Лекція 7-8.

ТЕОРІЯ ФОРМАЛЬНИХ ГРАМАТИК

4. ФОРМАЛЬНЫЕ ГРАММАТИКИ И ЯЗЫКИ

4.1. Общие сведения

В последние три десятилетия появилось большое количество работ по общей теории языков и грамматик. Можно выделить четыре научных направления, которые удалось объединить по методам их исследования в одну общую задачу теории языков.

Первое из этих направлений связано с построением формальной, или математической, лингвистики, которая начала особенно быстро развиваться в тот период, когда были сформулированы вопросы машинного перевода. Эта проблема требовала формализации понятий словарь, грамматика, язык, их классификации и умения относить конкретные словари, грамматики, языки к определенному классу.

Интерес к задачам такого рода еще более усилился с появлением искусственных языков программирования. С появлением трансляторов проблема перевода во многом определила построение общей теории вычислительных машин, а сами языки программирования стали все более приближаться к формально построенным математическим конструкциям.

Независимо от указанных двух направлений развивалось построение формальных моделей динамических систем. Для создания продуктивной теории эти модели должны были быть, с одной стороны, достаточно узкими, а с другой - достаточно широкими, чтобы охватить некоторый общий класс прикладных задач. Типичным примером такого рода является модель конечного автомата. Эта модель позволяет описать многие процессы, заданные на конечных множествах и развивающиеся в счетном времени, что позволило создать для нее продуктивную теорию. Если в эту модель ввести бесконечность по какому-либо параметру, то это приводит к общему классу систем, например, к машинам Тьюринга.

Независимо и параллельно развивалась общая теория алгоритмов как ветвь современной математики. Развитие вычислительной техники поставило перед математической теорией алгоритмов новую задачу: стало необходимым классифицировать алгоритмы по различным признакам. Эквивалентность понятий "алгоритм" и "машина Тьюринга" позволила предположить, что поиски классификации алгоритмов окажутся связанными с поисками промежуточных моделей между моделями конечного автомата и машиной Тьюринга.

Таким образом, перечисленные четыре направления оказались тесно связанными. Теория языков, порожденная чисто лингвистическими задачами, оказалась в центре интересов математиков, занимающихся теорией алгоритмов, и ученых, разрабатывающих абстрактные модели динамических систем и теоретические основы автоматики.

Теория формальных языков и грамматик является разделом математической лингвистики - специфической математической дисциплины, ориентированной на изучение структуры естественных и искусственных языков. По характеру используемого математического аппарата теория формальных грамматик и языков близка к теории алгоритмов и теории автоматов.

На интуитивном уровне язык можно определить как некоторое множество предложений с заданной структурой, имеющих определенное значение. Правила, определяющие допустимые конструкции языка, составляют синтаксис языка. Значение, или смысл фразы, определяется семантикой языка.

Если бы все языки состояли из конечного числа предложений, то не было бы проблемы синтаксиса. Но, так как язык содержит неограниченное (или довольно большое) число правильно построенных фраз, то возникает проблема описания бесконечных языков с помощью каких-либо конечных средств.

Различают следующие виды описания языков:

1) порождающее, которое предполагает наличие алгоритма, последовательно порождающего все правильные предложения языка;

2) распознающее, которое предполагает наличие алгоритма, определяющего принадлежность любой фразы данному языку.

4.2. Основные понятия порождающих грамматик

Алфавит - это непустое конечное множество. Элементы алфавита называются символами.

Цепочка над алфавитом $\Sigma=\{a_1, a_2, \ldots, a_n\}$ есть конечная последовательность элементов a_i. Множество всех цепочек над алфавитом Σ обозначается Σ^*.

Длина цепочки x равна числу ее элементов и обозначается $|x|$. Цепочка нулевой длины называется пустой и обозначается символом ε. Соответственно, непустая цепочка определяется как цепочка ненулевой длины. Пусть имеется алфавит $\Sigma=\{a, b\}$. Тогда множество всех цепочек определяется следующим образом: $\Sigma^*=\{ \varepsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, \ldots\}$.

