维普资讯 http://www.cqvip.com

6-12 程序设计 逻辑程序 语义问题 ⑥

计算机科学 1994 Vol. 21№. 1

逻辑程序的语义问题*)(I) TP311

王怀民
(国防科技大学 长沙 410073)

A

摘要 说明性语义是逻辑程序研究的重要内容，也是其作为说明性程序的基本特征。近年来由于人们对带否定前提的一般逻辑程序设计的关注，以及逻辑程序与非单调推理的结合，逻辑程序的语义研究出现了许多新结果。本文以早期的 Clark 语义和最小模型语义为起点，介绍这些新发展的部分内容，主要包括 Clark 语义的 Fitting 3-值扩充、理想模型语义、稳定模型语义、良基模型语义以及它们之间的关系，并在此基研上进一步讨论了说明性语义在逻辑程序设计中的地位和作用。

1. 引言

七十年代初，Colmerauer 和 Kowalski 提出了逻辑程序设计的思想。Roussel 在 1972 年实现了第一个 Prolog 解释器，使这一思想得以实现。如今逻辑程序设计已经成为 AI 领域广泛使用的程序设计风范。从计算机科学的角度讲，逻辑程序设计的重要意义在于它首先确立了说明性（declaration）程序设计的概念。这一概念基于 Kowalski 提出的逻辑与控制的分离原则，即在理想的情况下，一个程序员只需关心程序的说明性含义，而程序执行的过程性则被自动处理。实现这一理想的基本前提是将逻辑程序与独立于过程性的说明性语义联系在一起，这种说明性语义不仅提供了关于逻辑程序含义的理解，而且可以作为评估各种计算机制的基本规范。逻辑程序语义的研究始于七十年代中后期，代表性的工作包括 Van Emdem 和 Kowalski 的最小模型语义（1976）、Clark 的程序完全化（1978）和 Reiter 的封闭世界假设（1978）。这方面的工作在八十年代初得到了进一步的发展，终于形成了第一部逻辑程序设计基础的专著[2]。近十多年来，随着人们对常识推理中各种非单调现象的深入研究，逻辑程序的语义作为推理机制“终审官（final arbiter）”的含义更为广泛或更具多样性。寻找适当的说明性语义反映逻辑程序在常识背景下的意义（intended meaning）成为十分重要而又非常困难的研究课题。正因为如此，逻辑程序的语义问题的研究在八十年代中后期又出现了许多新的进展。

本文根据［1］的内容，讨论逻辑程序设计的语义问题，特别介绍八十年代中后期的部分研究成果。我们首先定义逻辑程序设计语义的一般概念和思想。然后分别介绍 Clark 语义及其 Fitting 的 3-值扩充、最小模型语

*) 国家 863 计划与国家自然科学基金资助课题。收到日期：93-01-12。

2. 研究水平仅处于跟踪水平；
3. 研究缺少新思想；
4. 缺少必要的国际交流和让别人了解我们工作的机会；
5. 英语表达差；
6. 基础研究水平弱。

实际上，每届 IJCAI 在基础研究方面的文章占相当大的比例，这方面应当是我们国家的优势。因此如果在上述诸多方面有所改进，我们的工作被国际人工智能学术界接收还是很有希望的。

