许多种说明语义,例如有 Clark 语义及其 Fitting 的三值扩充、完美模型语义、稳定模型语义和良基模型语义等等。本文比较了常规逻辑程序的主要说明语义,讨论其中模型论语义与非单调逻辑的联系,给出关于扩展逻辑程序和析取逻辑程序的说明语义的新结果,并介绍建立在自认知逻辑基础上的逻辑程序模型论语义的统一框架理论。

2 常规逻辑程序的语义方法的比较

逻辑程序的语义问题激起了众多人的研究兴趣,自然出现了许多处理方法,但归纳起来看,在人工智能领域里人们使用非单调理论来处理否定信息;逻辑程序领域里的总体思想是变换程序并把变换结果的模型作为程序的主观含义。这方面已有的工作自然分成两个方向:一个是"程序完整化"方法;另一个是"正则模型"方法。第一种方法是 Clark 于 70 年代末提出的[1,2],后来 Fitting、Kunen 都深入探讨了这一方法。这种方法的缺点是某些程序完整化以后可能变得非协调;有些虽然是协调的,但含义不自然——不符合人们对程序的直观理解[1]。第二种方法是模型论方法。为了描述如何抓住逻辑程序或演绎数据库的主观含义,Toper 和 Soneberg 两人非正式地提出"正则模型"的概念,它的含义是从一组极小模型中按一定标准选择某模型。但这种方法的问题是程序可能没有唯一的正则模型,并且即使正则模型存在,也不清楚它是否能反映程序的主观含义[1]。完美模型语义就是第二种方法的突出代表,它克服了"程序完整化"方法的缺点,其结果和程序的"自然"的主观含义是吻合的,但遗憾的是它只适用于一类很窄的逻辑程序(局部分层程序),由此对它有两种重要的扩展:稳定模型语义[2]和良基模型语义[1,2]。

了解了这些说明语义方法的发展轨迹,我们不想详细叙述这些成果,这部分内容请参见文[2]。下面给出两个浅显的程序例,从中可以对说明语义的主要能力有一个直观的认识。

例 2.1 程序 P 为 $a \leftarrow \neg b$

$b \leftarrow \neg a$

$c \leftarrow a, b$

$a \leftarrow \neg c$

Fitting 模型和良基模型为 $\varnothing$,但存在稳定模型 $\{a, \neg b, \neg c\}$。

例 2.2 程序 P 为 $p \leftarrow q$

$p \leftarrow r, \neg q$

$q \leftarrow q$

$r \leftarrow$

$s \leftarrow \neg s$

Fitting 模型和良基模型为 $\{(p, r), \varnothing\}$,稳定模型不存在。

可以看出,只有稳定模型语义准确反映了例 2.1 的含义,但对于例 2.2 它却无能为力(因为程序中最后一个规则),Fitting 语义和良基模型语义都抓住了该程序的主观含义。以上的实例暗示着可能不存在一个完美的说明语义。事实上,一个程序可能没有、或有一个、或有多个稳定模型,当程序具有唯一的稳定模型时,这一模型可以考虑为程序的"自然"的主观含义。与其相比,良基语义适用于全部的逻辑程序,它是一个部分模型(或称为三值模型,Fitting 模型也如此,稳定模型是二值的),已经证明完美模型的许多性质良基模型也具备,并且良基模型的最小不动点定义还导致了任意逻辑程序的自然的"动态分层"的概念,但对于表示"分情况证明"的逻辑程序,它还存在困难[4]。

我们想依据以下三条原则来比较具有代表性的说明语义:

1)说明语义反映常识和主观含义的能力(应注意到这个标准是比较模糊的)。

2)计算复杂性。

3)给定语义所刻画的语言的表达能力。

首先我们解释一下"语言表达能力"的含义。逻辑程序的一个重要应用是作为给定数据库上的提问语言,按照 Reiter 的术语,"外延数据库 EDB"是逻辑程序的事实集,逻辑程序中由规则定义的关系被称为"内涵数据库 IDB"。在这种意义下,一个逻辑程序(用不同的方式解释)定义了一个从 EDB 基例到 IDB 基例的映射(如图 2.1),称这样的映射为一个"概念"。提问语言的表达能力是指"是否存在该语言不能表达的概念?"。Kolaitis 证明了非分层程序类比分层程序类的表达能力强,Schlipf 研究了良基模型语义、稳定模型语义和 Fitting 语义的表达能力,他证明了在整数结构、Herbrand 结构和其他"合理"

结构上三者具有相同的表达能力。

图 2.1 概念——从 EDB 到 IDB 的映射

就原则 1 而论，稳定模型语义和良基模型语义具有更强的能力，并且二者具有密切的关系，Dung 的研究结果表明"稳定模型语义是二值的良基语义"、"二值强完整化确定稳定模型语义"、"三值强完整化确定良基语义"[3]。

