1972 год ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ Выпуск 51

УДК 681.326

ТРАНСЛЯТОР ДЛЯ ОДНОРОДНОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

В.И.Константинов

Исследуются особенности методики построения трансляторов для однородных вычислительных систем (ОВС) [1]на примере транслятора для ОВС типа "Минск-222" [2].

В основе входного языка транслятора (ОВС-языка) лежит диалоговый язык *BASIC* [3], пополненный операторами, реализующими взаимодействия машин в системе [4].Записываемые на ОВС - языке программы рассматриваются как идентичные параллельные ветви вычислительного процесса, получаемого с помощью методики распараллеливания по циклам [5]. Транслятор позволяет получать универсальные программы, настраивающиеся на число машин в системе и на входные данные как на параметр.

В соответствии с [6], в основу методики построения транслятора положены следующие принципы:

выделения языковой модели и разбиения на ее основе глобальной задачи построения транслятора на ряд задач, независимо реализуемых большой группой разработчиков;

универсального синтаксического анализатора;

семантического описания правил языковой модели, означающего, что каждое правило снабжается указаниями о действиях, выполняемых при распознавании соответствующей конструкции.

Грамматика входного языка описывается на метаязыке TR - грамматик (модификации R -грамматик [7,8]).

Построение языковой модели

Для построения языковой модели в качестве исходной формы задания грамматики входного языка используется модификация Бэкусово-Науровской формы (МБНФ), в которой отсутствуют леворе - курсивные элементы.

TR - грамматика-это шестерка вида

$$G = \left\{ \Sigma, V_N, P, R, \left\{ z_0^{\sigma_i} \right\}, R^* \right\},$$

где $\Sigma = \{a_0, \ldots, a_k\}$ - конечный терминальный алфавит;

$V_N = \{x_0, \ldots, x_n\}$ - конечный нетерминальный алфавит, $\Sigma \cap V_N = \phi$; $P = \{0,1\}$;

$$R = R^{\sigma_0} \cup \ldots \cup R^{\sigma_n}; \quad R^{\sigma_i} \cap R^{\sigma_j} = \phi \text{ при } i \neq j, \; 0 \leq i, j \leq n;$$

R^{σ_i} - множество имен подмножеств элементов из $(\Sigma \cup V_N) \times R^{\sigma_i} \times P$;

$R^{\sigma_i} = \{\phi, z_j^{\sigma_i}\}, \phi$ - имя пустого множества, $0 \leq j \leq \ell_i$, $0 \leq i \leq n$;

$z_0^{\sigma_i}$ - начальное множество (аксиома) для $x_i \in V_N$,

$$z_0^{\sigma_i} \in R^{\sigma_i}, \quad 0 \leq i \leq n;$$

$z_0^{\sigma_0}$ - аксиома грамматики G ;

R^* - подмножество R , возможно, пустое.

Элемент $(\sigma, \tau, p) \in ((\Sigma \cup V_N) \times R \times P$ называется правилом TR - грамматики G .

В результате перехода к TR -грамматикам исходная грамматика оказывается разложенной на множества подмножеств правил вида (σ , τ , p). Рассмотрим два уровня разложения:

терминальный уровень, соответствующий правилам (σ, τ, p) при $\sigma \in \Sigma$;

уровень конструкций, соответствующий правилам (σ, τ, p) при $\sigma \in V_N$;

Такая классификация правил определяет последующую тактику построения транслятора.

Уровни трансляции

Выделим три уровня трансляции:

терминальный уровень (уровень типа I);

уровень конструкций (уровень типа 2);

уровень грамматик (уровень типа 3).

Если первые два уровня возникают из способа построения языковой модели и являются внутренними по отношению к языку (и, соответственно, транслятору), то третий есть отражение некоторых внешних действий по отношению к транслятору. Применительно к описываемой методике эти действия состоят в выделении этапов синтаксического контроля, перекодировки на внутренний язык и собственно трансляции. Каждым этапом управляет соответствующая грамматика, внутри которой есть уровни типа I и 2.

Про уровень типа I можно сказать, что на нем осуществляется первичная обработка символов входной строки. Каждое терминальное правило грамматики снабжается соответствующей семантикой (директивой, подпрограммой), которая и осуществляет выполнение необходимых действий.

