递归程序变换及其实验*

蔡经球　郭艺勋**　许志端***

（厦门大学人工智能与计算机研究所）

摘　要

递归程序变换是软件自动化研究中程序变换途径的一种方法。本文介绍了递归程序等价变换的一系列模式以及根据递归程序变换基本思想而构造的人-机交互实验系统XDPTS的梗概。

软件自动化是计算机科学的前沿课题之一[11]，其主要研究途径有：演绎综合途径、程序变换途径、实例推广途径和过程化途径，其中程序变换是目前较为实用可行的途径，越来越受到计算机科学工作者的关注[1-8,10-16,19-21]，它大体上可分为纵向变换和横向变换两大类

所谓纵向变换是指将某一抽象级别的程序转换成较低抽象级别的程序，如西德慕尼黑技术大学F. L. Bauer教授主持的CIP项目和南京大学徐家福教授主持的NDAUTO系统[11,12]。

所谓横向变换是指在相似的抽象级别上将一个语言成分转换为另一个与之等价但效率更高的语言成分[11]。

横向程序变换的研究可追溯至1966年Cooper首次提出Cooper的变换[3]，而英国爱丁堡大学Burstall和Darlington的研究以及他们所研制的ZAP系统是目前这方面最有代表性的工作[2,7,9]。

将递归函数式程序变换成语义上等价而运行效率更高的迭代程序文本是横向变换的一种重要方式。我们主要针对递归函数式程序的变换，构造了一个人-机交互的递归函数式程序变换实验系统XDPTS，把某些类型的递归程序转化成等价的迭代程序。进行这项工作的意义在于，改进程序设计的时空效率（这对系统的核心程序尤为重要），搜索程序自动化的途径；随着匹配、变换效率的提高、若最终可以消除人机交互，那么XDPTS就可设计成一些程序设计语言编译系统的一个组成部分，用于消除一大类递归形式，提高编译后之可执行程序的运行时空效率。

一、递归函数式程序的变换

递归函数式程序是程序变换系统XDPTS的输入，我们将给出它的一种模型，后面所做的工作都是建立在递归函数式程序的这种表示模型的基础上。

我们讨论的是一种简化的递归函数式程序模型，其一般形式为[19]：

```
F (x1, x2, …, xn) ≡if p1 then E1
                    else if p2 then E2
                    ……
                    else if pm then Em
                    else Em+1        (*)
```

其中pi(i=1, 2, …, m)是测试谓词，Ei (i=1, 2, …, m+1)是表达式，F可以在Ei和Pi中出现，这种出现称为F的递归调用。

程序变换实质上就是根据某些变换模式把一个程序文本转换为另一个程序文本，而

*)本研究课题得到国家自然科学基金的资助。

**) 现在厦门大学经济学院计统系。

***)现在厦门大学经济学院工商管理教育中心。

这两个程序在语义上是等价的。变换模式是对同一类变换的抽象描述，每一变换模式由有序对（输入模式a，输出模式b），以及一个可用性条件c所组成[11]，其图示为图1：

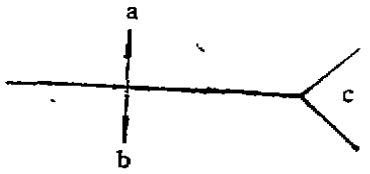


图1

"↔"表示两个样本可互推。

据有关文献介绍[19-27]的递归至尾递归的变换模式（通过手工变换得到的）主要有以下七类（十五种），我们的实验系统XDP-TS目前就是以这七类变换模式为知识基础，对满足这七类变换模式的可用性条件的递归函数式程序进行自动变换，得到相应的尾递归程序。所谓尾递归，是指在f的递归定义中，对f调用以后，在返回的结果值上不再出现其它任何形式的运算。针对这种尾递归程序，引入程序变量，利用循环、赋值语句等便能很容易地转化为等价的迭代程序，达到递归的目的，从而提高程序运行的时空效率。

二、递归程序等价变换模式

限于篇幅，我们在此只简要叙述各类变换的输入模式。输出模式及可用性条件、有关变换的应用示例及正确性证明（采用结构归纳法）将不列出，需要时可参阅文献[19~27]。

2.1 一阶递归程序的等价变换

以下讨论的递归函数只含有一个递归分支，称为一阶递归程序。

1. 一阶Cooper变换（A型）[3, 20, 22]

这相当于1966年Cooper提出的Cooper变换之原型。

(1) 输入模式：

f(x):=if b(x) then h(x) else F(f(k(x)), g(x))

(2) 输出模式：

f(x):=G(x, e).

