Глава 2

ГРАММАТИКИ

§ 2.1. Мотивировка

Имеется один класс порождающих систем, которые представляют для нас первейший интерес — системы, называемые *грамматиками*.

Первоначально понятие грамматики было формализовано лингвистами при изучении естественных языков. Они интересовались не только определением, что является или не является правильным предложением языка, но также искали способы описания структуры предложений. Одной из целей была разработка формальной грамматики, способной описывать естественный язык. Надеялись, что, заложив в компьютер формальную грамматику, например, английского языка, можно сделать его "понимающим" этот язык, по словесной формулировке проблем получать их решения, осуществлять с помощью компьютера перевод с одного языка на другой.

Как показывает практика использования общедоступных программ машинного перевода, по-настоящему хорошего решения этой проблемы мы пока не имеем. Но вполне удовлетворительные результаты достигнуты в описании и реализации языков программирования.

Из школьного опыта известно, что представляет собой грамматический разбор предложения. При таком разборе определяется, какое слово является подлежащим, какое используется в роли сказуемого, какие слова играют роль определения, дополнения, обстоятельства и т.д. При разборе мы имеем дело с грамматическими категориями: предложение, группа существительного, группа сказуемого, существительное, глагол, наречие и т.д. и пользуемся собственно словами, составляющими разбираемое предложение. Например, структуру английского предложения: "The little boy run quickly" можно изобразить в виде диаграммы (рис. 2.1).

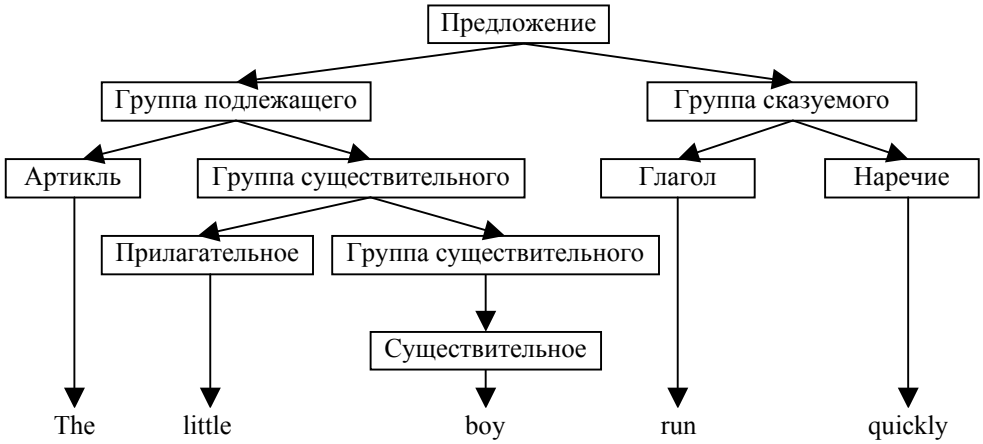

Рис. 2.1.

Грамматический разбор предложений подразумевает использование правил некоторой грамматики. Мы их будем представлять в следующей форме:

⟨ предложение ⟩ → ⟨ группа подлежащего ⟩ ⟨ группа сказуемого ⟩
⟨ группа подлежащего ⟩ → ⟨ артикль ⟩ ⟨ группа существительного ⟩
⟨ группа существительного ⟩→ ⟨ прилагательное ⟩ ⟨ группа существительного ⟩
⟨ группа существительного ⟩ → ⟨ существительное ⟩
⟨ группа сказуемого ⟩ → ⟨ глагол ⟩ ⟨ наречие ⟩
⟨ артикль ⟩ → The
⟨ прилагательное ⟩ → little
⟨ существительное ⟩ → boy
⟨ глагол ⟩ → run
⟨ наречие ⟩ → quickly

Здесь стрелочка отделяет левую часть правила от правой, а грамматические термины заключены в металингвистические скобки ⟨ и ⟩ для того, чтобы отличать их от слов, составляющих анализируемые предложения. По этим правилам можно не только проверять грамматическую правильность предложений, но также порождать грамматически правильные предложения. Механизм порождения прост. Начиная с грамматического термина ⟨ предложение ⟩, являющегося главным, каждый грамматический термин, замещается правой частью того правила, которое содержит его в левой части. Когда в результате таких замен в текущей цепочке не останется ни одного термина грамматики, мы получаем грамматически правильное предложение языка. Любое предложение языка, даже если он бесконечен, может быть выведено таким способом. И хотя большинство порождаемых предложений может не иметь смысла, тем не менее они считаются грамматически правильными.

Разумеется, приведенные здесь правила, не составляют грамматику настоящего английского языка, а служат лишь для иллюстрации разбора предложения, показанного на рис. 2.1.

§ 2.2. Формальное определение грамматики

В предыдущем параграфе речь шла о конкретной грамматике. Как видим, в ней имеются

1) грамматические термины (⟨ предложение ⟩, ⟨ группа подлежащего ⟩, ⟨ группа сказуемого ⟩ и т.д.), которые в теории формальных грамматик принято называть *нетерминалами;*

2) слова, составляющие предложения языка, они называются *терминалами*;

3) *правила*, левые и правые части которых состоят из нетерминалов и терминалов;

4) главный грамматический термин, называемый *начальным символом* или *начальным нетерминалом грамматики*, из него выводятся те цепочки терминалов, которые считаются предложениями языка (в рассмотренном в §2.1 примере начальным нетерминалом является символ ⟨ предложение ⟩).

