РЕФАЛ-КОМПИЛЯТОР*

С.А.Романенко, В.Ф.Турчин
(Москва)

3-6 февраля 1970

Резюме. В докладе описывается общая структура компилятора для языка РЕФАЛ. Основная часть работы по компиляции заключается в переводе текста на РЕФАЛе в текст на языке сборки (ЯС). Алгоритм перевода пишется на РЕФАЛе.

Введение

РЕФАЛ-интерпретатор, описанный в работе [1] является достаточно эффективным для решения многих практически важных задач, например, задачи трансляции с одного формализованного языка на другой с помощью программы, написанной на РЕФАЛе. Однако, определенная потеря эффективности при использовании РЕФАЛ-интерпретатора все-таки происходит за счет того, что при выполнении каждого шага РЕФАЛ-машины производится очистка некоторых вспомогательных таблиц и производится также ряд действий, связанных. с анализом применяемого предложения. Эти потери эффективности являются типичными для программы, работающей в режиме интерпретации. Транслятор, работающий в режиме компиляции, сопоставил бы каждому предложению участок программы, в котором особенности данного предложения учтены раз и навсегда, так что повторного анализа предложений при каждом использовании предложения уже не требуется. Кроме того, в РЕФАЛ-интерпретаторе [1] есть еще один специфический элемент (тоже, впрочем, непосредственно вытекающий из интерпретационного режима работы), устранив который можно повысить эффективность. В чем состоит этот элемент, покажем на следующем примере.

Пусть применяемое предложение имеет вид:

$$\S\ \underline{K}\varphi\underline{E}1\ \underline{S}2 + \underline{S}2\ \underline{E}3 \sim \underline{E}1\ \underline{S}2 + \underline{K}\varphi\underline{S}2\ \underline{E}3\ \underline{.}$$

*С.А.Романенко, В.Ф.Турчин. Рефал-компилятор. В сб.: “ВКП-2, Труды 2й Всесоюзной конференции по программированию, Заседание Б”, Новосибирск: 3-6 февраля 1970. Стр. 31-42.

Все, что нужно было бы сделать в данном случае после выполнения отождествления, - это перенести пару символов <u>K</u>φ из начала строки в середину, включив ее между символами + и вторым символом <u>S</u>2, то есть выполнить одну операцию исключения строки, и одну операцию включения (связь между <u>K</u> и φ разрывать не нужно). Между тем, РЕФАЛ-интерпретатор при замене левой части предложения на правую будет рассматривать каждый элемент правой части по отдельности, находить соответствующий элемент левой части и производить для каждого элемента операцию исключения а операцию включения. Чтобы устранить лишние действия, необходим компилятор, который сравнивал бы для каждого предложения левую и правую части, находил бы минимальную последовательность элементарных преобразований, переводящих левую часть в правую, и сопоставлял бы этой последовательности преобразований подпрограмму на выходном языке.

В настоящей работе описывается РЕФАЛ-компилятор, который, в частности, обладает указанной чертой, и позволяет существенно повысить эффективность выполнения программ, написанных на РЕФАЛе.

Расположение информации в памяти машины

В поле зрения РЕФАЛ-машины информация располагается, в общем, почти так же, как и в РЕФАЛ-интерпретаторе [1]. Под каждый символ отводится одна ячейка. В ячейке содержится, прежде всего, признак, указывающий, помещается ли в данной ячейке значащий символ, скобка, символа начала и конца конкретизации (<u>K</u> и <u>.</u>) или “представитель” (см. ниже), то есть указатель того, что следующая часть поля зрения вынесена во внешний память. Все ячейки содержат адреса предыдущего и последующего символов. Кроме того, ячейка со значащим символом содержит код этого символа, ячейка со скобкой содержит адрес парной скобки, ячейка с символом <u>K</u> или <u>.</u> - адрес парного символа, ячейка с представителем - адрес во внешнем запоминающем устройстве. В отличие от РЕФАЛ-интерпретатора, адреса связи между символами конкретизации, отражающие порядок выполнения конкретизации, не хранятся в поле зрения, а сосредоточены в специальном стэке - стэке конкретизации СК. Каждая строка СК содержит три адреса: адрес передачи управления на процедуру (рекурсивную функцию), соответствующую данному символу конкретизации, адрес символа конкретизации <u>K</u> и адрес парного символа <u>.</u> - конца области действия конкретизации. Поэтому хранение в тексте (то есть в поле зрения) детерминативов процедур становится ненужным, и они опускаются: в поле зрения вслед за символом конкретизации следует первый символ аргумента процедуры. В этом состоит второе отличие от РЕФАЛ-интерпретатора.

