知识程序设计语言的研究

胡运发　　陈火旺

（国防科技大学计算机系）

摘　要

本文提出了知识程序设计语言概念，即有实际效率能进行知识推理的知识表示语言，它有统一的逻辑基础又有当前若干有代表性人工智能语言的本质特征。本文还给出一种知识程序设计语言的有关研究内容和实际取得的进展。

人工智能的历史，也是人工智能程序设计语言发展的历史。由于知识在AI中具有特别的重要性，研究知识程序设计语言具有特别紧迫的意义。大家知道，AI的中心问题就是知识表示、知识利用和知识的获取问题，知识的表示有很多种，但有没有某些基本的表示方法呢？知识的利用，即知识的推理与知识的表示十分密切。知识的获取与知识的表示和推理亦有特殊的联系。一般说来，知识表示问题是语言问题，但不一定是知识程序设计语言。只有能进行知识推理并具有实际执行效率的知识表示语言才称为知识程序设计语言。从这个意义上说，在人工智能历史中有代表性知识程序设计语言有三种，(1)以LISP代表的函数语言。(2)以PROLOG为代表的逻辑语言。(3)以Smalltalk为代表的面向对象语言。每一种语言代表人们对客观世界的一种看法和思考方式。函数语言的基本观点是客观世界的描述可以映射为函数。函数是可以构造的，简单的函数可以复合成复杂的函数。函数语言的逻辑基础是等词逻辑。函数语言的操作语义是归约(Reduction)。逻辑语言的看法是客观世界的描述可以表达为对象之间所有可能的关系，和函数一样，关系也可以复合。逻辑语言的逻辑基础是一阶谓词逻辑。逻辑语言的操作语义是归结(Resolution)。对象语言看法则不同，它认为世界是由对象组成的，若干本质相同的对象可以构成类。若干本质相同的类可以构成超类。一个对象不能直接访问或改变另一对象的内部状态或操作。但是对象之间可以通过通讯而发生联系。一段时间内，人们创造了面向对象的语言，但并未弄清它的逻辑基础。现在人们倾向认为反射逻辑可以作为对象语言的逻辑基础。这样，高阶演绎就是对象语言的操作语义了。既然三种语言各从某个侧面描述了客观世界，那么我们能否从中选择一种语言，它更优于其他语言，从而成为知识程序设计的主导？我们研究结果表明，答案是倾向于否定的。我们还要问，能否产生一种新的知识表示语言，它具有上述三种语言的本质特征，又具有统一的逻辑基础和适当的执行效率？我们的研究表明，答案是肯定的。

为的科学家对探索在他们看来是广阔的、未揭开的、荒芜的领域，有着强烈的欲望。

我个人的看法是，神经网络的研究将对神经心理学作出重大的贡献，为在这一层次上的模拟提供有意义的概念。神经网络的研究最终将为象视觉这样的传感输入，建立某种快速的预符号处理装置。同时，神经网络的研究也会对涉及问题求解逻辑的中心认知功能模型产生一定影响。但我认为，它对从事专家系统研究的人们不会带来多少现实的帮助。

〔张少平、王怀民译自《Expert Systems》April 1989, Vol.6, №2, 陈火旺校〕

一、现有人工智能语言的本质特征

现在，有代表性的人工智能语言主要有三种：LISP，PROLOG，Smalltalk，它们的本质特征和优缺点如下：

1． 以LISP为代表的函数语言的优点。函数语言的理论基础是等词逻辑，本质特征是具有语义上的等价能力。其他重要特征还有：（1）采用和数学标记相似的嵌套式的函数标记方法，有时更加简便。（2）表示方法中包含了更多的控制信息，由此可导致更有效的实现技术，如确定性重写（Rewriting）。（3）函数语言历史较为悠久，已拥有一些成熟的实现技术，如编译和归约机。（4）函数语言中某些概念，如高阶函数，Lazy计算，流类型和多类等，在标准的逻辑框架下难于表达。

