1.3. Порождающие грамматики

[Гин, 1.1], [Сал, 2.1], [АхоУль, 2.1.2], [Гла, 1.2], [Лал, с. 159–161], [Бра, с. 32–36], [ГлаМел, с. 34–48], [ГорМол, с. 354–355, 367–370], [СокКушБад, с. 12–13], [ТраБар, 1.12], [LewPap2, 4.6], [Рей, с. 28–30], [КукБей, с. 264–268]

Определение 1.41. *Порождающей грамматикой* (*грамматикой типа* 0) (generative grammar, rewrite grammar) называется четвёрка $G \rightleftharpoons \langle N, \Sigma, P, S\rangle$, где N и Σ — конечные алфавиты, $N \cap \Sigma = \varnothing$, $P \subset (N \cup \Sigma)^+ \times (N \cup \Sigma)^*$, P конечно и $S \in N$. Здесь Σ — *основной алфавит* (*терминальный алфавит*), его элементы называются *терминальными символами* или *терминалами* (terminal), N — *вспомогательный алфавит* (*нетерминальный алфавит*), его элементы называются *нетерминальными символами*, *нетерминалами* или *переменными* (nonterminal, variable), S — *начальный символ* (*аксиома*) (start symbol). Пары $(\alpha, \beta) \in P$ называются *правилами подстановки*, просто *правилами* или *продукциями* (rewriting rule, production) и записываются в виде $\alpha \to \beta$.

Пример 1.42. Пусть даны множества $N = \{S\}$, $\Sigma = \{a, b, c\}$, $P = \{S \to acSbcS,\ cS \to \varepsilon\}$. Тогда $\langle N, \Sigma, P, S\rangle$ является порождающей грамматикой.

Замечание 1.43. Будем обозначать элементы множества Σ строчными буквами из начала латинского алфавита, а элементы множества N — заглавными латинскими буквами. Обычно в примерах мы будем задавать грамматику в виде списка правил, подразумевая, что алфавит N составляют все заглавные буквы, встречающиеся в правилах, а алфавит Σ — все строчные буквы, встречающиеся в правилах. При этом правила порождающей грамматики записывают в таком порядке, что левая часть первого правила есть начальный символ S.

Замечание 1.44. Для обозначения n правил с одинаковыми левыми частями $\alpha \to \beta_1, \ldots, \alpha \to \beta_n$ часто используют сокращённую запись $\alpha \to \beta_1 \mid \ldots \mid \beta_n$.

Определение 1.45. Пусть дана грамматика G. Пишем $\phi \underset{G}{\Rightarrow} \psi$, если $\phi = \eta\alpha\theta$, $\psi = \eta\beta\theta$ и $(\alpha \to \beta) \in P$ для некоторых слов α, β, η, θ в алфавите $N \cup \Sigma$.

Замечание 1.46. Когда из контекста ясно, о какой грамматике идёт речь, вместо $\underset{G}{\Rightarrow}$ можно писать просто $\Rightarrow$.

Пример 1.47. Пусть

$$G = \langle \{S\}, \{a, b, c\}, \{S \to acSbcS, cS \to \varepsilon\}, S\rangle.$$

Тогда $cSacS \underset{G}{\Rightarrow} cSa$.

Определение 1.48. Если $\omega_0 \underset{G}{\Rightarrow} \omega_1 \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} \omega_n$, где $n \geqslant 0$, то пишем $\omega_0 \overset{*}{\underset{G}{\Rightarrow}} \omega_n$ (другими словами, бинарное отношение $\overset{*}{\underset{G}{\Rightarrow}}$ является рефлексивным, транзитивным замыканием бинарного отношения $\underset{G}{\Rightarrow}$, определённого на множестве $(N \cup \Sigma)^*$). При этом последовательность слов $\omega_0, \omega_1, \ldots, \omega_n$ называется *выводом* (derivation) слова ω_n из слова ω_0 в грамматике G. Число n называется *длиной* (*количеством шагов*) этого вывода.

Замечание 1.49. В частности, для всякого слова $\omega \in (N \cup \Sigma)^*$ имеет место $\omega \overset{*}{\underset{G}{\Rightarrow}} \omega$ (так как возможен вывод длины 0).