Цепочки x и y равны, если они одинаковой длины и совпадают с точностью до порядка символов, из которых они состоят.

Над цепочками x и y определена операция сцепления (конкатенации, произведения), которая получается дописыванием символов цепочки y за символами цепочки x.

Язык L над алфавитом Σ представляет собой множество цепочек над Σ. Необходимо различать пустой язык L=0 и язык, содержащий только пустую цепочку: L={ε}≠0.

Формальный язык L над алфавитом Σ - это язык, выделенный с помощью конечного множества некоторых формальных правил.

Пусть M и L - языки над алфавитами. Тогда конкатенация LM = {xy|x∈L, y∈M}. В частности, {ε}L = L{ε}=L. Используя понятие произведения, определим итерацию L^* и усеченную итерацию L^+ множества L:

$$L^+ = \bigcup_{i=1}^{\infty} L^i ,$$

$$L^* = \bigcup_{i=1}^{\infty} L^i ,$$

где i - степень языка, L определяется рекурсивно следующим образом:

$L^0 = \{\varepsilon\}$;

$L^1 = L$;

$L^{n+1} = L^n L = L\, L^n$;

$\{\varepsilon\}L = L\{\varepsilon\} = L$.

Например, если задан язык L={a}, тогда L^*={ε, a, aa, aaa, …}, L^+={a, aa, aaa, …}.

Порождающая грамматика - это упорядоченная четверка $G= (V_T, V_N, P, S)$, где

V_T - конечный алфавит, определяющий множество терминальных символов;

V_N - конечный алфавит, определяющий множество нетерминальных символов;

P - конечное множество правил вывода, т.е. множество пар следующего вида:

$$u \to v, \quad \text{где } u, v \in (V_T \cup V_N)^*;$$

S - начальный символ (аксиома грамматики), $S \in V_N$.

Из терминальных символов состоят цепочки языка, порожденного грамматикой. Аксиомой называется символ в левой части первого правила вывода грамматики.

Для того чтобы различать терминальные и нетерминальные символы, принято обозначать терминальные символы строчными, а нетерминальные символы заглавными буквами латинского алфавита.

В грамматике G цепочка x непосредственно порождает цепочку y, если $x = \alpha u \beta$, $y = \alpha v \beta$ и $u \rightarrow v \in P$, т.е. цепочка y непосредственно выводится из x, что обозначается $x => y$.

Языком, порождаемым грамматикой $G = (V_T, V_N, P, S)$, называется множество терминальных цепочек, выводимых в грамматике G из аксиомы:

$L(G) = \{x \mid x \in V_T^*; S => *x\}$, где $=>^*$- выводимость.

Пример. Дана грамматика $G = (V_T, V_N, P, S)$, в которой $V_T = \{a, b\}$, $V_N = \{S\}$, $P = \{S \rightarrow aSb, S \rightarrow ab\}$. Определить язык, порождаемый этой грамматикой.

Решение. Используя рекурсию, выведем несколько цепочек языка, порождаемого данной грамматикой:

$S \rightarrow ab$;
$S \rightarrow aSb \Rightarrow aabb$;
$S \rightarrow aSb \Rightarrow aaSbb \Rightarrow aaabbb$; . . .

Определим язык, порожденный данной грамматикой:

$L(G) = \{a^n b^n \mid n > 0\}$.

Говоря о представлении грамматик, нужно отметить, что множество правил вывода грамматики может приводиться без специального указания множеств терминалов и нетерминалов. В таком случае обычно предполагается, что грамматика содержит в точности те терминальные и нетерминальные символы, которые встречаются в правилах вывода.

Также предполагается, что правые части правил, для которых совпадают левые части, можно записать в одну строку с использованием разделителя. Так, правила вывода из приведенного примера можно записать следующим образом: $S \rightarrow aSb \mid ab$.

