



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

С.А. Романенко

ПРОГОНКА ДЛЯ ПРОГРАММ НА РЕФАЛЕ - 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

В работе предлагается набор преобразований для программ, написанных на Рефале-4. Эти преобразования, представляют собой обобщение прогонки - 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.

СО Д Е Р Ж А Н И Е

Введение	3
1. Недостаточность раскрытия функциональных термов . .	4
2. Преобразование перестроек в сужения и присваивания .	5
3. Преобразование повторных VE-переменных	8
4. Преобразование открытых VE-переменных	10
5. Система преобразований перестроек	10
Заключение	17
Литература	18

В В Е Д Е Н И Е

В работах [ТУР 72], [ТУР 74] была предложена прогонка - 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:

$$E : \&X, R : D L \Rightarrow \\ \text{SUBST}(X, E, R) : D \text{SUBST}(X, E, L), E : \&X$$

где $\text{SUBST}(X, E, E')$ 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

$$E : \&X = R \Rightarrow = \text{SUBST}(X, E, R)$$

Кроме того, понадобится следующее очевидное преобразование:

$$E : \&X \#FENCE \Rightarrow \#FENCE E : \&X$$

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

Теперь, прогалкивая 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 EI", 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 EI EB : EB EA

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

Е1 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 E1 #FENCE EA : E1 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 EA", 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 жестким, 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 термины,

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.

РАЗБОР ПО ТЕРМАМ.

$MR : \#LNL \Rightarrow M : \#LN, R : \#LL$

$MR : \#RNL \Rightarrow R : \#RL, M : \#RN$

$RM : \#LLN \Rightarrow R : \#LL, M : \#LN$

$RM : \#RLN \Rightarrow M : \#RN, R : \#RL$

СОКРАЩЕНИЯ.

$(R) : D(L) \Rightarrow R : DL$

$TR : DTL \Rightarrow R : DL$

$RT : DLT \Rightarrow R : DL$

$[] : D[] \Rightarrow$ [пусто]

МАТЕРИАЛЬНЫЙ БАЛАНС.

Пусть перестройка имеет вид:

$X : DC_1XC_2$

где X - переменная, а C₁ и C₂ - такие последовательности символов, скобок и переменных, что C₁C₂ - выражение. Тогда ее можно заменить на перестройку

$[] : DC_1C_2$

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 V B L ⇒ #FAIL
 [] : D &VX L ⇒ #FAIL
 [] : D E B L ⇒ EB : [] , [] : D L

[] : D &EX L ⇒ [] : &EX, [] : D L

П У С Т О Й П О Л У Ч А Т Е Л Ь .

M R : D [] ⇒ #FAIL

V A R : D [] ⇒ #FAIL

E A R : D [] ⇒ E A : [], R : D []

П О В Т О Р Н Ы Е V E-П Е Р Е М Е Н Н Ы Е .

Пусть UA и UB - произвольные различные VE-переменные.

E A R : D L ⇒

#GATE #ALT

#PATH EA : [] #FENCE R : D L

#PATH EA : &VI #FENCE VI R : D L

#END

R E A : D L ⇒

#GATE #ALT

#PATH EA : [] #FENCE R : D L

#PATH EA : &VI #FENCE R VI : D L

#END

V A R : D N L ⇒ V A : &WI &E2 , WI E2 R : D N L

R V A : D L N ⇒ V A : &E2 &WI , R E2 WI : D L N

M R : D E B L ⇒

#GATE #ALT

#PATH EB : [] #FENCE M R : D L

#PATH EB : &VI #FENCE M R : D VI L

#END

```

R M : D L E B  =>
    #GATE #ALT
    #PATH E B : [ ] #FENCE R M : D L
    #PATH E B : &VI #FENCE M R : D L VI
    #END

```

```

M R : D V B L  =>  V B : D M &E I , R : D E I L

```

```

R M : D L V B  =>  V B : D &E I M , R : D L E I

```

```

U A R : D U B L  =>
    #GATE #ALT
    #PATH U A : U B &E I #FENCE E I R : D L
    #PATH U B : U A &E I #FENCE R : D E I L
    #END

```

```

R U A : D L U B  =>
    #GATE #ALT
    #PATH U A : &E I U B #FENCE R &E I : D L
    #PATH U B : &E I U A #FENCE R : D L E I
    #END

```

О Т К Р Ы Т Ы Е V E-П Е Р Е М Е Н Н Ы Е.

Пусть R есть $T_1 T_2 \dots T_N$, где T_i - терм. Тогда

```

R : #L &U X L  =>
    #ALT
    #PATH A 0
    #PATH B 0
    #PATH A i
    ...
    #PATH B N
    #PATH A N
    #END

```

где A_i есть #FAIL, если $T_{[i-1]}$ или $T_{[i+1]}$ являются VE-переменными. Иначе A_i есть

$$T_1 T_2 \dots T_i : \&UX, T[i+1] \dots T_N : \#L L$$

а V_i есть $\#FAIL$, если T_i не является VE -переменной. Если же T_i является VE -переменной, то V_i есть

$$T_i : \#L \&E1 \&E2, T_1 T_2 \dots T[i-1] E1 : \&UX, \\ E2 T[i+1] \dots T_N : \#L L$$

Для отождествления справа налево имеем аналогичное правило.

$$R : \#R L \&UX \Rightarrow \\ \#ALT \\ \#PATH A\emptyset \\ \#PATH B\emptyset \\ \#PATH A1 \\ \dots \\ \#PATH BN \\ \#PATH AN \\ \#END$$

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

$$T_i \dots T_2 T_1 : \&UX, T_N \dots T[i+1] : \#R L$$

а V_i есть $\#FAIL$, если T_i не является VE -переменной. Если же T_i является VE -переменной, то V_i есть

$$T_i : \#R \&E2 \&E1, E1 T[i-1] \dots T_2 T_1 : \&UX, \\ T_N \dots T[i+1] E2 : \#R L$$

ПОЛУЧАТЕЛЬ - V-ПЕРЕМЕННАЯ.

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

ПРОТАЛКИВАНИЕ ПРИСВАИВАНИЙ.

```
E : &X , R : DL =>
    SUBST(X,E,R) : D SUBST(X,E,L) , E : &X
```

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

УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ.

```
#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

```

ЗАКЛЮЧЕНИЕ

В тех случаях, когда программа на Рефале-4 удовлетворяет 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.

[БЭР 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.

Объем 0,9 уч.-изд.л. Цена 10 коп.

Отпечатано на ротавриантах в Институте прикладной математики АН СССР

Москва, Мясницкая пл. 4.

055 (02)2



Все авторские права на настоящее издание принадлежат Институту прикладной математики им. М.В. Келдыша АН СССР.

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.

Publication and distribution rights for this preprint are reserved by the Keldysh Institute of Applied Mathematics, the USSR Academy of Sciences.

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.

Address: USSR, 125047, Moscow A-47, Miusskaya Sq.4, the Keldysh Institute of Applied Mathematics, Ac.of Sc., the USSR, Information Bureau.

Цена 10 коп.