2． 以PROLOG为代表的逻辑语言的优点。逻辑语言的理论基础是一阶谓词逻辑，存在一个完备的SLD归结过程，它本质上是一种不确定的重写过程。语言系统本身就是一问题求解系统，比函数语言具有更强的表达能力。其他重要特征还有：（1）使用逻辑表达方式，和人们习惯方式接近。（2）标记上采用关系平坦化的"AND"的复合，增加了控制的不确定性。（3）基于搜索的计算机制，即允许don't know不确定性。（4）引入逻辑变量和一致化驱动方式。

函数语言与逻辑语言主要差异是变量的性质以及确定性。限制变量的性质和计算的不确定性，逻辑语言可蜕化为函数语言。

3． 以Smalltalk语言代表的面向对象语言的本质特征。面向对象语言的理论基础是反射逻辑（Reflection Logic）。操作语义是高阶的演绎过程。它本质上提供状态的局部性概念，既不象强制式语言那样状态是全局的，如破坏性赋值，引起程序语义的混乱，也不象函数和逻辑语言那样完全取消状态的概念，导致框架问题的低效。其他重要的特征还有：（1）对象的抽象统一了传统上两种抽象，即功能的抽象和数据的抽象。（2）计算的分布性和并行性。对象的执行是由消息启动的，消息传送的独立性，导致计算的并行性。（3）共享性。类和继承均是共享机制，可减少表达上的冗余性。

从上述分析可以看出，LISP，PROLOG，Smalltalk之所以受到人们的重视，是因为他们从不同的侧面表现了人们的思维方式的探索。他们各自包含一些符合人们的思维方式的优美特性，又各自存在一些不足。知识程序设计语言应积极继承这些优良的特征。

二、一种知识程序设计语言Unifol

我们提出的知识程序设计语言是在逻辑语义基础上统一函数，面向对象和逻辑语言三种范型（Paragim）的语言，执行的效率和任何其中一种范型的效率相当，我们把这种语言称为Unifol。我们的研究工作从四个方面进行：第一方面，理论的研究。以Horn逻辑为出发点，然后扩充等词逻辑，再扩充反射逻辑。第二方面语言系统实现技术的研究。第三方面是知识程序设计语言与现存软件资源的关系与合成的研究。第四方面是知识程序设计与新一代计算关系的研究。限于篇幅，本文重点叙述语言系统实现技术的研究。

1． 实现的策略

实现多范型知识程序设计语言有四种可能策略：（1）环境合成式。将现存的多种语言通过共同的环境合成成一个整体。例如RF-MAPLE合成了R-MAPLE和F-MAPLE，POPLOG合成了LISP，Pop-11，PROLOG以及Pascal等，此方法见效快，软件资源容易继承。缺点是语言成分复杂，语义不协调，不易形成好的计算模型。（2）语言扩展式在现存的语言中增加新的成分。例如LOGLISP在LISP的基础上扩充了PROLOG的功能。C+

＋在C的基础上，增加了面向对象的功能。优点是新的成份共享了主语言的数据结构，支撑环境，设计周期短，见效快，缺点是语言成分复杂。(3)理论合成式。在现存的语言中，选择有代表的几种语言，详细研究各语言的本质特征及其在理论上的反映，然后产生一种新的理论。并在此基础上重新定义语法、语义，从而产生一种新的语言，例如Qute等。这样做的好处是理论统一，语义清晰，语言成分简单，有助于产生新的计算模型，开发实用系统的把握较大。缺点是开发的工作量较大。(4)理论创新式。研究现存多种语言的不足，创造一种新的理论以弥补现存语言的不足。这种做法，步子很大，但见效很慢。我们选择第三种方式。

2. Unifol的语法

(1)Horn子句的定义(见GKD-PROLOG使用手册).

(2)等式的定义

(a) f, —, g.绝对等式.f=g.

(b) f, —p, g.条件等式, if p then f=q.

(3)类的定义

<Class>::=Class<Classname>
[Super<Classname>{,<Classname>};]
[Part<Class>{, <Classname>};]
[Class<Varname>{, <Varname>};]
[Instanceemethod<method>.{<method>.}]
[Local<method>. {<method>}]] End.

<Instance>::=Instance<instancename> OF <Classname>{ <instancevar value declaration>.}
End.

