Определение формальной грамматики и языка

Определение. Конечное множество символов, неделимых в данном рассмотрении, в *теории формальных грамматик* называется *словарем*, или алфавитом, а символы, входящие в множество, - *буквами алфавита*.

Например, алфавит A = {a, b, c, +, !} содержит 5 букв, а алфавит B = {00, 01, 10, 11} содержит 4 буквы, каждая из которых состоит из двух символов.

Определение. Последовательность букв алфавита называется *словом* или цепочкой в этом алфавите. Число букв, входящих в слово, называется его длиной.

Например, слово в алфавите A a = ab++c имеет длину l(a) = 5, а слово в алфавите B b = 00110010 имеет длину l (b) = 4.

Если задан алфавит A, то обозначим A* множество всевозможных цепочек, которые могут быть построены из букв алфавита A. При этом предполагается, что пустая цепочка, которую обозначим знаком $, также входит в множество A*.

Определение. *Формальной порождающей грамматикой* Г называется следующая совокупность четырех объектов: Г = { $V_т$, V_A, <I> $\in$ V_A, R }, где

Vт - *терминальный алфавит* (словарь); буквы этого алфавита называются *терминальными символами*; из них строятся цепочки, порождаемые грамматикой;

V_A - *нетерминальный, вспомогательный алфавит* (словарь); буквы этого алфавита (*нетерминальные символы*) используются при построении цепочек; они могут входить в промежуточные цепочки, но не должны входить в результат порождения;

<I> - *начальный символ грамматики* <I> $\in$ V_A;

R - *множество правил вывода*, или порождающих правил вида $\alpha \longrightarrow \beta$, где α и β - цепочки , построенные из букв алфавита $V_т \cup V_A$, который называют *полным алфавитом* (словарем) грамматики Г.

В множество правил грамматики могут также входить правила с пустой правой частью вида <E> $\longrightarrow$. Чтобы избежать неопределенности из-за отсутствия символа в правой части правила, условимся использовать символ пустой цепочки, записывая такое правило в виде <E> $\longrightarrow$ $.

Чтобы установить правила построения цепочек, порождаемых грамматикой , введем следующие понятия.

Определение. Пусть r = т $\rightarrow$ Υ - правило грамматики Г и α = χ' т χ'' - цепочка символов, причем χ', χ'' $\in$ ($V_т \cup V_A$) *. Тогда цепочка β = χ' Υ χ'' может быть получена из цепочки α путем применения правила r (т.е. заменой в цепочке α т на Υ). В этом случае говорят, что цепочка β *непосредственно выведена* из цепочки α и обозначают $\alpha \Rightarrow \beta$.

Определение. Если задана совокупность цепочек Ω = (ϖ_0 , ϖ_1 ,..., ϖ_n), таких что существует последовательность непосредственных выводов:

$$\varpi_0 \Rightarrow \varpi_1 , \varpi_1 \Rightarrow \varpi_2 , \dots , \varpi_{n-1} \Rightarrow \varpi_n ,$$

то такую последовательность называют *выводом* ϖ_n из ϖ_0 в грамматике Г и обозначают

$$\varpi_0 \Rightarrow^* \varpi_n$$

Определение. Множество конечных цепочек терминального алфавита $V_т$ грамматики Г, выводимых из начального символа <I>, называется *языком, порождаемым грамматикой* Г и обозначается L(Г).

$$L(\Gamma) = \{ \varpi \in V_т^* \mid <I> \Rightarrow^* \varpi \}$$

Определение. Если язык, порождаемый грамматикой Г, не содержит ни одной конечной цепочки (конечного слова), то он называется *пустым*.

Утверждение. Для того, чтобы язык L(Г) не был пустым, в множестве R должно быть хотя бы одно правило вида r = χ ⟶ ψ, где ψ ∈ $V_т^*$ и должен существовать вывод

$$<I> \Rightarrow^* \chi$$

Типы формальных языков и грамматик. Классификация по Хомскому

В теории формальных языков выделяются 4 типа *грамматик*, которым соответствуют 4 типа языков. Эти грамматики выделяются путем наложения усиливающихся ограничений на правила грамматики

Грамматики типа 0

Грамматики типа 0, которые называют грамматиками общего вида, не имеют никаких ограничений на правила порождения. Эти грамматики порождают естественные языки.