已有的研究结果表明：基于模型论语义的逻辑程序系统是一种非单调推理系统，因此每个模型论语义都可以确定一种非单调推理关系。Kraus 等人曾定义了非单调推理关系 $|\sim$ 的几种性质：令 Γ 是一理论，F 和 G 是任意公式，定义：

(1)cut：$\Gamma|\sim F$ 且 $\Gamma\cup\{F\}|\sim G$

(2)谨慎的单调性：$\Gamma|\sim F$ 且 $\Gamma|\sim G$ 蕴涵 $\Gamma\cup\{F\}|\sim G$

(3)合理性：$\Gamma|\sim\neg F$ 且 $\Gamma|\sim G$ 蕴涵 $\Gamma\cup\{F\}|\sim G$

其中性质(1)和(2)又被统称为累积性。Dix 为逻辑程序定义了一种推理关系 $|\sim_P^{SEM}$：给定任一逻辑程序 P，令 SEM 是它的某一模型论语义，A_i 和 L_j 分别是原子和文字，定义：

$\{A_1,\cdots,A_n\}|\sim_P^{SEM}\{L_1,\cdots,L_m\}$，如果 $SEM(P\cup\{A_1,\cdots,A_n\})|=\{L_1,\cdots,L_m\}$[1]。

针对逻辑程序，Dix 又定义了非单调推理关系的其它性质，它们的直观含义是：

(4)部分求值原理——即规则体中任意文字可以被它的定义(所有以该文字为头的规则的体组成的析取式)所代替。

(5)相干性(relevance)——即某原子的真值只由程序中它所依赖(规则头依赖于规则体)的部分决定。

就逻辑程序而言，直观上，累积性表明增加引理不改变程序的语义，如果程序语义所确定的推理关系满足该性质，那么该语义是可以迭代构造的；合理性表明增加对不可证结论的否定不改变程序的语义。良基模型语义所确定的推理关系满足以上全部性质，但稳定模型语义所确定的推理关系不满足累积性、合理性、相干性。就计算复杂性讲，稳定模型是非构造的，计算稳定模型是 NP 问题；Fitting 模型可迭代产生；良基模型的计算复杂性不高，可以在多项式时间内完成[1]。

3 非单调逻辑与逻辑程序的模型论语义

Przymusinski 于 90 年左右证明了良基模型语义与四种主要的非单调理论的三值形式是等价的[4]。从语义上讲，限制理论与逻辑程序都是基于最小化模型的，两者的区别在于，逻辑程序最小化所有的谓词外延，且谓词最小化的次序完全由句子的定义决定，例如给定一句子 $H\leftarrow L_1,\cdots,L_n$，其最小化的次序为 $L_1,\cdots,L_n,H$；而限制方法仅最小化理论中部分谓词的外延，并允许其它一部分谓词的外延随之变化。分层逻辑程序中谓词最小化的次序与它的谓词划分 $\{P_1,\cdots,P_n\}$ 是一致的，Lifschitz 的优先限制(prioritized circumscription)是在限制中明显引入谓词最小化序的概念，这与层次逻辑程序中谓词最小化的次序有异曲同工之妙，业已证明：分层程序 T 中，若谓词最小化的次序为 $\{P_1,\cdots,P_n\}$，它的极小模型完全等价于 $CIRC(T;P_1>\cdots>P_n)$，可以使用限制方法研究分层逻辑程序的说明语义。

基于非单调理论的逻辑程序语义研究方法还有：基于缺省理论的缺省语义和基于自认知逻辑的稳定语义。这两种方法都需一个翻译过程——即把逻辑程序转换成相应的缺省/自认知理论，然后程序语义根据它的转换结果的扩充/稳定扩充而定。给定一个常规逻辑程序 P，可以由 P 构造一个缺省理论 (D_p,W_p)：

(1)若 $A\leftarrow B_1,\cdots,B_k$ 是 P 中一个不含"失败即否定"的子句，则 $B_1,\cdots,B_k\supset A\in W_p$。

(2)若 $A\leftarrow B_1,\cdots,B_k,\mathbf{not}D_1,\cdots,\mathbf{not}D_m$ 是 P 中一个含"失败即否定"的子句，则缺省规则 $\dfrac{B_1\wedge\cdots\wedge B_k:\neg D_1,\cdots,\neg D_m}{A}$ 属于 D_p。

有了以上的翻译过程，令 Cn 是经典逻辑里的命题结果关系闭包，下面的定理反映了缺省逻辑与稳定模型间的联系：

定理 3.1[5] M 是常规逻辑程序 P 的稳定模型当且仅当 M 是一个极大原子集且它满足 $Cn(W_p\cup M)$ 是缺省理论 (D_p,W_p) 的一个扩充。

如何把一个常规逻辑程序P翻译成自认知理论

(1)SEM 可以返回单一模型、一个理论、一组模型。如果返回是一组模型，则采用谨慎推理(skeptical reasoning)：$SEM(P)|=L$，如果在 SEM(P)的所有模型中 L 都为真。