<method>::=<Horn Clause> | <Equality>

<Horn Clause>::=<Head>|<Head>:-<Body>

<Equality>:: = <Head> :-: <Body> |
<Head>:-C_1, …, C_2:<Body>

<Instancevar value declaration>::=<Instancevar name>:-:<value>

<Class var value declartion>::=<Classvar name>:-:<value>

<Message>: :=Send (receiver, massage)

<message>::=<Head>

[X]表示0个或1个

{X}表示0个或多个

<X>表示语法单位

3. Unifol的解释实现算法

我们已经用pascal 语言逐步实现了三个系统（1）GKD-PROLOG 逻辑系统。(2) GKD-FUNLOG 函数逻辑系统。(3)Unifol 函数逻辑和面向对象合成系统。规模是逐步扩大的。Unifol 解释实现的算法如果用逻辑语言来描述则十分简明：

Solve ((G_1, G_2)) : -Solve(G_1), Solve (G_2).

Solve ((G_1, G_2)) : -Solve (G_1); Solve (G_2).

Solve (Send(n, G)) , -Search (n, m), Slove (m, G).

Solve (G), -Clause (G, B), Solve(B).

Solve (m, G), -Clause0 (method, m, G, B), Solve(B).

Solve (m, G), -Super (m, m0), Solve (m0, G).

第1，2，4子句表达了逻辑语言的推理算法。当Clause使用的一致化是narowing或语义一致化，则第1，2，4子句表达了函数逻辑语义的推理算法。第3子句专为面向对象而设置的。其中，Search (n, m) 定义为查找实例n的所属类m。Send (n, G) 定义为给实例n发送消息G。Super (m, m0) 查找类m的超类m0。Clause，Clause 为索引原语，基本意思是从库中找到与目标相匹配的子句。上述算法仅供理解用，实际实现远比这复杂。

三、Unifol支撑环境的研究

知识程序设计语言不同于常规语言之处处于：（1）语言本身是一问题求解系统。（2）语言系统和支撑环境构成统一体。Unifol的环境包括：元级控制器和部分计算器；句法制导编辑器为中心的窗口系统，增量式动态编译系统，基于归纳的调试系统，基于演绎的说明系统。下面分别予以说明。

1. 元级控制器

作为知识程序设计语言只能提供若干系统和标准的控制策略。当用户需要某种比较复杂的控制策略时，可求助于元级控制器。元级控制的一般思想是。

```
Solve(G), -select_goal(G, SubG, H, Rest),
          Select_clause (SubG, H, B),
          Unify (SubG, H),
          Solve ([B|Rest]).
Solve ([ ]).
```

其中 Select__goal (__, __, __) 是目标的选择规则；Select__clause (__, __, __) 是子句的选择规则；unify (__, __) 是一致化算法，它们均由用户定义。但是元级控制的一个致命弱点是执行效率较低，为了提高执行效率，可使用部分计算技术。

2. 部分计算技术的研究

部分计算的思想早在1952年 Kleen 的 "Introduction to Metamathematics 就已经提出，对于函数F (X1,…, Xn)，如果X1=a1，…Xk=ak则令f' (X_{k+1}, …X_n)，f' 称为f的部分计算。

部分计算器对于逻辑程序来说更加适用，因为逻辑程序允许对不完全信息进行处理。部分计算器的主体部分是

```
Solve (_inital, _goallist, _r, _out),
Solve (_inital, [], _r, _r).
Solve (_initial, [_t|_q], _r, _out), -non-
        evaluable (_t), ! ,
        append_list (_r, [_t], _m),
        solve (_initial, _q, _m, _out).
solve (_initial, [_t|_q], _r, _out), -
        evaluable_Primitive(_t),!
        call(_t), solve (_inital, _q, _r,
        _out).
solve (_initial, [_t|_q], _r, _out), -
        meta (_t, _subguals),
        append_list (_subgoals, _q, new
              _goals).
         solve (_initial, new_goals, _r,
              _out).
```

上述定义说明，（1）如果没有子目标存在，则部分计算结果就是最终结果。（2）如果子目标表中第一个文字是不能部分计算的，则它和部分计算结果r拼接起来，然后求解剩余的子目标。（3）如果选择的文字是可以部分计算的，则计算它，然后求解其余子目标。（4）如果在程序中存在一个子句和当前文字相匹配，则将它的体拼接到子目标表上，然后求解它。其中 meta (__t, __subgoals) 为子句的元级表示。