4.3. Классификация грамматик

Правила порождающих грамматик позволяют осуществлять преобразования строк. Ограничения же на виды правил позволяют выделить классы грамматик. Рассмотрим классификацию, которую предложил Н. Хомский.

Грамматики типа 0 - это грамматики, на правила вывода которых нет ограничений.

Грамматики типа 1, или контекстные грамматики -это грамматики, все правила которых имеют следующий вид: $xAy \to x\varphi y$, где $A \in V_N$, $x, y, \varphi \in (V_N \cup V_T)^+$.

Грамматики типа 2 - это бесконтекстные, или контекстно-свободные грамматики (КС-грамматики). Правила вывода для этих грамматик имеют следующий вид: $A \to \varphi$, где $A \in V_N$, $\varphi \in (V_N \cup V_T)^*$.

Грамматики типа 3 - это автоматные грамматики, которые делятся на два типа:

а) леволинейные (леворекурсивные), правила вывода для которых имеют следующий вид: $A \to Aa \mid a$, где $A \in V_N$;

б) праволинейные (праворекурсивные), правила вывода для которых имеют следующий вид: $A \to Aa \mid a$.

Язык L называется языком типа i, если существует грамматика типа i, порождающая язык L.

4.3.1. Методика решения задач

Пример 1. Дана порождающая грамматика $G = (V_T, V_N, P, S)$, в которой $V_T = \{a, d, e\}$, $V_N = \{B, C, S\}$, $P = \{S \to aB, B \to Cd \mid dC, C \to e\}$. Выписать терминальные цепочки, порожденные данной грамматикой, и определить длину их вывода.

Решение. Получим следующие терминальные цепочки:

$S \to aB \to aCd \to aed$,

$S \to aB \to adC \to ade$,

длина вывода которых равна 3.

Пример 2. Дана грамматика $G = (V_T, V_N, P, C)$, в которой $V_T = \{a, b, c, d, e\}$, $V_N = \{A, B, C, D, E\}$, $P = \{A \to ed, B \to Ab, C \to Bc, C \to dD, D \to Ae, E \to bc\}$. Определить, принадлежит ли языку $L(G)$ цепочка eadabcb.

Решение. Выведем возможные терминальные цепочки из аксиомы с помощью заданных правил вывода:

$C \to Bc \to Abc \to edbc$,

$C \to dD \to dAe \to dede$.

Так как первые же терминальные символы полученных цепочек не совпадают с заданной, можно сделать вывод о том, что цепочка eadabcb не принадлежит языку $L(G)$.

Пример 3. Построить КС-грамматику (грамматику типа 2), порождающую цепочки из 0 и 1 с одинаковым числом тех и других символов.

Решение. Определим множества, задающие грамматику: $V_T = \{0, 1\}$; $V_N = \{S\}$; $P = \{S \to 0S1, S \to 1S0, S \to \varepsilon, S \to S01, S \to S10\}$.

Пример 4. Описать язык, порождаемый следующими правилами: $S \to bSS \mid a$.

Решение. Построим несколько терминальных цепочек, применяя заданные правила вывода:

$S \to a$;

$S \to bSS \to baa$;

$S \to bSS \to bbSSS \to bbaaa$;

$S \to bSS \to bbSSbSS \to bbaabaa$;

$S \to bSS \to bbSSbSS \to bbbSSbSSbbSSbSS \to \ldots$

Из полученных цепочек видно, что:

а) цепочки всегда начинаются с терминала b, кроме аксиомы, и заканчиваются терминалом a;

б) количество терминалов a в любой цепочке всегда на 1 больше, чем b.

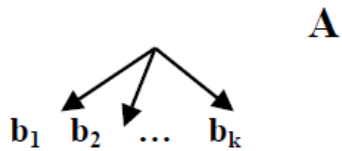
Исходя из этих выводов, определим язык, порождаемый заданными правилами, следующим образом:

$L(G) = \{\alpha \mid \alpha \in \{a, b\}^*\}$, $|a| = |b| + 1$, причем цепочки начинаются с терминала b и заканчиваются терминалом a.

