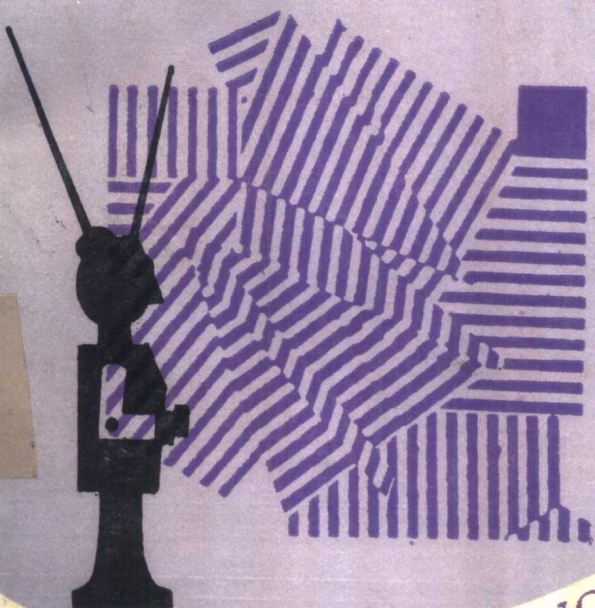
计算机科学丛书

人工智能与LISP程序设计

管 纪 文 编著



JISUANJI KEXUE CONGSHU

贵州人民出版社

计算机科学丛书

人工智能与LISP程序设计

管纪文 编著

中国科学院科学基金资助的课题

贵州人民出版社

封面设计　石俊生
技术设计　荀新馨

人工智能与LISP程序设计
管纪文　编著
贵州人民出版社出版发行
（贵阳市延安中路5号）
贵州新华印刷厂印刷　贵州省新华书店经销
787×1092毫米　32开本　10.25印张　220千字
1987年2月第1版　1987年2月第1次印刷
印数1——2,000册
书号13115·72　定价2.15元

《计算机科学丛书》

编委会

编者的话

为了加速发展我国的计算机科学技术，在贵州人民出版社的大力支持和协助下，中国科学院软件研究所、复旦大学、吉林大学、浙江大学、武汉大学、南京大学、上海交通大学、山东大学、哈尔滨科技大学、西北电讯工程学院、贵州大学等有关方面的同志经过多次磋商，组成了《计算机科学丛书》编委会。

这套《丛书》的作者，大多是长期从事计算机科学技术方面的科研、教学工作并在近几年内出国考察或学习过的中年同志。他们既有丰富的实践经验，又对国内外计算机科学的进展有比较清楚的了解。《丛书》将向读者介绍现代计算机科学方面的进展及其理论、方法和应用知识。每本书的内容也都自成体系，独立成册，集中介绍一个专题。为了便于学习，部分书后还列有少量习题，可供读者练习。在写作上，《丛书》力求做到篇幅短，内容新，重点突出，适于读者自学，并使读者在较短时间内对每一个专题的动向和发展趋势得到较为完整的了解。

这套《丛书》可作大专院校有关学科的教材和参考书。

《丛书》以大学生、研究生为主要读者对象，也可供大专院校教师、科研工作者和计算机工作者参考。

我们相信，这套《丛书》的出版，将对广大读者了解和掌握计算机科学知识有所裨益。

《计算机科学丛书》
编 委 会
一九八六年一月

前　言

远在十七世纪，人们就提出过这样的问题：能不能用机器来代替人的某些脑力劳动？直到二十世纪，电子计算机的诞生，才使这一问题有了正面回答的实际可能，而人工智能这一学科也才得以建立。

作为计算机科学的一个重要分支，人工智能的研究目的，是提高计算机应用的灵巧性，也就是说，使计算机具有更多的智能因素。因此，人工智能是计算机应用研究的最前沿，人工智能的研究应以在计算机上实现各种应用系统为目标。人工智能也有它自身的理论，但它的理论研究，是紧密围绕着这一目标而进行的。人工智能的一切成果都要经受实验的考验，只有用机器实现了的智能，才是人工智能。所以人工智能是一门实验性学科。

人工智能的研究领域极其广泛，它几乎涉及到人类创造的所有重要学科，诸如数学、物理学、心理学、生理学、医学、语言学、逻辑学、经济、法律、哲学以及计算机科学等等。因此，人工智能又是一门综合性的边缘学科。

人工智能的研究课题虽然很多，但从根本上来说，无非是如何处理知识的问题。即知识的获取、知识的表示和知识的利用这样三个基本问题。本书就这些基本问题作了一定的介绍，内容涉及自然语言理解、二值图象处理、定理证明、专家系统、生成系统、问题求解系统、博弈和智能机器人等

等．本书第二章介绍二值图象处理；第三章介绍定理证明；然后在第四章介绍积木世界，讨论智能机器人行动问题；第五章介绍游戏世界，讨论博弈方面的问题；第六章介绍符号模式匹配，涉及问题求解系统和自然语言理解；第七章介绍若干系统的LISP实现，讨论了增广转换网络的编译和生成系统；第八章介绍专家系统；第九章介绍数据库及其守护神．