Структура этих семантик обычно очень проста, а размеры не превышают нескольких десятков машинных слов.

На втором уровне происходит группировка символов входной строки в конструкции. Соответствующие правила грамматики также снабжаются семантиками, выполняемыми, если данная конструкция распознана.

Проиллюстрируем это на простом примере. Пусть на этапе перекодировки обрабатывается идентификатор. Тогда действие на терминальном уровне есть занесение очередного символа идентификатора в некоторое рабочее поле (сборка идентификатора). При появлении первого символа, не являющегося символом идентификатора, переходим на второй уровень и завершаем обработку конструкции, заменяя собственно идентификатор адресом ячейки рабочего поля, в которой он стоит. Если для первого уровня характерно "действие на символ", то для второго -"действие на группу символов".

Появление третьего уровня отражает итеративный принцип построения транслятора [6], согласно которому транслятор есть совокупность блоков, последовательно обрабатывающих символы

входной программы. Блок состоит из соответствующей грамматики и единой для всех блоков программы анализа, перенастраивающейся в зависимости от типа входного символа и этапа работы. Весь транслятор можно представить схемой (рис.I).

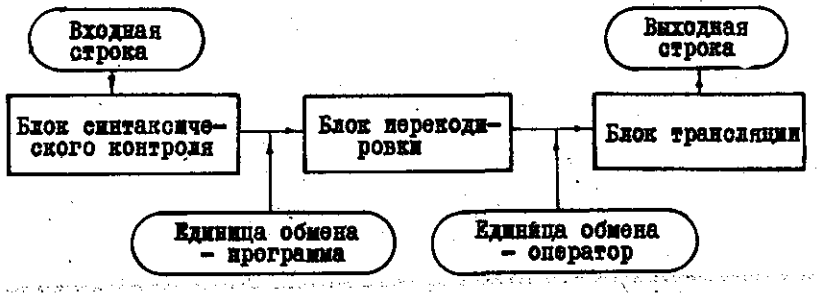

Рис. I

Роль блоков синтаксического контроля и трансляции ясна. Рассмотрим назначение блока перекодировки, выполняющего две основные функции:

приведение информации к некоторому стандартному виду (например, замена конструкций переменной длины, таких, как идентификатор, число и т.п., адресами);

выявление семантических ошибок, исключение комментариев.

При реализации на ЭВМ для блока синтаксического контроля можно использовать ту же грамматику, что и для блока перекодировки. Отметим также, что введение второго уровня трансляции позволяет осуществить локализацию ошибок по конструкциям.

Синтаксический анализ тор

Синтаксический анализатор позволяет осуществить объединение в единое целое правил, образующих грамматику входного языка. Форма этих правил такова, что по ним или через стек связи можно определять правила-преемники данного правила. Синтаксический анализатор выполняет следующие функции:

выделяет текущий анализируемый символ;

сравнивает его с левыми частями текущих правил и выполняет необходимые стековые операции;

определяет преемники данного правила или выходит на синтаксическую ошибку.

Высокая скорость синтаксического анализа и трансляции (линейная по отношению к длине входной программы) достигается за счет применения безвозвратных анализирующих схем.

На рис. 2 приведена блок-схема синтаксического анализатора, реализованного в ОВС-трансляторе.

Правила грамматики представляются в памяти ЭВМ строками таблицы следующей структуры:

T_1	T_2	T_3	T_4	T_5	T_6	T_7

T_1 - указатель конца уровня;
T_2 - указатель типа правила (терминальное либо нет);
T_3 - символ из алфавита $\Sigma \cup V_N$;
T_4 - адрес множества правил-преемников;
T_5 - выделитель разомкнутых правил;
T_6 - указатель заключительных правил;
T_7 - номер семантики;
$\Sigma[i]$ - текущий элемент входной строки;
$T[j]$ - содержимое текущей строки таблицы;
$ST[l]$ - содержимое вершины стека связи;
$\#$ - маркер конца входной строки.

Длина программы, реализующей приведенную блок-схему, 40 команд.