参考文献 IJCAI' 93 会议文集。

维普资讯 http://www.cqvip.com

义、理想模型语义、稳定模型语义和良基模型语义。最后，结合AI的研究说明我们对逻辑程序设计语义问题的基本观点。

2. 逻辑程序设计及其语义的基本概念

为使本文的描述连贯一致，我们首先给出以下一系列的基本概念，文中未加进一步说明的概念同[2]。

一阶语言$\mathscr{L}$的字母表$\mathscr{A}$是指（有限或可数无限的）常量、谓词和函数符号的集合*)，同时，字母表中含有可数无穷多个变量符号，连接词（∧，∨，¬，←）、量词（∀，∃）和通常的标点符号。$\mathscr{A}$上的项递归定义为常量、或变量或表达式$f(t_1, \cdots, t_n)$，其中，f是函数符号，t_i $(i=1, \cdots, n)$是项。$\mathscr{A}$上的一阶语义$\mathscr{L}$定义为所有合式的一阶公式的集合。一个原子A被称为一个正文字，它的否定¬A被称为负文字。一个表达式被称为基（ground）表达式，是指其中不含任何变量。$\mathscr{A}$上所有基项的集合称为$\mathscr{A}$上的Herbrand域，记为$\mathscr{B}$。$\mathscr{A}$上所有基原子的集合称为$\mathscr{A}$上的Herbrand基(base)，记为$\mathscr{H}$。设G是一个无量词约束的公式，G的一个基实例是指用基项代换G中所有变元所得到的基公式。对于$\mathscr{L}$中给定的公式G，它的全称闭公式是用全称量词"∀"约束G中所有非约束变量所得到的公式。除非另外说明，本文中涉及到的公式均假定为全称闭公式。

逻辑程序是指如下形式的全称闭子句的有限集合：

$$A \leftarrow L_1 \wedge \cdots \wedge L_m$$

其中$m \geqslant 0$，A是一个原子，L_i $(i=1, \cdots, m)$是文字，文字L_i被称为子句的前提，原子A被称为子句头（有时也称A中出现的谓词符号为子句头）。通常子句中的连接词"∧"用"，"代换。这样子句记为$A \leftarrow L_1, \cdots, L_m$。如果P是一个逻辑程序，除非另外说明，我们总是假设P的字母表$\mathscr{A}$是实际出现在P中的所有常量、谓词和函数符号，即$\mathscr{A}=\mathscr{A}_P$，$\mathscr{A}_P$是由P完全确定的字母表**)。相应地，我们规定$\mathscr{L}=\mathscr{L}_P$，$\mathscr{H}=\mathscr{H}_P$和$\mathscr{B}=\mathscr{B}_P$。

定义2.1 $\mathscr{L}$的一个（2-值）Herbrand解释是指Herbrand基$\mathscr{H}$的一个子集。

显然，任一（2-值）Herbrand解释I可以等价地视为一个2元序偶〈T，F〉，其中T，F是$\mathscr{H}$的子集，且$T \cap F=\phi$，$T \cup F=\mathscr{H}$，T是在I的解释下为真的原子（即属于I的原子）集合，F是在I的解释下为假的原子（即不属于I的原子）集合。称I为2-值的，因为$T \cup F=\mathscr{H}$，即I赋予每一个基原子的真值非真即假。

一个给定理论T的解释（或更精确地讲，它的一个模型）可以被看作一个"可能世界"，它表示了我们关于T的知识的可能状态。因为在某个特定的时刻，我们关于世界的知识可能不完全，因此需要描述这样的可能世界的能力，这里有一些事实非真非假，是不知道或未定义的。这就是说，我们需要3-值解释和模型描述我们知识的可能状态。3-值解释是2-值解释的直接推广，其中不再要求$\mathscr{H}=T \cup F$。

定义2.2 语言$\mathscr{L}$的3-值解释I是指一个序偶〈T，F〉，其中T，F是$\mathscr{H}$的互不相交的子集。集合T包含所有在I的解释下为真的基原子；集合F包含所有在I的解释下为假的基原子。$U=\mathscr{H}-(T \cup F)$中的原子的真值为不知道（或未定义）。

显然，一个3-值解释是2-值的当且仅当$\mathscr{H}=T \cup F$，或等价地，当且仅当$U=\phi$。

命题2.1 任一（3-值）解释$I=\langle T, F\rangle$可以等价地被视为一个如下定义的函数$I: \mathscr{H} \rightarrow V$，其中$V=\{0, 1, 1/2\}$：

$$I(A)=\begin{cases} 0 & \text{如果 } A \in F \\ 1/2 & \text{如果 } A \in U \\ 1 & \text{如果 } A \in T \end{cases}$$

