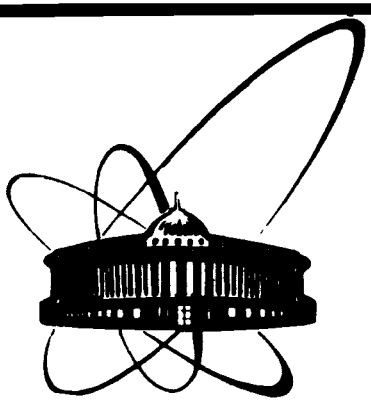

объединенный
институт
ядерных
исследований
дубна

К 356

Р11-88-402

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

ПРОГРАММНЫЕ СРЕДСТВА КОМПИЛИРОВАНИЯ ПРАВИЛ ПОДСТАНОВОК В СИСТЕМЕ REDUCE

Направлено на Международный симпозиум
по символьным и алгебраическим вычислениям
/4-8 июля 1988 г., Рим/

* НИИЯФ МГУ, Москва

1988

Введение

Сравнение с образцом является одним из важнейших этапов преобразования алгебраических выражений в современных системах аналитических вычислений на ЭВМ. Именно этот механизм делает системы гибкими, позволяя пользователю определять новые математические объекты и правила их обработки. В системе REDUCE [1] эти правила задаются с помощью специальной языковой конструкции /так называемых LET-предложений/, которая имеет следующий вид:

```
FOR ALL <список идентификаторов> SUCH THAT <услов>
 LET <образец> = <замена>;
```

Например, функцию "Факториал" можно определить следующим образом:

```
FOR ALL N SUCH THAT NUMBERP(N) AND N > 0
  LET FAC(N) = N*FAC(N-1);
FAC(0) := 1;
```

Подробно синтаксис системы REDUCE описан в /1-4/.

Чем сложнее алгоритм вычислений, тем большую роль в них играют правила подстановок. Поэтому совершенствование аппарата сопоставления с образцом имеет важное значение для развития систем аналитических вычислений. Так, введение в CAB REDUCE описанного в [6] нового типа подстановок для операторов с

Объединенный институт ядерных исследований БИБЛИОТЕКА

нефиксированным числом аргументов позволило легко реализовать алгоритм Цвитановича [7] вычисления группового веса диаграмм Фейнмана в неабелевых теориях поля. Настоящая работа посвящена вопросам повышения быстродействия аппарата подстановок системы REDUCE и основана на идеологии компиляции образцов. Теоретические основы этой работы и время выполнения характерных тестовых примеров рассмотрены в работе [8]. Ниже будут описаны реализация и способы применения компиляции образцов в системе REDUCE.

I. Сопоставление с образцом в CAB REDUCE

Обычно правила подстановок хранятся в системе REDUCE как значение свойства OPMTCH соответствующего оператора в виде списка, каждый элемент которого состоит из трех частей:

(<образец> <логическое условие> <подстановка>)

При обработке каждого оператора вызывается специальная функция OPMTCH, которая проверяет наличие данного свойства. Если такого свойства нет, то функция OPMTCH возвращает значение NIL, в противном случае обработка продолжается при помощи функции MCHARG. Эта функция имеет три аргумента: текущий список аргументов оператора, образец и имя оператора. Основная цель компиляции образцов в системе REDUCE заключается в том, чтобы вместо образца хранить функцию, осуществляющую проверку на сопоставимость именно с этим образцом. Эта функция имеет только один аргумент – текущий список аргументов оператора, и ее значение на данном образце совпадает со значением функции MCHARG:

$$[P,F](X) = \mathrm{MCHARG}(X,P,F)$$

Здесь P – образец, F – имя оператора, X – текущий список аргументов, $[P,F]$ – функция сопоставления с образцом P для оператора F. Тогда вместо функции MCHARG можно при обработке операторов вызывать скомпилированные функции образцов. Такой механизм имеет ряд преимуществ над традиционным. Уменьша-