呢？若 $A\leftarrow B_1,\cdots,B_k,$ **not** $D_1,\cdots,$ **not** D_m 是 P 中任一例化子句，则把它翻译成 $B_1\wedge\cdots\wedge B_k\wedge\neg LD_1\wedge\cdots\wedge\neg L\,D_m\supset A$（其中 L 是 Moore 的自认知逻辑 AEL 中的模态算子）。于是有：

定理 3.2[5] 令 M 是常规逻辑程序 P 的稳定模型，P 经翻译得到自认知理论 W_p，W_p 的稳定扩充 T 和 M 是一致的，即对于任意原子 A 都有：

$A\in M$ 当且仅当 $LA\in T$，

$\neg A\in M$ 当且仅当 $\neg LA\in T$。

4 扩展逻辑程序和析取逻辑程序的说明语义

最近几年，人们研究了扩展逻辑程序（包含强否定（亦称经典否定）和失败即否定）和析取逻辑程序（规则的头部是析取式）的说明语义问题，这种语法的逻辑程序具有更强的常识表达能力。这方面的结果主要是推广了良基模型语义和稳定模型语义。

扩展逻辑程序的规则都形如：

$$L_0\leftarrow L_1,\cdots,L_m,\ \textbf{not}\ L_{m+1},\cdots,\textbf{not}\ L_n \qquad (1)$$

其中 $L_i(i=0,\cdots,n)$ 为文字。

析取逻辑程序的规则都形如：

$$L_1|\cdots|L_k\leftarrow L_{k+1},\cdots,L_m,\ \textbf{not}\ L_{m+1},\cdots,\textbf{not}\ L_n \qquad (2)$$

其中 $L_i(i=1,\cdots,n)$ 为文字，注意式(2)中我们使用“|”而不是“∨”作为析取符号，这因为此处的析取和经典逻辑里的析取的含义是有区别的——粗略地讲规则头部的析取式中的原子的真性是排他的，式(2)可以分裂成 k 个以 $L_i(i=1,\cdots,k)$ 为头且规则体不变的子句，那么一个析取逻辑程序可以语义等价于多个扩展逻辑程序的集合，自然地，扩展逻辑程序的说明语义确定后，析取逻辑程序的说明语义就被确定了。当然，也有某些说明语义使用析取的经典含义。

如果程序中包含有强否定，通常的办法是将每个负文字一一对应替换成一个新文字，那么经过变换的程序就只包含失败即否定，求解这种程序的说明语义模型，然后把模型中的新文字替换成与它对应的负文字，此时如果模型包含有互补文字，则简单的办法是：“矛盾蕴涵一切”或“把矛盾的模型扔掉”，因此包含强否定并不需要新的说明语义。综上所述，我们只需详细给出一种说明语义在析取逻辑程序或扩展逻辑程序中的版本即可。

由于式(1)和(2)的头部既可以是正文字也可以是负文字，一般地，一个扩展逻辑程序可能是不相容的；实际上用扩展逻辑程序表示一个大规模知识基时可能保证不了一致性，一个知识基可能是局部不相容的，但它可能大体上有一个“合理的”主观含义，自然，如何处理局部矛盾也是一个有意义的问题，因此下文中在谈到某种说明语义时，也要谈它是如何处理矛盾模型的。

贯穿全文，将形如 **not** L 的文字称为缺省文字，并约定：一个逻辑程序 P 的 Herbrand 基记为 HB；给定一文字集合 S，把 S 中的每个文字取强否定（失败即否定）而得到的集合记为 $\neg$S(**not** S)。回忆一下直接结果算子的定义，令 P 是一个不含否定的常规逻辑程序，直接结果算子 $T_p(I)$ 的输入是一个 Herbrand 解释 I，$A\in T_p(I)$ 当且仅当存在 P 的一个例化规则，其头部是 A，而规则体里的所有文字在解释 I 下都为真。

4.1 回答集语义

Gelfond 和 Lifschitz 把稳定模型语义推广到扩展逻辑程序和析取逻辑程序中，并把它称为“回答集语义”[6]。

定义 4.1 令 P 是任意例化的扩展逻辑程序，S 是 P 的任意基文字集合，GL(S,P) 是由 P 删除情况(1)下的规则和情况(2)下的缺省文字而得到的不含 **not** 的新程序：①每个规则体中存在缺省文字 **not** L 满足 $L\in S$；②剩余规则体中的所有缺省文字。这种变换被称为 GL_变换。

定义 4.2 令 Q 是一个不含 **not** 的扩展逻辑程序，Q 的回答集合是满足如下条件的 $HB\cup\neg HB$ 的最小子集 S：(1)$S=T_Q^{\infty}(\varnothing)$，(2)若存在 $L\in S$ 且 $\neg L\in S$，则 $S=HB\cup\neg HB$。

