**Разновидности алгоритмов разбора**

Имея в виду интерпретацию Де Ремера, можно представить себе и различные подходы к решению задачи разбора.

Если подбор костей удается осуществлять так, что однажды поставленную кость никогда не придется убирать, мы имеем дело с детерминированным алгоритмом разбора — выбор применяемого правила грамматики всегда однозначен.

Если принятые решения о выборе типа домино приходится отменять — алгоритм работает с возвратами, он недетерминирован.

Детерминированные алгоритмы эффективней, и конечно, всегда существует стремление найти и использовать такой алгоритм для синтаксического анализа.

Если дерево строится сверху вниз от начального нетерминала в сторону терминальной цепочки, алгоритм относится к классу нисходящих; от цепочки в сторону корня дерева — он восходящий (рис. 5).

Можно вначале подбирать домино для левых символов терминальной цепочки, а можно вначале для правых - соответственно говорят о левосторонних или правосторонних алгоритмах разбора.

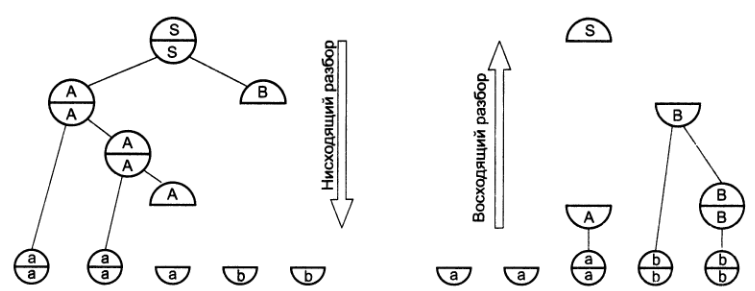


Рис.5 Начало нисходящего и восходящего разбора в грамматике G6

**Эквивалентность и однозначность грамматик**

Возьмем для примера грамматику арифметических выражений

G5:E→E + E|E-E|E\*E|E/E|a|b|c| (E).

Рассмотрим деревья вывода терминальных цепочек в этой грамматике.

На рис. 6 показаны два различных дерева разбора выражения а + b\* с.

Возможность построить эти деревья убеждает в том, что цепочка а + b\* с действительно принадлежит языку арифметических выражений.

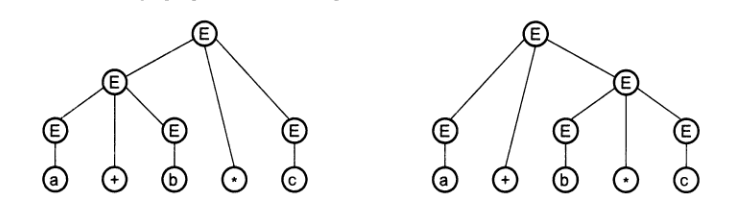


Рис.6 Деревья разбора цепочки а + b \* с в грамматике G5

Но два разных дерева дают две разные трактовки структуры этой цепочки.

Дерево слева объединяет а + b в одно подвыражение, которое затем участвует в роли операнда в операции умножения. Такая трактовка не соответствует общепринятому приоритету арифметических операций — умножение должно выполняться раньше сложения.

Дерево справа представляет структуру, соответствующую правильному порядку выполнения операций.

Грамматика G называется однозначной, если любой сентенции х∈ L(G) соответствует единственное дерево вывода.

Грамматика G5 неоднозначна, и это, безусловно, ее серьезный недостаток, который не позволяет применить ее на практике для определения языка выражений, поскольку эта грамматика не позволяет однозначным образом выявить структуру выражения.

