Номер 5

Сентябрь - Октябрь 1997

ISSN 0132-3474

РОССИЙСКАЯ АКАДЕМИЯ НАУК

ПРОГРАММИРОВАНИЕ

Главный редактор
В.П. Иванников

МАИК "НАУКА" "НАУКА"

ПРОГРАММИРОВАНИЕ, 1997, № 5, с. 33–35

ЯЗЫКИ И СИСТЕМЫ ПРОГРАММИРОВАНИЯ

УДК 681.3.06

БЫСТРЫЙ МЕТОД ЛЕКСИЧЕСКОГО АНАЛИЗА *

© **1997 г. В. М. Курочкин**
Вычислительный центр РАН
117967 Москва, ул. Вавилова, 40
Поступила в редакцию 02.04.96 г.

Рассматривается быстрый метод лексического анализа, работающий по принципу табличного автомата. Алгоритм анализа затрачивает минимальное количество тактов на обработку каждого входного символа и не предусматривает никаких поисков в таблицах. За счет склеивания совпадающих лексем достигается максимальная экономия в составляемых таблицах.

Хотя организация лексического анализа в принципе довольно проста, это тем не менее относительно трудоемкий и медленный процесс (если сравнивать его, например, с однопроходным синтаксическим анализом в LL- или LR-грамматиках). Объясняется это тем, что, во-первых, как правило, сам лексический анализ объединяется и проводится одновременно с рядом других действий – заполнением таблицы индентификаторов и констант, поиском в этих таблицах (с помощью функции хеширования или без нее), переводом числовых констант из системы записи их на входном языке в двоичную систему, экономией констант и идентификаторов в соответствующих таблицах и т.п., а во-вторых, длина входного файла для лексического анализатора существенно больше, чем длина файла из лексем для синтаксического анализа.

Предлагаемый здесь метод лексического анализа основан на использовании автомата, работа которого управляется двумерной таблицей. Каждый встретившийся во входном тексте символ определяет вход в таблицу, и тем самым, как правило, один такт работы автомата. Более или менее при любой системе команд компьютера выделение клетки в двумерной таблице требует одного и того же, причем небольшого (2–5) числа операций машины. К этому и сводится, в основном, работа, выполняемая машиной при лексическом анализе текста программы. Число столбцов в таблице в процессе анализа не меняется и равно числу символов входного языка (можно считать его равным 256). Число строк таблицы автомата, правда, может возрастать, и при обработке больших программ может достигать нескольких тысяч. Более аккуратно – оно равно числу разных начал лексем в программе, и, таким образом, по сравнению со стандартными методами, может увеличиться не более чем в число раз, равное максимальной длине используемых в программе идентификаторов. При этом время обработки одной лексемы зависит только от ее длины и не зависит от размера программы. Возможны два варианта организации и использования таблицы, в принципе не очень сильно отличающиеся один от другого. В одном из вариантов каждое появление нового, не встречавшегося до этого начала лексемы вызывает добавление в таблицу новой строки с некоторым начальным стандартным заполнением клеток этой строки. В другом варианте в самом начале работы анализатора для таблицы выделяется достаточно места в памяти ("с избытком") и производится ее стандартное заполнение. Если не учитывать времени "роста" таблицы и времен (необходимых и одинаковых при любом методе лексического анализа) для запоминания идентификаторов и (перевода и) запоминания констант, то время обработки всех встречающихся в дальнейшем, и, как правило, совпадающих с уже встретившимися ранее впервые лексем будет линейно зависеть от длины

*Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований, проект 96-0101757.

самих лексем и не будет зависеть от размера таблицы. При этом автоматически будет производиться слияние совпадающих лексем – как идентификаторов, так и констант, независимо от их типа, например, совпадающих по написанию строковых констант, чего не делает большинство из существующих лексических анализаторов.

