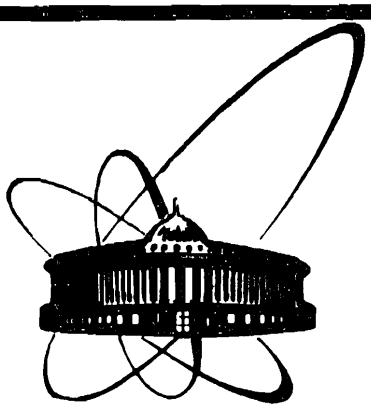

объединенный
институт
ядерных
исследований
дубна

Р11-87-302

А.П.Крюков*, А.Я.Родионов*, В.А.Ростовцев

КОМПИЛЯЦИЯ ОБРАЗЦОВ В СИСТЕМЕ REDUCE

Направлено в Оргкомитет Европейской
конференции по компьютерной алгебре
"ЕВРОКАЛ - 87", Лейпциг, ГДР, июнь 1987 г.

*Научно-исследовательский институт
ядерной физики МГУ, Москва

1987

I. ВВЕДЕНИЕ

Средства работы пользователя с правилами переписывания в том или ином виде реализованы в большинстве систем аналитических вычислений. Типичным примером являются описания подстановок в системе REDUCE [1]. Хотя последняя и не является системой, полностью основанной на правилах переписывания, они составляют ее важную часть. Дело в том, что при помощи правил переписывания удается задавать алгоритм упрощения алгебраических выражений в краткой и выразительной форме, а в некоторых случаях эти правила представляют собой практически единственный способ описания свойств различных математических объектов. В качестве примера широкого использования правил переписывания в системе REDUCE можно привести программу Шварца [2] для нахождения симметрий Ли дифференциальных уравнений.

Правила переписывания естественным образом находят широкое применение в прикладных программах. Однако следствием этого оказывается затрата значительной части процессорного времени на интерпретацию таких правил при решении конкретных задач. Поэтому актуальным является вопрос эффективной обработки правил переписывания в системах аналитических вычислений. Главный способ повышения скорости обработки правил переписывания состоит в переходе от интерпретации образцов к их компиляции.

При разработке компиляторов образцов наряду с классическим подходом [3] применяется также техника частичных вычислений [4]. В настоящей работе мы рассмотрим оба подхода и сравним некоторые результаты, полученные при их реализации в системе REDUCE.

2. Правила переписывания

При определении основных понятий мы следуем работам [5,6].

Множество правил переписывания есть конечное множество

$$R = \{ \rho_1, \dots \rho_n \} \qquad (1)$$

троек $\rho_i = \{ \ell_i, r_i, c_i \}$, где ℓ_i – левая часть правила, r_i – его правая часть и c_i – предикат, определяющий условие применимости данного правила. Мы будем записывать правила переписывания в виде

$$\rho_i = \ell_i \xrightarrow{c_i} r_i . \qquad (2)$$

Множество правил переписывания R действует на множестве термов

$$T = \{ t_1, \dots, t_k, \dots \} \qquad (3)$$

следующим образом.

Пусть $\tilde{t}$ есть либо терм t, либо его подтерм, и пусть левая часть ℓ правила ρ содержит переменные $v_1, \dots, v_m$, которые будем называть переменными образца. Подстановку терма t_j вместо переменной v_j обозначим

$$\sigma = \{ v_1/t_1, \dots, v_m/t_m \} . \qquad (4)$$

Тогда, если

$$\sigma(\ell) = \tilde{t} , \qquad (5)$$

то каждое вхождение $\tilde{t}$ в t должно быть заменено на $\sigma(r)$ при условии, что значение $\sigma(c)$ будет истинным.

Назовем эту операцию сопоставлением с образцом.