3. 动态增量式编译技术的研究

当然，以逻辑为基础，则相应的操作语义是解释性的。但是解释性在效率上不高。为了提高效率，研究相应的编译器是必要的。编译器可以看成解释器的加速装置。基础语言的编译器有许多不同于常规语言编译器的地方：

(1)逻辑程序要能回答所有蕴含的结论，在编译的时刻，也许还不知用户提什么样的问题，所以编译系统要拥有交互能力，要求用户提问，并动态编译成运行代码，给出运行结果。

（2）基础语言的特点是增量式，程序可以不断增加，功能也随之增强，增量式允许现场编译，现场装入。

（3）由于call，assert等元级功能的存在，要求编译程序实行动态编译，因此编译和运行程序紧密相依，原则上不允许编译程序在用户程序运行时，退出现场。

编译器的加速作用是十分明显的，实践

表明编译器可以比解释器快一个数量级。

4. 句法制导为中心的窗口技术

在句法分析或实际运行中，如果一旦发现句法错误或类型错，应立即终止检查或运行并转入现场窗口编辑器中，通过交互且编辑结束后，或保存，或再检查，或再运行。

关键技术是

（1）句法分析器或解释器或编译器必须具有一个统一的报错器。

（2）句法分析器、解释器，编译执行器和窗口系统相互独立，且可相互调用。

（3）要有良好的多个程序的统一界面设计。

5. 基于演绎的解释说明技术

由于基础语言是基于推理的，为了沟通系统和人的相互理解，解释说明推理过程的能力是十分重要的。过去存在于专家系统界面中的解释功能，其共性部分作为基础语言的解释基础。主要功能可概括为

- How为什么系统推出某结论？
- Why 为什么系统向用户提出如此问题？
- Whynot为什么系统否定了某结论？
- Assume假定改变某个数据，会出现什么结果？

具体实现算法请参考"人工智能系统原理与设计"。

6. 基于归纳的智能调试器

（1）在软件生存周期中，代价最大，费时最长的阶段就是调试，面向知识的程序设计，调试更加困难，因此智能调试器更加必要。

（2）能够发现错误

a. 检测循环。

b. 发现假解，即系统不正确。

c. 发现系统不完备，即丢掉真解。

（3）能够根据实例，从归纳中产生新的知识，从而导致校正错误。

a. 从实例中运行概念获取。

b. 根据实例和模型论语义信息运行模型推理。

c. 从统计中，运行概率推理，校正可信度。

兼有（2），（3）两项功能，称为智能调试器。

四、知识程序设计语言系统研究进展

按照上述研究内容：我们从事知识程序设计语言的研究大约经历了八年左右的时间，主要进展如下；

1. 在1985年，首先推出第一个阶段性成果逻辑型语言的解释系统GKD-PROLOG/VAX780，它是一种新型的人工智能系统语言，由于它的说明性能力，强有力的功能和较高的运行效率，受到参与鉴定的专家的一致好评，后来陆续在十多个单位得到使用，效果良好。该系统在1986年获部委级二等奖。以它为工具研究实现的'或'树林并行推理机模型系统获部委级一等奖。

2. 1986年，以GKD-PROLOG/VAX780和RDB数据库为基础，进一步推出具有知识演绎能力和知识管理能力的知识库系统KBS-1，获部委级二等奖。

3. 1989，在带等词的Horn 逻辑基础上，以我们首先提出的目标类型驱动法为规范，实现了函数语言和逻辑语言的合成系统GKD-Funlog。该系统的突出特点是函数语言和逻辑语言同时达到同样的运行效率，因而合成系统真正具有实用价值。该系统经鉴定达到国内领先水平。

4. 1989年，完成了逻辑语言和其支撑环境的合成，解释，编译，窗口合而为一。编译和执行效率平均提高15倍，并首次实现了编译和解释的完全相容。由于在该系统提出一系列先进技术，例如部分计算技术，启发式抽象机模型技术，半代码库的动态连结技术，保证了系统的高效率和实用性。鉴定后，立即受到用户的欢迎，并获得部委级二等奖。