ется количество вызовов функций, т.к. функции $[P,F]$ имеют полностью замкнутую структуру и не содержат обращения к другим функциям, в то время как функция MCHARG для проверки совпадения с образцом может обращаться, в зависимости от типа обрабатываемых операторов, к функциям MCHARG2, MTP, MCHSARG, MCHCOMB, MKBIN, каждая из которых также содержит новые вызовы. При компиляции возможна оптимизация получаемых функций сравнения с образцом. Наиболее существенной является возможность проверки логического условия сразу после того, как определены все входящие в него свободные переменные. Дальнейшая компиляция полученной функции образца при помощи компилятора системы REDUCE [9] позволяет использовать эффективные двоичные коды, создаваемые этим компилятором. Кроме того, нами реализован механизм создания файлов быстрой загрузки для скомпилированных правил подстановок. Этот механизм может быть особенно полезным при создании баз знаний системы REDUCE [10, 11] и создании пользовательских библиотек программ, написанных на алгебраическом уровне языка системы и использующих большое число LET-предложений.

2. Программные средства компиляции подстановок в системе REDUCE

Комплекс программ для компиляции подстановок в системе REDUCE хранится в трех файлах. В первом, с именем SEMIEVAL, содержится пакет программ для полувычисления функций. Во втором, SEMOPMT, – программы, переопределяющие ряд системных функций сопоставления с образцом и ряд дополнительных функций. В третьем файле, SEMTEST5, содержится чисто функциональный вариант программы сопоставления с образцом системы REDUCE. Первые два файла можно хранить в компилированном виде, годном для быстрой загрузки. Третий файл всегда используется в текстовом виде. Компиляция образцов происходит следующим образом. На вход программы полувычислений подается головная процедура функционального пакета сопоставления с образцом. На выходе получается процедура от одного аргумента, осуществляющая сравнение с этим образцом, и в тех случаях, когда оно имеется, – проверку выполнения логического условия. Имя созданной процедуры становится значением свойства OPMTCH вместо образца.

Пусть P,Q,F – операторы с заданными правилами подстановок. Для компиляции этих правил нужно загрузить файлы SEMIEVAL и SEMOPMT и ввести файл SEMTEST5 (это можно сделать как до, так и после определения правил подстановок). После этого можно использовать команду COMPLET . Ее формат

COMPLET <список имен функций> ;

Например:

```
COMPLET P,Q,F;
```

При этом компилируются все определенные к этому моменту правила подстановок для операторов P,Q,F . В частности, можно компилировать и правила дифференцирования. Для этого в качестве одного из аргументов команды COMPLET следует указать оператор дифференцирования DF . Компиляции не подлежат подстановки на симметричные и антисимметричные операторы, а также подстановки для операторов с нефиксированным числом аргументов. При попытке их компиляции выдается сообщение об ошибке:

"CANNOT BE COMPILED".

Следует заметить, что если симметричные или антисимметричные операторы встречаются в подстановке не на верхнем уровне, т.е. в качестве аргументов других операторов, то такая подстановка может быть скомпилирована. Запрет на компиляцию симметричных и антисимметричных функций введен вследствие того, что объем кодов, получаемых при их компиляции, получается слишком большим. Существует отличие в обработке правил подстановок в случаях компиляции и интерпретации для операторов, аргументы которых представляют собой выражения, содержащие арифметические операции: сложение и умножение. При компиляции эти операторы рассматриваются только как симметричные функции своих аргументов.

Приведем пример. Пусть задана подстановка

```
FOR ALL X,Y LET P(X*Y) = Q(Y);
```

Тогда выражение P (A * B * C) будет заменено в системе REDUCE на Q(B*C) . После компиляции этого правила такой замены не произойдет. Однако P (A * B) будет заменено на Q(B) в обоих случаях.