除非特别说明，3-值解释将被视为上述

*)函数符号集合可以是空集，但这时总是假定常量符号集合和谓词符号集非空。

**)如果P中不含常量符号，则在$\mathscr{A}_P$中加入一个常量符号。

维普资讯 http://www.cqvip.com

定义的3-值函数。函数 $I: \mathscr{A} \to V$ 可以递归地扩充为闭公式集 $\mathscr{C}$ 上的真值计值函数。

定义 2.3 设 I 是一解释，与 I 相对应的真值计值函数 $\hat{I}: \mathscr{C} \to V$ 是如下递归定义的函数：

- 如果 A 是原子，则 $\hat{I}(A)=I(A)$；
- 如果 p 是公式，则 $\hat{I}(\neg p)=1-\hat{I}(p)$；
- 如果 p 和 q 是公式，则 $\hat{I}(p \wedge q)=\min(\hat{I}(p),\hat{I}(q))$；$\hat{I}(p \vee q)=\max(\hat{I}(p),\hat{I}(q))$；

$$\hat{I}(q \leftarrow p)=\begin{cases} 1 & \text{如果 } \hat{I}(q) \geqslant \hat{I}(p) \\ 0 & \text{其它} \end{cases}$$

- 对于任何含一非约束变量的公式 p(x)，$\hat{I}(\forall x p(x))=\min\{\hat{I}(p(a)) \mid a \in \mathscr{B}\}$；$\hat{I}(\exists x p(x))=\max\{\hat{I}(p(a)) \mid a \in \mathscr{B}\}$。特别地，$\max\{\}=0, \min\{\}=1$。

真值计值函数赋予每一个公式 p 一个数字：0、或 1/2、或 1。这些数字反映了公式 p 的真值层次：0（表示假）层次最低，1/2（表示不知道）层次次之，1（表示真）层次最高。

定义 2.4 $\mathscr{L}$ 上的一个理论 R 是 $\mathscr{L}$ 上的闭公式的（有限或无限）集合。一个解释 I 是理论 R 的（2-值或 3-值）模型，如果对于每一个 $p \in R$，$\hat{I}(p)=1$。

显然每一个程序 P 是一个理论。不难得到以下明显的结论。

命题 2.2 一个（Herbrand）解释 M 是一个程序 P 的模型当且仅当对每个程序子句的每一个基实例 $A \leftarrow L_1, \cdots, L_m$ 满足：

$\hat{M}(A) \geqslant \min=\{\hat{M}(L_i) \mid i \leqslant n\}$。

这就是说，M 是程序 P 的模型当且仅当子句头的真值层次不低于子句前提并的真值层次。

推论 2.1 一个（Herbrand）解释 M 是一个程序的模型当且仅当 M 是该程序的基实例的模型。所谓一个程序的基实例是指该程序的所有子句的基实例构成的理论。

以后在讨论程序（Herbrand）模型时，我们经常把程序与程序的基实例不加区分地看待。

设 p，q 是任意两个公式，连接词 $\leftrightarrow$，$\Leftarrow$，$\Leftrightarrow$ 定义为：$p \leftrightarrow q \equiv (p \leftarrow p) \wedge (q \leftarrow p)$；$p \Leftarrow q \equiv \neg q \vee p$；$p \Leftrightarrow q \equiv (p \Leftarrow q) \wedge (q \Leftarrow p)$。显然"$p \leftarrow q$"和"$p \Leftarrow q$"在 2-值模型上是一致的，但在 3-值情况下是不同的。例如，对于任意的 3-值解释 M，$\hat{M}(p \leftarrow p)=1$。而 $\hat{M}(p \Leftarrow p)=1/2$，如果 $\hat{M}(q)=1/2$。

定义 2.5 设 I 和 J 是两个解释，我们说 $I \leqslant J$，如果对于任一基原子 A，$I(A) \leqslant J(A)$（或等价地，$\hat{I}(A) \leqslant \hat{J}(A)$）。设 $\mathscr{I}$ 是解释的集合。称一个解释 $I \in \mathscr{I}$ 是 $\mathscr{I}$ 中极小的，如果不存在 $J \in \mathscr{I}$ 使得 $J \leqslant I$ 且 $J \neq I$。称一个解释 $I \in \mathscr{I}$ 是 $\mathscr{I}$ 中最小的，如果对于任意 $J \in \mathscr{I}$，$I \leqslant J$。称理论 R 的一个模型 M 是极小的（或相应地，最小的），如果它在 R 的所有模型中是极小的（或最小的）。