5. 1990年，在以前工作成果的基础上，实现了知识程序设计语言核心部分，进一步丰富其使用环境，可投入实用。

现在再叙述一下Unifol语言系统的性能特点：

正如上述，Unifol系统是在逻辑语义基础上统一了逻辑型，函数型和面向对象型程序设计的风格和功能。

(1) 功能强。能方便地构造具有推理和函数计算的对象，对象中的方法和状态也都具有说明性。本语言系统特别支持软件工程中大型软件的设计，尤其是提供了信息隐蔽，抽象，模块化，分布式计算的良好手段。

(2) 使用方便。拥有Unifol，就象同时拥有逻辑、函数和面向对象三种语言系统。而且三者可以任意混合使用。系统还另外提供了一组功能强的原语和常用的类给用户，大大方便和简化用户的程序设计。

(3) 运行效率高。Unifol一个重要的特点是运行效率很高，函数，逻辑和面向对象具有同一数量级的运行效率。这是任何企图在某一种语言基础上建立的研究多种风格的合成系统方法所无法做到的。经测定本系统甚至比 smalltalk 系统运行的效率还要高5到10倍。

(4) 系统的扩充性能好。系统与需要的常规软件均有接口，因此可综合利用现有软件，例如利用网络系统和数据库系统，可以方便地建造面向对象的数据库系统。利用图形库软件，可以构造演绎的图形系统，把常规软件向智能化方向推进了一步。

(5) 推理机制统一。Unifol系统是以逻辑为基础，统一的索引机制和推理机制，孕育着有效的计算模型和结构（将另文叙述）。

致谢：本课题的一切成果均归于863-306-105课题组全体同志的共同努力。作者在此对他们表示谢意。

参考文献

1. K.Nygarrd,Basic Concepts in Object-Oriented Programming,SIGPLAN Notices, 21(10), 133-142
2. O.Dahl, B.Myhrhaug, and K.Nygarrd, The Simula67 Common Base Language, Technical Report,Norwegian Computing Center, Oslo, 1970, Publication s-22.
3. J. W. Lloyd, Foundation of Logic Programming, Springer-Verlay Berlin, 1984.
4. J. Goguen and J. Meseguer, Universal Realization, Abstract Modules, In Proc.9th International Conference on Automata,Languages and Programming, PP265-281, Spring-Verlag, 1982 LNCS 140.
5. J. Goguen and J. Meseguer, Unifying Functional, Object-Oriented and Relational Programming with Logical Semantics, In Research Direction in Object-Oriented Programming, PP417-477,MIT Press.1987
6. 邓铁清，金芝，胡运发，"Investigation and Implementation of the Partial Evaluation of PROLOG Programs"，The Fourth Japanese-Sino Sapporo International Conference on Computer Applications, 1990.10
7. 胡运发，"逻辑语言中启发式控制"，软件学报，1990，2
8. 胡运发,高洪奎,"智能机基础语言设计的研究"，89年全国人工智能及应用学术会议论文集，1989.11
9. 邓铁清，胡运发，"PROLOG 程序部分计算的研究与实现"，软件学报，1990.2
10. 邓铁清，胡运发，"GKD-PROLOG 编译系统的计算器"，89年全国人工智能及应用学术会议论文集，1989.11
11. 邓铁清，胡运发，高洪奎，"PROLOG 中非逻辑成分的部分计算"，中国第四届逻辑程序设计与人工智能语言会议论文集,1989.12
13. 陈宏盛，逻辑语言和函数语言的研究与合成，国防科技大学研究生硕士论文，1989。
14. 胡运发，高洪奎，胡子昂，卢肇川等，"逻辑语言及其环境合成系统WIK-PROLOG"，中国第四届逻辑程序设计与AI语言会议论文集，1989.12。
15. 吕国堂，胡运发，"在逻辑语言基础上统一逻辑函数型和面向对象的程序设计"，中国第四届逻辑程序设计与AI语言会议论文集，1989.12。
16. 胡运发，高洪奎，"人工智能系统原理与设计"，国防科技大学出版社，1988。