Госстрой СССР
ЦНИПИАСС

ФОНД АЛГОРИТМОВ
И ПРОГРАММ ДЛЯ ЭВМ
(В ОТРАСЛИ "СТРОИТЕЛЬСТВО")

РЕФАЛ

Базисный РЕФАЛ и его реализация на вычислительных машинах

(методические рекомендации)

СПЕЦИАЛЬНЫЙ РАЗДЕЛ

V
40
МОСКВА 1977

УДК 001(47):681.3.04:800.92

Дается полное описание подмножества универсального алгоритмического языка рекурсивных функций (базисного РЕФАЛа), предназначенного для символьных преобразований, а также приводятся методы его трансляции и особенности реализации на отечественных вычислительных машинах типа: ЕС ЭВМ, БЭСМ-6, "Минск-32", М-220, М-222, БЭСМ-4.

Одно из главных применений РЕФАЛа – использование его в качестве средства описания специализированных языков программирования и общения с ЭВМ, потребность в которых возникает во многих областях исследования, в частности при создании автоматизированных систем управления и проектирования в строительстве.

В подготовке рекомендаций участвовали: ЦНИПИАСС, МИФИ, ИПМ АН СССР, МГУ и другие организации.

Авторы:
гл. I, II и п.1 гл. III – В.Ф.Турчин
п.2 гл. III – И.Б.Щенков
п.3 гл. III и гл. V – С.А.Романенко
гл. IV – С.А.Романенко, Ант.В.Климов
гл. VI, VIII – Ант.В.Климов
гл. VII – Е.В.Травкина
гл. IX – Арк.В.Климов
гл. X – А.Г.Красовский, В.Ф.Хорошевский

виде числа +1, а значение false – в виде числа –1. С учетом этого обстоятельства должна быть определена и функция 'ПИУФ'.

Вернемся еще раз к вопросу об эффективности синтаксического анализа. Использованный нами способ расчленения арифметических и логических выражений на подвыражения в соответствии со старшинством операций является простым и достаточно эффективным для практического применения, хотя, вероятно, не оптимальным. Легко видеть, что даже при заведении специальной функции для каждого уровня старшинства он требует нескольких (по числу уровней) просмотров выражения. Существуют другие алгоритмы синтаксического анализа, которые устраняют повторные просмотры, правда, ценой увеличения числа операций, совершаемых в процессе просмотра, в то время как наш алгоритм требует весьма простых просмотров. Ясно, что эти алгоритмы также можно описать на рефале. Какой из возможных алгоритмов окажется в конечном счете эффективнее и насколько, об этом можно с уверенностью судить, только проведя специальное исследование.

Этим закончено описание функции 'АВЫР'. Напомним, что конкретизация

$$k\,'\text{АВЫР}'\mathscr{E}\perp$$

где $\mathscr{E}$ – арифметическое выражение, дает программу, заносящую на сумматор значение выражения $\mathscr{E}$. Теперь осталось только проверить, что в левой части оператора присваивания стоит идентификатор, и приписать команду засылки числа в соответствующую ячейку. Присвоим функции, которая решает задачу в целом, детерминатив 'ПОП'-программирование оператора присваивания:

$$k\,'\text{ПОП}'\,e_1 = e_2 \sim k\,'\text{ПОП}\,1'\,(k\,'\text{ИДЕНТ}'\,(\,)\,e_1\perp)\,e_2\perp$$

$$k\,'\text{ПОП}\,1'\,((e_1))\,e_2 \sim k\,'\text{АВЫР}'\,e_2\perp\ \text{ЗП},e_1;$$

Мы видели здесь использование рефала в качестве метаязыка для описания алгоритмического языка. В результате конкретизации функции 'АВЫР' получаем программу для машины-исполнителя. Рассмотрим теперь более общую задачу построения трансляторов с помощью рефала.

Пусть $\mathscr{P}$ – программа на некотором алгоритмическом языке, а $\mathscr{E}$ – начальные данные, т.е. объект, к которому применяется алгоритм $\mathscr{P}$. Опишем на рефале рекурсивную функцию с детерминативом 'L' (идентифицирующим данный язык) таким образом, что выполнение конкретизации

$$k\,'\text{L}'\,\mathscr{P}\,(\mathscr{E})\perp \qquad (1)$$

будет рассматриваться как применение алгоритма $\mathscr{P}$ к объекту $\mathscr{E}$, и, в частности, результат этой конкретизации (когда он существует)

будет результатом применения алгоритма. Описание рекурсивной функции 'L' есть наиболее прямое и непосредственное выражение семантики языка. Используя программистский термин, можно сказать, что здесь происходит интерпретация программы (текста $\mathscr{P}$). Функцию 'L' мы называем интерпретирующей функцией языка.