Алгоритмы трансляции

Все I9 операторов ОВС-языка, в том числе и операторы системных взаимодействий, можно отнести к одной из следующих групп:

операторы изменения порядка вычислений, сюда входят системные операторы обобщенного условного и безусловного переходов;

операторы описания и вызова функций, подпрограмм и библиотечных программ (программ, записанных в машинном коде);

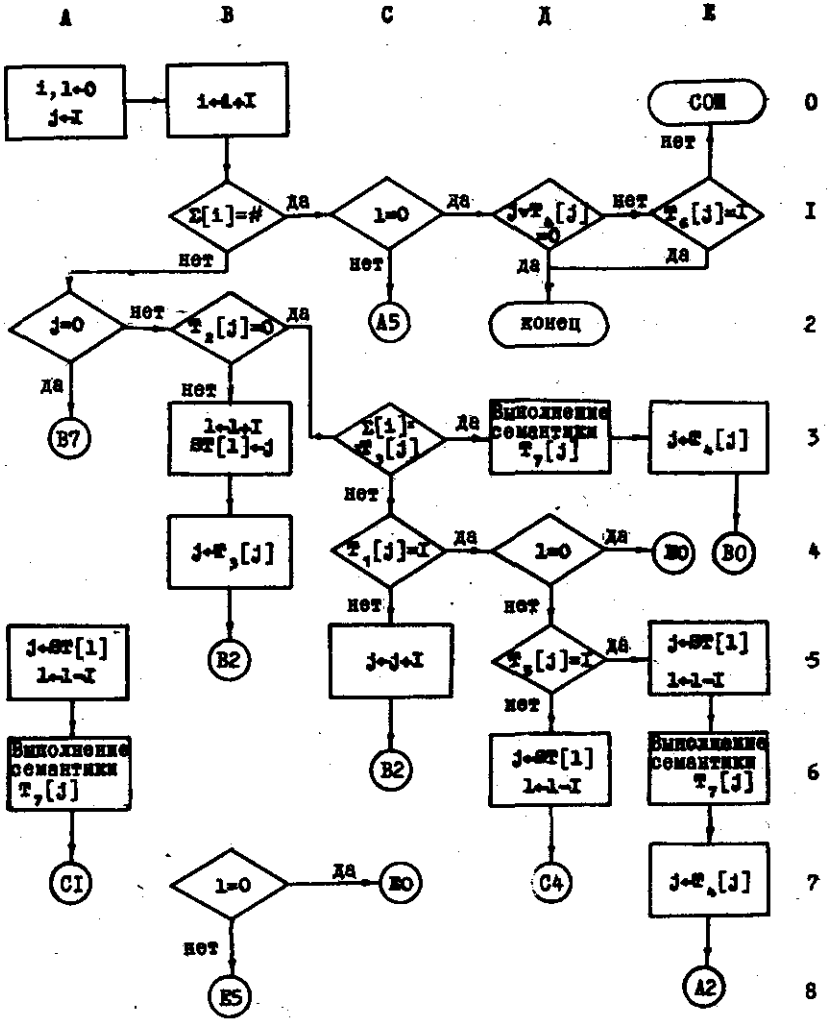

Рис. 2. Блок-схема синтаксического анализатора

операторы обмена информацией между ветвями параллельного процесса, ввода и вывода;

операторы описания и комментариев;

операторы организации циклов;

оператор присваивания;

Прежде всего опишем трансляцию формул, поскольку эта конструкция входит во многие операторы ОВС-языка. Для формирования команд готовой программы, реализующих вычисления по формулам, применяется широко распространенная польская инверсная запись [9]. В ОВС-трансляторе она получается с помощью специализированной *TR* - грамматики этапа трансляции, особенностью которой является порядок расположения конструкций, образующих формулу (аналог арифметического и булевского выражений); он совпадает с таблицей приоритетов соответствующих операций. Это позволяет отказаться от таблицы приоритетов (в отличие от [9]).

Роль стека для переупорядочения ограничителей играет стек связи правил грамматики. Действия по формированию команд готовой программы предельно просты. Так, если из стека связи вытолкнуто правило с семантикой, формирующей операцию умножения, то из поля операндов извлекаются два последних адреса, и требуемая команда посылается в массив для накопления готовой программы (ГП).