Как обычно, область определения подстановок σ расширяется на все множество термов:

$$\sigma(f(t_1, \dots, t_k)) = f(\sigma(t_1), \dots, \sigma(t_k)) . \qquad (6)$$

В системе REDUCE правила переписывания ρ_i задаются с помощью предложений LET. Например,

```
FOR ALL N,X SUCH THAT NUMBERP(N) AND N > 0
   LET SIN(N*X)=SIN(X)*COS((N-1)*X)+COS(X)*SIN((N-1)*X),
       COS(N*X)=COS(X)*COS((N-1)*X)-SIN(X)*SIN((N-1)*X);
```
(7)

где знак равенства "=" используется для разделения левых и правых частей правил, N, X – переменные образца, NUMBERP(N) AND N > 0 есть условие, которому должны удовлетворять оба правила.

Пусть результатом функции $\Theta(t, \ell)$ будет подстановка $\sigma = \{ v_1/t_1, \dots, v_m/t_m \}$, такая, что все переменные $v_1, \dots, v_m$ образца ℓ будут связаны с соответствующими подтермами из t. Тогда операцию сопоставления с образцом функционально можно определить следующим образом:

$$m(t) = \lambda t.(\lambda\sigma.\text{if } \sigma(c) \text{ then} \{\sigma(\ell)/\sigma(r)\} \ t \text{ else } t)(\Theta(t, \ell)). \qquad (8)$$

Представление (8) реализует функцию, действие которой эквивалентно действию заданного правила. Процесс построения функции (8) мы и назовем компиляцией образцов.

3. Частичные вычисления

Техника частичных вычислений в системах, основанных на языке Лисп, применяется для редукции сложных функций к более простым при использовании дополнительной информации об обстановке, в которой происходит конкретный вызов данной функции. Рассмотрим простой пример.

Пусть имеется функция **f(x,key)**, в теле которой содержится некоторое условие

```
.........
if key = 1 then g(x) else h(x);          (9)
.........
```

Если в момент вызова функции **f(x,key)** обстановка такова, что переменная **key** всегда связана со значением 1, то тело нашей функции может быть редуцировано к следующему виду:

```
.........
g(x) ;                                   (10)
.........
```

При этом можно вместо вызова функции **g(x)** подставить ее тело, настроенное соответствующим образом, и повторить процесс редукции. Такая обработка определений функций называется частичным вычислением.

Пусть **seval(x,env)** - функция, осуществляющая частичное вычисление выражения **x** в обстановке **env**. Легко видеть, что, применяя ее к функции (8):

$$m_i(t)=\text{seval}(\lambda t.(\lambda\sigma.\text{if } \sigma(c)\text{ then } \{\sigma(\ell)/\sigma(r)\}\ t \text{ else } t)(\Theta(t,\ell)),\rho_i), \qquad (11)$$

мы выполним редукцию последней к функции $m_i(t)$, которая настроена на сопоставление с конкретным образцом ℓ_i из правила ρ_i. При этом ρ_i может, конечно, иметь параметрическую свободу.

Например, если в качестве правила ρ при редукции выражения (9) взять правило для **SIN(N*X)** из (7), то полученная функция **m(t)** будет настроена на сопоставление с образцом, содержащим функцию SIN, хотя параметрическая свобода от N и X будет сохранена. Если вместо (7) задать правило

```
FOR ALL N SUCH THAT NUMBERP(N) AND N > Ø          (7')
   LET SIN(N*X)=SIN(X)*COS((N-1)*X)+COS(X)*SIN((N-1)*X);
```

то аналогичная редукция приведет к еще более простой функции, настроенной на работу с образцами, содержащими не просто функцию SIN, а функцию SIN от произведения некоторого числа N на переменную X. При этом X не будет переменной образца.

4. Прямая компиляция образцов

К сожалению, подход, основанный на технике частичных вычислений, обладает рядом недостатков. Прежде всего, в этом случае сама функция $\Theta(t,\ell)$ и связанные с ней функции не могут быть скомпилированы обычным Лисп-компилятором. Это связано с тем, что функция **seval** работает непосредственно с телом функции $\Theta(t,\ell)$. Следовательно, если даже в процессе работы мы откажемся от компиляции образцов, мы все же вынуждены работать с функцией $\Theta(t,\ell)$ в режиме интерпретации. Ниже мы остановимся и на других недостатках этого подхода. Поэтому возникает вопрос о реализации прямого компилятора образцов.