G(x,y):=if b(x) then F(h(x), y)
else G(k(x), F(g(x), y)

(3) 可用性条件：①F满足结合律：F(F(x,y), z)=F(x,F(y,z))；②F具有右单位元e，即F(x,e)=x；③b, h, g, k中均不含f

2. 一阶Cooper变换（B型）[22-23]

(1) 输入模式：

f(x):=if b(x) then h(x)else F(g(x), f(k(x)))

(2) 输出模式：

f(x):=G(e,x)

G(y,x):=if b(x) then F(y,h(x))else G(F(y,g(x)), k(x))

(3) 可用性条件：①F满足结合律；②F具有左单位元e，即F(e,x)=x；③ b, h, g, k中均不含f

3. CZ变换[23-24]

有些情况下，Cooper变换的可用性条件①即F满足结合律不能成立，但F可具有另外的性质，为此我们提出如下的CZ变换。

(1) 输入模式：

f(x):=if b(x)then h(x) else F(g(x), f(k(x)))

(2) 输出模式：

f(x):=G(x,h(x0))
(其中x0满足b(x0)为true)

G(x,y):=if b(x) then y else
G(k(x),F(g(x), y))

(3)可用性条件：①F满足如下性质
F(x,F(y,z)) =(y,F(x,z))；
②b, h, g, k中均不含f

4. 函数反演变换[16, 19]

在Cooper变换的输入模式中，若函数K(x)具有反函数$K^{-1}(x)$，则可使用下述函数反演变换：

(1) 输入模式：

f(x):=if b(x) then h(x) else F(f(k(x)), g(x))

(2) 输出模式：

f(x):=G(x,x0,h(x0))

G(x,y,z):=if y=x then z elseG(x, k^{-1}(y), F(z,g(k^{-1}(y))))

(3) 可用性条件：①b(x0)为true；②K(x)

存在反函数$K^{-1}(x)$，③b，h，k，g中均不含f

5． Cooper变换的改进型[18]

Cooper变换(A型)的可用性条件②即F具有右单位元e，这是为了保证在一开始时b(x)就为真时变换的正确性，因此只要将输出模式多加一个分支，将上述情况分离另作处理，则可用性条件②实际上可略去，于是[19]提出Cooper变换（A型）的改进型：

(1) 输出模式：

```
f(x):=if b(x) then h(x) else G(k(x),
      g(x))
G(x,y):=if b(x) then F(h(x),y)else
      G(k(x), F(g(x), y))
```

(2) 可用性条件：①F满足结合律；②b，h，k，g中均不含f

2.2 二阶递归程序的等价变换

以下讨论的递归函数含有两个递归分支，称为二阶递归程序。由于两个递归分支的相互影响，其变换将出现较复杂的情况。

1． 二阶Cooper变换（A型）[22]

(1) 输入模式：

```
f(x):=if b(x) then h(x) else
      if b1(x) then F1(f(k1(x)), g1
      (x)) else F2(f(k2(x)), g2(x))
```

(2) 输出模式：

```
f(x):=G(x,e)
G(x,y):=if  b(x) then  F1(h(x),y)
    else if b1(x) then G(k1(x),
    F1(g1(x),y))  else G(k2(x),
    F2(g2(x),y))
```

(3) 可用性条件：①F1满足结合律，即F1(F1(x, y), z)=F1(x, F1(y, z))；②F1关于F2满足结合律，即F1(F2(x, y), z)=F1(x, F2(y, z))；③F1具有右单位元e，即F1(x, e)=x；④b，b1，h，k1，k2，g1，g2中均不含f

2．二阶Cooper变换（B型）[22]

(1) 输入模式：

```
f(x):=if b(x) then h(x) else
      if b1 (x)then F1(g1(x), f(k1)
      (x)))  else F2 (g2(x),
      f(k2(x)))
```