Для контроля за процессом компиляции подстановок предусмотрен переключатель SEMIPR /нормально он выключен/. Если переключатель SEMIPR включить, то происходит печать тела полученной после компиляции функции. Например, пусть задана подстановка на оператор QQ :

```
FOR ALL X,Y SUCH THAT X > Y
  LET QQ(X,Y,Ø) = QQ(Y,X);
```

Тогда при включенном переключателе SEMIPR после команды COMPLET на дисплей будет выведен следующий текст:

```
COMPLET QQ;
  LAMBDA L;
    IF NULL THEN NIL;
     ELSE IF NULL CDR L THEN NIL
     ELSE IF EVALGREATERP(AEVAL CAR L,AEVAL CADR L)
      THEN IF NULL CDDR L THEN NIL
           ELSE IF CADDR L = Ø
            THEN IF CDDDR L THEN NIL
                 ELSE LIST('QQ,CADR L,CAR L);
```

Печать производится либо стандартными средствами системы REDUCE (PRETTYPRINTER), либо при помощи антитранслятора с языка RLISP [I2].

Для эффективной работы механизма подстановок полученные после компиляции функции должны быть скомпилированы при помощи стандартного компилятора системы REDUCE [9] . Это происходит автоматически при включенном переключателе COMP. Приведенные ниже промежуточные машинно-независимые коды демонстрируют результат двойной компиляции приведенного выше примера.

```
(]*ENTRY QQ] ] ] GØ133 EXPR 1)
(]*ALLOC 2)
(]*JUMPNIL ] ] ] GØ279)
(]*STORE 1 Ø)
(]*LOAD 1 (CDR 1))
(]*JUMPNIL ] ] ] GØ279)
(]*LOAD 1 (CAR Ø))
```

```
(]*LINK AEVAL EXPR 1)
(]*STORE 1 -1)
(]*LOAD 1 (CAR (CDR Ø)))
(]*LINK AEVAL EXPR 1)
(]*LOAD 2 1)
(]*LOAD 1 -1)
(]*LINK EVALGREATERP EXPR 2)
(]*JUMPNIL ] ] ] GØ279)
(]*LOAD 1 (CDR (CDR Ø)))
(]*JUMPNIL ] ] ] GØ279)
(]*LOAD 2 (QUOTE Ø))
(]*LOAD 1 (CAR 1))
(]*LINK EQUAL EXPR 2)
(]*JUMPNIL ] ] ] GØ279)
(]*LOAD 1 (CDR (CDR (CDR Ø))))
(]*JUMPNIL ] ] ] GØ276)
(]*LOAD 1 (QUOTE NIL))
( *JUMP ] ] ] GØ279)
( *LBL ] ] ] GØ276)
( *LOAD 3 (CAR Ø))
( *LOAD 2 (CAR (CDR Ø))
( *LOAD 1 (QUOTE QQ))
( *LINK LIST3 EXPR 3)
( *DEALEXT 2)
```

Выше уже отмечалась важность создания файлов быстрой загрузки для скомпилированных правил подстановок. Создание файлов быстрой загрузки происходит в два этапа. На первом этапе создается промежуточный файл, содержащий программу на языке LISP , состоящую из текстов функций для правил подстановок и некоторых дополнительных команд. Для этого следует включить переключатель SFAP /по умолчанию он выключен/, переключить вывод в файл и воспользоваться командой COMPLET . Например

```
ON SFAP;
OUT OUTF;
COMPLET QQ;
OUT T;
SHUT OUTF;
```

Следующим шагом является создание файла быстрой загрузки (FAP-файла). Для этого следует использовать стандартные средства системы REDUCE (команда FASLOUT):

```
FASLOUT QQRULE;
IN OUTF;
FASLEND;
```

После выполнения этих команд в файле QQRULE будет записана готовая к загрузке командой LOAD двоичная версия программы.

3. Использование пакета полувычислений в системе REDUCE

В описываемой реализации компиляция правил подстановок производится путем полувычисления функции сопоставления с образцом при некотором конкретном значении образца. Пакет SEMIEVAL,

реализующий полувычисления, может быть использован и независимо. Для полувычисления функций существуют процедуры SPUTD и EXSEVAL.Первая из них служит для определения новой функции, получаемой из заданной путем ее полувычисления. Процедура SPUTD имеет три аргумента:

```
SPUTD(NF,FN,FAL);
```

где NF - имя создаваемой функции(результата), FN- имя полувычисляемой(исходной) функции, FAL- ассоциативный список:

```
( ... (AN . AV) ... )
```

где AN - имя аргумента, AV - его значение; аргументы, не вошедшие в этот список, считаются неопределенными. Значением функции SPUTD является имя создаваемой функции. Текст, полученный в результате полувычисления функции, может быть выведен на дисплей включением переключателя SEMIPR . Как и в случае компиляции образцов, переключатель SFAP используется для создания файлов, содержащих тексты, получаемые после полувычисления функций.