另外，LISP语言是一种著名的程序设计语言，特别适用于人工智能以及某些非数值问题．这是一种函数型语言，在程序设计语言中占有特殊地位．因此，本书也简明地介绍了LISP语言及其程序设计技术．

本书可作高等院校有关专业开设人工智能或LISP程序设计课程的教材或参考书，也可供有关专业的教师、学生与科研人员自学或参考．

目　录

第一章　LISP程序设计

对于人工智能，计算机程序设计是重要的。其理由如下。

1）程序能检验，而人工智能，作为“实验性”科学，其理论正需要通过实验来加以验证。

2）程序设计是表示知识的新的强有力的手段。程序语言中的词汇和语言，已经丰富和加强了人工智能方面的词汇和表达的能力。

3）程序的实现使理论严格化。

4）程序本身能说明本任务所要求的信息处理能力，程序本身所需的空间和时间等数据，多少说明了本任务的内在复杂性。

5）程序能明快又准确地说明理论。程序本身也就是理论，因为它本身也反映着客观的事物及其规律性。程序设计对于人工智能的作用，颇象数学对于自然科学的作用。

那么，人工智能的程序设计，宜采用什么语言呢？计算机语言种类繁多，但对于人工智能而言，最适宜的语言莫过于 LISP 了。因为，LISP 已广泛采用于人工智能领域，而且 LISP 很容易掌握，其语法特别简单。还有一些语言，诸如 SAIL 和 POP-2 等，形式类似于 LISP，所以在掌握了 LISP 之后，很容易学会其他语言。

总之，就作用上来讲，我们把 LISP 当作“人工智能的

数学”，用 LISP 来编码智能过程。有人说，没有 LISP 语言的人工智能是诗人的物理学。意思是说这种物理学是没有数学描述的，是不严谨的。

以下就来介绍基本的 LISP 程序设计。§1 介绍 LISP 语句符号表达式，以及组成 LISP 程序的基本函数。§2 介绍 LISP 程序设计。

LISP 有许多版本，诸如 MACLISP，INTERLISP，LISP 机 LISP，UCILISP，UNIVAC 1108 LISP等等。这里介绍的 LISP，是 MACLISP，而且是对诸版本都通用的函数。

§1 符号处理

开始，让我们来看几个简单的 LISP 程序，第一个是处理算术的：

```
(PLUS 3.14 2.71)      人问：3.14 + 2.71 = ?
5.85                  机答：5.85
```

处理符号的与此相类似。设想要让机器记住DICK，JANE，SALLY 是朋友，则可打入：

```
(SETQ FRIENDS '(DICK JANE      人：建立朋友名单以后，要询问朋友名单时即可打入：
SALLY))

FRIENDS                        人问：朋友名单？
```

则机器就会打出：

```
(DICK JANE SALLY)                    机答：列出名单
```

同样，也可以记住敌人的名单

```
(SETQ ENEMIES '(TROLL GRINCH         人：建立
              GHOST))                敌人名单
ENEMIES                              人问：敌人名单？
  (TROLL GRINCH GHOST)               机答：列出名单
```

在敌友情况转化时，还可以修改敌友名单，通过打入：

```
(SETQ ENEMIES (DELETE                人：从ENEMIES
            'GHOST ENEMIES))         删去GHOST
(SETQ FRIENDS (CONS'GHOST            把 GHOST 加入
            FRIENDS))                FRIENDS。
```

于是，此后询问敌友名单时，已更新成：

```
ENEMIES                              人问：敌人名
                                     单？
  (TROLL GRINCH)                     机答：列出敌人
                                     名单
FRIENDS                              人问：朋友名
                                     单？
  (GHOST DICK JANE SALLY)            机答：列出朋友
                                     名单
```

为了更新敌友名单，把敌人转化成朋友的工作，也可以专门编一个程序 NEWFRIEND 来实现，这程序通过函数 DELETE(删去) 和 CONS(加入) 定义如下：

```
(DEFINE (NEWFRIEND                   定义函数NEWFRIEND,
          NAME)                      变元是NAME
```

```
(SETQ ENEMIES (DELETE NAME
 ENEMIES))          从ENEMIES删去NAME
(SETQ FRIENDS (CONS NAME
 FRIENDS))          把NAME加入FRIENDS
```

这样，利用这个新定义的 NEWFRIEND，把 GHOST从敌人转化到朋友名单来就可以简单地打入一行：

```
(NEWFRIEND 'GHOST)          人：把GHOST从
                                ENEMIES转
                                化到FRIENDS
```

