ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

С. А. Романенко

Прогонка для программ на РЕФАЛЕ-4

Рекомендуемая форма библиографической ссылки: Романенко С. А. Прогонка для программ на РЕФАЛЕ-4 // Препринты ИПМ им. М.В.Келдыша. 1987. № 211. 23 с.
https://library.keldysh.ru/preprint.asp?id=1987-211

Ордена Ленина
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В. Келдыша
Академии наук СССР

С.А. Романенко

ПРОГОНКА ДЛЯ ПРОГРАММ НА Р Е Ф А Л Е – 4

Препринт № 211 за 1987г.

Москва

Ордена Ленина
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
им. М.В.Келдыша
Академии Наук СССР

С.А.Романенко

ПРОГОНКА ДЛЯ ПРОГРАММ НА РЕФАЛЕ-4

Москва
1987

УДК 681.3.06

В работе предлагается набор преобразований для программ, написанных на Рефале-4. Эти преобразования, представляют собой обобщение п р о г о н к и - системы преобразований, применимых к программам на "Ограниченном Рефале". Рефал-4 является расширением Базисного Рефала и Рефала-2, в то время как "Ограниченный Рефал" является их подмножеством.

КЛЮЧЕВЫЕ СЛОВА И ФРАЗЫ: обработка символьной информации, преобразование программ, рекурсия, рефал, сопоставление с образцом, функциональное программирование.

С О Д Е Р Ж А Н И Е

В В Е Д Е Н И Е

В работах [ТУР 72], [ТУР 74] была предложена п р о г о н к а - система преобразований, обеспечивающая возможность выполнять метавычисления (в частности - частичные вычисления) для программ, написанных на языке Рефал [ТУР 66], [БЗР 77], [КР 87]. (Позднее подобная система преобразований была рассмотрена в [БД 77].) Прогонка является основой "суперкомпилятора" - системы автоматического анализа и преобразования рефал-программ [ТУР 86].

В том виде, как она была известна до настоящего времени, прогонка применима только к программам, написанным на так называемом "ограниченном рефале" [ТУР 72], [ТУР 74]. Главной особенностью ограниченного рефала является то, что в нем запрещено употребление открытых VE-переменных, а также - повторных W- и VE-переменных.

Как представляется автору, прогонка была сформулирована только для ограниченного рефала не по той причине, что ее невозможно выполнять для Базисного Рефала [БЗР 77] или Рефала-2 [КР 87], а в силу того, что ее результаты средствами самого Рефала-2 невыразимы.

В [РОМ 87] описано расширение Рефала-2 - Рефал-4, а также одно из преобразований над программами на Рефале-4 - раскрытие функциональных термов. Раскрытия функциональных термов вполне достаточно, чтобы достичь формальной цели прогонки - соединения двух шагов рефал-машины в один. Однако, содержательная цель прогонки состоит в том, чтобы слить два процесса отождествления в один и при этом избежать промежуточного построения выражений. Ниже будет показано, что раскрытия функциональных термов недостаточно для достижения этой цели. Затем будет сформулирован набор локальных

преобразований, которые (вместе с раскрытием функциональных термов) позволяют добиться тех же результатов, что и "классическая" прогонка.

Рефал-4 является только одним из возможных способов расширения Базисного Рефала и Рефала-2. Другие проекты расширения рефала можно найти, например, в работах [ТУР 71], [БЕЛ 83], [БЕЛ 85].

1. НЕДОСТАТОЧНОСТЬ РАСКРЫТИЯ ФУНКЦИОНАЛЬНЫХ ТЕРМОВ

Рассмотрим функции

```
#FUN F
   # EX #FENCE = <G 'A' EX>
#END F
#FUN G
   # EB SA #FENCE = (EB) SA
#END G
```

Раскроем обращение к "G" в описании функции "F". Получается

```
#FUN F
   # EX #FENCE
      #GATE #ALT
         #PATH 'A' EX : EB SA #FENCE = (EB) SA
      #END
#END F
```

Чтобы не загромождать дальнейшее изложение, сразу же очистим это описание функции от лишних #GATE, #ALT и #FENCE.

```
#FUN F
   # EX, 'A' EX : EB SA #FENCE   = (EB) SA
#END F
```