Значением процедуры EXSEVAL является тело функции после ее полувычисления. Она имеет два аргумента.

```
EXSEVAL(FN,FAL);
```

Назначение и формат этих аргументов те же, что и назначение и формат аргументов FN и FAL процедуры SPUTD . В качестве примера приведем полувычисление функций сопоставления с образцом, предложенных в [I3] , текст которых приведен в приложении.

```
SPUTD('F1,'PATMATCH,'((PAT A ]& D ]-)));

    LAMBDA L;
       IF NULL L THEN NIL
        ELSE IF 'A=CAR L
         THEN IF NULL CDR L THEN NIL
                 ELSE IF NULL CDDR L THEN NIL
                 ELSE IF 'D=CADDR L THEN T
                 ELSE NIL
        ELSE NIL;
F1
```

Приведем также временные характеристики получаемых в процессе полувычисления программ. Возможны четыре различных случая:

I) используется непосредственно приведенный в приложении пакет сопоставления с образцом; 2/ используется его скомпилированный LISP-компилятором [9] вариант; 3) используются полученные после полувычисления функции сопоставления с образцом; 4) используются эти же функции, но после их дополнительной компиляции LISP-компилятором. Результаты вычислений приведены в таблице. Из этой таблицы видно, что в случае интерпретации полувычисленные функции производят сопоставление с образцом примерно в два раза быстрее исходного пакета. После компиляции соотношение времен выполнения тестовых примеров практически сохраняется.

Отметим, что для того, чтобы вместо вызова функции при полувычислении было подставлено ее полувычисленное тело, для нее нужно определить свойство SEMIEVAL со значением SEMIEXPR, как это сделано в рассматриваемом примере для функции PATMATCH. Свойство SEMIEVAL определяет имя функции, осуществляющей полувычисления. SEMIEXPR - имя функции, предназначенной для полувычисления функций типа EXPR. В целях оптимизации или при необходимости обрабатывать функции других типов пользователь может определить специальную функцию для полувычислений. Ее имя необходимо задать как значение свойства SEMIEVAL функции, подлежащей специальной обработке. Эта функция должна иметь четыре аргумента. Первый аргумент - имя вызываемой функции; второй аргумент - список ее текущих аргументов; третий аргумент - список, содержащий точечные пары (<переменная> . <значение>); Четвертый аргумент - список переменных для данного шага полувычислений. Например, приведем специальный обработчик для функции NULL.

```
SYMBOLIC PROCEDURE SEMINULL(FN,ARG,AL,FV);
  (LAMBDA Z;
     IF NULL CAR Z                          %ARGUMENTS RESOLVE
       THEN IF (CADR Z EQ 'CONS OR CADR Z EQ 'QUOTE OR
                CADR Z EQ 'LIST) THEN T . NIL
            ELSE IF (CADR Z EQ 'NOT OR CADR Z EQ 'NULL)
                   THEN SEMEVAL(CADDR Z,AL,FV)
                 ELSE SEMIAPPLY(FN,CAR Z . (CDR Z . NIL))
       ELSE SEMIAPPLY(FN,CAR Z . (CDR Z . NIL)))
   SEMEVAL(CAR ARG,AL,FV);

PUT('NOT,'SEMIEVAL,'SEMINULL);

PUT('NULL,'SEMIEVAL,'SEMINULL);
```


Значением таких функций должны быть точечная пара (BOOL . VAL), где BOOL имеет значение T , если выражение полностью вычислено, и NIL в противном случае, VAL - значение вычисляемого выражения (оно, разумеется, может быть вычислено не до конца). Функция SEMIAPPLY аналогична функции APPLY .