Любое правило

$$r = \eta \longrightarrow \psi$$

может быть построено с использованием произвольных цепочек η , ψ ∈ $(V_т \cup V_A)^*$. Например,

<T><W> ⟶ <W><T> или x<A>b<C><D> ⟶ x<H><D>

Грамматики типа 1

Грамматики типа 1, которые называют также контекстно-зависимыми грамматиками, не допускают использования любых правил. Правила вывода в таких грамматиках должны иметь вид:

χ_1<A>χ_2 ⟶ χ_1 ω χ_2,

где χ_1, χ_2 - цепочки, возможно пустые, из множества $(V_т \cup V_A)^*$, символ <A> $\in V_A$ и цепочка ω $\in (V_т \cup V_A)^*$. Цепочки χ_1 и χ_2 остаются неизменными при применении правила, поэтому их называют контекстом (соответственно левым и правым), а грамматику - контекстно-зависимой.

Грамматики типа 1 значительно удобнее на практике, чем грамматики типа 0, поскольку в левой части правила заменяется всегда один нетерминальный символ, который можно связать с некоторым синтаксическим понятием, в то время как в грамматике типа 0 можно заменять сразу несколько символов, в том числе и терминальных.

Например, грамматика:

$Г_{1.4}$:

$V_т$ = {a, b, c, d}, V_A = {<I>, <A>, <B>}

R = { <I> ⟶ aA<I>,

A<I> ⟶ AA<I>,

AAA ⟶ A<B>A,

A ⟶ b,

b<B>A ⟶ bcdA,

b<I> ⟶ ba }

является контекстно-зависимой, поскольку второе и шестое правила имеют непустой левый контекст, а третье и пятое правила содержат оба контекста. Вывод в такой грамматике может иметь вид:

<I> ⇒ a<A><I> ⇒ a<A><A><I> ⇒ ab<A><I> ⇒ abb<I> ⇒ abba

Грамматики типа 2

Грамматики типа 2 называют контектно-свободными и бесконтекстными грамматиками (КС-грамматики или Б-грамматики).

Правила вывода таких грамматик имеют вид:

<A> ⟶ α,

где <A> $\in V_A$ и α $\in (V_т \cup V_A)^*$.

Очевидно, что эти правила получаются из правил грамматики типа 1 при условии $\chi_1 = \chi_2 = \$$. Поскольку контекстные условия отсутствуют, то правила КС-грамматик получаются проще, чем правила грамматик типа 1. Именно такие грамматики используются для описания языков программирования.

Примером КС-грамматики может служить следующая:

$Г_{1.5}$:

$V_т$ = {a, b}, V_A = {<I>},

R = { <I> ⟶ a<I>a,

<I> ⟶ b<I>b,
<I> ⟶ aa,
<I> ⟶ bb}

Эта грамматика порождает язык, который состоит из цепочек, каждая из которых в свою очередь состоит из двух частей, цепочки $\beta \in V_т^*$ и зеркального отображения этой цепочки β'.

$L(\Gamma_5) = \{ bb' \mid b \in V_т^+ \}$,

где $V_т^+$ - это множество $V_т^*$ без пустой цепочки. С помощью правил этой грамматики может быть построена, например, следующая цепочка:

<I> ⇒ a<I>a ⇒ ab<I>ba ⇒ aba<I>aba ⇒ ababbaba

Грамматики типа 3

Грамматики типа 3 называют автоматными грамматиками (А - грамматиками). Правила вывода в таких грамматиках имеют вид:

<A> ⟶ a или <A> ⟶ a<B> или <A> ⟶ <B>a,

где $a \in V_т$, <A>, <B> $\in V_A$, причем грамматика может иметь только правила вида <A> ⟶ a<B> - правосторонние правила, либо только вида <A> ⟶ <B>a - левосторонние правила. Примерами автоматных грамматик могут служить правосторонняя грамматика $\Gamma_{1.6}$ и левосторонняя грамматика $\Gamma_{1.7}$.