***Определение* 2.1.** *Грамматикой* называется четверка $G = (V_N, V_T, P, S)$, где V_N, V_T — *алфавиты* (*словари*) *нетерминалов* и *терминалов* соответственно, причем $V_N \cap V_T = \varnothing$, P — конечное множество *правил*, каждое из которых имеет вид $\alpha \to \beta$, где $\alpha \in V^* V_N V^*$, $\beta \in V^*$, $V = V_N \cup V_T$ — *объединенный алфавит* (*словарь*) грамматики; S — *начальный нетерминал*.

В дальнейшем нетерминалы мы будем обозначать заглавными латинскими буквами, например S, A, B, C; терминалы — строчными буквами из начала латинского алфавита, например, a, b, c; цепочки терминалов — строчными буквами из конца латинского алфавита, например x, y, z; цепочки символов из объединенного алфавита — строчными греческими буквами, например α, β, γ.

Мы представили грамматику $G = (V_N, V_T, P, S)$, но не определили язык, который она порождает. Для этого нам потребуется ввести отношение непосредственной выводимости $\Rightarrow_G$, его транзитивное $\overset{+}{\Rightarrow}_G$ и рефлексивно-транзитивное замыкание $\overset{*}{\Rightarrow}_G$. Когда очевидно, в какой грамматике производится вывод, имя грамматики G можно не указывать.

***Определение* 2.2.** Пусть $\alpha \to \beta \in P$ — правило, γ, δ — любые цепочки из множества V^*. Тогда $\gamma\alpha\delta \Rightarrow_G \gamma\beta\delta$ читается следующим образом: “из $\gamma\alpha\delta$ *непосредственно выводится* $\gamma\beta\delta$ в грамматике G (при помощи данного правила)”.

Другими словами, символ $\Rightarrow_G$ связывает две цепочки тогда и только тогда, когда вторая цепочка получается из первой применением единственного правила.

***Определение* 2.3.** Пусть α_1, α_2, ..., α_m — цепочки из множества V^* и $\alpha_1 \Rightarrow_G \alpha_2$, $\alpha_2 \Rightarrow_G \alpha_3$, ..., $\alpha_{m-1} \Rightarrow_G \alpha_m$. Тогда мы пишем: $\alpha_1 \overset{*}{\Rightarrow}_G \alpha_m$ и говорим, что “из α_1 *выводится* α_m в грамматике G”.

Другими словами, $\alpha \overset{*}{\Rightarrow}_G \beta$, если β может быть получена из α путем применения некоторого числа правил из множества правил P. Считается, что $\alpha \overset{*}{\Rightarrow}_G \alpha$ для любой цепочки $\alpha \in V^*$ (рефлексивность) и для этого не требуется никаких правил.

Если мы хотим подчеркнуть, что такой вывод использует по крайней мере одно правило грамматики, то мы пишем: $\alpha \overset{+}{\Rightarrow}_G \beta$. Если мы хотим указать, что такой вывод происходит за n шагов, т.е. посредством применения n правил грамматики, то пишем: $\alpha \overset{n}{\Rightarrow}_G \beta$. Значок $\overset{n}{\Rightarrow}_G$ обозначает n-ю степень отношения непосредственной выводимости.

Напомним, что для любого отношения R имеют место следующие тождества:
$R^0 = E = \{(\alpha, \alpha) \mid \forall\, \alpha \in V^*\}$, $R^n = R R^{n-1} = R^{n-1} R$ для $n > 0$; в частности, $R^1 = R$;
$R^* = \bigcup_{k=0}^{k=\infty} R^k$, $R^+ = \bigcup_{k=1}^{k=\infty} R^k$.

Они, разумеется, применимы и к отношению непосредственной выводимости $\Rightarrow_G$.

***Определение* 2.4.** *Язык*, *порождаемый* грамматикой G, определим как

$$L(G) = \{w \mid w \in V_T^*,\ S \overset{*}{\Rightarrow}_G w\}.$$

Другими словами, язык есть множество терминальных цепочек, выводимых из начального нетерминала грамматики.

***Определение* 2.5.** Любая цепочка α, такая, что $\alpha \in V^*$ и $S \overset{*}{\underset{G}{\Rightarrow}} \alpha$, называется *сентенциальной формой*.

***Определение* 2.6.** *Грамматики* G_1 *и* G_2 называются *эквивалентными,* если

$$L(G_1) = L(G_2).$$

Пример 2.1. Рассмотрим грамматику $G = (V_N, V_T, P, S)$, где $V_N = \{S\}$, $V_T = \{0, 1\}$, $P = \{S \to 0S1,\ S \to 01\}$. Здесь S — единственный нетерминал, он же — начальный; 0 и 1 — терминалы; правил два: $S \to 0S1$ и $S \to 01$.

Применив первое правило $n-1$ раз, а затем второе правило, получим:

$$S \underset{G}{\Rightarrow} 0S1 \underset{G}{\Rightarrow} 00S11 \underset{G}{\Rightarrow} 0^3S1^3 \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} 0^{n-1}S1^{n-1} \underset{G}{\Rightarrow} 0^n1^n.$$

Здесь мы воспользовались обозначением $w^i = \underbrace{w \ldots w}_{i \text{ раз}}$, причем $w^0 = \varepsilon$.

Таким образом, эта грамматика порождает язык $L(G) = \{0^n1^n \mid n > 0\}$. Действительно, при использовании первого правила число символов S остается неизменным и равно 1. После использования второго правила число символов S в сентенциальной форме уменьшается на единицу. Поэтому после использования правила $S \to 01$ в результирующей цепочке не останется ни одного символа S. Так как оба правила имеют в левой части по символу S, то единственно возможный порядок их применения — сколько-то раз использовать первое правило, а затем один раз использовать второе.