Следует иметь в виду, что в процессе полувычислений функции (и, следовательно, при компиляции правил подстановок) в силу рекурсивного характера программ пакета полувычислений происходит интенсивное использование стека, что может привести к прерыванию работы программы с диагностикой

PUSHDOWN STAK OVERFLOW

В этом случае следует увеличить объем стека (это можно сделать командой CONDENSE) и провести вычисления заново. Рекомендуемый размер стека - I2000 слов.

Заключение

Развитие систем аналитических вычислений и, в частности, REDUCE, по-видимому, вступило в период создания больших библиотек прикладных программ. Программы эти часто пишут пользователи, не являющиеся специалистами по компьютерной алгебре, поэтому черезвычайно актуальным становится вопрос программной поддержки процесса создания, хранения и использования программ, написанных на алгебраическом уровне языка REDUCE. Часть проблем, таких, как создание файлов быстрой загрузки [I] , автоматическая подкачка процедур [I4] , использование виртуальной памяти для лисповских функций [I5] , создание динамического отладчика для системы REDUCE [I6] и системы средств, повышающих эффективность диалоговой работы [I7], уже нашли свое решение. Данная работа посвящена повышению быстродействия при обработке правил подстановок. Правила подстановок часто используются для реализации алгебраических алгоритмов в системе REDUCE . Как было показано в работе [8] на ряде тестовых примеров, компиляция подстановок может привести к значительному уменьшению времени упрощения алгебраических выражений. Следующим важным шагом по увеличению быстродействия программ, написанных на языке REDUCE, явилось бы, по нашему мнению, создание эффективного механизма компиляции алгебраических процедур.

Приложение

Текст программы сопоставления с образцом [I3] .

```
SYMBOLIC  PROCEDURE PATMATCH(PAT,L);
   IF NULL PAT THEN NULL L
    ELSE IF CAR PAT = '1-
           THEN IF CDR PAT THEN SEGMATCH(CDR PAT,L)
                 ELSE T
    ELSE IF NULL L THEN NIL
    ELSE IF CAR PAT = '1& THEN PATMATCH(CDR PAT,CDR L)
    ELSE IF CAR PAT = CAR L THEN PATMATCH(CDR PAT,CDR L)
    ELSE NIL;

SYMBOLIC PROCEDURE SEGMATCH(PAT,L);
   BEGIN SCALAR NEWL;
         NEWL := L;
   LP:
         IF PATMATCH(PAT,NEWL) THEN RETURN T
          ELSE IF NULL NEWL THEN RETURN NIL
          ELSE NEWL := CDR NEWL;
         GO LP;
  END;

PUT('PATMATCH,'SEMIEVAL,'SEMIEXPR);

PUT('SEGMATCH,'SEMIEVAL,'SEMIEXPR);
```

Таблица

Время выполнения в секундах тестовых примеров для пакета сравнения с образцом [I3] и функций, полученных в результате полувычислений (функции сравнения вызывались I000раз). Времена приведены с учетом накладных расходов.

Образец	Входное выражение	Случаи:			
		I	2	3	4
'(A B C & C)	'(A B C C C)	II	0,6	5	0,3
- " -	'(A B C C C C)	I3	0,7	7	0,4
'(A B)	'(A B C C C C)	5,6	0,6	2,5	0,3

Случаи: I. Сравнение выполняется при помощи функции PATMATCH.
2. Сравнение выполняется при помощи компилированной функции PATMATCH.
3. Сравнение выполняется при помощи компилированного образца.
4. Сравнение выполняется при помощи дважды компилированного образца.

Литература

I. Hearn A.C. REDUCE User's Manual. The RAND Publ., 1982, USA.

2. Боголюбская А.А., Жидкова И.Е., Ростовцев В.А.Система программирования REDUCE , ОИЯИ Б1-11-83 - 512 , Дубна, 1983.

3. Еднерал В.Ф., Крюков А.П., Родионов А.Я. Язык аналитических вычислений REDUCE , ч.I. М: МГУ, 1983.

4. Еднерал В.Ф., Крюков А.П., Родионов А.Я. Язык аналитических вычислений REDUCE , ч.2. М: МГУ, 1986.