根据该定义，R 的极小（或最小）模型是其原子真值层次的极小（或最小）化。

命题 2.3 设 $I=\langle T; F\rangle$ 和 $I'=\langle T'; F'\rangle$ 是两个解释，则 $I \leqslant I'$ 当且仅当 $T \subseteq T'$ 且 $F \supseteq F'$。特殊情况下，对于 2-值解释，$I \leqslant I'$ 当且仅当 $I \subseteq I'$。

这意味着，模型的极小（或最小）化反映为真原子集合 T 的极小化和假原子集合的极大化。

为了与下面定义的 F-序相对，我们称这里定义的解释上的序关系为标准序。

通常人们认为，真值为"不知道"的命题传达的信息比真命题或假命题传达的信息少得多。Fitting 依此提出了基于信息层次的解释上的序关系。

定义 2.6 设 $I=\langle T; F\rangle$ 和 $I'=\langle T'; F'\rangle$ 是两个解释。$I \leqslant_F I'$ 当且仅当 $T \subseteq T'$ 且 $F \subseteq F'$。我们称 $\leqslant_F$ 为 F-序。设 $\mathscr{I}$ 是解释的集合，$I \in \mathscr{I}$。称 I 为 $\mathscr{I}$ 中的 F-极小的，如果不存在 $J \in \mathscr{I}$，$J \neq I$ 使得 $J \leqslant_F I$；称 I 是 $\mathscr{I}$ 中 F-最小的，如果对于任意 $J \in \mathscr{I}$，$I \leqslant_F J$。理论 R 的模型 M 称为 F-极小（或 F-最小）的，如果 M 是 R 的所有模型中 F-极小（或 F-最小）的。

显然，$I=\langle \phi; \phi\rangle$ 没有明确肯定或否定

维普资讯 http://www.cqvip.com

任何原子命题，因此没有传达任何信息，在信息层次上是最小的。实际上，$I=\langle\phi;\phi\rangle$ 是所有解释中F-最小的。由F-序的定义不难看出，基于信息层次的F-极小（或F-最小）化是真原子命题集合T和假原子命题集合F的极小化，而真值为"不知道"的原子集合U的极大化。

逻辑程序的说明性语义常常用作用在解释的序集上的某个自然操作的不动点定义。设$\leqslant$是一给定语言的解释集$\mathscr{I}$上的序关系，$\mathscr{J}$是$\mathscr{I}$的子集，Ψ：$\mathscr{I}\to\mathscr{I}$是$\mathscr{I}$上的操作。

定义 2.7 如果$I\leqslant J$蕴含$\Psi(I)\leqslant\Psi(J)$，那么操作Ψ称为单调的。一个解释$I\in\mathscr{I}$是Ψ的一个不动点，如果$\Psi(I)=I$。$\mathscr{J}$的最小上限$\Sigma\mathscr{J}$（相应的，最大下限$\Pi\mathscr{J}$）是指一个解释$I\in\mathscr{I}$使得对任何$J\in\mathscr{J}$，$J\leqslant I$，且对于任何具有该性质的J'，$I\leqslant J'$（相应地，对任何$J\in\mathscr{J}$，$I\leqslant J$，且对于任何具有该性质的J'，$J'\leqslant I$）。

单调操作Ψ的最小不动点常常可以从最小的解释I_0开始用操作Ψ迭代获得的（可能是超穷的）序列产生：

$\Psi\uparrow 0=I_0$；

$\Psi\uparrow\alpha+1=\Psi(\Psi\uparrow\alpha)$；

$\Psi\uparrow\lambda=\Sigma_{\alpha<\lambda}\Psi\uparrow\alpha$；$\lambda$是极限序数。

显然，一个迭代$\Psi\uparrow\alpha$是一个不动点当且仅当

$$\Psi\uparrow\alpha=\Psi\uparrow\alpha+1$$

下面各节我们仅考虑前面定义的两个序关系：标准序关系和F-序关系。为简便起见，我们将作用于标准序的解释集合上的操作记为Ψ或Θ，而作用于F-序的解释集合上的操作记为Φ或Ω。不难验证，$I_0=\langle\phi;\mathscr{H}\rangle$（相应地，$I_0=\langle\phi;\phi\rangle$）是所有解释中在标准序（相应地，在F-序）的意义下的最小元素。