4.4. Грамматический разбор

В КС-грамматике может быть несколько выводов, эквивалентных в том смысле, что в них применяются одни и те же правила к одинаковым в промежуточном выводе цепочкам, но в различном порядке. Например, для грамматики G с правилами вывода $S \to ScS|b|a$ возможны следующие эквивалентные выводы терминальной цепочки acb: $S \to ScS \to Scb \to acb$ и $S \to ScS \to acS \to acb$.

Деревом вывода цепочки x в КС-грамматике $G = (V_T, V_N, P, S)$ называется упорядоченное дерево, каждая вершина которого помечена символом из множества $V \cup V_N \cup \{\varepsilon\}$ так, что каждому правилу $A \to b_1b_2b_3 \ldots b_k$, использующемуся при выводе цепочки x, в дереве вывода соответствует поддерево с корнем A и прямыми потомками b_1, b_2, b_3, ..., b_k, которое приведено на рисунке:



В силу того, что цепочка x ∈ L(G) выводится из аксиомы S, корнем вывода всегда является аксиома. Внутренние узлы вывода соответствуют нетерминальным символам. Концевые вершины дерева вывода - листья - это вершины, не имеющие потомков. Такие вершины соответствуют либо терминалам, либо пустым символам (ε) при условии, что среди правил грамматики имеются правила с пустой правой частью. При чтении слева направо концевые вершины дерева вывода образуют цепочку x.

Дерево вывода часто называют деревом грамматического разбора, или синтаксическим деревом, а процесс построения дерева вывода - грамматическим разбором (синтаксическим анализом). Одной цепочке языка может соответствовать больше одного дерева, так как эта цепочка может иметь разные выводы, порождающие разные деревья.

КС-грамматика $G == (V_T, V_N, P, S)$ называется неоднозначной (неопределенной), если существует цепочка $x \in L(G)$, имеющая два или более дерева вывода.

Рассмотрим пример.

Пусть даны две КС-грамматики:

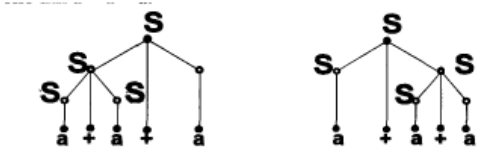
$G_1 = (V_T, V_N, P_1, S)$, $V_N = \{S\}$,

$P_1 = \{S \to S+S \mid S \mid S \cdot S \mid (S) \mid a\}$;

$G_2 = (V_T, V_N, P_2, S)$, $V_N = \{S, A, B\}$,

$P_2 = \{S \to S + A \mid A \mid A,\ A \to A \cdot B \mid A \ / \ B \mid B,\ B \to a \mid (S)\}$, содержащие в множестве терминальных символов знаки операций, круглые скобки и символ a. Определить, являются ли грамматики однозначными. Если какая-либо из них неоднозначна, привести пример цепочки, для которой существует два различных дерева вывода.

Решение: Грамматика G_1 является неоднозначной, так как она имеет правила с правой частью, содержащей два вхождения нетерминала S и знак арифметической операции. Построим два дерева вывода для простейшей цепочки a + a + a:



Грамматика G_2 является однозначной, так как не содержит правил с двойным вхождением нетерминального символа. Так, для цепочки a + a + a она имеет только одно дерево вывода.

В некоторых случаях неоднозначность в грамматиках может устраняться путем эквивалентных преобразований. Однако в общем случае проблема устранения неоднозначности неразрешима. На практике от неоднозначности избавляются путем задания словесных ограничений, называемых контекстными условиями языка, которые включаются в его семантику.