Попробуем построить однозначную грамматику выражений. Для этого используем еще один нетерминал, обозначив его X

G7: E→X|E + X|E-X|E\*X|E/X

Х→а|b|с| (Е)

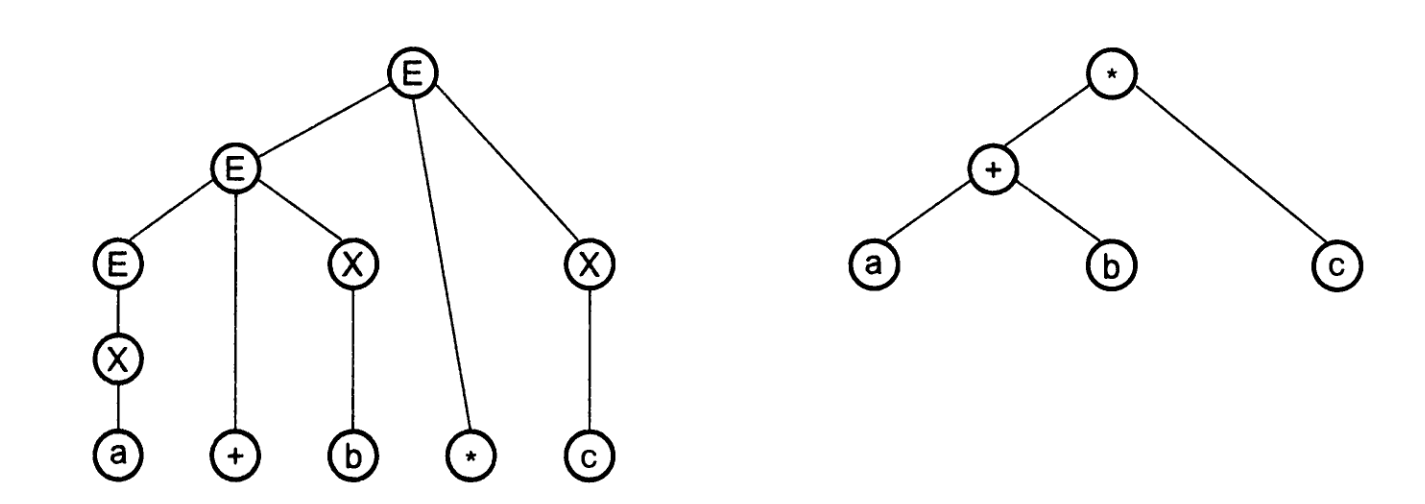


Рис.7 Дерево разбора выражения а + b \* с в грамматике G7

Содержательно X можно понимать как операнд выражения. Теперь для цепочки а + b \* с можно построить единственное дерево разбора. Оно показано на рис. 7 слева. Можно убедиться, что в грамматике G7 единственное дерево вывода соответствует любому правильному выражению. Грамматика G7 однозначна.

Справа на рисунке показано редуцированное (упрощенное) дерево того же выражения в грамматике G7. Такие деревья могут служить для представления структуры выражений в трансляторе. Будем называть их семантическими деревьями.

Семантическое дерево может быть получено из дерева разбора устранением нетерминальных вершин и помещением знаков операций во внутренние вершины, в то время как операнды остаются листьями дерева.

Несмотря на однозначность, G7 непригодна для использования в трансляторе, поскольку приписывает выражениям неподходящую структуру.

Уже на примере цепочки а + b\* с (см. рис. 7) видно, что операции и операнды связываются неправильно. Нетрудно заметить, что G7 предполагает выполнение операций без учета их приоритета в порядке слева направо.

Исправить положение можно, определив нетерминалы для двух категорий подвыражений — слагаемых и множителей. После этого выражение представляется как последовательность слагаемых, разделенных знаками плюс и минус. В свою очередь слагаемые образуются из элементарных операндов — множителей, соединенных знаками умножения и деления.

Слагаемые обозначим Т (от term — элемент), множители — М(от multiplier).

G8: E→T|E+T|E-T

T→M|T\*M|T/M

М→ а |b|с| (Е)

Дерево вывода цепочки а + b \* с в грамматике G8 показано на рис. 8. Эта грамматика однозначна и приписывает арифметическим выражениям структуру, соответствующую правильному порядку выполнения операций.