(2) 输出模式：

```
f(x):=G(e,x)
G(y,x):=if b(x) then F1(y, h(x))else
    if b1(x) then G(F1(y,g1(x)),
    k1(x))  else G(F2(y, g2(x)),
      k2(x))
```

(3) 可用性条件：①同A型（除③外）；②F1具有左单位元e，即F1(e, x)=x

3． C变换[21]

(1) 输入模式：

```
f(x):=if b(x) then h(x) else
if b1(x) then F1(f(k1 (x)), g1(x))
else F2(f2(x)), g2(x)
```

(2) 输出模式：

```
f(x):=G(x, e)
G(x, y):=if b(x) then F1(y, h(x))
      else  if b1(x) then  G(k1(x),
      F1(y, g1(x)))  else G(k2(x),
      F2(y, g2(x)))
```

(3) 可用性条件：①F1满足，F1(F1(x, y), z)=F1(x, F1(z, y))；②F1,F2满足，F1(F2(x, y), z)=F1(x, F2(z, y))；③F1具有右单位元e；④b，b1，h，k1，k2，g1，g2中均不含f

4． T1变换[20]

(1) 输入模式：

```
f(x):=if b(x) then h(x) else
      if b1(x) then  F1(f(k1(x)),
      g1(x))  else F2(f(k2(x)),
      g2(x))
```

(2) 输出模式：

```
f(x):=G(x, e1, e2)
G(x,y,z):=if b(x) then F1(F2(h(x),
      z), y) else  if b1(x)
      then G(k1(x),F1(F2(g1
      (x),z),y),z)else G(k
      2(x), y, F2(g2(x), z))
```

(3) 可用性条件：①F1，F2满足结合律，并分别具有右单位元e1，e2；②F2关于F1满足分配律，即：F2(F1(x,y),z)=F1(F2(x,z), F2(y,z))；③b，b1，h，k1，k2，g1，g2中均不含f

2.3 T2变换[20]

在Cooper变换的输入模式中，决定分支条件的布尔表达式均未出现递归调用，当布尔表达式中出现递归调用将使消除递归的问题变换更加复杂。

(1) 输入模式

```
f(x):=if b(x) then h(x) else
        if B(f(k(x)), d(x)) then D(x)
                            else f(k(x))
```

(2) 输出模式：

```
f(x):=if b(x) then G(k(x), d(x), D
        (x))
G(x,y,z):=if b(x) then[if B(h(x),y)
        then x else h(x)]
        else [if B(D(x),y) then
        G(k(x), y, z)
        else G(k(x), d(x), D(x))]
```

(3) 可用性条件：①$\forall t$：B(t, d(x))∧B(D(x), y)⇒B(t,y)=true ¬B(t, d(x))∧¬B(D(x), y) ⇒B(t, y)=false，②b，h1，d，D，k中均不含f

2.4 RR变换[21,25]

(1) 输入模式：

```
f(x):=if b(x) then h(x) else f(f(k
      (x)))
```

(2) 输出模式：

```
f(x):=G(x,0)
G(x,y):=if b(x) then [if y=0 then
        h(x) else G(h(x), y-1)]
        else G(k(x), y+1)
```

(3) 可用性条件：

①b，h1，d，D，k中均不含f

2.5 多步递归变换[24]

(1) 输入模式：

```
f(x):=if b1(x) then h1(x) else
      if b2(x) then h2(x) else
        F1(F2(f(k(x)), g1(x)),
        F2(f(k(k(x))), g2(x)))
```

(2) 输出模式：

```
f(x):=if b1(x) then h1(x) else
      if b2(x) then h2(x) else G(x,
        e2, e1)
G(x, y, z) :=if b2(x) then F1(F2
      (h2(x), y), F2(h1(k(x)), z))
      else G(k(x), F1(F2(g1(x),
      y, z), F2(g2((x),y)))
```

(3) 可用性条件：①F2满足结合律，即F2(F_2(x,y),z) =F2(x, F2(y, z))，② F1(F1(x,y), z) =F1(F1(x, y), z)，③F2关于F1满足分配律，即F2(F1(x, y), z) =F1(F2(x, z), F2(y, z))，④F1，F2分别具有右单位元e1，e2；⑤b2(x)⇐⇒b1(k(x))；⑥b1，b2，g1，g2，h1，h2中均不含f