Интерпретация происходит на рефал-машине, которая, в свою очередь, моделируется на определенной машине-исполнителе типа описанной выше (в дальнейшем машину-исполнитель или компьютер мы будем обозначать $M_\mathscr{P}$). В целях эффективности необходимо скомпилировать программу, осуществляющую алгоритм $\mathscr{P}$ для машины $M_\mathscr{P}$, т.е. перевести $\mathscr{P}$ с рефала на язык машины $M_\mathscr{P}$. Это значит, что мы должны проследить, как происходит конкретизация (1) с заданным $\mathscr{P}$, но произвольным $\mathscr{E}$, и отобразить объекты и действия рефал-машины на объекты и действия машины $M_\mathscr{P}$.

Решение этой задачи в соответствии с теорией компиляции осуществляется в два этапа: анализ конфигураций и отображение.

Конфигурацией мы называем обобщенное состояние рефал-машины, описываемое находящимся в ее поле зрения рефал-выражением со свободными переменными. Иначе говоря, конфигурация есть множество рабочих выражений в поле зрения, получающееся заменой свободных переменных на их всевозможные значения. Например, множество (1) с заданным $\mathscr{P}$, но произвольным $\mathscr{E}$ есть конфигурация

$$k\,'\mathrm{L}'\mathscr{P}\ (e_1)\perp$$

которая представляет собой начальную конфигурацию для компиляции программы $\mathscr{P}$.

Эквивалентные преобразования рефал-программ, в частности прогонка (см. [24-25]), позволяют выполнить последовательные шаги рефал-машины над обобщенным полем зрения и проследить, как конфигурации расщепляются и превращаются одна в другую. Результатом этого прослеживания является граф конфигураций.

На этапе отображения рефал-машины на машину $M_\mathscr{P}$ различные конфигурации отображаются на различные состояния точки управления в машине $M_\mathscr{P}$, а свободные переменные отображаются на информационные поля $M_\mathscr{P}$. В результате граф конфигураций отображается на программу для $M_\mathscr{P}$. Те операции, которые могут осуществляться машиной $M_\mathscr{P}$, но не описываются на рефале, рассматриваются как "машинные процедуры" и в процессе компиляции переводятся в соответствующие коды машины $M_\mathscr{P}$. Например, описанная машина имела команду сложения, записываемую в виде:

СЛ, А (*)

в результате которой содержимое ячейки А складывается с сумматором и сумма помещается в сумматор. На рефале вводится, соответ-

ственно, функция с детерминативом СЛ и форматом обращения:

$$k\,'\text{СЛ}'\,(e_1)(e_2)\perp \qquad (**)$$

причем характер выражений e_1 и e_2 и результата конкретизации не уточняется, пока речь идет о рефал-машине. Необходимо только отметить, что переход конфигурации (**), где e_1 отображено на сумматор, а e_2 – на ячейку А, в конфигурацию e_3, отображенную снова на сумматор, описывается на языке машины M_r командой (*).

В частном случае, когда машина M_r представляет собой рефал-машину, применение изложенных методов приводит к оптимизации программы на рефале. Если же алгоритм, осуществляющий процесс компиляции, также описан на рефале, то он становится одним из объектов своей работы, в результате чего мы автоматически получаем важнейшие системные программы.

Пусть рекурсивная функция 'СР', описанная на рефале, выполняет компиляцию заданной рефал-программы в программу для некоей машины M_r. Так как объектом работы функции 'СР' являются рефал-предложения, можно воспользоваться каким-либо метакодом, превращающим предложения в выражения. Это будет метакод, близкий к метакоду-А (см. главу X, разд.2).
Вот ключ к нему:

e_1	→* E1	k	→* K( )
A	→ A	( )	→( )
*	→* V	'OMEGA'	→'OMEGA'

Метакод предложения имеет вид:

$$(\mathcal{L})=\mathcal{R})$$

где $\mathcal{L}$ и $\mathcal{R}$ – метакоды левой и правой частей соответственно.