Этот пример грамматики очень прост: было почти очевидно, какие предложения выводятся в ней. В общем случае определить, что порождается грамматикой, бывает очень трудно. Дадим более сложный пример.

Пример 2.2. Пусть $G = (V_N, V_T, P, S)$, где $V_N = \{S, B, C\}$, $V_T = \{a, b, c\}$, $P = \{(1)\ S \to aSBC,\ (2)\ S \to aBC,\ (3)\ CB \to BC,\ (4)\ aB \to ab,\ (5)\ bB \to bb,$
$(6)\ bC \to bc,\ (7)\ cC \to cc\}$.

Язык $L(G)$ содержит цепочки вида $a^nb^nc^n$ для каждого $n \geq 1$, так как мы можем использовать правило (1) $n-1$ раз, чтобы получить $S \overset{*}{\Rightarrow} a^{n-1}S(BC)^{n-1}$. Затем мы используем правило (2), чтобы получить $S \overset{*}{\Rightarrow} a^n(BC)^n$. Правило (3) дает нам возможность расположить B и C так, чтобы все B предшествовали всем C. Таким образом, $S \overset{*}{\Rightarrow} a^nB^nC^n$. Далее мы используем один раз правило (4) и получаем $S \overset{*}{\Rightarrow} a^nbB^{n-1}C^n$. Затем, используя правило (5) $n-1$ раз, получаем $S \overset{*}{\Rightarrow} a^nb^nC^n$. Применив один раз правило (6), получим $S \overset{*}{\Rightarrow} a^nb^ncC^{n-1}$. Наконец, применив $n-1$ раз правило (7), окончательно получим $S \overset{*}{\Rightarrow} a^nb^nc^n$.

Легко показать, что в общем случае возможен только такой порядок применения правил, т.е. что $L(G) = \{a^nb^nc^n \mid n \geq 1\}$.

§ 2.3. Типы грамматик

Грамматику, определенную в предыдущем параграфе, вслед за Н.Хомским назовем *грамматикой типа* 0. Им введено еще три типа грамматик, различающихся ограничениями, накладываемыми на вид правил.

***Определение* 2.7.** Грамматика $G = (V_N, V_T, P, S)$ является *грамматикой типа* 1 или *контекстно-зависимой грамматикой*, если для каждого ее правила $\alpha \to \beta \in P$ выполняется условие $|\beta| \geq |\alpha|$.

Часто вместо термина "контекстно-зависимая грамматика" используют аббревиатуру csg (context-sensitive grammar). Очевидно, что грамматики, приведенные в примерах 2.1. и 2.2, являются контекстно-зависимыми грамматиками, поскольку правые части их правил не короче левых частей.

Замечание 2.1. Некоторые авторы требуют, чтобы правила контекстно-зависимой грамматики имели вид: $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$, где $\alpha_1, \alpha_2, \beta \in V^*$, причем $\beta \neq \varepsilon$, а $A \in V_N$. Это мотивирует название "контекстно-зависимая", так как правило $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$ позволяет заменять A на β только, если A появляется в сентенциальной форме в контексте α_1 и α_2. В отечественной литературе для таких грамматик чаще используется термин НС-грамматики — *грамматики непосредственных составляющих*, а грамматики типа 1 называются *неукорачивающими грамматиками*.

Можно показать, что это требование не изменяет класса порождаемых языков. Действительно, любая НС-грамматика является неукорачивающей. Однако любое правило неукорачивающей грамматики может быть преобразовано так, чтобы все символы, его составляющие, были нетерминалами. Для этого достаточно каждое вхождение терминала $a \in V_T$ заменить на новый нетерминал Z, пополнить словарь нетерминалов этим символом и включить правило $Z \to a$ в множество правил грамматики. Правила вида $Z \to a$ допустимы для НС-грамматик.

Правило же вида

$X_1 X_2 \ldots X_m \to Y_1 Y_2 \ldots Y_{m+q}$, где $m > 0,\ q \geq 0,\ X_i, Y_j \in V_N,\ 1 \leq i \leq m,\ 1 \leq j \leq m + q$

эквивалентно группе правил:

$$
\begin{aligned}
&X_1 X_2 \ldots X_m \to A_1 X_2 \ldots X_m,\\
&A_1 X_2 \ldots X_m \to A_1 A_2 \ldots X_m,\\
&\qquad\qquad \ldots\\
&A_1 A_2 \ldots X_m \to A_1 A_2 \ldots A_m,\\
&A_1 A_2 \ldots A_m \to Y_1 A_2 \ldots A_m,\\
&Y_1 A_2 \ldots A_m \to Y_1 Y_2 \ldots A_m,\\
&\qquad\qquad \ldots\\
&Y_1 Y_2 \ldots A_{m-1} A_m \to Y_1 Y_2 \ldots Y_{m-1} A_m,\\
&Y_1 Y_2 \ldots Y_{m-1} A_m \to Y_1 Y_2 \ldots Y_{m-1} Y_m Y_{m+1} \ldots Y_{m+q},
\end{aligned}
$$

где $A_1, A_2, \ldots, A_m$ — дополнительные нетерминалы. Каждое из этих правил имеет требуемый НС-грамматикой вид: $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$.

***Определение* 2.8.** Грамматика $G = (V_N, V_T, P, S)$ является *грамматикой типа* 2 или *контекстно-свободной грамматикой*, если каждое ее правило имеет вид $A \to \beta \in P$, где $A \in V_N$, $\beta \in V^+$.

Вместо термина “контекстно-свободная грамматика” часто используют аббревиатуру cfg (context-free grammar) или сокращение КС-грамматика.

Замечание 2.2. Правило вида $A \to \beta$ позволяет заменить A на β независимо от контекста, в котором появляется A.