Под прямым компилятором образцов мы понимаем такую функцию cpat(ρ), которая строит функцию $m_\rho(t)$, аналогичную (11), без обращения к **seval**. Такой подход является более трудоемким, но, как будет показано ниже, позволяет получить более эффективные коды функций.

5. Реализация в системе REDUCE

Нами были реализованы оба подхода и проведено их сравнительное исследование. При этом выяснился ряд особенностей системы REDUCE, которые, по-видимому, не позволяют надеяться на увеличение скорости обработки подстановок в сотни раз, как это указывается в работах [3,7].

Основная причина этого состоит в том, что "накладные расходы" при обработке алгебраических выражений и применении к ним правил переписывания в системе REDUCE весьма велики. В некоторых случаях эти расходы полностью перекрывают выигрыш от компиляции образцов.

Мы ставили перед собой цель исследовать возможность применения компиляции образцов в системе REDUCE, поэтому мы ограничивались довольно узким подмножеством всех допустимых в этой ситуации подстановок. А именно: обе реализации допускают в настоящее время компиляцию образцов следующего вида:

$$f(x_1,\ldots,x_n)\xrightarrow{c(x_1,\ldots,x_k)} e\ , \qquad (12)$$

где e - произвольное выражение, а x_i могут быть либо переменными, либо некоторыми функциями $y(z_1,\ldots,z_m)$ и так далее. Например,

```
FOR ALL X,Y,Z LET FF(X,GG(Y,Z)) = EXPR .
```

При этом все функции, входящие в левую часть правила, не должны обладать свойствами симметрии, не заданными явно через соответствующие подстановки.

Остановимся на особенностях реализации описанных выше двух подходов к компиляции образцов в системе REDUCE.

5.1. Техника частичных вычислений

Пакет программ, основанный на технике частичных вычислений, в значительной степени повторяет работу[4]. Отличия связаны как с учетом конкретных целей использования этой техники, так и с наложением ряда ограничений для упрощения программы.

Важной чертой реализации I является обработка условий c_i, накладываемых на правила. Так как весьма важно как можно раньше убедиться в их справедливости, то мы специально следим за теми переменными образца ℓ_i, которые входят в условие c_i. Как только все они будут отождествлены с некоторыми значениями, происходит немедленное вычисление этих условий. Для этого в соответствующие места генерируемой функции вставляются необходимые коды. Наибольший выигрыш это дает в случае, когда число переменных образца значительно превышает число переменных, входящих в соответствующее условие.

5.2. Прямая компиляция

Наиболее важные отличия этого варианта от реализации I состоят в следующем:

а) коды, генерируемые данным компилятором, имеют структуру конструкции PROG, а не чисто рекурсивную;

б) в реализации I обрабатываются только левые части правил ρ_i и условия c_i, а правые части r_i правил обрабатываются стандартным образом. В реализации II компилятор образцов обрабатывает все правило полностью, в соответствии с формулой (8). Это дает существенный выигрыш в тех случаях, когда условия c_i задаются в правой части правила ρ_i.

Поясним это на примере симметричных функций. Пусть функция $ff(x_1,x_2,x_3,x_4,x_5,x_6)$ симметрична по своим аргументам. Мы ограничимся случаем, когда x_i принимают числовые значения. Заметим, что работать с такой функцией в системе REDUCE, просто объявив ее симметричной, нельзя, так как у нее больше пяти аргументов. Однако мы можем задать свойства симметрии, используя следующий ряд правил:

```
FOR ALL A,B,C,D,X,Y SUCH THAT X < Y
   LET FF(X,Y,A,B,C,D) = FF(Y,X,A,B,C,D) ,
       FF(A,X,Y,B,C,D) = FF(A,Y,X,B,C,D) ,
       .......
       FF(A,B,C,D,X,Y) = FF(A,B,C,D,Y,X) .
```
(I3)