Начальное заполнение таблицы автомата. Перед работой лексического анализатора производится стандартная для данного входного языка настройка автомата в соответствии с перечисленным в описании лексики языка набором служебных слов. К служебным словам целесообразно отнести (или по крайней мере чисто формально так же с ними работать) многосимвольные ограничители, такие, как $**$, $:=$, $>=$ *b* и т.п. Для каждого начала служебного слова в таблице автомата отводится строка, а в клетке, соответствующей символу возможного продолжения этого начала, пишется номер строки продолжения и ключ "продолжить чтение" (о ключах будет сказано ниже). Так, если номера строк, соответствующих началам *e*, *el*, *els*, *else*, *elsi*, *elsif*, *en*, *end*, *ent*, *entr*, *entry*, *ex*, *exi*, *exit* получили номера 2, 3, 4, 5, 6, 7, 12, 13, 14, 15, 16, 21, 22, 23, то в клетках *l*, *n*, *x* строки 2 пишутся номера 3, 12, 21; в клетке *s* строки 3 – номер 4; в клетках *e*, *i* строки 4 – номера 5, 6; в клетке *f* строки 6 – 7; в клетках *d*, *t* строки 12 – 13, 14 и т.д.; в клетке *t* строки 22 – 23. В клетках, соответствующих ограничителю, например, +, >, "пробел" и т.п., означающих конец служебного слова, пишется признак конца лексемы и ставится номер соответствующей лексемы – служебного слова. Для языка Ada, который имеет 63 служебных слова и на котором моделировался описываемый метод, построенная таким образом таблица содержит 230 строк.

Регулярная работа автомата. Программа лексического анализа считывает из входного потока символ за символом. Считывание одного символа вызывает выполнение одного такта работы автомата, определяемого содержимым одной клетки таблицы автомата. Нужная клетка находится на пересечении текущей строки таблицы автомата и столбца, соответствующего считанному символу. В этой клетке находится ключ, определяющий вид работы автомата, и, возможно, номер строки одной из таблиц (с прямым входом!), содержащих необходимую дополнительную информацию. Возможные значения ключей:

- ошибочный символ во входном потоке (дополнительная информация – адрес программы реакции на соответствующую ошибку);
- продолжить чтение символов входного потока (этот вид работы выполняется в тех случаях, когда прочитанная часть лексемы, включая и последний символ, совпадает с уже встречавшимся ранее началом лексемы или с помещенным в таблицу автомата началом одного из служебных слов; дополнительная информация – номер строки таблицы автомата, соответствующей прочитанному началу лексемы; эта информация позволяет непосредственно продолжить работу автомата);
- конец служебного слова (дополнительная информация – номер служебного слова, которые предварительно были пронумерованы каким-либо произвольным способом; в выходной поток посылается тип лексемы, равный номеру служебного слова);
- конец идентификатора (дополнительная информация – новый или уже встречавшийся идентификатор; если новый, то он помещается в таблицу идентификаторов, а его номер записывается в данную клетку и в клетку дополнительного специального столбца таблицы; если идентификатор уже встречался, то его номер уже находится в клетке дополнительного столбца; в выходной поток посылается тип лексемы – идентификатор и его номер);
- конец числовой константы (дополнительная информация аналогична той, которая стоит при окончании обработки идентификатора; дополнительно производится перевод числовой константы в нужную систему счисления, а сама константа как в исходном, так и в конечном, т.е. двоичном представлении помещается в таблицу констант).

Если необходимо по синтаксису различать константы различных типов, например, *real*,

integer и др., то соответствующие сведения содержатся в ключе; со строковыми константами действуют совершенно аналогично. Все числовые константы, идентификаторы, строки и т.п. можно помещать в объединенную таблицу, а можно по тем или иным соображениям их разъединить. В любом случае номер лексемы означает прямой адрес этой лексемы в соответствующей таблице, а тип (т.е. ключ) определит, в какой именно таблице находится лексема. Возможны и небольшие вариации в организации таблицы автомата (например, можно не помещать в таблицу строковые константы, тем самым не производя слияния совпадающих строковых констант и экономя на размерах таблицы). Необходимо лишь сохранять основной принцип: каждый считанный из входного потока символ вызывает выполнение одного такта работы автомата, причем никаких поисков в таблицах не производится и нужная информация получается прямым входом в соответствующие таблицы.

Таким образом, предлагаемый метод позволяет значительно сократить время выполнения всех этапов лексического анализа, связанных с обработкой таблиц, что особенно существенно для программ большого объема. На основе описанного метода был реализован генератор лексических анализаторов, который был проверен на лексике языка Ada83.