$\Gamma_{1.6}$
$V_т$ = {a, b}, V_A = {<I>, <A>},
R = { <I> ⟶ a<I>,
<I> ⟶ a<A>,
<A> ⟶ b<A>,
<A> ⟶ b<Z>,
<Z> ⟶ $ }

$\Gamma_{1.7}$
$V_т$ = {a, b}, V_A = {<I>, <A>},
R = { <I> ⟶ <A>b,
<A> ⟶ <A>b,
<A> ⟶ <Z>a,
<Z> ⟶ <Z>a,
<Z> ⟶ $ }

Эти грамматики являются эквивалентными и порождают язык

$L(\Gamma_7) = \{a...ab...b \mid n,m > 0\}$

Между множествами языков различных типов существует отношение включения:

$\{ L_{типа\ 3} \} \subset \{ L_{типа\ 2} \} \subset \{ L_{типа\ 1} \} \subset \{ L_{типа\ 0} \}$

Доказано, что существуют языки типа 0, не являющиеся языками типа 1, языки типа 2, не являющиеся языками типа 1, и языки типа 3, не являющиеся языками типа 2.

Учитывая, что наибольшее практическое применение находят грамматики типа 2 и типа 3, дальнейшее изложение посвящается рассмотрению именно этих типов грамматик.

Вывод в КС-грамматиках и правила построения дерева вывода

Формальные грамматики позволяют задавать языки, представляющие множества цепочек, построенных по определенным правилам. Используемый способ задания позволяет построить любую цепочку, принадлежащую языку. Чтобы сделать процесс построения, называемый выводом, наглядным, его изображают в виде графа, точнее, в виде дерева, которое называют *синтаксическим деревом* или деревом вывода. Учитывая, что вывод любой цепочки языка, принадлежащей языку, порождаемому заданной грамматикой, должен начинаться с начального символа, правила построения дерева можно сформулировать так:

1. В качестве *начальной вершины* или корня дерева возьмем вершину, которую обозначим начальным символом грамматики <I>; эта вершина образует нулевой ярус дерева
2. Если при выводе цепочки на очередном шаге используется правило грамматики <A> $\longrightarrow$ α и вершина, помеченная нетерминалом <A>, расположена на ярусе с номером k-1, то к построенному дереву нужно добавить столько вершин, сколько содержится символов в цепочке α, расположить эти вершины на ярусе k, обозначить их символами цепочки a и соединить эти вершины дугами с вершиной <A>. Результатом вывода является множество конечных узлов - листьев, которые выписываются при обходе дерева слева - вниз - направо - вверх . Рассмотрим, например, грамматику $Г_{1.8}$:

$Г_{1.8}$:
Vт = {a, b}, Va = {<I>},
R = {<I> $\longrightarrow$ a<I>b,
<I> $\longrightarrow$ ab },

которая порождает язык $L(Г_8)$ = {aa...abb...b}, где a и b повторяются по n раз, n=1,2,...

Вывод цепочки с помощью правил этой грамматики имеет вид:

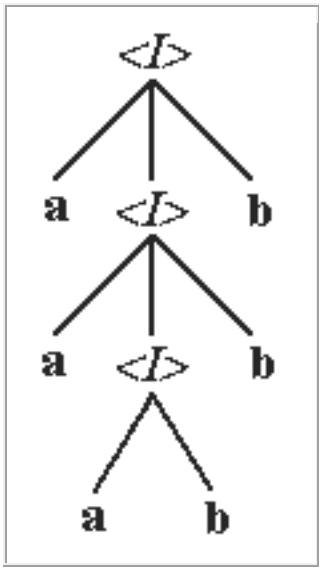

Рис. 1.

Синтаксический разбор

Вывод цепочки с помощью правил грамматики может быть задан не только в виде синтаксического дерева. Если пронумеровать правила грамматики, то последовательность номеров используемых правил также задает вывод.

Определение. Последовательность номеров правил грамматики Г, применение которых позволяет построить вывод рассматриваемой цепочки σ из начального символа грамматики, называется *синтаксическим разбором* σ.

Например, в следующей грамматике

$Г_{1.9}$:

$V_т$ = { i, +, * , (,) }, V_A = {<E>, <T>, <P>}

R ={ (1) <E> ⟶<E> +<T>,
(2) <E> ⟶<T>,
(3) <T> ⟶<T> *<P>,
(4) <T> ⟶<P>,
(5) <P> ⟶(E),
(6) <P> ⟶ i },

правила которой пронумерованы, вывод

<E> ⇒<E> +<T> ⇒ <T> +<T> ⇒ <T> *<P> +<T> ⇒
<P> *<P> +<T> ⇒ i * <P> +<T> ⇒ i * i +<T> ⇒
i * i +<P> ⇒ i * i + i

имеет синтаксический разбор [1,2,3,4,6,6,4,6].