定义 4.3 给定一扩展逻辑程序 P，假设 S 是 P 的一个基文字集合，令 $Q=GL(S,P)$，若有 $S=T_Q^{\infty}(\varnothing)$，则称 S 是稳定的，S 是程序 P 的回答集。

对于析取逻辑程序，需把定义 4.2 修改一下：

定义 4.4 令 Q 是一个不含 **not** 的析取逻辑程序，Q 的回答集合是由满足如下条件的 $HB\cup\neg HB$ 的最小子集 S 组成的集合：①对于每个例化规则 $L_1|\cdots|L_k\leftarrow L_{k+1},\cdots,L_m$，若有 $L_{k+1},\cdots,L_m\in S$，则对某个 $i=1,\cdots,k$，$L_i\in S$；②若存在 $L\in S$ 且 $\neg L\in S$，则 $S=HB\cup\neg HB$。

类似地，给定一个析取逻辑程序 P，假设 S 是一个基文字集合，可以对 P 应用 GL_变换，令 $Q=GL(P,S)$，定义 4.4 确定了 Q 的回答集合，如果 S 是其中的一员，则 S 是稳定的，S 是扩展逻辑程序 P 的一个回答集。应注意，定义 4.2 和定义 4.4 的②表明，

回答集语义对矛盾的处理是简单的——局部矛盾毁坏了所有有意义的信息。

显然，回答集语义是二值的，它与缺省理论的缺省扩充和自认知逻辑的稳定扩充的等价关系与第3节类同(以下各小节中推广的稳定模型语义都有类似的结论)，只不过翻译方法可能不同，这里我们不想展开讨论。下面给出一个例子，容易看出该程序例中析取的含义是排他的。

例4.1 考虑文[6]里的实例。

employed(Jack,Stanford)|employed(Jack,SRI)←

adequate_income(x)←employed(x,y)

¬employed(x,y)←not employed(x,y)

这个程序有两个回答集：{employed(Jack,Stanford)，¬employed(Jack,SRI)，adequate_income(Jack)}和{employed(Jack,SRI)，¬employed(Jack,Stanford)，adequate_income(Jack)}。

4.3 三值稳定模型语义

本节的内容是建立在三值逻辑基础上的，原子的真值有三种：t(真)、f(假)、u(未定义)，也把这三种真值视为命题原子。Przymusinski提出的三值稳定模型语义[7]既推广了良基模型语义也推广了稳定模型语义。

定义4.5 三值Herbrand解释I是一个对偶I=(T,F)，其中T和F是HB的两个不相交的子集，集合T由在I下为真的基原子组成，集合F由在I下为假的基原子组成。令U=HB−T∪F，U中基原子在I下的真值是未定义。

显然，U为空时，I就退化成二值的了。Przymusinski稍微修改了GL_变换：

定义4.6 令P是一个析取逻辑程序，M是一个三值Herbrand解释，P的扩展GL_变换结果GL(P,M)定义为：对于P中任意例化规则，若该规则中缺省文字在M下为真(假/未定义)，则把它替换为命题原子t(f/u)，对P的所有例化规则都执行这种替换而得到GL(P,M)。

易见，对于程序GL(P,M)，如果t出现于某规则体中，则可把t删掉；如果f出现于某规则体中，则可删除此规则；但对于剩余的规则，规则体中虽包含u也要保留下来。既然GL(P,M)不包含not，它的极小模型存在(可能不唯一，并且不考虑那些包含矛盾的模型)，如果M也是GL(P,M)的一个相容的极小模型，则M就是P的一个三值稳定模型。令STBMOD(P)为P的三值稳定模型的集合，程序P的三值稳定语义是这样的：

定义4.7 给定一析取逻辑程序P，P的三值稳定语义STABLE3(P)由STBMOD(P)决定：公式F在STABLE3(P)下为真(假)当且仅当F在STBMOD(P)的所有模型中都为真(假)；否则F在STABLE3(P)下的真值未定义。

例4.2 考虑程序P={1. B|C← 2. A←not B 3. A←not C}，它有两个二值稳定模型M_1=({A,B},{C})，M_2=({A,C},{B})，则STABLE3(P)蕴涵A，B∨C，¬B∨¬C。

4.3 静止语义

Przymusinski的静止语义[8]中析取式的语义是附和经典逻辑的，因此，在本小节，规则的析取式头部使用经典的析取符号∨。为方便起见，假定本节语法只考虑失败即否定(事实上，静止语义中对矛盾的处理也是扔掉不相容的模型)。本质上，静止模型语义是基于GCWA(广义封闭世界假设)的——给定程序P，$P\models_{GCWA}\neg A$当且仅当¬A在P的所有极小模型中都为真。