Различают две стратегии грамматического разбора: восходящую и нисходящую, которые соответствуют способу построения синтаксических деревьев. При нисходящей стратегии разбора дерево строится от корня аксиомы вниз, к терминальным вершинам. Главной задачей при нисходящем разборе является выбор того правила, которое следует применить на рассматриваемом шаге. При восходящем разборе дерево строится от терминальных вершин к корню дерева (аксиоме).

Преобразование цепочки, обратное порождению, называется редукцией.

4.4.1. Представление грамматики в виде графа

Дерево грамматического разбора не следует путать с представлением грамматики в виде графа. Граф грамматики в качестве вершин содержит сентенциальные формы (любые цепочки, выводимые из аксиомы).

Рассмотрим представление грамматики G в виде графа: $G = (V_T, V_N, P, S)$, в которой $V_T = \{a, b, c\}$, $V_N = \{S\}$, $P=\{S \rightarrow aSa \mid bSb \mid c\}$.

4.5. Преобразования КС-грамматик

Часто требуется изменить грамматику таким образом, чтобы она удовлетворяла определенным требованиям, не изменяя при этом порождаемый грамматикой язык. Для этого используются эквивалентные преобразования КС-грамматик, некоторые из которых рассмотрены ниже.

4.5.1. Удаление правил вида $A \to B$

Преобразование первого типа состоит в удалении правил $A \to B$, или <нетерминал> → <нетерминал>.

Покажем, что для любой КС-грамматики можно построить эквивалентную грамматику, не содержащую правил вида: $A \to B$, где A и B - нетерминальные символы.

Пусть имеется КС-грамматика $G=(V_T, V_N, P, S)$, где множество нетерминалов $V_N=\{A_1, A_2, \ldots, A_n\}$. Разобьем P на два непересекающихся множества: $P = P_1 \cup P_2$. В P_1 включены все правила вида $A_i \to A_k$, в P_2 включены все остальные правила, т.е. $P_2 = P \setminus P_1$. Затем для каждого A_i определим множество правил $P(A_i)$, включив в него все такие правила $A_i \to \varphi$, что $A_i \to^* A_j$ и $A_j \to \varphi$, где $A_j \to \varphi \in P_2$. Построим эквивалентную КС-грамматику $G_э = (V_T, V_N, P_э, S)$, в которой множества терминальных и нетерминальных символов, а также аксиома совпадают с теми же объектами исходной грамматики G, а множество правил $P_э$ получено объединением правил множества P_2 и правил $P(A_i)$ для всех $1 \le i \le n$:

$$P_э = \bigcup_{i=1}^{n} P(A_i) \cup P_2 .$$

Пример. Пусть задана грамматика G со следующими правилами вывода S → aFb | A; A → aA | B; B → aSb | S; F → bc | bFc.

Построим множества правил P2, P(S), P(A), P(B), P(F).

Определим правила для P2: P2 = {S → aFb; A → aA; B → aSb; F → bc | bFc}.

Определим правила для P(S): S => A => B или S =>*A; S=>B, где =>* обозначает непосредственную выводимость. P(S) = {S → aA; S → aSb).

Определим правила для P(A): A => B => S или A =>* B; A => S. P(A) = {A → aSb; A → aFb}.

Определим правила для P(B): B => S => A или B =>* S; B =>*A. P(B) = {B → aFb; B → aA}.

Определим правила для P(F): так как непосредственно выводимых нетерминалов не существует, то P(F) = 0.

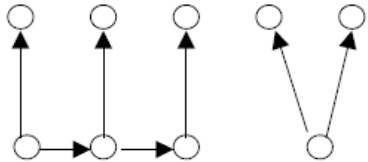
Объединив полученные правила, можно записать грамматику Gэ, эквивалентную исходной:

S → aFb | aSb | aA; A → aA | aSb | aFb;

B → aA | aSb | aFb; F → bc | bFc.