Стэк конкретизации заполняется таким образом, что последняя строка соответствует ведущему символу конкретизации, предпоследняя - тому символу, который станет ведущим следующим, и т.д. Первая строка соответствует самому внешнему символу конкретизации.

Использование внешней памяти

Если использовать только оперативное запоминающее устройство, то это наложит существенные ограничения на возможности компилятора. Так, у БЭСМ-6 длина поля зрения не сможет превышать 15-20 тыс. символов, в то время как для многих задач (трансляция с АЛГОЛа, алгебраические преобразования) этого совершенно недостаточно.

В связи с этим становится необходимым использование внешних запоминающих устройств для хранения части поля зрения, не используемой в какой-то момент выполнения алгоритма.

Специфика РЕФАЛа как языка рекурсивных функций такова, что не позволяет заранее предусмотреть размеры массивов обрабатываемой информации, поэтому система обмена с внешней памятью должна быть полностью независимой от пользователя, чтобы дать ему возможность работать исключительно на уровне языка.

Так как с внешними устройствами возможен обмен только большими массивами фиксированных размеров, вынос информации из поля зрения должен также осуществляться определенными квантами.

При этом, каждый такой квант выносимой информации должен представлять собой выражение РЕФАЛа, так как в противном случае будут разорваны адреса связи у скобок. Это требование не позволяет выносить кванты строго фиксированной длины l_0, так как в поле зрения только в редких случаях можно будет подобрать выражение, пригодное для выноса и имеющее длину ровно l_0. Целесообразно допустить, чтобы квант имел длину от $l_0/2$ до l_0, где l_0 - максимальная длина кванта, определяемая параметрами внешних устройств машины, (например, 1024 символа). На месте вынесенного кванта информации в ОЗУ подшивается его представитель, который оформляется как было указано выше. Таким образом в ОЗУ по-прежнему остается выражение РЕФАЛа, но содержащее, однако, представители.

Для облегчения работы системы на кванты накладывается еще два ограничения. Во-первых, они не должны содержать представителей, во-вторых, они не должны содержать К и . . Благодаря второму ограничению все К и . остаются в ОЗУ.

Если при выполнении шага РЕФАЛ-машины возникает необходимость в просмотре вынесенной информации, представитель разворачивается, то есть на его место в ОЗУ вшивается соответствующий отрезок поля зрения, записанный во внешнем запоминающем устройстве.

Задачей системы автоматического обмена с внешними ЗУ является поиск квантов информации в ОЗУ, пригодных для вынесения на внешние устройства и его осуществление в случае нехватки оперативной памяти. Кроме того в задачу системы входит развертка представителей, необходимая в процессе синтаксического отождествления.

Поиск кванта информации для выноса начинается в области действия самого внешнего символа конкретизации, то есть того, который станет ведущим в последнюю очередь. Если в нем не нашлось выражения, подходящего для выноса, аналогичный поиск производится в области действия предпоследнего $\underline{\texttt{К}}$ и т.д. вплоть до ведущего $\underline{\texttt{К}}$, . Таким образом, в данном случае порядок движения по символам конкретизации обратен порядку, принятому при выполнении последовательных шагов. Он осуществляется путем просмотра стэка конкретизации СК в направлении от первой ячейки к последней.

Случай, когда приходится сворачивать информацию в ведущей области конкретизации, будет рассмотрен особо.

В описанном процессе главную роль играет блок выделения кванта БВК, который по данным адресам начала АНАЧ и конца АКОН выражения ищет в нем квант информации, подходящий для выноса, и, в случае удачного поиска, выдает адрес начала и адрес конца кванта. При этом, как отмечено выше, квант не должен содержать $\underline{\texttt{К}}$, или представителей.

БВК может работать в двух режимах: выделение слева направо и выделение справа налево. При выделении слева направо всегда выделяется самый левый квант информации, а справа налево - наоборот. Оба процесса аналогичны, поэтому в дальнейшем будет описан только процесс выделения слева направо.

Если выделить квант информации невозможно, блок выдает сигнал об этом.

Выделение кванта осуществляется путем просмотра выражения слева направо. При этом заводится стэк потенциальных квантов СПК.