2.6 并行递归的消除[25-27]

我们先约定两个概念：将所有表元素都是原子的表称为单表，例如表(A B C D)就是一个单表。而复表则是指表元素可以是原子也可以是表的表，如(A (B C (D F)) K)。

在对复表进行处理时，经常会出现并行递归的情况[9]，例如，计算复表L中所有原子个数的函数

```
COUNTATOM(L):=if L=NIL then 0
                else if ATOM(CD
                R(L)) then COUN-
                TATOM(CDR(L))+1
                else COUNTATOM
                (CAR(L))+COUN-
                TATOM(CDR(L))
                                (**)
```

的最后一个分支就是并行递归。对于并行递归目前还无有效变换成尾递归的变换模式[10]。

对于相当广泛的一类并行递归程序可以采用一种间接消除递归的方法[25-27]：

①将复表改造成伪单表形式 例如可约定在复表内层用%代替左括号，&代替右括号，而把%和&也当成一个特殊的元素，称这种表形式为伪单表。

这样复表 (A (B (C)) (D)) 可改造为伪单表(A % B % C & & % D &)

②设计相应的处理伪单表的递归程序，并选用适

当变换。它只需将处理单表的递归程序略加修改就可得到，且不会再出现并行递归调用。选用适当的变换技术可把这个程序转化为等价的尾递归形式。

③如有必要，可将处理后的结果表(伪单表形式)再改造成复表形式。

2.7 高阶递归变换[20]

(1)输入模式：

```
f(x):=if b(x) then h(x) else
      if b1(x) then F1(f(k1(x)),
         g1(x)) else
          ...      ...
      if bn(x) then Fn(f(kn(x)),
      gn(x)) else  F0(f(k0(x)),
      g0(x))
```

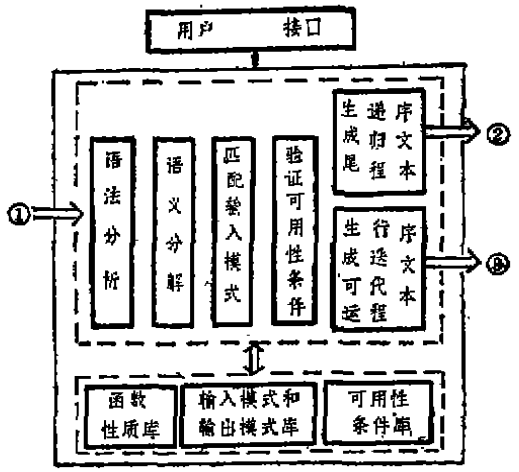
(2)输出模式：

```
f(x):=if b(x) then h(x)
      else if b1(x) then  G1(k1(x),
      g1(x))
           ...    ...
      else if bn(x) then Gn(kn(x),
      gn(x))
      else G0(k0(x), g0(x))
Gi(x, y):=if b(x) then Fi(h(x), y)
      else if b1(x) then  Gi(k1(x),
      F1(g1(x), y))
        ...        ...
      else if bn(x) then Gi(kn(x),
      Fn(gn(x), y))
      else Gi(k0(x), F0(g0(x), y)
                    (0≤i≤n)
```

(3)可用性条件：①Fi满足结合律，即Fi(Fj(x, y), z)=Fi(x, Fj(y, z)), 0≤i, j≤n; ②b, bj, h, gi, ki中均不含f, 0≤i≤n, 1≤j≤n

三、一个实例—XDPTS实验变换系统

我们的实验变换系统XDPTS是使用编译型汉字dBASE Ⅲ语言在IBM-PS Ⅱ/30机上实现的。该系统以第一节介绍的模型(*)为基础，其总体结构如图2所示。其中函数性质库、输入模式与输出模式库、可用性条件库等几种知识库。在XDPTS运行的最初，可以是空的。以后随着系统的运行，便可不断地扩充或修改，亦即系统的知识、经验不断积累、丰富，变换的能力也越来越强。在用户接口方面，为用户提供对程序变换过程的不同干预，包括以下五种交互窗口：(1)输入窗，(2)语义解释窗，(3)条件验证窗，(4)知识获取窗，(5)跟踪窗。这五个窗口为用户提供了全屏幕的提示操作方式。