Графическая модификация метода

Аналитическое преобразование по рассмотренному алгоритму оказывается довольно сложным. При автоматизированном преобразовании грамматик проще применить графическую модификацию этого метода. С этой целью каждому нетерминальному символу и каждой правой части правил из множества P2 поставлена в соответствие вершина графа. Из вершины с меткой U в вершину с меткой V направлено ребро, если в грамматике существует правило $U \rightarrow V$.



В эквивалентную грамматику будут включены правила вида $A \to w$, $A \in V_N$; $w \in (V_T \cup V_N)^*$, если из вершины с меткой A существует путь в вершину с меткой w.

$S \to aFb \mid aSb \mid aA;$

$A \to aA \mid aSb \mid aFb;$

$B \to aA \mid aSb \mid aFb;$

$F \to bc \mid bFc.$

Получено то же множество правил P, что и аналитическим методом.

4.5.2. Построение неукорачивающей грамматики

Грамматика, не содержащая правил с пустой правой частью, называется неукорачивающей грамматикой.

В грамматике с правилами вида $A \to \varepsilon$ длина выводимой цепочки при переходе от k-го шага к (k+1)-му уменьшается. Поэтому грамматики с правилами вида $A \to \varepsilon$ называются укорачивающими. Восходящий синтаксический разбор в укорачивающих грамматиках сложнее по сравнению с разбором в неукорачивающих грамматиках, т.к. при редукции необходимо отыскать такой фрагмент входной цепочки, в которую можно вставить пустой символ.

Покажем, что для произвольной КС-грамматики, порождающей язык без пустой цепочки, можно построить эквивалентную неукорачивающую КС-грамматику.

Построим множество всех нетерминальных символов грамматики $G=(V_T, V_N, P, S)$, из которых выводится пустая цепочка, выделив следующие множества:

$W_1 = \{A \mid A \to \varepsilon \in P\}$,

$W_{n+1} = W_n \cup \{B \mid B \to \varphi \in P, \varphi \in W^*_n\}$.

Затем найдем множество правил эквивалентной грамматики в два этапа:

а) удалив из множества P исходной грамматики правила с пустой правой частью $P_1' = P \setminus \{A \to \varepsilon \mid A \to \varepsilon \in P\}$;

б) получив новые правила $A \to \varphi'$ после удаления из каждого правила исходной грамматики $A \to \varphi \in P$ те нетерминалы, которые вошли в множество W_n по правилу:

$P_1'' = \{ A \to \varphi' \mid A \to \varphi \in P'; \varphi = \varphi_1 B \varphi_2 \mid B \in W_n; \varphi' = \varphi_1 B \varphi_2\}$.

Повторив п.б) для каждого нетерминала, принадлежащего множеству W_n, получим эквивалентную грамматику $G = (V_T, V_N, P_э, S)$.

Пример. Пусть задана грамматика G со следующими правилами вывода: $S \to AbA \mid cAb \mid Bb$; $A \to aAb \mid \varepsilon$; $B \to AA \mid a$. Необходимо:

1) построить множество нетерминалов, из которых выводится ε;

2) построить неукорачивающую грамматику, эквивалентную исходной.

Для того чтобы построить множество всех нетерминалов грамматики, из которых выводится пустая цепочка, выделим следующие множества:

$W_1 = \{A \mid A \to \varepsilon \in P\}$;

$W_{m+1} = W_m \cup \{B \mid B \to \varphi \in P, \varphi \in W^*_m\}$.

Так как мы имеем правило $A \to \varepsilon \in P$, то можно построить множество $W_1 = \{A\}$, включающее нетерминал A.

Построим множество W_2. С нетерминалом A связан нетерминал B, т.е. существует правило $B \to AA$ и $A \in W_1$. Следовательно, $W_2 = \{A,B\}$.

$W_3 = W_2$, т. к. множество $W_3 = \{B \mid B \to \varphi \in P, \varphi \in W^*_m\}$ является пустым.