Если в процессе построения вывода появляются промежуточные цепочки, содержащие несколько нетерминальных символов, то можно продолжать вывод, заменяя любой из них. Таким образом, одни и те же правила могут быть использованы при выводе цепочки в разном порядке.

Например, вывод цепочки i + i в грамматике $Г_{1.9}$ может быть получен десятью различными способами.

Левый и правый выводы

Среди всевозможных выводов наибольший интерес представляют следующие два типа выводов.

Определение. Если при построении вывода цепочки α при каждом применении правила заменяется самый левый нетерминальный символ, то такой вывод называется *левым* или левосторонним выводом α. Если при построении вывода α, всегда заменяется самый правый нетерминальный символ промежуточной цепочки, то вывод называется *правым* или правосторонним выводом α.

Например, приведенный выше вывод цепочки i * i + i в грамматике $Г_{1.9}$ является левосторонним выводом. Следует отметить, что различным выводам цепочки i+i в грамматике $Г_{1.9}$ соответствует одно и то же синтаксическое дерево. Аналогичная ситуация имеет место и при выводе цепочки i * i + i.

Неоднозначные и эквивалентные грамматики

Существуют грамматики, в которых одна и та же цепочка может быть получена с помощью различных выводов. Например, в грамматике $Г_{1.10}$ цепочка abc может быть получена с помощью двух различных выводов, и ей соответствуют два различных *синтаксических дерева*.

$Г_{1.10}$

$V_т$ = {a, b, c, d}, V_A = {<I>, <A>, <B>},

R = { <I> ⟶ <A><B>,

<A> ⟶ ac,

<B> ⟶ b,

<B> ⟶ cb}.

Первый вывод этой цепочки имеет вид:

1) <I> ⇒ <A><B> ⇒ <A>b ⇒ acb,

а второй можно получить так:

2) <I> ⇒ <A><B> ⇒ <A>cb ⇒ acb

Этим выводам соответствуют разные синтаксические деревья и разборы:

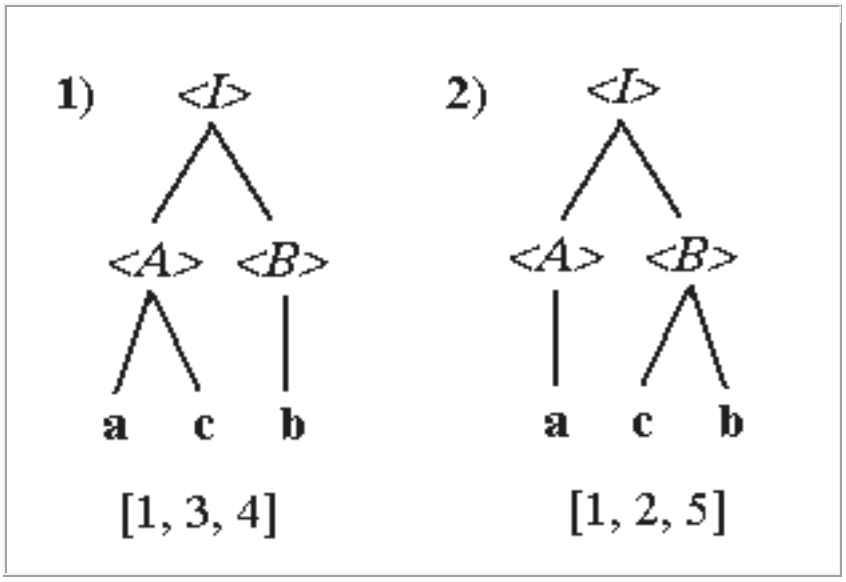

Рис. 1.

Следующая грамматика также допускает построение одной и той же цепочки с помощью двух выводов, имеющих разные синтаксические деревья.