Грамматика, приведенная в примере 2.1, является контекстно-свободной.

Пример 2.3. Рассмотрим интересную контекстно-свободную грамматику $G = (V_N, V_T, P, S)$, где $V_N = \{S, A, B\}$, $V_T = \{a, b\}$, $P = \{S \to aB,\ S \to bA,\ A \to a,\ A \to aS,\ A \to bAA,\ B \to b,\ B \to bS,\ B \to aBB\}$.

Индукцией по длине цепочки покажем, что $L(G) = \{x \in \{a, b\}^+ \mid \#_a x = \#_b x\}$, где $\#_a x$ обозначает число букв a в цепочке x. Другими словами, язык, порождаемый этой грамматикой, состоит из непустых цепочек, в которых число букв a и b одинаково.

Достаточно доказать, что для $x \in V_T^+$

1) $S \overset{*}{\Rightarrow} x$ тогда и только тогда, когда x состоит из равного числа букв a и b;

2) $A \overset{*}{\Rightarrow} x$ тогда и только тогда, когда x имеет на одно a больше, чем b;

3) $B \overset{*}{\Rightarrow} x$ тогда и только тогда, когда x имеет на одно b больше, чем a.

База. Очевидно, что все три утверждения выполняются, если $|x| = 1$, поскольку $A \overset{*}{\Rightarrow} a$, $B \overset{*}{\Rightarrow} b$ и никакая терминальная цепочка не выводима из S. Кроме того, никакие строки единичной длины, отличающиеся от a и b, не выводимы из A и B соответственно.

Индукционная гипотеза. Предположим, что утверждения 1–3 выполняются для всех x, длина которых меньше k ($k \geq 1$).

Индукционный переход. Покажем, что они истинны для $|x| = k$.

Необходимость. Если $S \overset{*}{\Rightarrow} x$, то вывод должен начинаться либо с правила $S \to aB$, либо с правила $S \to bA$. В первом случае $x = ax_1$, причем $|x_1| = k - 1$ и $B \overset{*}{\Rightarrow} x_1$. По индукционному предположению число букв b в цепочке x_1 на единицу больше, чем число букв a, так что в цепочке x букв a столько же, сколько букв b. Аналогичное рассуждение достигает цели, если вывод начинается с правила $S \to bA$.

Достаточность. Теперь нужно доказать обратное утверждение, т.е. что если $|x| = k$ и $\#_a x = \#_b x$, то $S \overset{*}{\Rightarrow} x$. Либо первый символ x есть a, либо он есть b. Предположим, что $x = ax_1$. Теперь $|x_1| = k - 1$ и цепочка x_1 имеет на одну букву b больше, чем букв a. По индукционной гипотезе $B \overset{*}{\Rightarrow} x_1$. Но тогда

$$S \Rightarrow aB \overset{*}{\Rightarrow} ax_1 = x.$$

Аналогичное рассуждение достигает цели, если первый символ x есть b.

Для завершения доказательства аналогичным образом должны быть доказаны утверждения 2 и 3 (предоставим это читателю).

***Определение* 2.9.** Грамматика $G = (V_N, V_T, P, S)$ является *грамматикой типа* 3 или *регулярной грамматикой* (rg — regular grammar), если каждое ее правило имеет вид $A \to aB$ или $A \to a$, где $a \in V_T$; $A, B \in V_N$.

Замечание 2.3. В гл. 3 будет определено абстрактное устройство, называемое *конечным автоматом*, и показано, что языки, порождаемые грамматиками типа 3, являются в точности теми множествами, которые распознаются (допускаются) этими устройствами. Поэтому такой класс грамматик и языков часто называют *конечно-автоматными* или просто *автоматными*.

Пример 2.4. Рассмотрим грамматику $G = (V_N, V_T, P, S)$, где $V_N = \{S, A, B\}$, $V_T = \{0, 1\}$, $P = \{S \to 0A,\ S \to 1B,\ S \to 0,\ A \to 0A,\ A \to 0S,\ A \to 1B,\ B \to 1B,\ B \to 1, B \to 0\}$.

Ясно, что G — регулярная грамматика. Определить, какой язык порождает данная грамматика и доказать это, предоставляем читателю.

Очевидно, что каждая регулярная грамматика — контекстно-свободна; каждая контекстно-свободная грамматика — контекстно-зависима; каждая контекстно-зависимая грамматика является грамматикой типа 0. Каждому классу грамматик соответствует класс языков. Языку приписывается тип грамматики, которой он порождается. Например, контекстно-свободные грамматики (cfg) порождают контекстно-свободные языки (cfl), контекстно-зависимые грамматики (csg) порождают контекстно-зависимые языки (csl).

В соответствии с текущей практикой язык типа 3 или регулярный язык часто называют *регулярным множеством* (rs — regular set). Язык типа 0 называют *рекурсивно перечислимым множеством* (res — recursively enumerable set). Далее будет показано, что языки типа 0 соответствуют языкам, которые интуитивно могут быть перечислимы конечно описываемыми процедурами.

§ 2.4. Пустое предложение

Легко заметить, что по тому, как определены грамматики, пустое предложение (ε) не находится ни в контекстно-свободном (cfl), ни в контекстно-зависимом (csl) языках, ни в регулярном множестве (rs). Мы расширим данные ранее определения csg, cfg и rg, допустив порождение пустого предложения посредством правила вида $S \to \varepsilon$, где S — начальный символ, при условии, что S не появляется в правой части никакого правила. В этом случае ясно, что правило $S \to \varepsilon$ может использоваться только на первом шаге вывода и только на нем.

Имеет место следующая лемма.