Легко видеть, что формальная цель прогонки достигнута: устранен один шаг рефал-машины (вызов вспомогательной функции "G"). Между тем, содержательная цель прогонки состоит в том, чтобы суметь сделать два отождествления подряд, не делая промежуточного синтеза выражений. И эта цель не достигнута, ибо в результате прогонки возникла перестройка

```
'A' EX : EB SA
```

которая не является сужением! Исполнение этой перестройки как раз и заключается в том, что сначала строится выражение 'A' EX, а затем это выражение подвергается анализу.

Таким образом, чтобы достичь содержательной цели прогонки, требуется проделать над получившейся программой некоторые преобразования. Цель этих преобразований - превращение перестроек общего вида (не содержащих в источнике вызовы функций) в программы, состоящие из сужений и управляющих конструкций.

Промежуточным этапом при достижении этой цели будет программа, в которой все перестройки (не содержащие вызовов функций) преобразованы в сужения и присваивания.

2.ПРЕОБРАЗОВАНИЕ ПЕРЕСТРОЕК В СУЖЕНИЯ И ПРИСВАИВАНИЯ

Рассмотрим перестройку

```
'A' EX : &EB &SA
```

Переменные EB и SA в получателе этой перестройки помечены знаком "&" в знак того, что они получают значение во время исполнения этой перестройки. В рефал-программе "новые" переменные не помечаются, так как их всегда можно отличить от "старых" по контексту. Однако, при выполнении преобразований мы будем использовать пометки, чтобы преобразования можно было делать локально, не используя широкий контекст.

Итак, попробуем преобразовать данную перестройку в сужения и присваивания. Нам пока мешает то обстоятельство, что источник состоит не из одной переменной.

Значение переменной EX станет известно только во время исполнения перестройки. Однако, оно обязательно окажется либо пустым, либо непустым. Причем, если оно непусто, то сопоставление с образцом будет успешным, только если EX будет оканчиваться на символ. Поэтому, если EI - переменная, которая нигде больше в описании функции не используется, то перестройку можно заменить на

```
#GATE #ALT
   #PATH EX : #FENCE 'A' : &EB &SA
   #PATH EX : &EI &SA #FENCE 'A' EI : &EB
#END
```

Теперь преобразуем перестройку 'A' : &EB &SA в последовательность

```
: &EB , 'A' : &SA
```

Таким образом, описание функции "F" преобразуется к виду:

```
#FUN F  # EX
  #GATE #ALT
     #PATH EX : #FENCE : EB, 'A' : SA
     #PATH EX : EI SA #FENCE 'A' EI : EB
  #END
  #FENCE = (EB) SA
#END F
```

Итак, первая промежуточная цель достигнута: остались только присваивания и сужения.

Теперь постараемся устранить присваивания. Это будет делаться проталкиванием присваиваний вправо. Пусть присваивание имеет вид E:&X, где X - новая переменная. Тогда, если

вслед за присваиванием находится перестройка вида R : D L, где R - источник, D - указатель направления отождествления (всегда #L или #R), а L - типовое выражение, мы будем применять следующее преобразование:

```
E : &X , R : D L  ⇒
   SUBST(X,E,R) : D SUBST(X,E,L) , E : &X
```

где SUBST(X,E,E') обозначает выражение, которое получается из выражения E', если в последнем заменить все вхождения переменной X на выражение E.

Если присваивание оказывается перед выходным присваиванием, будем применять преобразование

```
E : &X  = R  ⇒  = SUBST(X,E,R)
```

Кроме того, понадобится следующее очевидное преобразование:

```
E : &X #FENCE  ⇒  #FENCE E : &X
```

Теперь наша задача ясна - присваивания нужно продвигать вплоть до выходного присваивания, где они и исчезнут. Но этому препятствует закрывающий #END конструкции #ALT. Поэтому предварительно следует "втянуть" конец описания функции внутрь #ALT-а.

```
#FUN F  # EX
   #GATE #ALT
      #PATH  EX :          #FENCE : EB, 'A' : SA
                                    #FENCE = (EB) SA
      #PATH  EX : E1 SA  #FENCE 'A' E1 : EB
                                    #FENCE = (EB) SA
   #END
#END F
```