Для того чтобы задание на компиляцию было полностью определенным, необходимо, кроме описания всех участвующих функций, указать еще начальную и конечную конфигурации и их отображение на машину M_r. Это будет достигаться введением двух условных функций: 'OMA' (т.е. ω^a – начальная конфигурация рефал-машины) и 'OMZ'(ω^z – конечная конфигурация). Для рассмотренного выше случая (компиляция программы $\mathcal{P}$ на языке 'L') начальная конфигурация будет иметь вид:

$$k\,'\text{OMA}'\;e_1 \sim k\,'\text{L}'\mathcal{P}\,(e_1)\perp$$

Для спецификации отображения при переводе этого предложения в метакод сделаем в левой части такую замену:

E1 →(E1 'IN' M1,'M2)

Это означает, что значение переменной e_1 располагается в машине M_r в поле с границами M1 и M2.

Метакод предложения, описывающего выходную конфигурацию, будет иметь вид:

(('OMZ' (*E1 'IN' M1,M2)) = *E1)

Это означает только то, что результат конкретизации (скомпилированная программа) рассматривается как значение одной свободной переменной выражения, которое размещается в машине M_r в том же поле (M1,M2).

Итак, для того чтобы осуществить перевод программы $\mathscr{P}$, надо выполнить конкретизацию:

$$k\,'CP'\mathscr{D}\,['L']\,(('OMA'\,(*E1\;'IN'\;M1,M2)) = *K('L'\mathscr{M}[\mathscr{P}]\,(*E1)))\,(('OMZ'\;(*E1\;'IN'\;M1,M2)) = *E1)\perp \qquad (IC)$$

Через $\mathscr{D}$['L']мы обозначили описание функции 'L' в метакоде, через $\mathscr{M}[\mathscr{P}]$ – метакод текста $\mathscr{P}$.

Эта конкретизация выполняется с помощью рефал-интерпретатора, т.е. мы получаем компилятор, работающий в режиме интерпретации. Конкретизация требует $\mathscr{P}$ и дает программу для компьютера, которая помещает $\mathscr{E}$ в поле (M1,M2) и в нем же вырабатывает результат.

Следующее выражение описывает процесс компиляции интерпретатора:

$$k\,'CP'\mathscr{D}\,['L']\,(('OMA'\,(*E1\;'IN'\;M1,M2)\;(*E2\;'IN'\;N1,N2)) = *K('L'\;*E2\,(*E1)))\;(('OMZ'\,(*E1\;'IN'\;M1,M2)) = *E1)\perp \qquad (C1)$$

Конкретизация требует только описания 'L' и дает программу для компьютера, которая помещает $\mathscr{P}$ в поле (N1,N2), $\mathscr{E}$ – в поле (M1, M2) и вырабатывает путем интерпретации результат в поле (M1,M2).

Совершая еще один метасистемный переход, т.е. компилируя процесс конкретизации (IC) для произвольного $\mathscr{P}$, получаем:

$$k\,'CP'\mathscr{D}\,['CP']\,(('OMA'\;(*E1\;'IN'\;N1,N2)) = *K('CP'\mathscr{M}[\mathscr{D}\,['L']]\,(('OMA'\;(*VE1\;'IN'\;M1,M2)) = *VK('L'\;*E1\,(*VE1)))\;(('OMZ'\;(*VE1\;'IN'\;M1,M2)) = *VE1)))\;(('OMZ'\;(*E1\;'IN'\;N3',N4)) = *E1)\perp \qquad (CC)$$

что представляет собой скомпилированный компилятор. Конкретизация требует только описания 'L' и дает программу, которая помещает в поле (N1,N2) и порождает в поле (N3,N4) скомпилированную программу, такую же как в случае (*IC*).

Сделав еще один метасистемный переход, можно написать выражение (CCC в нашей системе обозначений). конкретизация которого дает скомпилированный для машины компилятор компиляторов.

БАЗИСНЫЙ РЕФАЛ И ЕГО РЕАЛИЗАЦИЯ НА ВЫЧИСЛИТЕЛЬНЫХ МАШИНАХ (методические рекомендации). М., ЦНИПИАСС 258 с. (Фонд алгоритмов и программ для ЭВМ в отрасли "Строительство". Вып.У-40).

Л-91371. Подписано к печати 3/Ш-77 г. Формат 60x84/16.
Объем 16 п.л. Тир.800. Зак.273. Цена 2 руб.17 коп.

ОТРД ЦНИПИАСС
117393, ГСП-1, Москва, В-393, Новые Черемушки, квартал 28, корпус 3