Таким образом, всякое выражение $FF(I_1,\ldots,I_6)$ система приведет к виду $FF(I_{j1},\ldots,I_{j6})$, где $I_{jk} > I_{jk+1}$. Нам пришлось задать пять отдельных правил. Следовательно, при обработке исходного выражения иногда будет необходимо сопоставить до пяти левых частей правил (I3). Если же аргументы уже упорядочены, сопоставление всегда будет выполняться 5 раз. Чтобы этого не происходило, модифицируем правила (I3) следующим образом:

```
FOR ALL A,B,C,D,E,F LET
    FF(A,B,C,D,E,F) → FF(B,A,C,D,E,F) WHEN A < B
                    → FF(A,C,B,D,E,F) WHEN B < C
                    → FF(A,B,D,C,E,F) WHEN C < D
                    → FF(A,B,C,E,D,F) WHEN D < E
                    → FF(A,B,C,D,F,E) WHEN E < F
                    → FAIL.
```
(I4)

Здесь использован новый синтаксис задания подстановок, который свободен от указанного недостатка, так как позволяет связать несколько правых частей с одной левой частью. Введены два изменения: во-первых, условия, которые задаются в виде конструкций SUCH THAT, перенесены в правые части и отделяются зарезервированным идентификатором WHEN, и, во-вторых, введен новый оператор FAIL, определяющий, что сопоставление не удалось и, следовательно, выражение не подлежит дальнейшему преобразованию.

Форма (I4) хотя и похожа на конструкцию IF-THEN-ELSE, отличается от нее большей наглядностью и наличием оператора FAIL. Если вместо (I4) использовать обычную для системы REDUCE форму

```
FOR ALL A,B,C,D,E,F LET
   FF(A,B,C,D,E,F)=IF A < B THEN FF(B,A,C,D,E,F) ELSE
                   IF B < C THEN FF(A,C,B,D,E,F) ELSE
                   ........
                   IF E < F THEN FF(A,B,C,D,F,E) ,
```
(I5)

то результат упрощения выражения $FF(I_1,\ldots,I_6)$ всегда будет равен нулю, так как NIL в алгебраической моде системы REDUCE имеет значение 0.

6. Тестовые примеры

Для тестирования средств компиляции правил переписывания, добавленных в систему REDUCE, мы использовали правила подстановок (I3), (I4) для функции от шести аргументов и подстановку

```
FOR ALL K,N SUCH THAN K > Ø LET
                EE(K,N) = EE(K-1,N+1).
```
(I6)

Результаты тестирования приведены в таблицах I и 2.

Кроме того, в таблице 3 приводится результат выполнения авторского теста программы Шварца LIE4 при компиляции подстановок.

Все тесты выполнялись на ЭВМ ЕС-1045. Под систему REDUCE выделялся раздел оперативной памяти размером 700 Кбайт. Время приводится в секундах, время сборки мусора не учитывалось.

Таблица 1

Тест	Правила подстановок	Вариант компиляции правил	Время выполнения 100 вызовов, с				$\frac{\Delta_2}{\Delta_1}$
			t_1	$\Delta_1=t_1-t'$ (**)	t_2	$\Delta_2=t_2-t'$	
FF(6,5,4,3,2,1)	I3	0 (*)	2,6	I,9			
-"-	I3	I			2,4	I,7	0,89
-"-	I4	0	I,9	I,2			
-"-	I4	I			I,2	0,5	0,42
-"-	I4	II			0,85	0,I5	0,I25
FF(1,2,3,4,5,6)	I3	0	I8,7	I8,0			
-"-	I3	I			8,0	7,3	0,4I
-"-	I4	0	I7,2	I6,5			
-"-	I4	I			6,7	6,0	0,36
-"-	I4	II			3,9	3,2	0,I9

(*) - правила не компилируются,

(**) - $t' = 0,7$ с - накладные расходы.