Теперь, проталкивая присваивания вправо, получаем

```
#FUN F  # EX
   #GATE #ALT
      #PATH  EX :         #FENCE  #FENCE = () 'A'
      #PATH  EX : EI SA   #FENCE  #FENCE = ('A' EI) SA
   #END
#END F
```

И, наконец, устраняя лишние #GATE и #ALT, получаем

```
#FUN F
   #          #FENCE = () 'A'
   #  EI SA   #FENCE = ('A' EI) SA
#END F
```

3. ПРЕОБРАЗОВАНИЕ ПОВТОРНЫХ VE-ПЕРЕМЕННЫХ

Рассмотрим перестройку

EA EB : EB EA

Как преобразовать ее в цепочку сужений и присваиваний? Можно сделать это, руководствуясь следующими соображениями. Предположим, что "EA EB" успешно сопоставляется с "EB EA". В данном случае это означает, что источник перестройки совпадает с получателем после замены переменных EA и EB на их значения.

Каковы бы ни были значения переменных EA и EB, могут встретиться только три случая: либо EA и EB содержат одинаковое число термов, либо значение EA короче значения EB, либо значение EB короче значения EA. Таким образом, если сопоставление "EA EB" с "EB EA" возможно, то либо EA начинается с EB, и тогда EA можно представить в виде "EB E1", либо EB начинается с EA, и тогда EB можно представить в виде "EA E2". Пусть, например, EA начинается с EB. Тогда исходную перестройку можно представить в виде

EB E1 EB : EB EA

Видно, что теперь можно "сократить" EB в обеих частях перестройки, после чего получается перестройка

```
EI EB : EA
```

которую мы можем превратить в сужение поменяв местами источник и получатель. Если же EB начинается с EA, исходную перестройку можно представить в виде

```
EA EB : EA E2 EA
```

Сокращая EA, получаем сужение

```
EB : E2 EA
```

Таким образом, исходную перестройку можно преобразовать в следующую конструкцию Рефала-4:

```
#GATE #ALT
  #PATH  EA : EB EI #FENCE  EA : EI EB
  #PATH  EB : EA E2 #FENCE  EB : E2 EA
#END
```

Для прогонки, описанной в [ТУР 72], [ТУР 74], повторные VE-переменные представляли неодолимое препятствие. Однако, программы на Рефале-4, могут анализировать значения одних и тех же переменных много раз различными способами во время одного шага рефал-машины. Благодаря этому и удается свести сопоставление двух повторных VE-переменных к конечному числу сужений и присваиваний.

4. ПРЕОБРАЗОВАНИЕ ОТКРЫТЫХ VE-ПЕРЕМЕННЫХ

Рассмотрим перестройку

```
EA EB : #L &EX &EY
```

Если сопоставление образца со значением источника возможно, то либо длина значения переменной EX не превышает длину значения переменной EA, и тогда EA можно представить в виде "E1 E2", где E1 равно EX, либо длина значения переменной EX не меньше, чем длина значения выражения EA, и тогда EB можно представить в виде "E3 E4", где "EA E3" равно EX. Таким образом, исходную перестройку можно преобразовать к виду

```
#ALT
  #PATH  EA : #L &EX &E2, E2 EB : &EY
  #PATH  EB : #L &E3 &EY, EA E3 : &EX
#END
```

5. СИСТЕМА ПРЕОБРАЗОВАНИЙ ПЕРЕСТРОЕК

Пусть описание некоторой функции содержит только такие перестройки, в источниках которых нет вызовов функций. Ниже описан набор преобразований, с помощью которого можно изгнать из описания функции все перестройки, которые не являются сужениями.

Будем говорить, что терм является ж е с т к и м, если этот терм - символ, или S-переменная, или W-переменная, или выражение, заключенное в структурные скобки.

Будем использовать следующие обозначения.

[] - пустое выражение.

D - указатель направления отождествления (#L или #R).

T - произвольный терм.

M и N - произвольные жесткие термы,

R и L - типовые выражения.