对于$\mathscr{I}$的子集$\mathscr{J}$，$\Sigma\mathscr{J}$（对应地，$\Pi\mathscr{J}$）表示$\mathscr{J}$的在标准序$\leqslant$意义下的最小上界（对应地，最大下界）。类似的，$\Sigma_F\mathscr{J}$（对应地，$\Pi_F\mathscr{J}$）表示$\mathscr{J}$的在F-序$\leqslant_F$意义下的最小上界（对应的最大下界）。

设$\mathscr{J}=\{J_s|s\in S\}$，$J_s=\langle T_s;F_s\rangle$，显然

$$\Sigma\mathscr{J}=\langle\bigcup_{s\in S}T_s;\bigcap_{s\in S}F_s\rangle$$

$$\Pi\mathscr{J}=\langle\bigcap_{s\in S}T_s;\bigcup_{s\in S}F_s\rangle$$

$$\Sigma_F\mathscr{J}=\langle\bigcup_{s\in S}T_s;\bigcup_{s\in S}F_s\rangle$$

$$\Pi_F\mathscr{J}=\langle\bigcap_{s\in S}T_s;\bigcap_{s\in S}F_s\rangle$$

不难看出，$\Sigma\mathscr{J}$，$\Pi\mathscr{J}$，$\Pi_F\mathscr{J}$是良定的解释，而$\Sigma_F\mathscr{J}=\langle T;F\rangle$可能不是一个解释，因为T与F可能有交。但如果$\mathscr{J}$是$\mathscr{I}$的F-直接子集，则$\Sigma_F\mathscr{J}$总是良定的解释。所谓$\mathscr{J}$是F-直接的，如果$\forall J$，$J'\in\mathscr{J}$，存在$J''\in\mathscr{J}$使得$J\leqslant_F J''$且$J'\leqslant_F J''$。

并没有所谓逻辑程序合理语义的明确定义，更没有所谓最好的语义标准。但学者们一般普遍认为程序P的语义SEM(P)应该反映P在常识背景下的意义，而非仅仅是纯逻辑的内容。例如，我们将自然数定义为：1是自然数；如果n是自然数，则n+1是自然数。这时，直觉地讲，我们应该认为0和米老鼠不是自然数，即不在自然数的定义之列，尽管这些事实不是上述自然数定义的逻辑结论。如果我们基于程序的内在意义寻找其说明性语义，那么语义定义的多样性甚至局限性是不可避免的，这一点将反映在以下各节的讨论中。

逻辑程序的说明性语义可以分为两大类。一种称为证明论的，即把程序P的语义规定为另一个可以完整表达其意义的理论COMP(P)，例如COMP(P)为Clark的谓词完全化comp(P)。说一个公式p被其语义蕴含当且仅当p是COMP(P)的逻辑结论：$COMP(P)\vdash p$。

另一种说明性语义称为模型论的。程序P的语义由反映P的意图的模型集合MOD(P)（特别情况下，由一个模型M_P）决定。例如，MOD(P)是P的所有极小模型的集合或最小模型。说一个公式p被其语义蕴含当且仅当p被MOD(P)满足：$MOD(P)\models p$（特别地，$M_P\models p$）。

显然证明论途径是模型论途径的特例。但这两种途径反映了人们对程序的不同看

维普资讯 http://www.cqvip.com

法：前者表明我们倾向于把程序视为一种理论；而后者表明我们倾向于把程序视为一类模型。

3. Clark 的谓词完全化语义

逻辑程序的 Clark 谓词完全化语义是一种被广泛采用的基于证明论的语义规定。具体地讲，程序 P 的语义指称用 Clark 谓词完全化 P 后所得到的理论。基本过程是这样的。