Лемма 2.1. *Если $G = (V_N, V_T, P, S)$ есть контекстно-зависимая, контекстно-свободная или регулярная грамматика, то существует другая грамматика G_1 такого же типа, которая порождает тот же самый язык и в которой ни одно правило не содержит начальный символ в своей правой части.*

Доказательство. Пусть S_1 — символ, не принадлежащий ни алфавиту нетерминалов, ни алфавиту терминалов граммматики G. Положим $G_1 = (V_N \cup \{S_1\}, V_T, P_1, S_1)$. Множество правил P_1 состоит из всех правил P и правил вида $S_1 \to \alpha$ при условии, что $S \to \alpha \in P$. Поскольку символ $S_1 \notin V$ ($V = V_N \cup V_T$), то он не появляется в правой части никакого правила из множества P_1.

Докажем, что $L(G) = L(G_1)$.

I. Покажем сначала, что $L(G) \subseteq L(G_1)$.

Пусть $x \in L(G)$, т.е. $S \overset{*}{\underset{G}{\Rightarrow}} x$. Пусть первое используемое правило есть $S \to \alpha \in P$. Тогда $S \underset{G}{\Rightarrow} \alpha \overset{*}{\underset{G}{\Rightarrow}} x$. По построению P_1 правило $S_1 \to \alpha \in P_1$, так что $S_1 \underset{G_1}{\Rightarrow} \alpha$. Поскольку любое правило грамматики G является также правилом грамматики G_1, то $\alpha \overset{*}{\underset{G_1}{\Rightarrow}} x$. Таким образом, имеем $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} x$, т.е. $x \in L(G_1)$ и тем самым доказано, что $L(G) \subseteq L(G_1)$.

II. Покажем теперь, что $L(G_1) \subseteq L(G)$.

Пусть $x \in L(G_1)$, т.е. $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} x$. Пусть первое используемое правило есть $S_1 \to \alpha$. Но такое правило существует во множестве P_1 только потому, что в правилах P имеется правило $S \to \alpha$. Следовательно, $S \underset{G}{\Rightarrow} \alpha$. С другой стороны, $\alpha \overset{*}{\underset{G_1}{\Rightarrow}} x$ и α не содержит символа S_1. Поскольку ни одно правило из множества P_1 не содержит справа символа S_1, то ни одна сентенциальная форма этого вывода также не содержит символа S_1. Значит, в этом выводе используются только такие правила, которые имеются в множестве P. Поэтому $\alpha \overset{*}{\underset{G}{\Rightarrow}} x$. С учетом того, что $S \underset{G}{\Rightarrow} \alpha$, получаем вывод $S \underset{G}{\Rightarrow} \alpha \overset{*}{\underset{G}{\Rightarrow}} x$. Это и означает, что $L(G_1) \subseteq L(G)$.

Из утверждений I и II следует, что $L(G) = L(G_1)$.

Очевидно, что грамматики G и G_1 имеют один и тот же тип. Действительно, все правила грамматики G являются правилами грамматики G_1. Те же новые правила, которые имеются в грамматике G_1, но отсутствуют в грамматике G, отличаются от прототипа лишь символом слева, что не может изменить тип грамматики. Что и требовалось доказать.

Теорема 2.1. *Если L — контекстно-зависимый, контекстно-свободный или регулярный язык, то языки $L \cup \{\varepsilon\}$, $L \setminus \{\varepsilon\}$ также являются соответственно контекстно-зависимым, контекстно-свободным или регулярным языком.*

Доказательство. Согласно лемме 2.1 существует грамматика G, порождающая язык L, начальный нетерминал которой не встречается в правых частях ее правил.

Если язык $L = L(G)$ не содержит пустого предложения, то мы можем пополнить грамматику G еще одним правилом вида $S \to \varepsilon$. Обозначим пополненную грамматику G_1. Правило $S \to \varepsilon$ может использоваться только как первое и единственное правило вывода в G_1, поскольку начальный нетерминал S не встречается в правых частях правил. Любой вывод в грамматике G_1, не использующий правило $S \to \varepsilon$, есть также вывод в G, так что $L(G_1) = L(G) \cup \{\varepsilon\}$.

Если же $L = L(G)$ содержит пустое предложение, то среди правил грамматики G имеется правило вида $S \to \varepsilon$, с помощью которого только пустое предложение и выводится. Ни в каком другом выводе это правило не используется, так что, если его отбросить, то получим грамматику G_1, которая порождает все предложения языка L, кроме пустого. Следовательно, $L(G_1) = L(G) \setminus \{\varepsilon\}$.

Согласно лемме 2.1 типы грамматик G и G_1 одинаковы, поэтому одинаковы и типы языков, порождаемых этими грамматиками. Что и требовалось доказать.

Пример 2.5. Рассмотрим грамматику G из примера 2.2. Перестроим ее согласно лемме 2.1 так, чтобы начальный нетерминал не встречался в правых частях правил. Обозначим перестроенную грамматику G_1.

Очевидно, что $G_1 = (V_N, V_T, P_1, S_1)$, где $V_N = \{S_1, S, B, C\}$, $V_T = \{a, b, c\}$,

$P_1 = P \cup \{S_1 \to aSBC,\ S_1 \to aBC\} = \{S_1 \to aSBC,\ S_1 \to aBC$ (дополнительные правила),

1–7 — старые правила:

(1) $S \to aSBC$, (2) $S \to aBC$, (3) $CB \to BC$, (4) $aB \to ab$, (5) $bB \to bb$, (6) $bC \to bc$, (7) $cC \to cc$ }.