SUBST(X,E,E') - результат замены всех вхождений переменной X на выражение E в выражении E'.

Преобразования, выполняемые над программой, разбиваются на группы.

Р А З Б О Р П О Т Е Р М А М.

```
M R : #L N L   ⇒   M : #L N, R : #L L
M R : #R N L   ⇒   R : #R L, M : #R N
R M : #L L N   ⇒   R : #L L, M : #L N
R M : #R L N   ⇒   M : #R N, R : #R L
```

С О К Р А Щ Е Н И Я.

```
(R) : D (L)    ⇒   R : D L
T R : D T L    ⇒   R : D L
R T : D L T    ⇒   R : D L
[] : D []      ⇒   [пусто]
```

М А Т Е Р И А Л Ь Н Ы Й Б А Л А Н С.

Пусть перестройка имеет вид:

X : D C1 X C2

где X - переменная, а C1 и C2 - такие последовательности символов, скобок и переменных, что C1 C2 - выражение. Тогда ее можно заменить на перестройку

[] : D C1 C2

П Е Р Е С Т Р О Й К И Т И П А Т Е Р М - Т Е Р М.

Эти преобразования делаются только после того, как оказались неприменимы предыдущие преобразования.

Пусть A, B - произвольные различные символы, а SA, SB, WA, WB, EA, EB, VA, VB - произвольные различные переменные, уже имеющие значение. Пусть E' и E'' - произвольные выражения. Тогда выполнимы следующие преобразования:

```
A : B     ⇒  #FAIL
A : SB    ⇒   SB : A
A : WB    ⇒   WB : A
A : EB    ⇒   EB : A
A : VB    ⇒   VB : A
A : (E'')     ⇒   #FAIL
SA : WB   ⇒    WB : SA
SA : EB   ⇒    EB : SA
SA : VB   ⇒    VB : SA
SA : (E'')    ⇒   #FAIL
WA : EB   ⇒    EB : WA
WA : VB   ⇒    VB : WA
VA : EB   ⇒    EB : VA
(E') : B      ⇒   #FAIL
(E') : SB     ⇒   #FAIL
(E') : EB     ⇒   EB : (E')
(E') : VB     ⇒   VB : (E')
```

Во всех этих правилах в получателях перестроек опущен указатель направления отождествления, поскольку он не меняется при выполнении этих преобразований.

П У С Т О Й И С Т О Ч Н И К.

```
[] : D M L      ⇒  #FAIL
[] : D VB L     ⇒  #FAIL
[] : D &VX L    ⇒  #FAIL
[] : D EB L     ⇒  EB :[] , [] : D L
```

```
[]: D &EX L   ⇒   []: &EX,  []: D L
```

ПУСТОЙ ПОЛУЧАТЕЛЬ.

```
M R : D []    ⇒  #FAIL
VA R : D []   ⇒  #FAIL
EA R : D []   ⇒  EA : [], R : D []
```

ПОВТОРНЫЕ VE-ПЕРЕМЕННЫЕ.

Пусть UA и UB - произвольные различные VE-переменные.

```
EA R : D L  ⇒
        #GATE #ALT
           #PATH  EA : []  #FENCE  R : D L
           #PATH  EA : &VI #FENCE  VI R : D L
        #END

R EA : D L  ⇒
        #GATE #ALT
           #PATH  EA : []  #FENCE  R : D L
           #PATH  EA : &VI #FENCE  R VI : D L
        #END

VA R : D N L  ⇒  VA : &WI &E2 , WI E2 R : D N L

R VA : D L N  ⇒  VA : &E2 &WI , R E2 WI : D L N

M R : D EB L  ⇒
        #GATE #ALT
           #PATH  EB : []  #FENCE  M R : D L
           #PATH  EB : &VI #FENCE  M R : D VI L
        #END
```

```
R M : D L EB  ⇒
        #GATE #ALT
           #PATH  EB : []   #FENCE  R M : D L
           #PATH  EB : &V1  #FENCE  M R : D L V1
        #END

M R : D VB L  ⇒  VB : D M &E1 , R : D E1 L

R M : D L VB  ⇒  VB : D &E1 M , R : D L E1

UA R : D UB L  ⇒
        #GATE #ALT
           #PATH  UA : UB &E1  #FENCE  E1 R : D L
           #PATH  UB : UA &E1  #FENCE  R : D E1 L
        #END

R UA : D L UB  ⇒
        #GATE #ALT
           #PATH  UA : &E1 UB  #FENCE  R &E1 : D L
           #PATH  UB : &E1 UA  #FENCE  R : D L E1
        #END
```

О Т К Р Ы Т Ы Е vE–П Е Р Е М Е Н Н Ы Е.