В каждую ячейку СПК заносится количество набранных символов кванта КСК и адрес начала кванта АНК. В процессе работы глубина СПК зависит от глубины уровня скобочной структуры, на котором ведется просмотр. Первоначально в первую ячейку СПК заносится КСК = 0 и АНК = АНАЧ. Затем в процессе просмотра различаются следующие случаи.

а) Очередной символ - значащий. В этом случае в последней ячейке стэка СПК КСК увеличивается на единицу. Если после этого КСК = l_0, выделение кванта считается законченным. При этом АНК дает начало кванта, а адрес текущего символа - адрес конца кванта. Если КСК < l_0, просмотр продолжается.

б) Очередной символ - левая скобка. В этом случае, если КСК $\geq l_0/2$, выделение кванта считается законченным. АНК дает адрес начала кванта, а символ, предшествующий скобке - адрес конца. Если КСК < $l_0/2$, глубина стэка СПК увеличивается. В стэк заносится КСК = 0 и АНК, равное адресу символа, следующего за скобкой. Просмотр продолжается.

в) Очередной символ - правая скобка. Если КСК $\geq l_0/2$, поиск кванта считается законченным. АНК дает начало, символ, предшествующий скобке - конец. Если КСК $< l_0/2$, рассматривается два случая.

Если внутри, скобочной структуры есть K или представители (при этом ЗАПР = 1) глубина стэка уменьшается на единицу, в КСК заносится 0, в АНК - адрес символа, следующего за правой скобкой, и просмотр продолжается. Если же K или представителей внутри нет (при этом ЗАПР = 0), стэк СПК уменьшается на единицу. Если КСК$'$ + КСК + 2 $=< l_0$, просмотр продолжается, если же КСК$'$ + КСК + 2 $= l_0$, поиск кванта заканчивается. Случая КСК$'$ + КСК + 2 $> l_0$ быть не может, так как КСК$' \leq l_0/2 - 1$ и КСК $\leq l_0/2 - 1$.

г) Очередной символ - представитель. Если КСК $\geq l_0/2$, поиск кванта окончен. Если КСК $< l_0/2$, в КСК заносится 0, а в АНК заносится адрес следующего символа. В ЗАПР заносится 1. Просмотр продолжается.

д) Очередной символ K. Этот случай аналогичен предыдущему, за тем исключением, что если КСК $< l_0/2$, в АНК заносится адрес символа, следующего за ., парной K. Таким образом, все внутренние области конкретизации обходятся и остаются в ОЗУ .

Если найти подходящий квант вне ведущей области конкретизации не удается, необходимо выносить информацию из ведущей области конкретизации. В этом случае мы возвращаемся к началу шага, который выполнить не удалось, и максимально сжимаем все выражение, составляющее область действия ведущего символа конкретизации, вынося несколько квантов. Затем приступаем к выполнению отложенного на время шага, развертывая представители, когда в этом возникает необходимость.

Язык сборки

Трансляция осуществляется в два этапа. Сначала текст на РЕФАЛе Преобразуется в текст на языке сборки (ЯС), затем текст на ЯС переводится на язык машины (или автокода).

Текст на ЯС имеет вид последовательности операторов, отделенных друг от друга точкой с запятой, из которых некоторые могут быть снабжены меткой, отделенной двоеточием, подобно тому, как это записывается на АЛГОЛе. Список операторов включает оператор присваивания, который записывается как на АЛГОЛе, оператор перехода ПЕРЕХОД и еще 34 специфических оператора,которые будут описаны ниже. При переводе на язык машин операторам будут соответствовать подпрограммы на языке машины. Кроме того, в языке сборки определены пять “существительных”, которым при переводе на язык машины будут соответствовать поля в памяти машины. Это, прежде всего две таблицы СК и ТЭ. СК - это стэк конкретизации, о котором говорилось выше. Число ячеек, которые надо отвести под СК равно максимальной глубине вложения друг в друга символов конкретизации. Хотя нетрудно написать на РЕФАЛе программу, которая приведет к сколь угодно большой глубине в процесс выполнения конкретизации, под СК "за глаза" достаточно сотни ячеек, ибо каждую рекурсивную функцию можно определить такими предложениями, что глубина символов <u>К</u> будет оставаться небольшой. ТЭ - таблица адресов элементов. Под элементом понимается элемент левой части предложения, то есть либо символ, либо свободная переменная либо <u>К</u> либо <u>.</u>. В процессе отождествления элементы левой части получают, номера (в том порядке, как они отожествляются), и в таблицу ТЭ заносятся *конечные* адреса соответствующих элементов. Очевидно, под ТЭ нужно отвести столько ячеек, каково максимальное число элементов в левой части предложений. Здесь также практически вполне достаточно сотни ячеек.