为了定义静止语义，要扩充语言包含一个新的基原子集合HB'，使得：①HB与HB'的交集为空。②对于每个$p\in HB$，存在与它对应的信念原子$Bp\in HB'$。给定一析取程序P，通过将它的每个缺省文字not p变成$\neg Bp$而得到的新程序记为P_B。为方便起见，在本节以下的叙述中，假定析取逻辑程序P都是类似P_B的程序，这样析取逻辑程序的子句都形如：$A_1\vee\cdots\vee A_n\leftarrow B_1\wedge\cdots\wedge B_m\wedge\neg BC_1\wedge\cdots\wedge\neg BC_k$。

定义4.8 一个Herbrand解释I是相容的，如果对于任意基原子A，I中不同时存在A和$\neg BA$。

例如解释I={A,BA,¬C,BC}就是相容的。应该强调：本节中的联结符←是"构造蕴涵"(constructive implication)，F←G不等价于F∨¬G，若G非真且F非假，则该子句总为真。假如I={A,¬BC}，容易验证I⊨B←¬B，I⊨A←B，但I弄假¬A←B。如果把Herbrand解释I看成对偶(I^+,I^-)，其中I^+代表正原子集合，I^-代表负原子集合，则有如下定义：

定义4.9 令I=(I^+,I^-)和J=(J^+,J^-)为两个Herbrand解释，定义I≤J，如果$I^+\subseteq J^+$且$I^-\supseteq J^-$。

以下我们使用的极小模型、最小模型都是指≤-极小模型、≤-最小模型。解释间的序关系≤反映这样的直观含义"对正原子极小化的同时，尽可能极大化负原子(包括负信念原子)"。

例 4.3　考虑程序 P={A←¬BC}，它有三个极小模型{¬C，¬BC，A}、{¬C，¬BA}、{¬C，BC，¬A，¬BA}。应注意到如果在第二个极小模型中加入文字¬A，那它会演变成第三个极小模型。

定义 4.10　称信念原子的析取式 $BA_1 \vee \cdots \vee BA_n$ 为正信念析取式，称负信念原子的析取式 $\neg BA_1 \vee \cdots \vee \neg BA_n$ 为负信念析取式。析取逻辑程序 P 的一个 Herbrand 状态 S 是任何一个相容的正、负信念析取式的集合。

对于 Herbrand 状态，我们定义如下序列：

$S_0=\varnothing$，

$S_{\alpha+1}=POS(P_\alpha)\cup NEG(P_\alpha)$，若 $\alpha+$ 是一持续序数(successor ordinal)。

$S_\lambda=\bigcup_{\alpha<\lambda}S_\alpha$，若 λ 是极限序数(limit ordinal)。

其中 $P_\alpha=P\cup S_\alpha$，$POS(P_\alpha)=\{Bp_1\vee\cdots\vee Bp_n \mid P_\alpha \models_{GCWA} p_1\vee\cdots\vee p_n\}$，

$NEG(P_\alpha)=\{\neg Bp_1\vee\cdots\vee\neg Bp_n \mid P_\alpha\models_{GCWA}\neg p_1\vee\cdots\vee\neg p_n\}$。

这个序列是单调的，因而最终能到达它的不动点 $S_k=S_{k+1}$，称 S_k 为析取逻辑程序 P 的静止状态，我们定义该状态 $S_p=S_k$ 为程序 P 的正则静止状态。

定义 4.11　析取逻辑程序 P 的静止语义 STAT(P)定义为静止状态 S_k 的所有演绎结果的集合，即 $STAT(P)=\{p_1\vee\cdots\vee p_n \mid S_k\models Bp_1\vee\cdots\vee Bp_n\}\cup\{\neg p_1\vee\cdots\vee\neg p_n \mid S_k\models\neg Bp_1\vee\cdots\vee\neg Bp_n\}$

例 4.4　令程序 P={1. A∨C 2. D←¬BA 3. D←¬BC}。

显然第一条规则的极小模型是{A，¬C，¬BC}和{C，¬A，¬BA}，所以 P 有极小模型：{C，BC，BA，¬A，¬D，¬BD}、{A，BA，BC，¬C，¬D，¬BD}、{C，D，¬A，¬BA}和{A，D，¬C，¬BC}。因 $S_0=\varnothing$，$P_1=P\cup S_0=P$，$P_1\models_{GCWA}A\vee C$，$P_1\models_{GCWA}\neg A\vee\neg C$，有 $S_1=\{BA\vee BC,\neg BA\vee\neg BC\}$。$P_2=P\cup S_1$，$P_2$ 有极小模型{A，D，BA，¬C，¬BC}和{C，D，BC，¬A，¬BA}，因为 $P_2\models_{GCWA}D$，$P_2\models_{GCWA}A\vee C$，$P_2\models_{GCWA}\neg A\vee\neg C$，有 $S_2=\{BD,BA\vee BC,\neg BA\vee\neg BC\}$。可以验证 $S_2=S_3$，所以有 $S_p=S_2$。