首先将 P 中的子句 $q(t_1, \cdots, t_n) \leftarrow L_1, \cdots, L_m$ (1) 改写为 $q(Y_1, \cdots, Y_n) \leftarrow E$ (2)，其中 q 是谓词符号，t_i 是一阶项，L_i 是文字，$X_1, \cdots, X_k$ 是子句 (1) 中出现的变量，Y_i 是子句 (1) 中没有出现过的变量，$E = \exists X_1, \cdots, X_k (Y_1 = t_1 \wedge \cdots \wedge Y_n = t_n \wedge L_1 \wedge \cdots \wedge L_m)$。由此得到 P 中以 q 为头的被改写的子句集合（可能为空集，即 m=0）：

$$q(Y_1, \cdots, Y_n) \leftarrow E_1$$
$$\vdots$$
$$q(Y_1, \cdots, Y_n) \leftarrow E_m$$

最后得到以下全称量词约束的公式，称为 q 的完全化定义：

$$q(Y_1, \cdots, Y_n) \leftrightarrow E_1 \vee \cdots \vee E_m$$

这里，如果 P 中没有以 q 为头的子句，则 q 的完全化定义为 $\neg q(Y_1, \cdots, Y_n)$。

程序 P 的完全化就是其中出现的所有谓词符号的完全化定义加上以下等式理论（称为 Clark 等式理论）构成的一阶理论，记为 comp (P)。

CET1　$x = x$；

CET2　$x = y \Rightarrow y = x$；

CET3　$x = y \wedge y = z \Rightarrow x = z$；

CET4　对于任何函数符号 f，$x_1 = y_1 \wedge \cdots \wedge x_m = y_m \Rightarrow (f(x_1, \cdots, x_m) = f(y_1, \cdots, y_m))$；

CET5　对于任何谓词符号，$x_1 = y_1 \wedge \cdots \wedge x_m = y_m \Rightarrow (p(x_1, \cdots, x_m) \Rightarrow p(y_1, \cdots, y_m))$；

CET6　对于任何互不相同的函数符号 f、g，$f(x_1, \cdots, x_m) \neq g(y_1, \cdots, y_m)$；

CET7　对于任何函数符号 f，$f(x_1, \cdots, x_m) = f(y_1, \cdots, y_m) \Rightarrow x_1 = y_1 \wedge \cdots \wedge x_m = y_m$；

CET8　对于任何不同于 x 且含有 x 出现的一阶项 t [x]，$t[x] \neq x$。

其中 CET1～CET5 称为相等公理，CET6～CET8 称为单名公理。由于 CET1～CET8 是程序完全化所共有的，因此通常我们无需显式列出上述公理。

例如，我们用以下的程序 P_1 描述自然数：

natural-number (0)

natural-number (succ (x)) ←natural-number (x) .

P_1 的完全化 comp (P_1) 为：

natural-number (Y) ↔

$\exists x (Y = 0 \vee (Y = succ(x) \wedge natural\text{-}number(x)))$

Clark 的完全化方法在数学上是精美的，同时也反映了人们在日常活动中陈述问题的一种习惯："如果……"的意思实际指"当且仅当……"的意思。例如 P_1 的意思实际是指 comp(P_1)的意思。这时，虽然 ¬natural-number (mickey-mouse) 不是 P_1 的逻辑结论，但该公式为 P_1 的 Clark 语义蕴含，即：comp (P_1) ⊢ ¬natural-number (mickey-mouse)

Clark 完全化方法也存在一系列问题，其中之一就是 Clark 完全化可能导致不协调。例如，程序 $p \leftarrow \neg p$ 的完全化是 $p \leftrightarrow \neg p$，它是不协调的。另外，程序的完全化还会产生种种奇怪现象。例如设程序 P_2 为：$p \leftarrow \neg q$，$\neg p$。comp (P_2) 为：$p \leftrightarrow \neg q$，$\neg p$；$\neg q$。

显然 comp (P_2) 是不协调的。但是如果在 P_2 中加入一条重言式 $q \leftarrow q$，则完全化为：$p \leftrightarrow \neg q$，$\neg p$；$q \leftrightarrow q$。这是协调的，其模型为 $\{\{q\}, \{p\}\}$。而在 P_2 中如果加入另一条重言式 $p \leftarrow p$，则完全化为：