Исключив правило, содержащее пустую цепочку в правой части, получим неукорачивающую грамматику G1 следующего вида:

$S \to AbA \mid cAb \mid Bb \mid bA \mid Ab \mid cb \mid b$;

$A \to aAb \mid ab$;

$B \to AA \mid A \mid a$.

4.5.3. Построение грамматики с продуктивными нетерминалами

Нетерминальный символ A называется непродуктивным (непроизводящим), если он не порождает ни одной терминальной цепочки, т.е. не существует вывода $A \to^* x$, где $x \in V^*_T$. Поэтому представляет интерес удаление из грамматики всех непродуктивных нетерминальных символов. Рассмотрим, как для произвольной КС-грамматики можно построить эквивалентную КС-грамматику, все нетерминальные символы которой продуктивны. С этой целью выделяется множество $W_1=\{A \mid A \to \varphi \in P, \varphi \in V^*_T\}$. Затем строится множество $W_1, W_2, \ldots, W_{n+1}$ по следующим правилам:

$W_{n+1} = W_n \cup \{B \mid B \to x \in P, x \in (V_T \cup W_n)^*\}$.

Пусть задана грамматика G со следующими правилами вывода: $S \to SA \mid BSb \mid bAb$; $A \to aSa \mid bb$; $B \to bBb \mid BaA$. Построим множество продуктивных нетерминалов:

$W_1 = \{A\}$, т.к. в множестве P есть правило $A \to bb$;

$W_2 = \{A, S\}$, т.к. имеется правило $S \to bAb$ и $A \in W_1$;

$W_3 = W_2$.

Все символы множества $V_N \backslash W_n$ являются непродуктивными, не используются в выводе никакой терминальной цепочки и их можно удалить из грамматики вместе со всеми правилами, в которые они входят. Грамматика G_1, эквивалентная исходной грамматике, будет включать следующее множество правил:

$S \to SA \mid bAb$; $A \to aSa \mid bb$.

В грамматике G_1 все нетерминалы продуктивны.

4.5.4. Построение грамматики, аксиома которой зависит от всех нетерминалов

Существует еще один тип нетерминальных символов, которые можно удалять из грамматики вместе с правилами, в которые они входят. Например, в грамматике G, заданной множеством правил P: $S \to aSb \mid ba$; $A \to aAa \mid bba$, нетерминал A не участвует ни в каком выводе, т.к. из аксиомы нельзя вывести цепочку, содержащую этот нетерминал. Поэтому из заданной грамматики можно удалить нетерминал A. Рассмотрим, как для произвольной КС-грамматики можно построить эквивалентную КС-грамматику, аксиома которой зависит от всех нетерминальных символов.

Для этого построим множество нетерминалов, от которых зависит аксиома. С этой целью выделим множества $W_1, W_2, \ldots, W_{n+1}$ по следующим правилам:

$W_1 = \{S\}$,

$W_{n+1} = W_n \cup \{B \mid A \to xBy \in P, A \in W_n\}$.

Нетерминал $B \in W_n$ тогда и только тогда, когда аксиома S зависит от B. Все нетерминалы, не содержащиеся в W_n, можно удалить из грамматики вместе с правилами, в которые они входят.

Пример. Пусть дана КС-грамматика G:

$S \to abS \mid ASa \mid ab;\ A \to abAa \mid ab;\ B \to bAab \mid bB.$

Найдем нетерминалы, от которых зависит аксиома:

$W_1 = \{S\}$,

$W_2 = \{S, A\}$, т.к. имеется правило $S \to Asa$ и $S \in W_1$;

$W_3 = W_2$.