Построенная грамматика отличается от исходной грамматики G только дополнительным нетерминалом S_1, используемым в качестве нового начального символа, и двумя дополнительными правилами, его определяющими. Согласно лемме 2.1 $L(G_1) = L(G) = \{a^n b^n c^n \mid n \geq 1\}$.

Мы можем добавить пустое предложение к $L(G_1)$, пополнив грамматику G_1 еще одним правилом: $S_1 \to \varepsilon$. Обозначим пополненную грамматику G_2. Тогда $G_2 = (V_N, V_T, P_2, S_1)$, где $V_N = \{S_1, S, B, C\}$, $V_T = \{a, b, c\}$, $P_2 = P_1 \cup \{S_1 \to \varepsilon\}$.

Очевидно, что $L(G_2) = L(G_1) \cup \{\varepsilon\} = \{a^n b^n c^n \mid n \geq 0\}$.

§ 2.5. Рекурсивность контекстно-зависимых грамматик

Напомним, что грамматика $G = (V_N, V_T, P, S)$ — *рекурсивна*, если существует алгоритм, который определяет (за конечное время), порождается ли любая данная цепочка $x \in V_T^*$ данной грамматикой G.

Пусть $G = (V_N, V_T, P, S)$ — контекстно-зависимая грамматика. Проверить, порождается пустое предложение данной грамматикой или нет, просто. Достаточно посмотреть, имеется ли в ней правило $S \to \varepsilon$, поскольку $\varepsilon \in L(G)$ тогда и только тогда, когда $S \to \varepsilon \in P$. Если $\varepsilon \in L(G)$, то мы можем образовать новую грамматику: $G_1 = (V_N, V_T, P_1, S)$, отличающуюся от исходной лишь тем, что в ней не будет правила $S \to \varepsilon$, т.е. $P = P_1 \setminus \{S \to \varepsilon\}$. Согласно теореме 2.1 грамматика G_1 тоже контекстно-зависима, и $L(G_1) = L(G) \setminus \{\varepsilon\}$. Следовательно, в любом выводе длина последовательных сентенциальных форм разве лишь возрастает (т.е. не убывает).

Пусть объединенный словарь V грамматики G_1 имеет k символов. Предположим, что $S \overset{*}{\underset{G_1}{\Rightarrow}} x$, $x \neq \varepsilon$. Пусть этот вывод имеет вид: $S \underset{G_1}{\Rightarrow} \alpha_1 \underset{G_1}{\Rightarrow} \alpha_2 \underset{G_1}{\Rightarrow} \ldots \underset{G_1}{\Rightarrow} \alpha_m = x$, причем $|\alpha_1| \leq |\alpha_2| \leq \ldots \leq |\alpha_m|$. Предположим, что сентенциальные формы $\alpha_i, \alpha_{i+1}, \ldots, \alpha_{i+j}$ все одной и той же длины, скажем, p. Предположим также, что $j \geq k^p$. Тогда среди этих сентенциальных форм, по крайней мере, какие-то две одинаковы, так как число всевозможных различных непустых цепочек длиной p,

составленных из символов алфавита V, в котором k символов, равно k^p. В рассматриваемой же последовательности мы имеем $j + 1$ цепочек, где $j \geq k^p$. Пусть, например, $\alpha_q = \alpha_r$, где $q < r$. Тогда

$$S \underset{G_1}{\Rightarrow} \alpha_1 \underset{G_1}{\Rightarrow} \alpha_2 \underset{G_1}{\Rightarrow} \ldots \underset{G_1}{\Rightarrow} \alpha_q \underset{G_1}{\Rightarrow} \alpha_{r+1} \ldots \underset{G_1}{\Rightarrow} \alpha_m = x$$

является более коротким выводом цепочки x в грамматике G_1, чем первоначальный.

Интуитивно ясно, что если существует какой-нибудь вывод терминальной цепочки, то существует и "не слишком длинный" ее вывод.

В следующей теореме описывается алгоритм распознавания, в котором существенно используется это соображение.

Теорема 2.2. *Если грамматика* $G = (V_N, V_T, P, S)$ *— контекстно-зависима, то она — рекурсивна.*

Доказательство. В предыдущих параграфах было показано, что существует простой способ узнать, действительно ли $\varepsilon \in L(G)$, и если это так, то, исключив из грамматики правило $S \to \varepsilon$, получим грамматику, которая порождает тот же самый язык, но без пустого предложения. Любое же непустое предложение языка выводимо без использования этого правила. Поэтому, предполагая, что P не содержит правила $S \to \varepsilon$, рассмотрим произвольную цепочку $x \in V_T^+$. Наша задача найти алгоритм, разрешающий вопрос: $x \in L(G)$?

Пусть $|x| = n$ $(n > 0)$. Определим множество T_m следующим образом:

$$T_m = \{\alpha \mid S \overset{i}{\Rightarrow} \alpha,\ i \leq m,\ \alpha \in V^+,\ |\alpha| \leq n\}.$$

Другими словами, T_m содержит сентенциальные формы, выводимые не более, чем за m шагов, и не длиннее, чем n символов. Очевидно, что $T_0 = \{S\}$ и $T_m = T_{m-1} \cup \{\alpha \mid \beta \Rightarrow \alpha, \beta \in T_{m-1},\ |\alpha| \leq n\}$, т. е. T_m есть результат пополнения множества T_{m-1} цепочками, выводимыми из его цепочек за один шаг, длина которых не превосходит n.