$p \leftrightarrow p \wedge (\neg q \wedge \neg p)$；$\neg q$。

其模型为 $\{\{p\}, \{q\}\}$。Clark 完全化过程导致的这种模型的偏移是不自然的。

Fitting (1985) 证明如果 comp (P) 采用

维普资讯 http://www.cqvip.com

3-值 Herbrand 模型，则可以消除 comp (P) 的不协调性以及相关的一些问题。

定理 3.1 (Fitting，1985) 任何程序 P 的 Clark 完全化至少有一个 3-值 Herbrand 模型。进一步，comp (P) 的所有 3-值 Herbrand 模型中存在 F-最小模型 M_P。

例如 comp (P_2) 的 F-最小模型为 {φ; {q}}。

由此 Fitting 提出了 Clark 完全化语义的 3-值扩充。

定义 3.1 (Fitting 语义) Clark 的谓词完全化语义的 Fitting3-值扩充由 M_P 或 comp (P) 的所有 3-值 Herbrand 模型集合 MOD (P) 决定。

Fitting 还给出了 comp (P) 的 3-值模型的一个很漂亮的不动点性质。

定义 3.2 (Fitting 操作) 设 P 是一个逻辑程序，$\mathscr{I}$ 是 comp (P) 的 3-值解释的集合，$I \in \mathscr{I}$，A 是基原子。Fitting 操作 $\Phi: \mathscr{I} \to \mathscr{I}$ 定义如下：

(1) $\Phi(I)(A)=1$，如果 P 中存在子句 $A \leftarrow L_1, \cdots, L_n$ 使得 $\hat{I}(L_i)=1$，$i=1, \cdots, n$；

(2) $\Phi(I)(A)=0$，如果对于 P 中所有 $A \leftarrow L_1, \cdots, L_n$ 均存在一个 L_i 使得 $\hat{I}(L_i)=0$；

(3) $\Phi(I)(A)=1/2$，其它。

定理 3.2 comp (P) 的一个解释 I 是 comp (P) 的模型当且仅当 I 是 Φ 的不动点。特别地，M_P 是 Φ 的 F-最小不动点。进一步，模型 M_P 可以由 Φ 迭代生成，即序列 $\{\Phi^{\uparrow\alpha}\}$ 是单调增的，且有一个不动点 $\Phi^{\uparrow\alpha}=M_P$。

Kunen (1987) 提出，被 Fitting 语义蕴含的公式集不是递归可枚举的。为此他对 Fitting 语义作了如下改进。

定义 3.3 (Kunen 语义) Clark 谓词完全化语义的 Kunen3-值扩充由 comp (P) 的所有 3-值模型（自然包括非 Herbrand 模型）的集合 MOD (P) 决定。

定理 3.3 (Kunen，1987) 一个闭公式 p 被 Kunen 3-值扩充的 Clark 谓词完全化语义蕴含当且仅当 p 被 Fitting 操作 Φ 的有限迭代 $\Phi^{\uparrow n}$ (n=0，1，2，…) 所满足。进一步，这种语义蕴含的公式集合是递归可枚举的。

不难看出 Fitting 语义比 Kunen 语义强，即被 Kunen 语义蕴含的闭公式也被 Fitting 语义蕴含。

尽管 Clark 的谓词完全化在许多方面反映了逻辑程序的自然语义，但对于许多程序，这种语义规定（包括 3-值扩充在内）还有种种弱点。这里仅举几个例子说明这个问题。

例 3.1 P'_1 是程序 P_1 加上下面的子句得到的程序。

natural-number (X) ←natural-number (X)。

直观地讲，P'_1 与 P_1 的意义没有差别。然而 comp (P'_1) 为：

natural-number (Y) ↔ (natural-number (Y) ∨ Y=0 ∨ ∃ x (Y=succ (x) ∧ natural-number (x))

显然，comp (P_1) 与 comp (P'_1) 在逻辑上是不等价的。例如，comp (P'_1) 不再蕴含 ¬natural-number (mickey-mouse)。