Оставшиеся три существительных - это НЭЛ - номер текущего элемента, ПЕР - адрес переход при неудачном отождествлении и АОТМ - адрес отметки отождествленной части при обратном ходе в процессе отождествления закрытой свободной переменной выражения (см. [2]).

Кроме того, в языке сборки фигурируют в качестве аргументов символы (в своем "натуральном" виде), которым в действительности должны быть сопоставлены адреса ячеек, содержащих их коды. Аргументы операторов записываются через запятые.

Операторы (специфические) делятся на две группы. Операторы первой группы работают в процессе отождествления, В конечном счете они служат для того, чтобы заполнить таблицу адресов, элементов ТЭ и, разумеется, установить, возможно ли отождествление. Каждый специфический оператор относится к определенному элементу. Все специфические операторы увеличивают на единицу НЭЛ и заносят в ячейку ТЭ[НЭЛ] (НЭЛ - уже новое) адрес первого свободного символа для следующего (то есть имеющего номер НЭЛ) элемента. Операторы, относящиеся к отождествлению справа налево (в частности, при обратном ходе в процессе обычного - слева направо - отождествления) помечаются звездочкой.

Элементу, имеющему вид конкретного значащего символа соответствуют операторы СИМ и СИМ* . Действие оператора

```
СИМ,B;
```

таково. Он проверяет, является ли текущий символ в поле зрения символом B. Если это так, то выполняются указанные выше изменения НЭЛ и ТЭ и работа оператора закончена. Если символ не совпадает с B , то управление передается в ячейку ПЕР.

Свободной переменной символа соответствуют два оператора без аргументов: ЗНАЧ и ЗНАЧ*, действующих аналогично. Если свободная переменная (значащего) символа встречается второй раз, то используется оператор СТЗН (и СТЗН*, соответственно) с одним аргументом, который указывает номер той свободной переменной, с которой данная должна совпадать, (СТ означает "старая"). При выполнении этих операторов производится дополнительно необходимое сравнение.

Свободной переменной терма сопоставлены аналогичные четыре оператора: ТЕРМ, ТЕРМ* СТТ и СТТ*. Скобкам соответствуют операторы СКОБ и СКОБ*. Свободной переменной выражения, встречающейся повторно, соответствуют операторы СТВЫР и СТВЫР*. Открытой свободной переменной выражения соответствуют операторы ИСК и ИСК* - для случая, когда имеется значащий опорный символ, и УДЛ и УДЛ* для случая, когда значащего опорного символа нет. Оператор ИСК продвигает НЭЛ на две единицы (вторая - за счет определения адреса опорного символа). Кроме того, он отличается еще тем, что требует, чтобы в ячейку НЭЛ+1 (соответствующую опорному символу) было заранее занесено значение адреса, предшествующего первой просматриваемой на предмет поиска опорного символа ячейке. Операторы ОТМ и ОТМ* служат для установки отметки в поле зрения при обратном ходе. Такой отметкой служит признак символа $\underline{K}$ и $\underline{.}$ соответственно. Оператор СНОТМ снимает отметку. Наконец, операторы КОНК и ТОЧКА служат для перехода на символ $\underline{K}$ и $\underline{.}$ соответственно, с занесением их адресов в ТЭ.

Операторы второй группы (они обозначаются набором латинских букв) служат для преобразования поля зрения. Оператор трансплантации TPL,n,m,l; переставляет участок, начинающийся *непосредственно* за адресом ТЭ[n] и кончающийся адресом ТЭ[m] (включительно), вшивая его непосредственно за символом с адресом ТЭ[l]. Оператор MS,n,m; - движение символа (одного) аналогично предыдущему оператору. Оператор SS,α,n; - подстановка символа α. (в натуральном коде) *вместо* символа по адресу ТЭ[n]. Оператор NS,α,n,m; вставляет *новый* символ α (используя ячейку из свободной памяти). Оператор BRA,n,m; ставит пару скобок. Оператор размножения MULT,n,m,l; действует так же, как TPL, с той лишь разницей, что участок не вынимается из своего места, а репродуцируется. Для управления стэком СК служат операторы ST,*varphi*,A_1,A_2; - занести в стэк тройку адресов, и оператор STMIN - уменьшить стэк на единицу.