不难发现，静止状态是程序 P 的所有极小模型的紧凑形式(这类同于三值稳定语义)。

4.4　扩展良基语义 AFSX

对于扩展逻辑程序中的矛盾的处理，回答集语义是矛盾蕴含任何公式，三值稳定模型语义的办法是把有矛盾的模型扔掉，事实上局部矛盾不应毁掉其它的可靠信息。我们在文[9]中，以 Belnap 的格值逻辑(原子的四种真值是⊥(未定义)、t(真)、f(假)和⊤(矛盾))为基础，一个原子 A 可以有 A、¬A、not A 和 not¬A 四种语义形式——它们都被视为原子(not 是模态算子)，提出一种扩展良基语义 AFSX。给定一扩展逻辑程序 P 后，AFSX 在两个层次上处理矛盾，先将它变换为含义更清晰的程序 P^c，在此层次上可以避免一些明显的矛盾；由于 AFSX 是可迭代构造的，可以标记那些依赖于矛盾的结论。

由于篇幅所限，我们想结合实例只给出 AFSX 的思想。在四值格逻辑下，对于任意文字 L，¬L⊃not L 成立，把 not 读作"证不出"。

定义 4.12　给定一个扩展逻辑程序 P，它的解释 I 可表示成对偶 I=(Pr，Df)，其中 Pr 是基文字的集合，Df 是由缺省基文字组成的集合，并且 I 满足：对于任意基文字 L∈Pr，若¬L∉Pr，则 not¬L∈Df；对于某些基文字 L∈HB-Pr，或者 not L∈Df 或者 not L∈Df 且 not¬L∈Df。若解释 I 满足 P 的所有规则，则称 I 是 P 的一个(部分)模型。

直观上，Pr 代表可证文字的集合，它等同于 Herbrand 解释；Df 代表缺省文字集，它依赖于 Pr。具体地说，令 L 是一正文字，如果 L(¬L)属于 Pr，则 L 在 I 下为真(假)，特别地，如果 L∈Pr 且¬L∈Pr 则 L 在 I 下是矛盾的。

定义 4.13　给定一程序 P，假设正文字 L 满足 L∈Head(P)并且¬L∈Head(P)，则定义 L-类规则变换：令任意以 L 为头的规则为 L←B(其中 B 是一个合取式)，定义该规则的对应物是 L←B，not¬L，所有以 L 为头的规则都变成它的对应物，这种变换称 L-类规则变换。对所有满足假设条件的任意文字 L 执行 L-类规则变换而得到程序 P^c。

给定一个经过 L-规则变换的例化扩展逻辑程序 P，下述序列描述了 AFSX 的思想：

$I\uparrow 0=(\varnothing,\varnothing)$

$I\uparrow\alpha+1=(T_Q(\varnothing)\uparrow\omega,Df_{\alpha+1})$

其中 $Q=GL(P,I\uparrow\alpha)$，α 是一持续序数，ω 为超限数，$Df_{\alpha+1}$按照定义 4.12 由 $T_Q(\varnothing)\uparrow\omega$ 构造而得。

当存在 λ 满足 $I\uparrow\lambda+2=I\uparrow\lambda$ 时，$I\uparrow\lambda$ 和 $I\uparrow\lambda+0$ 的交集就是 P 的 AFSX 语义。

下面给出一个简单实例说明 AFSX 的含义。

例 4.5　令扩展逻辑程序 P={1. Fly←Bird，not¬fly 2. Bird←Penguin 3. ¬Fly←Pengiun 4. Penguin← 5. Active_in_the_Daytime←not Active_in_the_Night 6. Active_in_the_Night ←not Active_in_the_Daytime}

$I\uparrow 0=(\varnothing,\varnothing)$，于是 $GL(P,I\uparrow 0)$={1. Bird←

Penguin 2. →Fly←Pengiun 3. Penguin← 4. Active_in_the_Daytime← 5. Active_in_the_Night←}，为书写简便，以下用原子中的大写字母代替该谓词，显然，GL(P,I↑0)的最小模型(即为直接结果算子的最小不动点)为{P,B,→F,AD,AN}，有 I↑1=({P,B,→F,AD,AN},not{→P,→B,F,→AD,→AN})。GL(P,I↑1)={1. Bird←Penguin 2. →Fly←Pengiun 3. Penguin←}，GL(P,I↑1)的最小模型是{P,B,→F}，于是 I↑2=({P,B,→F},not{→P,→B,F,AD,AN})。显然有，GL(P,I↑2)=GL(P,I↑0)，I↑3=I↑1，因此，若继续迭代下去，就变成一个"等幅振荡"——这也是 AFSX 的含义(alternating fixpoint semantics for extended logic porgrams)。