$Г_{1.11}$:

$V_т$ = {0, +}, V_A = {<I>},

R = { <I> ⟶ 0,

<I> ⟶ <I> + 0,

<I> ⟶ 0 +<I> }

Два вывода этой грамматики, порождающие одинаковые цепочки, имеют вид:

1) <I> ⇒ <I> + 0 ⇒ <I> + 0 + 0 ⇒ 0 + 0 + 0,

2) <I> ⇒ 0 + <I> ⇒ 0 + 0 +<I> ⇒ 0 + 0 + 0,

а синтаксические деревья, соответствующие этим выводам, можно изобразить так:

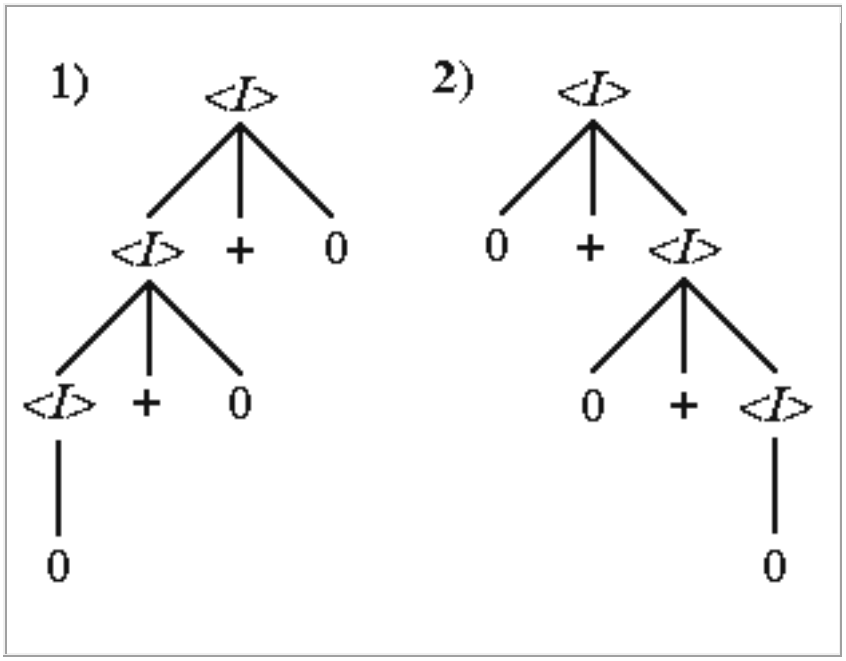

Рис. 2.

Рассмотренное свойство грамматик называется неоднозначностью. Оно может быть определено следующим образом.

Определение. Цепочка языка L(Г) называется *неоднозначной*, если для её вывода существует более чем одно синтаксическое дерево. Если грамматика Г порождает неоднозначную цепочку, то она называется неоднозначной.

Свойство неоднозначности является крайне нежелательным для искусственных языков, поскольку оно не позволяет однозначным образом восстановить дерево вывода по заданной цепочке языка.

В общем случае можно сделать следующий вывод:

1. каждой цепочке, выводимой в грамматике, может соответствовать одно или несколько синтаксических деревьев,
2. каждому синтаксическому дереву могут соответствовать несколько выводов,
3. каждому синтаксическому дереву соответствует единственный правый и единственный левый выводы.

Кроме того, следует подчеркнуть, что один и тот же язык может быть получен с помощью различных грамматик

Определение. Две грамматики $Г_1$ и $Г_2$ называются *эквивалентными*, если они порождают один и тот же язык, т.е.

$L(Г_1) = L(Г_2)$

Способы задания схем грамматик. Форма Наура-Бэкуса

Схема грамматики содержит правила вывода, определяющие синтаксис языка, или, другими словами, возможные компоненты и конструкции цепочек порождаемого языка. Для задания правил используются различные формы описания: *символическая*, *форма Наура-Бэкуса*, *итерационная форма* и *синтаксические диаграммы*.

В работах, связанных с рассмотрением общих свойств грамматик, обычно применяют символическую форму задания правил. Эта форма была рассмотрена в предыдущем параграфе. Она предусматривает использование в качестве элементов нетерминального словаря отдельных символов и стрелки в качестве разделителя правой и левой частей правила.