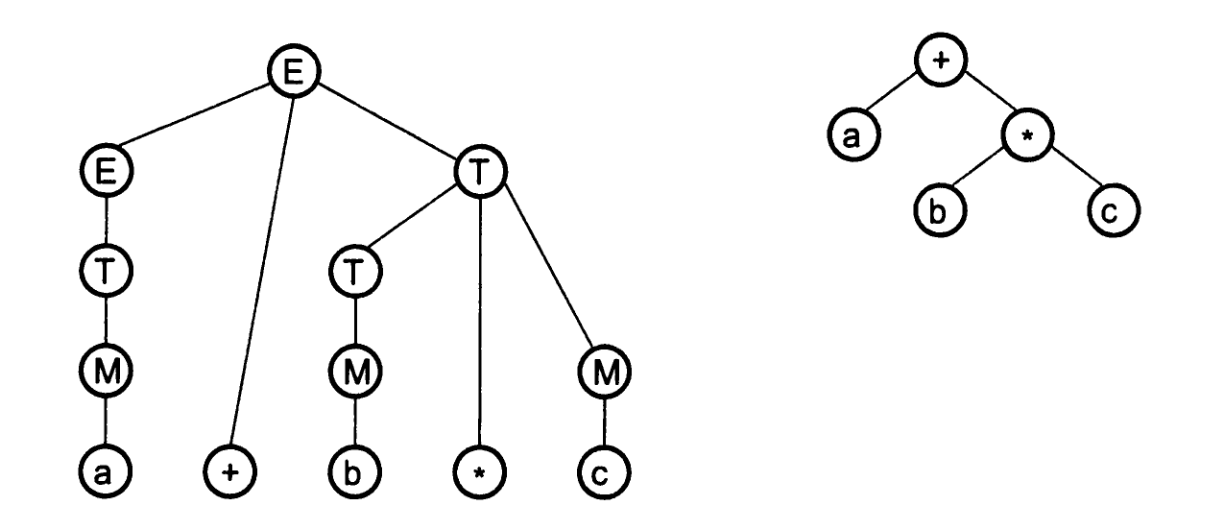


Рис.8 Дерево выражения а + b\*с в грамматике G8

Все три рассмотренные грамматики выражений (G5, G7, G8), хотя и различаются и обладают разными свойствами, задают один и тот же язык.

***Грамматики называются эквивалентными, если порождают один и тот же язык***.

Грамматики G5, G7, G8 эквивалентны, поскольку L(G5) = L(G7) = L(G8).

**Иерархия грамматик Н. Хомского**

Н. Хомский предложил разделение порождающих грамматик на четыре типа в зависимости от вида их правил.

**Тип 0**. Произвольные грамматики. На вид их правил не накладывается каких- либо ограничений. Правила имеют вид:

**α→β**,

где **α,β** — цепочки терминалов и нетерминалов. Цепочка **α** не должна быть пустой.

**Тип 1**. Контекстно-зависимые грамматики. Правила таких грамматик имеют вид:

**αAβ → αγβ**,

где **α**, **β**, **γ**— цепочки терминалов и нетерминалов; А — нетерминальный символ.

Такой вид правил означает, что нетерминал А может быть заменен цепочкой **γ** только в контексте, образуемом цепочками **α** и **β**.

**Тип 2**. Контекстно-свободные грамматики. Их правила имеют вид:

**А → γ**,

где, **А** — нетерминал; **γ** — цепочка терминалов и нетерминалов. Характерная особенность — в левой части правил всегда один нетерминальный символ.

**Тип 3**. Автоматные грамматики. Все правила автоматных грамматик имеют одну из трех форм:

**А→аВ**

**А→а**

**А →ε,**

где **А,В** — нетерминалы; **а** — терминал; **ε** — пустая цепочка.