例 3.2 假设我们想描述一个图中的结点到给定结点 a 的可达关系。我们可能得到下面的程序 P_3：

edge (a，b)；edge (c，d)；edge (d，c)；reachable (a)；

reachable (x) ←reachable (y)，edge (y，x)。

直觉地讲，我们希望 P_3 的语义反映这样的事实：c 和 d 均不可到达 a。然而 comp (P_3) 均不蕴含 ¬reachable (c) 和 ¬reachable (d)。这里 comp (P_3) 中的谓词 reachable 的完全化定义为：

reachable (x) ↔ (x=a ∨ ∃ y (reachable (y) ∧ edge (y，x))。由于存在对称子句 edge (c，d) 和 edge (d，c)，comp (P_3) 无法导出 ¬reachable (c) 和 ¬reachable (d)。

例 3.3 设 P_4 是以下的子句组成的程序：

bird (tweety)；

fly (x) ←bird (x)，¬abnormal (x)；

abnormal (x) ←irregular (x)；

irregular (x) ←abnormal (x)。

最后两个子句是说“不规则的情况”与“反常的情况”是同义的。这两个子句并不能使我们相信 tweety

维普资讯 http://www.cqvip.com

计算机网络 CSCW

12-16

计算机科学 1994 Vol. 21 №. 1

计算机辅助协同工作*)

TP393

杜 兴 谢 立 孙钟秀
(南京大学计算机科学与技术系 南京 210008)

摘要 Computer Supported Cooperative Work (CSCW) is a rapidly growing field in computer science. This paper discusses its concepts, research fields and problems. Two typical CSCW examples are also outlined. Finally, the human-computer interaction aspect of CSCW is analyzed.

关键词 computer supported cooperative work, human-computer interaction

一、什么是计算机辅助协同工作?

随着高性能计算机及其网络的广泛应用，计算机系统正由传统的问题求解，执行一定的计算，朝着帮助用户进行交互并支持协同工作的方向发展。分工协作是一种完善化的标志[1]，传统的单用户使用计算机的模式，限制了可以解决问题的复杂度和求解的效率，目前正受到分散在不同计算机上由多个用户分工协作以高效求解复杂问题（包括科学计算、日常办公自动化）的挑战。为支持这种新的工作方式，计算机辅助协同工作应运而生，受到越来越多的重视，并成为目前计算机领域内发展最快的研究方向之一。

计算机辅助协同工作(Computer Supported Cooperative Work，简称 CSCW）是这样的计算机系统，它支持一组用户协同执行一项共同的任务，并为其提供访问共享环境的接口。CSCW 又常称为组件（groupware)[2]。共同的任务及共享环境是 CSCW 有别于其他（如多用户分时系统）的关键点。

显然 CSCW 根据计算机辅助的程度，可以是简单到 Unix 下的 talk 及电子邮件，也可以是下几节将介绍的包括提供声文图在内的复杂交互系统。

CSCW 的应用领域十分广阔。举一公司管理为例，假设该公司有两位秘书，一位分管公司日常事务，另一位负责公司的业务活动。两秘书的工作均涉及公司经理的日常时间安排。事务秘书和业务秘书必须通过协同工作，才能完成管好经理日程这个共同任务。一个支持这样工作的 CSCW 系统可以方便地让两秘书交换各自在其计算机上用有关软件产生的经理日程安排信息；在计算机及网络的帮助下进行类似面对面的交互协商，最后合理地安排好经理的日程。

二、CSCW 发展概况

*) 863 部分资助项目。

是反常情况。因此，基于封闭世界假设的观点，我们有理由相信“tweety 会飞”的断言。但是 comp (P_4) 为：

fly (y) ↔ (bird (y) ∧ ¬abnormal (y))；

abnormal (y) ↔irregular (y) .

由此不能推出 fly (tweety)。另一方面，去掉最后两个子句（或其中的一条），Clark 语义就可以导出希望的结论。

下面几节讨论模型论途径的说明语义，可以克服 Clark 语义的上述缺点。

（未完待续）