При описании синтаксиса конкретных языков программирования приходится вводить большое число нетерминальных символов, и символическая форма записи теряет свою наглядность. В этом случае применяют форму Наура-Бэкуса (ФНБ), которая предполагает использование в качестве нетерминальных символов комбинаций слов естественного языка, заключенных в угловые скобки, а в качестве разделителя - специального знака, состоящего из двух двоеточий и равенства. Например, если правила

<L> ⟶ <L> и <L> ⟶ <E> записаны в символической форме, и символ <L> соответствует синтаксическому понятию "список", а символ <E> - "элемент списка", то их можно представить в форме Наура-Бэкуса так:

<список>::= <элемент списка><список>,

<список>::= <элемент списка>

Чтобы сократить описание схемы грамматики, в ФБН разрешается объединять правила с одинаковой левой частью в одно правило, правая часть которого должна включать правые части объединяемых правил, разделенные вертикальной чертой. Используя объединение правил, для рассматриваемого примера получаем:

<список>::=<элемент списка><список>|<элемент списка>

Итерационная форма

Для получения более компактных описаний синтаксиса применяют *итерационную форму* описания. Такая форма предполагает введение специальной операции, которая называется итерацией и обозначается парой фигурных скобок со звездочкой. Итерация вида {a}* определяется как множество, включающее цепочки всевозможной длины, построенные с использованием символа a, и пустую цепочку.

{a}* = {$, a, aa, aaa, aaaa,...}

Используя итерацию для описания множества цепочек, задаваемых символическими правилами, для списка получаем:

<L> ⟶ <E> {<E>}*

Например, описание множества цепочек, каждая из которых должна начинаться знаком # и может состоять из произвольного числа букв x и y, может быть представлено в итерационной форме так:

<I> ⟶ #{x | y}*

В итерационных формах описания наряду с итерационными скобками часто применяют квадратные скобки для указания того, что цепочка , заключенная в них, может быть опущена. С помощью таких скобок правила:

<A> ⟶ x<A>y<B>z ***и*** <A> ⟶ x<B>z

могут быть записаны так:

<A> ⟶ x[<A>y]<B>z.

Синтаксические диаграммы

Для того, чтобы улучшить зрительное восприятие и облегчить понимание сложных синтаксических описаний, применяют представление правил грамматики в виде *синтаксических диаграмм*. Правила построения таких диаграмм можно сформулировать в следующем виде:

1. Каждому правилу вида <A> ⟶ a_1 | a_2 |...| a_k ставится в соответствие диаграмма, структура которой определяется правой частью правила.
2. Каждое появление терминального символа x в цепочке a_i изображается на диаграмме дугой, помеченной этим символом x, заключенным в кружок.

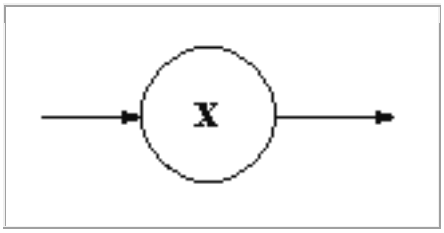

Рис. 1.

3. Каждому появлению нетерминального символа <A> в цепочке a_i ставится в соответствие на диаграмме дуга, помеченная символом, заключённым в квадрат.

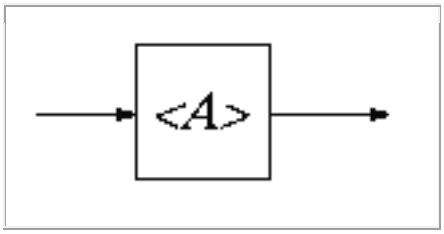

Рис. 2.

4. *Порождающее правило*, имеющее вид:

<A> ⟶ $a_1a_2...a_n$

изображается на диаграмме следующим образом:

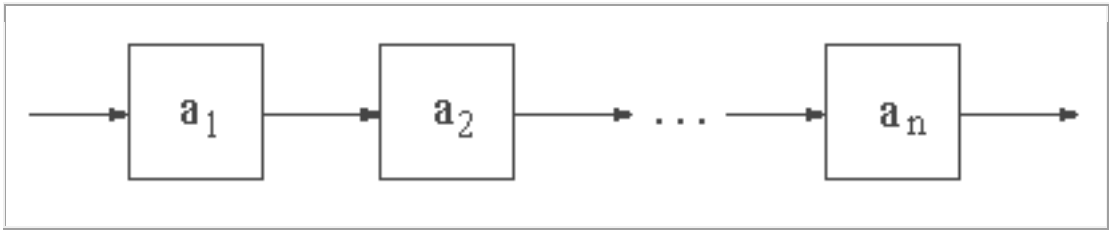

Рис. 3.