Пусть R есть T1 T2 ... TN, где Ti – терм. Тогда

```
R : #L &UX L  ⇒
        #ALT
           #PATH  A0
           #PATH  B0
           #PATH  A1
            ...
           #PATH  BN
           #PATH  AN
        #END
```

где Ai есть #FAIL, если T[i–1] или T[i+1] являются vE–переменными. Иначе Ai есть

T1 T2 ... Ti : &UX, T[i+1] ... TN : #L L

а Bi есть #FAIL, если Ti не является VE-переменной. Если же Ti является VE-переменной, то Bi есть

Ti : #L &E1 &E2, T1 T2 ... T[i-1] E1 : &UX,
E2 T[i+1] ... TN : #L L

Для отождествления справа налево имеем аналогичное правило.

```
R : #R L &UX  ⇒
        #ALT
           #PATH  A0
           #PATH  B0
           #PATH  A1
            ...
           #PATH  BN
           #PATH  AN
        #END
```

где Ai есть #FAIL, если T[i-1] или T[i+1] являются VE-переменными. Иначе Ai есть

Ti ... T2 T1 : &UX, TN ... T[i+1] : #R L

а Bi есть #FAIL, если Ti не является VE-переменной. Если же Ti является VE-переменной, то Bi есть

Ti : #R &E2 &E1, E1 T[i-1] ... T2 T1 : &UX,
TN ... T[i+1] E2 : #R L

П О Л У Ч А Т Е Л Ь - V-П Е Р Е М Е Н Н А Я.

```
EA R : &VX  ⇒
        #GATE #ALT
           #PATH  EA : []   #FENCE  R : &VX
           #PATH  EA : &VI  #FENCE  VI R : &VX
        #END
```

П Р О Т А Л К И В А Н И Е П Р И С В А И В А Н И Й.

```
E : &X , R : D L  ⇒
        SUBST(X,E,R) : D SUBST(X,E,L) , E : &X

E : &X = R  ⇒
        = SUBST(X,E,R)
```

У П Р А В Л Я Ю Щ И Е К О Н С Т Р У К Ц И И.

```
#ALT #END  ⇒  #FAIL

#ALT P #PATH #ABORT Q #END  ⇒
        #ALT P #PATH #ABORT #END

#GATE #ALT P #PATH #FENCE Q #END  ⇒
        #GATE #ALT P #END

#ALT #PATH A #END  ⇒  A

#GATE #FENCE  ⇒  [пусто]

#ALT
   #PATH  A1
   #PATH  A2
    ...
   #PATH  AN
#END B  ⇒
```

```
#ALT
   #PATH  A1 B
   #PATH  A2 B
    . . .
   #PATH  AN B
#END
```

З А К Л Ю Ч Е Н И Е

В тех случаях, когда программа на Рефале-4 довлетворяет тем же ограничениям, что и программы на "ограниченном рефале", предлагаемый набор преобразований позволяет получить те же результаты, что и "классическая" прогонка, описанная в [ТУР 72], [ТУР 74]. Поэтому эти преобразования естественно считать обобщением и переформулировкой прогонки.

В то же время, нетрудно усмотреть сходство между этими преобразованиями и методами преобразования цепочек сужений и присваиваний, рассмотренными в [ТУР 86]. Разница, однако, заключается в том, что в указанной работе рассматриваются только такие образцы, которые не вызывают перебора в процессе отождествления, а в качестве входного языка системы опять таки служит "ограниченный рефал".