5 逻辑程序的模态语义

为了增强 Moore 的自认知逻辑 AEL，Przymusinski 提出了一种知识和信念自认知逻辑 AELB (autoepistemic logic of knowledge and bliefs)[10]。AELB 构成了逻辑程序各种模型论语义的统一框架理论，本节简要介绍了 AELB 的主要内容。

AEL 定义了一个模态算子 L，AELB 比 AEL 多一个新的"信念算子 B"。在 AELB 中，模态原子 LF 的含义是"F 是可证的"或者"在稳定自认知扩充中，F 被逻辑蕴涵"，可以把 LF 读作"知道 F"；模态原子 BF 的含义是"F 在所有的极小模型中都为真"或者"在稳定自认知扩充中，F 被极小蕴涵"——BF 体现了 GCWA 的思想，把 BF 读作"相信 F"。

定义 5.1 一个信念理论是一个子句集合，它的子句都形如：$B_1 \wedge \cdots \wedge B_m \wedge BG_1 \wedge \cdots \wedge BG_n \wedge LH_1 \wedge \cdots \wedge LH_s \wedge \neg BF_1 \wedge \cdots \wedge \neg BF_n \wedge \neg LH_1 \wedge \cdots \wedge \neg LH_t \supset A_1 \wedge \cdots \wedge A_l$，其中的蕴涵符号是经典蕴涵。

注意本文不讨论定义 5.1 中 l=0 时的子句(这样的子句退化为一个约束)。AELB 的公理如下：

(D)相容公理：$\neg B\perp$ 和 $\neg L\perp$，其中 $\perp$ 表示矛盾。

(K)常规公理：对于任意公式 F 和 G，有

$B(F\supset G)\supset(BF\supset BG)$ 和 $L(F\supset G)\supset(LF\supset LG)$。

(N)必然规则：对于任意公式 F，有

$$\frac{F}{BF} \text{ 和 } \frac{F}{LF}$$

容易验证，算子 B 和算子 L 对于合取满足分配律。

定义 5.2 称信念理论 T° 为信念理论 T 的静态自认知扩充，如果它满足不动点方程：$T^\circ=Cn^*(T\cup\{LF\mid T^\circ\models F\}\cup\{\neg LF\mid T^\circ\not\models F\}\cup\{BF\mid T^\circ\models_{min}F\})$，其中 Cn^* 是指在命题结果关系和必然规则下封闭，$T^\circ\models_{min}F$ 表示 F 在 T° 的所有极小模型中都为真。

定义 5.3 信念理论 T 的静态语义是一个公式集合，其中的所有公式都属于 T 的每个静态自认知扩充 T°。

逻辑程序到 AELB 理论的翻译过程只需考虑缺省原子即可，有意义的方法只有以下两种：令 C 是原子公式，(1)**not** $C\equiv\neg LC$，和(2)**not** $C\equiv B\neg C$。如同第 3 节一样，有了翻译方法，可以建立逻辑程序的模型与静态自认知扩充的对应关系，不过不同的语义可能要求附加一些新公理。下面列出这些附加公理，并列表说明：在什么附加公理下，不同模型论语义与静态自认知扩充是一致的。

(BCA)信念完备公理：$LBA\vee LB\neg A$，其中 A 为原子。

(DBA)析取信念公理：$B(F\vee G)\equiv BF\vee BG$，其中 F 和 G 为公式。

(PIA)正反省公理：$LA\supset A$，其中 A 为原子。

(S)强否定公理：$\sim A\supset\neg A$，其中 A 为原子，$\sim$ 表示 AELB 里的强否定。

应强调的是，逻辑程序中 **not**(失败即否定)翻译后，与经典逻辑或非标准逻辑中的经典否定 $\neg$ 对应，但是，为了翻译逻辑程序里的强否定 $\neg$，AELB 引入符号 $\sim$ 与之对应，公理(S)把 AELB 中的两种否定联系了起来。直观上，公理(BCA)暗含着，对于任意原子，或者它为真或者它为假；公理(PIA)反映了可证的结论可以作为"定理"使用。

结束语 本文阐述了逻辑程序中的否定问题与非单调逻辑的紧密联系——可以用非单调逻辑的方法来研究逻辑程序的说明语义，同时后者的研究结果也促进了模态非单调逻辑的发展，例如除 AELB 外，还有 Lifshitz 的极小信念与失败即否定逻辑 MBNF 和 Schwarz 的自反自认知逻辑 RAEL 等等。

回顾以上各节的内容，如果从研究方式的角度看，逻辑程序的语义可以分成"不动点语义"和"模型论语义"两种。前者建立在不动点理论上，不断对规则施用事实而达到的不动点即为语义模型；后者关注满足什么性质的极小模型才符合程序的主观含义，虽然如此，两者事实上是交融在一起的。应该强调不动点语义具有很强的计算特点，但我们也有理由选择模型论语义：一是说明语义指出我们需要什么。现考虑程序 $P=\{p\ q\leftarrow\rightarrow p\}$，则 $T_P(\varnothing)=\{p,q\}$，直觉上，p产生于第一条单元规则，由于使用了事实