①递归函数式程序文本；
②尾递归程序文本；
③可运行的迭代程序。

图2

总之，XDPTS的实现涉及了许多繁杂而琐细的工作，工作量颇大。在我们的工作中，对dBASE Ⅲ的许多功能进行了拓广，探索并研制了一些解决人工智能问题的程序。XDPTS的源程序约100k，编译后的运行文件200k。它可以不作任何修改直接移植到IBM-PC/XT机上运行。

最后，感谢陈火旺教授对本文的悉心指导。

参 考 文 献

[1] Broy, M., 等 程序变换:程序设计的一种形式化方法，计算机科学，1988.2，65-73

[2] Burstall, R. M., Darlington, J., A Transformation System for Develop-

ing Recursive Programs, JACM, 1977, 1, 44-67

[3] Cooper, D. C., The Equiralence of Certain Computations, Comp. J., 1966. 9, 184-206

[4] Gries, D., The Science of Programming Methodology, Prentice Hall, 1977

[5] Linger, R. C., Mills, H. D. and Witt, B. I., Structured Programming: Theory and Practice, Reading, Mass., Addison-Weslay, 1979

[6] Manna, Z., Mathematical Theory of Computation, McGraw-Hill, New York, 1974

[7] Manna, Z. & Waldinger, R., The Logic of Computer Programming, IEEE Trans on Software Engineering, SE-4, 1978, 3

[8] Partsch, H. & Stembruggen, R., A Comprehensive Survey of Program Transformation, TUM Instftut Fur Informatik, July 1981

[9] Winston, P. H等著, 黄昌宁等译, LISP程序设计, 清华大学出版社, 1983

[10] Yeh, R. T. ed, Current Trends in Programming, Springer-verlag, 1981

[11] 徐家福, 软件自动化, 计算机研究与发展, 1988.11, 7-13

[12] 徐家福, 戴敏, 袁峰, 基于NDAUTO系统的软件开发过程, 计算机研究与发展, 1988. 6, 1-7

[13] 吕建, 算法设计自动化研究综述, 计算机科学, 1988.3, 15-20

[14] 樊哲民, 程序的形式说明、推导和变换, 计算机科学, 1982.2

[15] 樊哲民, 程序变换概论, 电子学报, 1983. 1

[16] 仲萃豪, 冯玉琳, 陈友君, 程序设计方法学, 北京科学技术出版社, 1985

[17] 陈火旺, 罗朝晖, 马庆鸣, 程序设计方法学基础, 湖南科学技术出版社, 1987

[18] 陈火旺 等编, 编译原理, 国防工业出版社, 1980

[19] 胡正国, 蔡经球, 程序设计方法学, 西北工业大学出版社, 1987

[20] 蔡经球, 周松, 递归算法的若干等价变换, 厦门大学学报, 1983.2, 165-174

[21] 蔡经球, 递归算法等价变换的若干新模式, 人工智能学报, 1983.3, 18-30

[22] 蔡经球, 关于Cooper变换的研究, 厦门大学学报, 1987. 1, 33-40

[23] 蔡经球, 张克均, 递归程序变换的一种新模式——CZ变换, 厦门大学学报, 1988. 4

[24] 张克均, 蔡经球, 关于多步递归算法的等价变换模式, 福建电脑, 1988.2 17-18

[25] 蔡经球, 关于并行递归程序变换的若干研究, 小型微型计算机, 1989, 11

[26] 蔡经球, 关于横向程序变换的若干研究, (待发表)

[27] 蔡经球, 关于并行递归的消除, (待发表)

[28] 涂序彦, 人工智能及其应用, 电子工业出版社, 1988

[29] 陈增武 等编著, 汉字dBASE Ⅲ及其程序设计技巧, 浙江省计算机学会、《计算机时代》编辑部

[30] 郭艺勋, 许志端, 蔡经球, 利用汉字dBASE Ⅲ实现人工智能技术, (待发表)

[31] 郭艺勋, 递归程序变换实验系统XDPTS, 厦门大学硕士研究生毕业论文

[32] 蔡经球, 郭艺勋, 许志端, 递归程序变换及其实验系统XDPTS, 福建省计算机1989年学术年会论文