Наконец, в языке сборки определены еще две функции СЛЕД и ПРЕД сопоставляющие адресу значения адресов предыдущей и последующей ячейки.

Приведем в качестве примера перевод на язык сборки, следующего описания процедуры φ[1]:

РЕФАЛ:

$$\S\ \overset{1}{\underline{\mathtt{K}}}\ \varphi\ \overset{2}{\underline{\mathtt{E}}}\mathtt{1}\ \overset{4}{\underline{\mathtt{W}}}\mathtt{2}\overset{3}{*}\overset{5}{(}\ \overset{9}{\underline{\mathtt{E}}}\mathtt{3}\overset{10}{+}\ \overset{11}{\underline{\mathtt{E}}}\mathtt{4}\overset{6}{)}\ \overset{8}{\underline{\mathtt{E}}}\mathtt{5}\ \overset{7}{\sim}$$

$$\underline{\mathtt{E}}\mathtt{1}\,(\,\underline{\mathtt{W}}\mathtt{2} * \underline{\mathtt{E}}\mathtt{3} + \underline{\mathtt{W}}\mathtt{2} * \underline{\mathtt{E}}\mathtt{4})\,\underline{\mathtt{K}}\varphi\,\underline{\mathtt{E}}\mathtt{5}\ \underline{.}$$

$$\S\,\underline{\mathtt{K}}\varphi\underline{\mathtt{E}}\mathtt{1} \sim \underline{\mathtt{E}}\mathtt{1}$$

```
ЯЗЫК СБОРКИ:
     НЭЛ := 1;
     ПЕР := М1;
     КОНК;
     ОТМ;
     ТЭ[3] := ТЭ[1];
М2:  НЭЛ := 2;
     ПЕР := М1;
     ИСК,*;
     ПЕР := М2;
     ТЕРМ*;
     СНОТМ;
     ТЭ[5] := СЛЕД(ТЭ[3]);
     СКОБ;
     ТОЧКА;
     ТЭ[10] := ТЭ[5];
М3:  НЭЛ := 9;
     ПЕР := М2;
     ИСК,+;
     ПЕР := М3;
     ТЭ[11] := ПРЕД(ТЭ[6]);
     MS,5,2;
     MULT,2,3,10;
     MS,1,6;
     GNS;
М1:  MS,1,0;
     MS,7,0;
     STMIN;
     GNS;
```

Здесь GNS - оператор перехода к очередному шагу.

[1]Номерами сверху помечены элементы в порядке отождествления

Перевод на язык сборки

Перевод текста на РЕФАЛе в текст на языке сборки - задача, состоящая из двух частей (для каждой процедуры). Первая часть получающейся подпрограммы соответствует процедуре отождествления. Как можно видеть из приведенного выше примера, эта часть работы выполняется без труда, так как каждому элементу левой части соответствует либо один оператор, либо определенная комбинация операторов. Определение порядка отождествления также не представляет трудностей.

Вторая ("латинская") часть подпрограммы требует более сложной работы по-определению набора трансформаций, переводящих левую часть в правую. Однако к эта часть может быть выполнена достаточно хорошо с помощью алгоритма.

Алгоритмы перевода написаны на РЕФАЛе (для управляющих символов используется специальный код), и реализуются с помощью РЕФАЛ-интерпретатора. В момент написания доклада алгоритм перевода находится в стадии отладки. В дальнейшем авторы надеются перевести с помощью РЕФАЛ-интерпретатора сам алгоритм перевода и полученную скомпилированную программу использовать в дальнейшем в РЕФАЛ-компиляторе. Этот акт явится в своем роде "актом самоубийства" РЕФАЛ-интерпретатора. Впрочем, самоубийство это не является полным, так как в режиме отладки, по-видимому, удобнее использовать РЕФАЛ-интерпретатор.

Литература

1. С.Н.Флоренцев, В.Ю.Олюнин, В.Ф.Турчин. *РЕФАЛ-интерпретатор.* Труды 1-ой Всесоюзной Конференции по программированию. Киев, ноябрь 1968 г.

2. С.Н.Флоренцев, В.Ю.Олюнин, В.Ф.Турчин. *Эффективный интерпретатор для языка РЕФАЛ*, Препринт ИПМ АН СССР, № 29,1969.