Таблица 2

Тест	Вариант компиляции правил	Время выполнения 20 вызовов, с				$\frac{\Delta_2}{\Delta_1}$
		t_1	$\Delta_1=t_1-t'$ (**)	t_2	$\Delta_2=t_2-t'$	
EE(100,0)	0 (*)	I9,3	I9,2			
-"-	I			I6,9	I6,8	0,88
-"-	II			I7,2	I7,I	0,89

(*) - правила не компилируются

(**) - $t' = 0,I$ с - накладные расходы.

Таблица 3

	Время счета, с	Время компиляции, с	Объем кодов, слов
Правила не компилируются	65,2		
Правила компилируются	36,4	~40	~700

7. Заключение

Полученные результаты показывают, что компиляция подстановок дает в системе REDUCE выигрыш по времени обработки функций, не требующих дополнительных алгебраических преобразований, до 5-6 раз. Для выражений типа (I6) выигрыш оказывается значительно более скромным и составляет I0-I5%, что связано с большими затратами на непосредственно алгебраические преобразования. Следует также отметить, что использование нового синтаксиса правил подстановок и введение оператора FAIL позволили в случае правил (I4) получить даже без компиляции выигрыш на I0-40%, в зависимости от начального упорядочения.

По нашему мнению, введение в систему REDUCE компиляции подстановок и синтаксиса для них, подобно приведенному в (I4), вполне оправдано. Описание решения задачи на основе подстановок является более естественным, чем при помощи процедур. В этом случае пользователь сообщает системе, что надо сделать, а процесс выполнения берет на себя система. Можно утверждать, что, используя компиляцию правил подстановок, мы не только не проиграем в эффективности (а в ряде случаев существенно выиграем), но и перейдем на более высокий технологический уровень разработки программ.

Авторы благодарны П.М.Петрову за плодотворное обсуждение рассматриваемой в данной работе проблемы.

Литература

1. Hearn A.C. REDUCE User's Manual, Version 3.2, The RAND Corp., Santa Monica, 1985 .

2. Schwarz F. A REDUCE Package for Determining Lie Symmetries of Ordinary and Partial Differential Equations, C.P.C., 1982, v.27, p.179 .

3. Jenks R.D. A Pattern Compiler,Proc.of SYMSAC'76,A.C.M., New York 1976,p.60 .

4. Emanuelson P.,Haraldsson A.On Compiling Embedded Languages in LISP,Proc.of LISP'80 Conf., The LISP Conference,California, 1980,p.208 .

5. Hornfeldt L. A Sum-Substitutor used as Trigonometric Simplifier, Proc.of EUROCAM'82,LNCS v.144,Springer Verlag,1982,p.188 .

6. Musser D.R.,Kapur D.Rewrite Rule Theory and Abstract Data Type Analisys,Proc.of EUROCAM'82,LNCS v.144,Springer Verlag,1982,p.77.

7. Cowen R.M.,Griss M.Hashing - the Key to Rapid Pattern Matching, Proc.of EUROSAM'79,LNCS v.72,Springer Verlag,1979,p.266 .

Рукопись поступила в издательский отдел
29 апреля 1987 года.

Крюков А.П., Родионов А.Я., Ростовцев В.А. Р11-87-302

Компиляция образцов в системе REDUCE

Рассматриваются два подхода к реализации компилятора образцов, а именно метод частичных вычислений и прямая компиляция. Результаты, полученные при реализации этих подходов в системе REDUCE, показывают, что в некоторых случаях компиляция подстановок дает выигрыш во времени обработки функций до 5-6 раз.

Работа выполнена в Лаборатории вычислительной техники и автоматизации ОИЯИ.

Препринт Объединенного института ядерных исследований. Дубна 1987

Перевод авторов

Kryukov A.P., Rodionov A.Ya., Rostovtsev V.A. P11-87-302

Pattern Compilation in REDUCE

Two approaches to the pattern compiler implementation are threated. They are the partial compilation technique and the direct compilation. The results obtained upon their implementation in the REDUCE system show that the compilation of substitutions provides the gain in processing time of functions by 5-6 tymes in some cases.

The investigation has been performed at the Laboratory of Computing Techniques and Automation, JINR.

Preprint of the Joint Institute for Nuclear Research. Dubna 1987