***Автоматные грамматики называют также регулярными.***

Как можно видеть, грамматики типа 1 являются частным случаем грамматик типа 0, грамматики типа 2 — частный случай контекстно-зависимых грамматик, автоматные — частный случай контекстно-свободных. То есть грамматика типа 3 является и грамматикой типа 2, и типа 1, и типа 0.

Однако в дальнейшем, если не оговорено особо, будет иметься в виду, что, например, контекстно-свободной называется грамматика, не являющаяся автоматной.

Языки, порождаемые грамматиками типа 0-3, называются соответственно языками без ограничений, контекстно-зависимыми, контекстно-свободными и автоматными (регулярными) языками.

Но контекстно-свободным языком называют язык, для которого существует порождающая его контекстно-свободная, но не автоматная грамматика.

Такой же подход применяется к контекстно-зависимым языкам и языкам без ограничений.

**Примеры грамматик различных типов**

Рассмотрим грамматику G2, порождающую язык L2 = {anbncn | n > 0}.

G2: S → aSBc (тип 2)

S → abc (тип 2)

сВ → Bс (тип 0)

bВ→bb (тип1)

S → е (тип 3)

Справа около каждого правила помечен тип грамматики, к которой оно может быть отнесено. Типом грамматики естественно считать минимальный из типов ее правил. Следовательно, грамматика G2 — это грамматика типа 0 — произвольная.

Примером контекстно-свободной грамматики может служить грамматика арифметических выражений. С помощью контекстно-свободных грамматик задается и синтаксис языков программирования. Грамматики этого класса будут подробно обсуждаться и в дальнейшем, сейчас же возьмем конкретный пример.

G9: N→a (1)

N→Na (2)

N→Nb (3)

Это грамматика типа 2, поскольку правила (2) и (3) относятся именно к этому типу.

Рассмотрим язык L9 = L(G9), порождаемый этой грамматикой. Цепочка а принадлежит L9 по правилу (1). Если к правильному предложению N языка приписать справа символ a, то снова получится правильное предложение (по правилу (2)). Аналогично, приписывание к N символа b снова дает предложение языка L9.

Принадлежащие языку L9 цепочки начинаются символом а, за которым могут следовать а и b в произвольном порядке. Если под а понимать любую латинскую букву, a b воспринимать как цифру, то G9 можно считать грамматикой идентификаторов. Она порождает последовательности букв и цифр, начинающиеся с буквы, которые используются в языках программирования в роли имен переменных, типов и т. д.

Опираясь на такую содержательную трактовку G9 и L9, попытаемся сконструировать автоматную грамматику, порождающую язык идентификаторов.

Первое правило грамматики G9 может быть сохранено. Оно, во-первых, соответствует одному из допустимых видов правил автоматных грамматик, во-вторых, определяет, что идентификатор, состоящий из одного символа, может быть только буквой.

N→a (1)

Нетерминал N — это начальный символ нашей грамматики. Он и обозначает само понятие «идентификатор». Как синоним термина «идентификатор» будем использовать также слово «имя». Можно считать, что название N происходит от Name — имя.

Обозначим В часть идентификатора, которая может следовать за первой буквой. Тогда можно записать правило (2):

N→aB (2)

Запишем правила для В. «Хвост» может быть буквой или цифрой:

В→а (3)

В→b (4)

Если В — это «хвост» идентификатора, то, записав его за буквой или цифрой, снова получим правильный «хвост»:

В →аВ (5)

В → bВ (6)

Обозначим сконструированную грамматику G10.

Она автоматная, поскольку все ее правила удовлетворяют ограничениям автоматных грамматик.

G10: N→a (1)

N→aB (2)

В→а (3)

В→b (4)

В→аВ (5)

B→bВ (6)

Грамматика G10 эквивалентна G9, поскольку порождает тот же язык:

L(G10)=L(G9).

Этот язык — язык идентификаторов — следует считать автоматным.