(下转第 58 页)

表 并行处理实验结果

处理机数目	处理时间(s)	加速比
1	486	1
2	253	1.85
3	176	2.66
4	133	3.52

可以观察到，单处理机在处理大规模数据库时，由于资源上的限制，不得不频繁地将数据从内存卸载到外存上，再从外存上载入内存，降低了处理效率。即便给单处理机配置大容量的内存，但由于其内存的使用效率很低，因而极不经济。目前，国际上并行机技术发展迅速，巨型并行处理机的处理能力远远超过理论上的单处理机的极限能力。而且从资源上的使用效率而言，并行机也胜于单处理机。我们实验用的国产曙光一号有四个 CPU，共享64MB 内存，CPU 采用的是摩托罗拉公司的 RISC MC88100。

结束语 基于客户/服务器结构的数据开采对于大型数据库的处理确实具有其优越性，由于客户端与服务器端分工明确，能够各自发挥所长。客户端着重开发可视化的用户界面与规则生成分析部分；服务器端着重开发数据并行处理部分。而相对于整个从数据库中发现知识过程，数据的前处理与后处理也可以集中在客户端来处理。特别是在当今世界并行机的飞速发展，已经不再是单处理机所能企及，因而基于客户/服务器结构的数据开采技术也就具有非常重要的实际应用意义。

参考文献

[1] Fayyad U. M. et al., Knowledge Discovery and Data Mining: Towards a Unifying Framework. Proc. 2nd Intl. Conf. on Knowledge Discovery and Data Mining (KDD-96). Menlo Park, CA: AAAI Press. 1996

[2] Piatetsky-Shapiro G. et al., An Overview of Issues in Developing Industrial Data Mining and Knowledge Discovery Applications. Same to [1]

[3] Fayyad U. M. et al., From Data Mining to Knowledge Discovery: An Overview. in AKDDM. AAAI/MIT Press. 1996

[4] Holsheimer M. et al., Data Surveyor: Searching the Nuggets in Parallel. Same to [3]

（上接第22页）

——空 Herbrand 解释满足→p——于是第二条规则派生出 q。空解释的含义是计算程序 P 语义的初始点，正常地它应表示我们对 P 定义的正和负事实一无所知，于是这成为坚持模型论语义的第二点理由。

虽然人们针对逻辑程序的语义提出了多种的方法，但似乎总可以找到实例使已知方法不能抓住程序的"自然"的主观含义，于是人们一边定义了各种满足某种性质的程序类，一边无休止地继续"繁殖"语义方法，虽然人们对各种语义模型的关系做了不少的研究，对语义模型的统一框架的研究也有完整的结果，但主要的说明语义如何"结合"和"转换"是一个实际的问题，这种"结合体"也许会有更强的反映常识和主观含义的能力。

表5.1 静止语义统一了各种模型论语义[1]

语义名称	not C 的翻译	附加公理
良基	B→C	
静态	B→C	
稳定	B→C	(BCA)
稳定	→LC	
静止	B→C	
扩展静态	B→C	(S)
扩展静态	B→C	(S),(DBA)
扩展稳定	→LC	(S)
扩展静止	B→C	(S),(DBA)
折取静态	B→C	(S),(DBA)
折取稳定	→LC	(S),(PIA)

参考文献

[1] Van Gelder. A. et al. The well-founded semantics for general logic programs. J. ACM 38(3). 1991

[2] 王怀民，逻辑程序的语义问题(I)(II)，计算机科学，21(1)，21(2)，1994

[3] Dung. P. M., On the relation between stable and well founded semantics of logic programs. Theoretical Computer Science. 105(1). 1992

[4] Przymusinski. T. C., Three-valued non-monotonic formalisms and semantics of logic programs. Artificial intelligence 49. 1991

[5] Marek. W. and Subrahmanian. V. S., The relationship between stable, supported, default and autoepistemic semantics for general logic programs. Theoretical Computer Science. 103. 1992

[6] Gelfond. M., Lifschitz. V., Classical negation in logic programs and disjunctive databases. New generation computing. 1991

[7] Przymusinski. T., Extended stable semantics for normal and disjunctive programs. Proc. of 1990 intl. conf. on logic programming. 1990

[8] Przymusinski. T., Stationary semantics for disjunctive logic programs and deductive database. Proc. of the 1990 American conf. on logic progcamming. 1990

[9] 陈莽，孙吉贵，不相容逻辑程序的矛盾处理及其语义 AFSX(待发表)

[10] Przymusinski. T., Semantics of normal disjunctive logic programs--a unifying framework. ICLP' 94 workshop. 1994

(1)语义名称前可能增加了"折取"或"扩展"的字样，这对应于折取或扩展逻辑程序；否则，表示该语义是常规逻辑程序的语义。