5. Крюков А.П. Родионов А.Я. COLOR - программа для вычисления группового фактора диаграмм Фейнмана в неабелевых калибровочных теориях. Препринт ИФВЭ, 86-72, Серпухов, 1986.

6. Крюков А.П., Родионов А.Я. Применение системы REDUCE для вычисления группового фактора диаграмм Фейнмана в QCD. В сб.: "Совещание по системам и методам аналитических вычислений на ЭВМ и их применению в теоретической физике, Дубна 17-20 сентября 1986 г., ОИЯИ, Д11-85-791, Дубна, 1986, с.388.

7. Cvitanovic P. Group theory for Feynman diagrams in non-Abelian gauge theories. Phys. Rev. D, 1976, v.D14, p.1536.

8. Крюков А.П., Родионов А.Я., Ростовцев В.А.Компиляция образцов в системе REDUCE . Препринт ОИЯИ, Р11-87-302, Дубна, 1987.

9. Griss M.L., Hearn A.C., A portable LISP compiler. Software-Prac. and Expr., 1982, v.11, p.554.

10. Спиридонова М.Н. Организация и применение баз символьных преобразований на основе системы REDUCE. В сб.: "Совещание по системам и методам аналитических вычислений на ЭВМ и их применению в теоретической физике", Дубна, 17-20 сентября 1986 г., Д11-85-791, Дубна, 1986, с.28.

11. Крюков А.П., Родионов А.Я. Проблема контроля и модификации базы знаний системы аналитических вычислений REDUCE. В сб.: "Теория и практика автоматизированных систем аналитических преобразований". Изд. Института повышения квалификации народного хозяйства ЛитССР, Вильнюс, 1984, с.79.

12. Крюков А.П. Антитранслятор языка RLISP. В сб.: "Совещание по системам и методам аналитических вычислений на ЭВМ и их применению в теоретической физике. Дубна, 21-23 сентября 1982 г.", Д11-83-511, Дубна, 1983, с.191.

13. Emanuelson P., Haraldson A., On compiling embeded languages in LISP. Proc. LISP'80 Conf., The LISP Conf., California, 1980, p.208.

I4. Бутенко В.А., Крюков А.П., Родионов А.Я. Реализация автоматической загрузки процедур в системе аналитических вычислений REDUCE . Программирование, I985, № 4, с. 89.

I5. Ростовцев В.А. Использование вторичной памяти в системах программирования аналитических преобразований. В сб.: "Совещание по системам и методам аналитических вычислений на ЭВМ и их применению в теоретической физике", Дубна, 2I-23 сентября I982 г., ДII-83-5II, Дубна, I983, с.I07.

I6. Kryukov A.P., Rodionov A.Ya. Dynamic debugging system for REDUCE programs. SIGSAM Bull., 1985, v.19, p.34.

I7. Крюков А.П. Опыт развития диалоговых средств системы аналитических вычислений REDUCE. В сб.: "Совещание по системам и методам аналитических вычислений на ЭВМ и их применению в теоретической физике", Дубна, I7-20 сентября I986 г., ОИЯИ, ДII-85-79I, Дубна, I986, с.90.

Рукопись поступила в издательский отдел
6 июня I988 года.

Крюков А.П., Родионов А.Я., Ростовцев В.А. P11-88-402

Программные средства компилирования правил подстановок в системе REDUCE

Описываются новые программные средства, позволяющие производить компиляцию образцов в системе REDUCE /1/ , приводятся руководство по их использованию и примеры работы.

Работа выполнена в Лаборатории вычислительной техники и автоматизации ОИЯИ.

Препринт Объединенного института ядерных исследований. Дубна 1988

Перевод О.С.Виноградовой.

Kryukov A.P., Rodionov A.Ya., Rostovtsev V.A. P11-88-402

New Programming Tools for Computing Substitution Rules in REDUCE System

New programming tools allowing to compile patterns in REDUCE /1/ system are described. A guide for using these tools and examples of their working are presented.

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

Preprint of the Joint Institute for Nuclear Research. Dubna 1988