5. Порождающее правило, имеющее вид:

<A> ⟶ a_1 | a_2 | ... | a_n

изображается на диаграмме так:

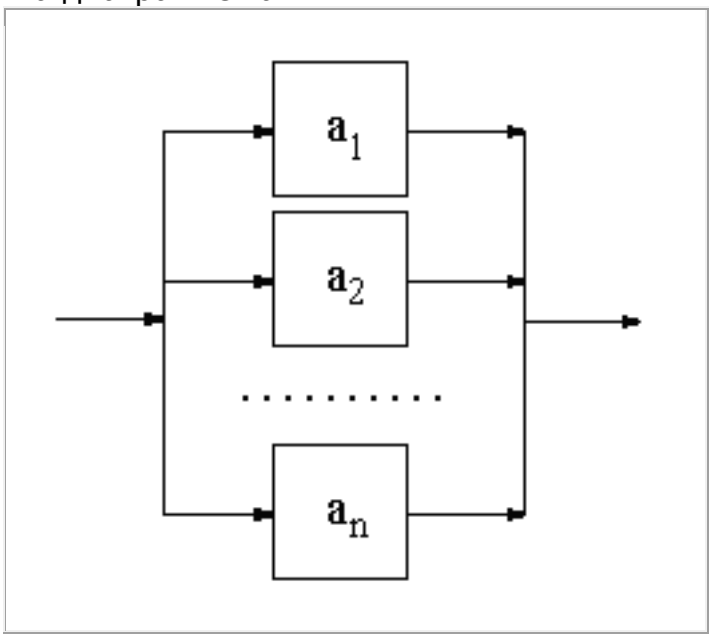

Рис. 4.

6. Если порождающее правило задано в виде итерации:

<A> ⟶ {a}*,

то ему соответствует диаграмма:

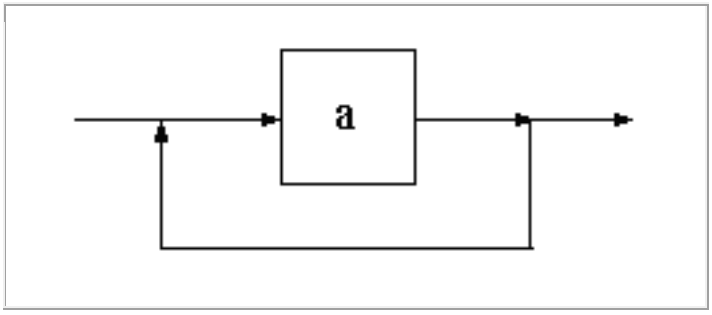

Рис. 5.

Число синтаксических диаграмм, которые можно построить для заданной схемы грамматики, определяется числом правил. Чтобы сократить число диаграмм, их объединяют, заменяя нетерминальные символы, входящие в диаграмму, построенными для них диаграммами.

Правила 3-6 предусматривают, что в качестве цепочки a_1 на объединенной диаграмме могут быть использованы диаграммы построенные для этих цепочек. В качестве примера рассмотрим следующую грамматику с начальным символом <A>:

$Г_{1.14}$:

$V_т$ = { x, +, (,) }, V_A = {<A>, <B>, <C>},

R = {<A> ⟶ x | (<B>),

<B> ⟶ <A><C>,

<C> ⟶ {+<A>}*}

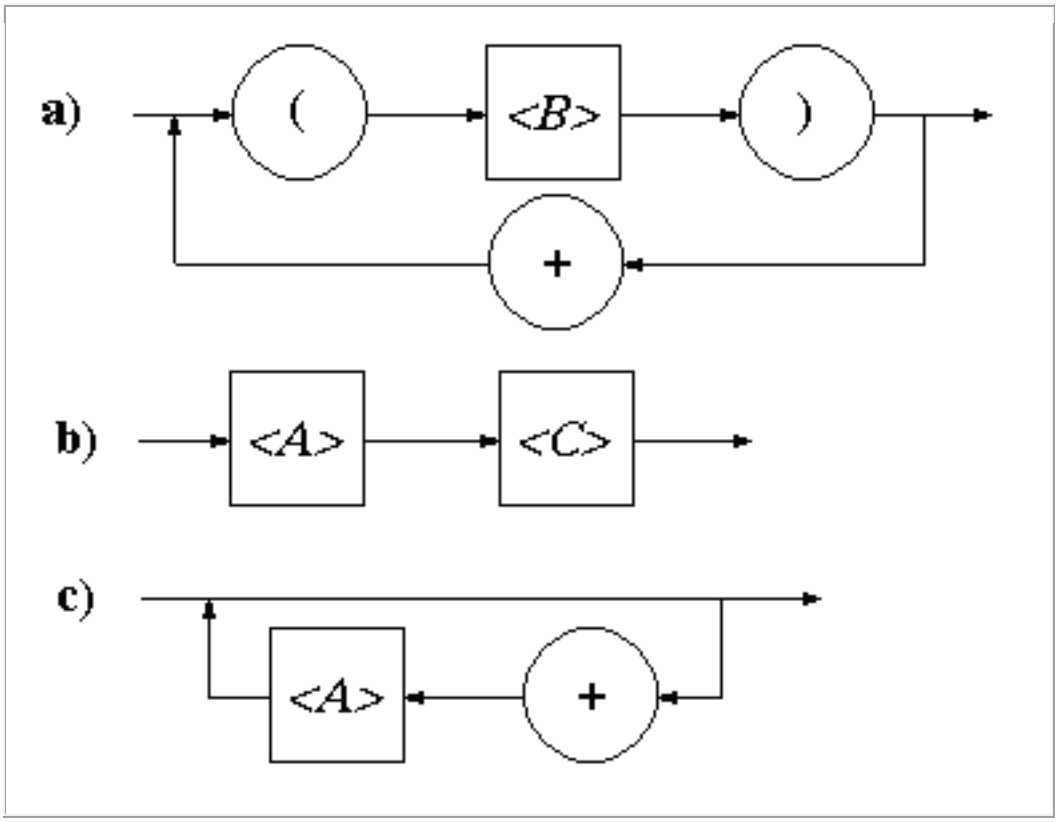

Рис. 6.

Заменяя нетерминальные символы, соответствующими диаграммами, получаем объединенную диаграмму в виде:

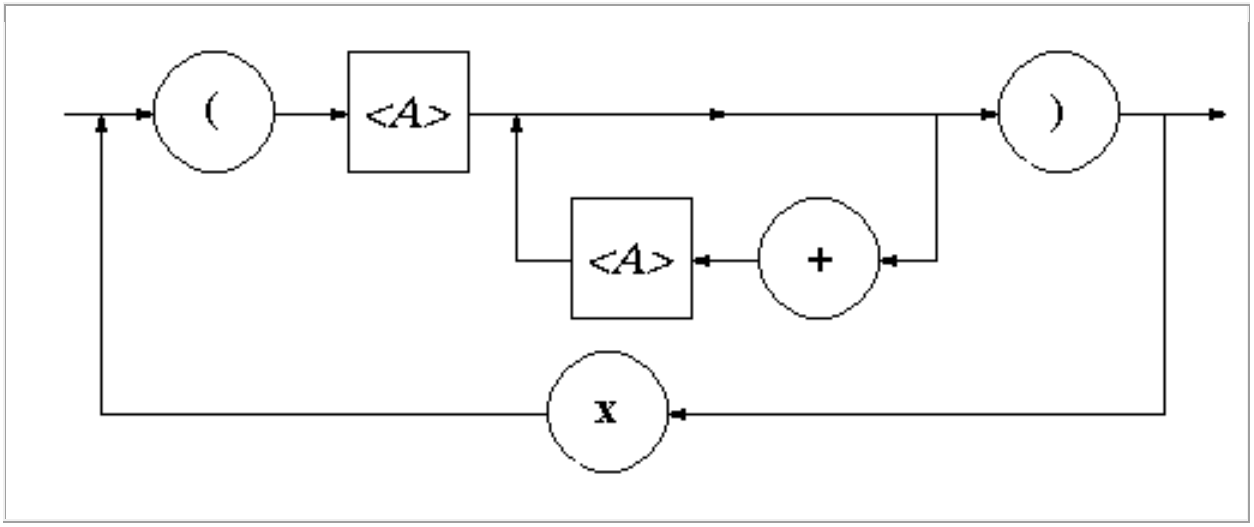

Рис. 7.