Эквивалентная грамматика G_1, аксиома которой зависит от всех нетерминальных символов:

$S \to abS \mid ASa \mid ab;\ A \to abAa \mid ab.$

4.5.5. Удаление правил с терминальной правой частью

Пусть в грамматике G имеется с терминальной правой частью $A \to \beta$, где $\beta \in V_T^*$. Тогда любой вывод с использованием этого правила имеет вид: $S \to^* xAy \to x\beta y$. Здесь нетерминал A в сентенциальной форме xAy появился на предыдущем шаге вывода $B \to uAv$. Если в это правило вместо A подставить β, получим правило $B \to u\beta v$. Тогда длина вывода некоторой цепочки сократится на один шаг. Следовательно, для того чтобы удалить терминальное правило грамматики $A \to \beta$, необходимо учесть следующие правила:

а) если для нетерминала A больше нет правил, тогда во всех правых частях A заменяется на β;

б) если для нетерминала A есть другие правила, тогда добавляются новые правила, в которых A заменяется на β.

Пример. Пусть дана КС-грамматика G:

$S \to aSb \mid bAa \mid B$; $A \to ABa \mid b \mid \varepsilon$; $B \to ab \mid ba$.

Удалим правила для нетерминала B. Тогда эквивалентная грамматика G_1 будет включать следующие правила:

$S \to aSb \mid bAa \mid ab \mid ba$; $A \to Aaba \mid Abaa \mid b \mid \varepsilon$.

4.5.6. Построение эквивалентной праворекурсивной КС-грамматики

Некоторые специальные методы грамматического разбора неприменимы к леворекурсивным или праворекурсивным грамматикам, поэтому рассмотрим устранение левой или правой рекурсии. В общем случае избавиться от рекурсии в правилах грамматики невозможно, т.к. бесконечные языки порождаются грамматиками с конечным числом правил только благодаря рекурсии. Поэтому можно говорить лишь о преобразовании одного вида рекурсии в другой. Рассмотрим, как для любой леворекурсивной КС-грамматики построить эквивалентную праворекурсивную КС-грамматику.

Пусть нетерминал A - леворекурсивен, т.е. для него имеются правила следующего вида:

$A \to Ax_1 \mid Ax_2 \mid \ldots \mid Ax_p \mid w_1 \mid \ldots \mid w_k$, где x_i и w_j - цепочки над множеством $V_T \cup V_N$.

Введем дополнительные нетерминалы B и D и указанные правила заменим на эквивалентные им:

$A \to AB \mid D$;

$B \to x_1 \mid x_2 \mid \ldots \mid x_p$;

$D \to w_1 \mid w_2 \mid \ldots \mid w_k$.

В результате замены получим вывод, который имеет вид:

$A \to AB \to ABB \to ABBB \to \ldots \to AB^* \to DB^*$.

Таким образом, для нетерминала A можно определить эквивалентные правила:

$A \to DK$;

$K \to BK \mid \varepsilon$.

Выполняя подстановку D и B в эти правила, получим следующие праворекурсивные правила:

$A \to w_1K \mid w_2K \mid \ldots \mid w_kK$;

$K \to x_1K \mid x_2K \mid \ldots \mid x_pK \mid \varepsilon$.

Пример. Пусть задана грамматика G со следующими правилами вывода: $S \to Sa \mid ba$. Требуется построить эквивалентную ей праворекурсивную грамматику.

Для решения задачи воспользуемся рассмотренным выше алгоритмом. В исходной грамматике один леворекурсивный нетерминал S. Построим для него вывод:

$S \to SB \mid D$;

$B \to a$; $D \to ba$. Вывод из S имеет вид: $S \to SB \to SBB \to \ldots \to DB^*$.

Запишем эквивалентные правила для S:

$S \to DK$; $K \to BK \mid \varepsilon$.

Подставим в эти правила B и D и получим эквивалентную праворекурсивную грамматику G_1:

$S \to baK$; $K \to aK \mid \varepsilon$.

Рассмотрим вывод цепочки baaaa в исходной леворекурсивной грамматике G:

$S \to Sa \to Saa \to Saaa \to baaaa$.

Эту же цепочку можно вывести в эквивалентной праворекурсивной грамматике G_1:

$S \to baK \to baaK \to baaaK \to baaaaK \to baaaa$.