Для трансляции операторов изменения порядка вычислений (*GOTO*- безусловного перехода, *IF-THEN* - условного перехода, *GUP* - обобщенного безусловного перехода, *GCP* - обобщенного условного перехода) формируется таблица соответствий номеров операторов адресам команд готовой программы. Общий прием, применяемый при трансляции операторов этой группы, заключается в следующем. Выделяются два типа передач управления (ПУ):

на оператор, уже оттранслированный (завершенная ПУ);

на оператор, еще не оттранслированный (незавершенная ПУ);

В первом случае команду ПУ можно сформировать, взяв адрес α ПУ из таблицы соответствий. При этом оператор *GOTO* транслируется в команду:

K+0) -30 00 α . 0000.

Оператор *GUP* транслируется в две команды:

K+0) -02 00 0000 *K+1*

K+1) -30 00 α 0000 .

Оператор *IF - THEN* транслируется в последовательность команд вычисления формулы, определяющей условие, после чего ставится команда условного перехода:

$k+0$) . . . . . .

$k+\beta$) −34 00 α $k+\beta+1$.

Оператор *GCP* транслируется в две команды:

$k+0$) I5 00 z_1 0000

$k+1$) −65 00 4000 α ,

где z_1 – адрес простой переменной, значение которой участвует в выработке обобщенного условия. Единица в тринадцатом разряде команды k +I означает, что команда −65 00 AI A2 реагирует на признак знака.

Операторы ПУ второго типа транслируются с использованием стека незавершенных переходов (НП). В момент появления такого оператора в ГП засылается незавершенная команда ПУ, а в стек НП записывается номер оператора, на который передается управление, и адрес γ ГП незавершенной команды ПУ. Если появляется еще одна команда ПУ на тот же оператор, стоящая в ГП по адресу μ , то в стек НП вместо γ ставится μ , а γ записывается в ГП по адресу μ и т.д.

В момент поступления текущего оператора анализируется наличие незавершенных команд ПУ на него. При положительном исходе цепочка раскручивается в обратном порядке: из стека НП считывается адрес μ последней команды ПУ на данный оператор. Затем анализируется содержимое первого адреса ячейки μ ГП. Если оно пусто, то незавершенная команда ПУ единственна, завершаем ее; иначе считываем содержимое первого адреса (пусть это адрес γ), завершаем ПУ по адресу μ и все повторяем для ячейки γ ГП.

Структура команд безусловного и условного переходов ЭВМ "Минск-22", а также команды обобщенного безусловного перехода ОВС"Минск-222" такова, что позволяет иметь один стек НП для всех этих команд и при завершении не делать между ними различий. Для команды обобщенного условного перехода заводится свой стек той же структуры, однако если бы адрес ПУ в команде −65 00 AI A2 стоял по первому адресу, то при завершении её можно было бы не отличать от команд ПУ других типов. Заметим, что поскольку все

ветви параллельной программы считаются идентичными, анализа регистров настройки не требуется, так как все машины системы настроены одинаковым образом.

Оператор системных обменов реализуется с помощью команд синхронизации, передачи и приема ОВС "Минск-222". Трансляция такого оператора состоит из следующих действий:

определения передающей и принимающих машин, что соответствует схеме обмена "одна – всем";

построения программ передающей и принимающей ветвей.

Для этого формируются команды ГП, которые вычисляют целое значение ℓ простой переменной, определяющей номер передающей машины. Считается, что во время выполнения ГП в каждой машине системы в двух определенных ячейках n_1 и n_2 содержатся номер машины N_1 и число машин в системе N_2 , причем адреса n_1 и n_2 известны транслятору. Для определения передающей и принимающих ветвей формируются команды готовой программы, сравнивающие ℓ с N_1 , и та машина, в которой произошло совпадение, объявляется передающей. Кроме того, анализируется содержимое n_2 , и если $N_2 \geq 2$, то остальные машины объявляются принимающими. Программа передающей и принимающей ветви начинается с команды синхронизации −65 00 0000 0000, после которой ставится команда −56 i 0000 0000 для передающей ветви и команда −57 i 0000 0000 для принимающей ветви. Перед этим в ГП засылаются команды, формирующие содержимое индексной ячейки i . В передающей ветви в i заносится 0000 $z\alpha$, где z –число передаваемых кодов, а α – адрес первой передаваемой ячейки. В принимающей ветви в i заносится 0000 $h\beta$, где h –число принимаемых кодов, β –адрес первой принимающей ячейки.