Если $S \overset{*}{\Rightarrow} \alpha$ и $|\alpha| \leq n$, то $\alpha \in T_m$ при некотором m. Если $S \overset{*}{\Rightarrow} \alpha$ не имеет места или $|\alpha| > n$, то $\alpha \notin T_m$ ни при каком m. Также очевидно, что $T_{m-1} \subseteq T_m$ для всех $m \geq 1$. Поскольку T_m зависит только от T_{m-1}, то $T_m = T_{m+1} = T_{m+2} = \ldots$, если окажется на некотором шаге вычислений членов этой последовательности $T_m = T_{m-1}$.

Наш алгоритм будет вычислять T_1, T_2, T_3, ... до тех пор, пока для некоторого m не окажется $T_m = T_{m-1}$. Если цепочки x нет во множестве T_m, то ее нет и в $L(G)$, потому что $T_j = T_m$ для $j > m$. Конечно, если $x \in T_m$, то $S \overset{*}{\Rightarrow} x$.

Осталось доказать, что для некоторого m непременно будет $T_m = T_{m-1}$. Вспомним, что $T_i \supseteq T_{i-1}$ для каждого $i \geq 1$. При $T_i \neq T_{i-1}$ число элементов во множестве T_i, по крайней мере, на 1 больше, чем во множестве T_{i-1}. Если алфавит V имеет k символов, то число строк длиной n или меньше во множестве V^+ равно $k + k^2 + k^3 + \ldots + k^n \leq (k + 1)^n$. И это единственно возможные строки, которые могут быть в любом множестве T_i. Таким образом, при некотором $m \leq (k+1)^n$ непременно случится $T_m = T_{m-1}$. Следовательно, процесс вычисления

множеств $T_i\,(i>0)$ гарантированно закончится за конечное число шагов, и он тем самым является алгоритмом.

Замечание 2.4. Нет нужды доказывать, что алгоритм, описанный в теореме 2.2, применим также к контекстно-свободным и регулярным грамматикам.

Пример 2.6. Рассмотрим грамматику G из примера 2.2. С помощью только что описанного алгоритма проверим: $abac \in L(G)$? Получим, что

$T_0 = \{S\}$.
$T_1 = \{S, aSBC, aBC\}$.
$T_2 = \{S, aSBC, aBC, abC\}$.
$T_3 = \{S, aSBC, aBC, abC, abc\}$.
$T_4 = T_3$.

Поскольку $abac \notin T_3$, то $abac \notin L(G)$.

§ 2.6. Деревья вывода в контекстно-свободных грамматиках

Рассмотрим теперь наглядный метод описания любого вывода в контекстно-свободной грамматике. Фактически мы его уже могли наблюдать в §2.1.

***Определение* 2.10.** Пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика. Дерево есть *дерево вывода* для G, если оно удовлетворяет следующим четырем условиям:

1) каждый узел имеет метку — символ из алфавита V;
2) метка корня — S;
3) если узел имеет по крайней мере одного потомка, то его метка должна быть нетерминалом;
4) если узлы $n_1, n_2, \ldots, n_k$ — прямые потомки узла n, перечисленные слева направо, с метками $A_1, A_2, \ldots, A_k$ соответственно, а метка узла n есть A, то $A \to A_1A_2\ldots A_k \in P$.

Пример 2.7. Рассмотрим КС-грамматику $G = (\{S, A\}, \{a, b\}, P, S)$, где $P = \{S \to aAS, S \to a, A \to SbA, A \to ba, A \to SS\}$.

На рис. 2.2 изображено дерево, представляющее вывод:
$S \Rightarrow aAS \Rightarrow aSbAS \Rightarrow aabAS \Rightarrow aabbaS \Rightarrow aabbaa$.

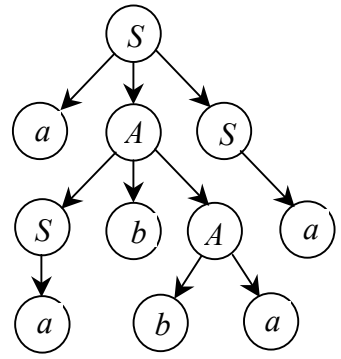

Рис. 2.2.

Результат *aabbaa* этого дерева вывода получается, если выписать метки листьев слева направо.

Имеет место следующая

Теорема 2.3. *Пусть* $G = (V_N, V_T, P, S)$ *— контекстно-свободная грамматика. Вывод* $S \overset{*}{\underset{G}{\Rightarrow}} \alpha$, *где* $\alpha \in V^*$, $\alpha \neq \varepsilon$, *существует тогда и только тогда, когда существует дерево вывода в грамматике G с результатом* α.

Доказательство. Будем доказывать аналогичное утверждение для грамматик $G_A = (V_N, V_T, P, A)$ с одними и теми же V_N, V_T и P, но с разными начальными символами: $A \in V_N$. Если это вспомогательное утверждение будет доказано для любой грамматики G_A, то справедливость утверждения теоремы будет следовать просто как частный случай при $A = S$.

Поскольку, как было сказано, во всех грамматиках одни и те же правила, то утверждение $A \overset{*}{\underset{G_A}{\Rightarrow}} \alpha$ эквивалентно утверждению $A \overset{*}{\underset{G_B}{\Rightarrow}} \alpha$ для любого $B \in V_N$, в частности при $B = S$, так как $G_S = G$, имеем также $A \overset{*}{\underset{G}{\Rightarrow}} \alpha$.

Все узлы дерева, не являющиеся листьями, будем называть *внутренними*.

I. Пусть $\alpha \in V^+$ есть результат дерева вывода для грамматики G_A. Индукцией по числу внутренних узлов k в дереве вывода покажем, что тогда $A \overset{*}{\underset{G_A}{\Rightarrow}} \alpha$.

База. Пусть $k = 1$, тогда имеется только один внутренний узел. В этом случае дерево имеет такой вид, как показано на рис. 2.3.