以上是两个 LISP 的例子，一个处理算术，一个处理符号。当然，处理算术也是在处理符号。通过这些例子我们看到，LISP 中出现的语句，多数都带有圆括号，这样的带圆括号的符号串称为**表**；也有不带圆括号的，这样的符号串称为**原子**；原子和表都称为**符号表达式**。原来，LISP 的程序和数据都是符号表达式。举例来说，(PLUS 3.14 2.71)是表，它有三个元素 PLUS，3.14，2.71，它们都是原子。PLUS是 LISP 中的基本函数，要写在最前头，然后跟着写变元3.14 2.71，这是大家熟知的**前缀记法**。在 LISP 中，函数一律采用前缀记法。这种记法，与数学中的函数记法一致：如函数 $f(x,y,z)$相当于(FXYZ)，函数符号F在前头，但是，却与算术运算符的记法不同，因为，如3.14＋2.71中算符"＋"在变元3.14与2.71之间而不是之前，这是中缀记法。

再看一些 LISP 算术函数的例子。

```
(DIFFERENCE 3.14 2.71)      人：3.14－2.71＝?
 .43                        机：.43
```

(TIMES 9 3)	人：$9 \times 3 = ?$
27	机：27
(QUOTIENT 9 3)	人：$9 \div 3 = ?$
3	机：3
(MAX 2 4 3)	人：$\max(2, 4, 3) = ?$
4	机：4
(MIN 2 4 3)	人：$\min(2, 4, 3) = ?$
2	机：2
(ADD 1 6)	人：$6 + 1 = ?$
7	机：7
(SUB 1 6)	人：$6 - 1 = ?$
5	机：5
(SQRT 4)	人：$\sqrt{4} = ?$
2	机：2
(EXPT 2 3)	人：$2^3 = ?$
8	机：8
(MINUS 8)	人：$-8 = ?$
-8	机：-8
(ABS -8)	人：$\lvert -8 \rvert = ?$
8	机：8

LISP 程序是由一些函数组成的．函数又由40来个基本函数通过复合或递归等手段构成．这些基本函数，大半就象这里介绍的 PLUS，DIFFERENCE，TIMES，…等等，可借英文的词意助记．以下将介绍几个最重要的基本函数．

再明确一下符号表达式的概念．例如

(PLUS(TIMES 2 2)(QUOTIENT 2 2))

人：2 × 2 + 2 / 2 = ?

作为一个符号表达式来看，它的第1，2，3个元素分别是

PLUS，(TIME 2 2)，(QUOTIENT 2 2)。

这三个元素中后两个又是表，这就是说，表的元素还可以是表，表本身又可以是别的表的元素。至于原子，就是不带圆括号的字符串，诸如PLUS，3.14，FOO，B27，123XYZ，等等。

头四个最基本的函数是：CAR(取头)，CDR(留尾)，APPEND(合并)，CONS(加入)，这是对符号表达式的最基本操作。头一对作分解操作，后一对作合并操作，例如：

```
(CAR '(FAST COMPUTERS ARE        人：取头
      NICE))
  FAST                           机：FAST
(CAR '(ABC))                     人：取头
  A                              机：A
(CAR '((A B) C))                 人：取头
  (A B)                          机：(AB)
```

函数CDR完成与CAR相补的操作：CAR是取头，CDR则留尾，即是留下被CAR取了头之后剩下的表，连原表的圆括号也留下。

```
(CDR '(FAST COMPUTERS ARE NICE)
                                 人：留尾
  (COMPUTERS ARE NICE)           机：(COMPUTERS
                                     ARE NICE)
(CDR '(A B C))                   人：留尾
  (B C)                          机：(B C)
```

(CDR '((A B) C))　　人：留尾

(C)　　机：(C)

这里用了引号“'”，它的作用就象自然语言中的标点符号一样，关系重大。原因在于，在LISP语句中，操作函数与被操作数据都是原子，它们齐集于一个符号表达式中，不加标点符号就无从辨别。例如(CAR (CDR (A B C)))就可以有多种解释：

(CAR(CDR '(A B C))) = (CAR(B C)) = B

但

(CAR '(CDR(A B C))) = CDR

或

(CAR (CDR (A 'B C))) = ?

在第一种解释下CDR是函数，在第二种解释下CDR是变元。在第三种解释下A是函数（也许是用户定义的），B和C是它的变元。标号符号“'”指示其前是函数，其后是变元表，其前是一系列操作，其后是被操作的数据。这样，在第一种解释中，那里的标点符号“'”就防止了LISP插手(ABC)而作第三种解释，也防止了LISP作出第二种解释。在第三种解释下，A或者是LISP的基本函数，或者是用户定义的函数，否则LISP将告知无定义，函数出错。注意，'S也可以写成(QUOTE S)。例如，'(ABC)可写成(QUOTE (A B C))。

总之，通过取头、留尾操作CAR和CDR，可以提取表中的某一项。为了从多层嵌套的表中提取某一项，往往需要多次操作CAR和CDR。它们的复合函数可以简记为C-R，C--R，C---R，C----R的形式，其中“-”或为A或为D，分