Мы рассмотрели трансляцию операторов системных взаимодействий более подробно ввиду малоизученности этих вопросов. Можно отметить, что она на самом деле весьма проста и может проводиться (например, операторы ПУ) теми же средствами, что и трансляция обычных операторов языка.

Трансляция остальных операторов ОВС-языка имеет много общего с трансляцией подобных операторов в других трансляторах , и на ней мы останавливаться не будем. Отметим лишь одну особенность описываемой методики, которая заключается в представлении процесса трансляции в виде последовательности простых по

структуре действий, выполняемых при распознавании соответствующих конструкций языка.

Характеристики транслятора

ОВС-транслятор-это синтаксически управляемый транслятор итеративного типа [6]. Для улучшения сервисных характеристик транслятора ввод программы и внесение исправлений реализованы в режиме диалога с пользователем. Применение метаязыка *TR* - грамматик и связанного с ним синтаксического анализатора позволило достичь необходимой компактности транслятора (один блок памяти "Минск-22", дополнительно к которому прилагается библиотека транслятора, хранящаяся на магнитной ленте).

Входной язык транслятора - ОВС-язык - позволяет записывать как последовательные, так и параллельные процессы. Формируемые транслятором программы вместе с таблицами идентификаторов, констант, функций и т.д. размещаются во втором блоке памяти (под таблицы - 1024 ячейки, под программы - 3072 ячейки). При работе готовой программы под массивы данных отводится первый блок, за исключением первых 64 ячеек.

Выводы

1. Опыт разработки транслятора показал высокую эффективность методики построения математического обеспечения, основанной на идеях, разработанных в [6,7,8]. Введение в метаязык нетерминальных символов способствовало увеличению независимости блоков транслятора, что позволило выполнять их в виде стандартных блоков, объединенных грамматикой входного языка. Объем транслятора оказался всего 4000 ячеек (без библиотеки транслятора), срок разработки - 2 человеко-года.

2. Выяснилось, что трансляция операторов системных взаимодействий и по структуре, и по методам трансляции практически не отличается от реализации обычных операторов и не оказывает существенного влияния на сроки разработки и общий объем транслятора (трансляция системных взаимодействий занимает порядка 300 машинных слов).

3. Отладка транслятора, поиск и исправление неудачных решений существенно упростились, так как стало возможным осуществить принцип локализации изменений по конструкциям грамматики входного языка [2].

4. Полученные результаты позволяют считать, что описанная выше методика может оказаться полезной при разработке других трансляторов для ОВС.

Литература

1. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. Однородные универсальные вычислительные системы высокой производительности. Новосибирск, "Наука" СО, 1966.

2. ЕВРЕИНОВ Э.В., ЛОПАТО Г.П. Универсальная вычислительная система "Минск-222". -"Вычислительные системы", Новосибирск, "Наука" СО, 1966, вып. 23, стр. 13-20.

3. A Pocket Guide to Hewlett-Packard Computers. Palo Alto, California, 1969.

4. КОСАРЕВ Ю.Г., ПЫХТИН В.Я., ЖУКОВ Е.Н., ГОЛОВЯШКИНА Л.В.; КОЛОСОВА Ю.И. Особенности употребления команд системы "Минск-222". -"Вычислительные системы", Новосибирск, "Наука" СО, 1967, вып. 24, стр. 41-54.

5. КОСАРЕВ Ю.Г. Распараллеливание по циклам. -"Вычислительные системы", Новосибирск, "Наука" СО, 1967, вып. 24, стр. 3-20.

6. ВЕЛЬБИЦКИЙ И.В., КОСАРЕВ Ю.Г. Системный подход к построению трансляторов для вычислительных систем. -"Вычислительные системы", Новосибирск, 1970, вып. 42, стр. 12-21.

7. ВЕЛЬБИЦКИЙ И.В. О метаязыке синтаксически управляемого транслятора. -"Вычислительные системы", Новосибирск, 1970, вып. 42, стр. 22-31.

8. ВЕЛЬБИЦКИЙ И.В., ЮЩЕНКО Е.Л. Метаязык, ориентированный для синтаксического анализа и контроля. -"Кибернетика", Киев, 1969, № 3, стр. 50-53.

9. РЕНДЕЛЛ Б., РАССЕЛ Л. Реализация АЛГОЛа-60, М., "Мир", 1967.

Поступила в ред.-изд.отд.
20. VI. 1972 г.