В заключение следует сказать, что предлагаемый набор преобразований представляет собой и с ч и с л е н и е, в то время как для выполнения преобразований с помощью компьютера необходимы а л г о р и т м ы. Таким образом, вопрос о том, когда и какие преобразования применять, нуждается в дополнительном рассмотрении.

Л И Т Е Р А Т У Р А

[БД 77]

R.M.BURSTALL, J.DARLINGTON. A TRANSFORMATION SYSTEM FOR DEVELOPING RECURSIVE PROGRAMS. J.ACM 24 (1977), 44-67.

[БЕЛ 83]

А.П.Бельтюков. Язык дедуктивного программирования. – Теория языков программирования. Сборник научных трудов. Ижевск, 1983, с.3-18.

[БЕЛ 85]

А.П.Бельтюков. Эффективное расширение языка Рефал. – Вычислительные методы и информационное обеспечение пакетов прикладных программ. Сборник научных трудов. Устинов, 1985, с.50-54.

[БЗР 77]

Базисный рефал и его реализация на вычислительных машинах. М., ЦНИПИАСС, 1977, с.92-95.

[КР 87]

Ан.В.Климов, С.А.Романенко. Система программирования Рефал-2 для ЕС ЭВМ. Описание входного языка. ИПМ им.М.В.Келдыша АН СССР, М., 1987.

[РОМ 87]

С.А.Романенко. Рефал-4 – расширение Рефала-2, обеспечивающее выразимость результатов прогонки. Препринт ИПМ им.М.В.Келдыша АН СССР, М., 1987, N 147.

[ТУР 66]

В.Ф.Турчин. Метаязык для формального описания алгоритмических языков. – В сб.: Цифровая вычислительная техника и программирование, М.: Сов. радио, 1966, с.116-124.

[ТУР 71]

В.Ф.Турчин. Программирование на языке Рефал. Препринты ИПМ АН СССР N 41, 43, 44, 48, 49. М., 1971.

[ТУР 72]

В.Ф.Турчин. Эквивалентные преобразования рекурсивных функций, описанных на языке Рефал. - В сб.: Теория языков и методы построения систем программирования. Труды симпозиума, Киев-Алушта: 1972, с. 31-42.

[ТУР 74]

В.Ф.Турчин. Эквивалентные преобразования программ на Рефале. - В сб.: Автоматизированная система управления строительством. М.: ЦНИПИАСС, 1974, вып.4, с.36-68.

[ТУР 86]

V.F.TURCHIN. THE CONCEPT OF A SUPERCOMPILER. ACM TRANSACTIONS ON PROGRAMMING LANGUAGES AND SYSTEMS, VOL.8, NO.3, JULY 1986, PP.292-325.

С.А. Романенко " Прогонка для программ на Рефале - 4."
Редактор Вс.С. Штаркман. Корректор А.В. Климов.

Подписано в печать 16.10.87г. № Т-14598. Заказ № 435.
Формат бумаги 60X90 1/16. Тираж 230 экз.
Объём 0,9 уч.-изд.л. Цена 10 коп.

055 (02)2

Отпечатано на ротапринтах в Институте прикладной математики АН СССР
Москва, Миусская пл. 4.

Ссылки на издание рекомендуется делать по следующей форме: и.о., фамилия, название, препринт Ин. прикл. матем. им. М.В. Келдыша АН СССР, год, №.

Распространение: препринты института продаются в магазинах Академкниги г. Москвы, а также распространяются через Библиотеку АН СССР в порядке обмена.

Адрес: СССР, 125047, Москва-47, Миусская пл. 4, Институт прикладной математики им. М.В. Келдыша АН СССР, ОНТИ.

The references should be typed by the following form: initials, name, title, preprint, Inst.Appl.Mathem., the USSR Academy of Sciences, year, N(number).

Distribution. The preprints of the Keldysh Institute of Applied Mathematics, the USSR Academy of Sciences are sold in the bookstores "Academkniga", Moscow and are distributed by the USSR Academy of Sciences Library as an exchange.

Adress: USSR, I25047, Moscow A-47, Miusskaya Sq.4, the Keldysh Institute of Applied Mathematics, Ac.of Sc., the USSR, Information Bureau.

Цена 10 коп.