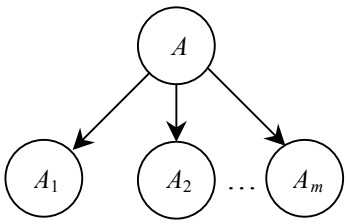

Рис. 2.3.

По определению дерева вывода $A \to A_1 A_2 \ldots A_m$ должно быть правилом грамматики G_A и, следовательно, вывод $A \overset{*}{\underset{G_A}{\Rightarrow}} \alpha$ существует.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех $k \le n$ ($n \ge 1$).

Индукционный переход. Пусть α есть результат дерева вывода с корнем, помеченным нетерминалом A, в котором k внутренних узлов, причем $k = n + 1$. Рассмотрим прямых потомков корня данного дерева вывода. Они не могут быть все листьями, так как в противном случае дерево имело бы только одну внутреннюю вершину — корень, а их должно быть не меньше двух.

Пусть метки прямых потомков корня, перечисленных в порядке слева направо: $A_1, A_2, \ldots, A_n$. Перенумеруем эти узлы в том же порядке: 1, 2, ... , n. По определению дерева вывода $A \to A_1 A_2 \ldots A_n \in P$.

Если узел i — не лист, то он — корень некоторого поддерева, в котором внутренних узлов не больше n. По индукционному предположению результат этого поддерева, обозначим его α_i, выводим из A_i, где A_i — нетерминал. В обо-

значениях это можно записать так: $A_i \overset{*}{\underset{G_{A_i}}{\Rightarrow}} \alpha_i$. Если же A_i — лист, то $A_i = \alpha_i$ и в этом случае $A_i \overset{*}{\underset{G_{A_i}}{\Rightarrow}} \alpha_i$.

Легко видеть, что если $i < j$, то узел i и все его потомки должны быть левее узла j и всех его потомков, и потому $\alpha = \alpha_1\alpha_2 \ldots \alpha_n$.

Мы можем теперь, используя правило $A \to A_1 A_2 \ldots A_n$ и все частичные выводы, выстроить вывод:

$A \underset{G_A}{\Rightarrow} A_1 A_2 \ldots A_n \overset{*}{\underset{G_A}{\Rightarrow}} \alpha_1 A_2 \ldots A_n \overset{*}{\underset{G_A}{\Rightarrow}} \alpha_1\alpha_2 \ldots A_n \overset{*}{\underset{G_A}{\Rightarrow}} \ldots \overset{*}{\underset{G_A}{\Rightarrow}} \alpha_1\alpha_2 \ldots \alpha_n = \alpha$.

Итак, $A \overset{*}{\underset{G_A}{\Rightarrow}} \alpha$. Утверждение I доказано.

II. Пусть $A \overset{*}{\underset{G_A}{\Rightarrow}} \alpha$. Индукцией по длине вывода l покажем, что существует дерево вывода в грамматике G_A, результат которого есть α.

База. Пусть $l = 1$. Если $A \underset{G_A}{\Rightarrow} \alpha$, то на этом единственном шаге вывода используется правило $A \to \alpha \in P$. Если $\alpha = A_1 A_2 \ldots A_m$, то по определению дерево, показанное на рис. 2.3, есть дерево вывода в грамматике G_A. Очевидно, что его результат есть α.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех $l \le n$ ($n \ge 1$).

Индукционный переход. Пусть $A \overset{l}{\underset{G_A}{\Rightarrow}} \alpha$, где $l = n + 1$. Этот вывод имеет длину, по крайней мере, 2. Следовательно, имеется первый шаг и n других шагов ($n \ge 1$), т. е. вывод имеет вид

$A \underset{G_A}{\Rightarrow} A_1 A_2 \ldots A_m \overset{l_1}{\underset{G_A}{\Rightarrow}} \alpha_1 A_2 \ldots A_m \overset{l_2}{\underset{G_A}{\Rightarrow}} \alpha_1\alpha_2 \ldots A_m \ldots \overset{l_m}{\underset{G_A}{\Rightarrow}} \alpha_1\alpha_2 \ldots \alpha_m = \alpha$.

Здесь $l_1 + l_2 + \ldots + l_m = n$, причем $l_i \le n$ ($1 \le i \le m$).

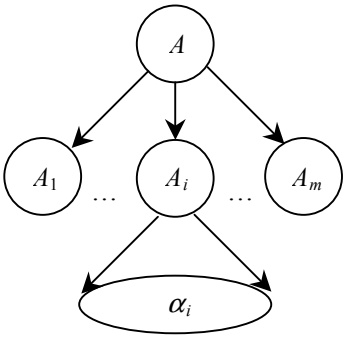

Рис. 2.4.

Если $l_i = 0$, то $\alpha_i = A_i$. Если $l_i > 0$, то по индукционному предположению существует дерево вывода T_i с корнем, имеющем метку A_i и результатом α_i. Но первый шаг вывода предполагает существование правила $A \to A_1 A_2 \ldots A_m \in P$. Следовательно, можно построить дерево вывода, верхняя часть которого будет иметь такой же вид, как на рис. 2.3.

Далее, те вершины, которые помечены символами A_i и для которых существуют выводы вида $A_i \overset{l_i}{\underset{G_A}{\Rightarrow}} \alpha_m$ при $l_i > 0$, заменим деревьями вывода T_i с корнями, помеченными A_i, и результатами α_i. То, что получится — см. рис. 2.4, является деревом вывода $A \overset{l}{\underset{G_A}{\Rightarrow}} \alpha$ в грамматике G_A. Утверждение II доказано.

Из I и II при $A = S$ следует утверждение теоремы.