Пример 1.50. Пусть $G = \langle \{S\}, \{a, b\}, \{S \to aSa,\ S \to b\}, S\rangle$. Тогда $aSa \overset{*}{\underset{G}{\Rightarrow}} aaaaSaaaa$. Длина этого вывода — 3.

Определение 1.51. Язык, *порождаемый* грамматикой G, — это множество $L(G) \rightleftharpoons \{\omega \in \Sigma^* \mid S \overset{*}{\underset{G}{\Rightarrow}} \omega\}$. Будем также говорить, что грамматика G *порождаёт* (generates) язык $L(G)$.

Замечание 1.52. Существенно, что в определение порождающей грамматики включены два алфавита — Σ и N. Это позволило нам в определении 1.51 “отсеять” часть слов, получаемых из начального символа. А именно, отбрасывается каждое слово, содержащее хотя бы один символ, не принадлежащий алфавиту Σ.

Пример 1.53. Если $G = \langle \{S\}, \{a, b\}, \{S \to aSa,\ S \to bb\}, S\rangle$, то $L(G) = \{a^n bba^n \mid n \geqslant 0\}$.

Определение 1.54. Две грамматики *эквивалентны*, если они порождают один и тот же язык.

Пример 1.55. Грамматика $S \to abS$, $S \to a$ и грамматика $T \to aU$, $U \to baU$, $U \to \varepsilon$ эквивалентны.

1.4. Классы грамматик

[Гин, с. 23–24, 78–79], [АхоУль, 2.1.3, с. 191], [Сал, 2.1, с. 94], [Гла, 1.2, 1.3], [Бра, с. 39–45], [ГлаМел, с. 54, 63, 69–70], [ГорМол, с. 361–367], [ТраБар, 1.12], [КукБей, с. 268–271], [ЛПИИ, 5.2.1]

Определение 1.56. *Контекстной грамматикой (контекстно-зависимой грамматикой, грамматикой непосредственно составляющих, НС-грамматикой, грамматикой типа* 1) (context-sensitive grammar, phrase-structure grammar) называется порождающая грамматика, каждое правило которой имеет вид $\eta A\theta \to \eta\alpha\theta$, где $A \in N$, $\eta \in (N \cup \Sigma)^*$, $\theta \in (N \cup \Sigma)^*$, $\alpha \in (N \cup \Sigma)^+$.

Пример 1.57. Грамматика $S \to TS$, $S \to US$, $S \to b$, $Tb \to Ab$, $A \to a$, $TA \to AAT$, $UAb \to b$, $UAAA \to AAU$ не является контекстной (последние три правила не имеют требуемого вида).

Определение 1.58. *Контекстно-свободной грамматикой (КС-грамматикой, бесконтекстной грамматикой, грамматикой типа* 2) (context-free grammar) называется порождающая грамматика, каждое правило которой имеет вид $A \to \alpha$, где $A \in N$, $\alpha \in (N \cup \Sigma)^*$.

Пример 1.59. Грамматика $S \to ASTA$, $S \to AbA$, $A \to a$, $bT \to bb$, $AT \to UT$, $UT \to UV$, $UV \to TV$, $TV \to TA$ является контекстной, но не контекстно-свободной (последние пять правил не имеют требуемого вида).

Определение 1.60. *Линейной грамматикой* (linear grammar) называется порождающая грамматика, каждое правило которой имеет вид $A \to u$ или $A \to uBv$, где $A \in N$, $u \in \Sigma^*$, $v \in \Sigma^*$, $B \in N$.

Пример 1.61. Грамматика $S \to TT$, $T \to cTT$, $T \to bT$, $T \to a$ является контекстно-свободной, но не линейной (первые два правила не имеют требуемого вида).

Определение 1.62. *Праволинейной грамматикой (рациональной грамматикой, грамматикой типа* 3) (right-linear grammar) называется порождающая грамматика, каждое правило которой имеет вид $A \to u$ или $A \to uB$, где $A \in N$, $u \in \Sigma^*$, $B \in N$.

Пример 1.63. Грамматика $S \to aSa$, $S \to T$, $T \to bT$, $T \to \varepsilon$ является линейной, но не праволинейной (первое правило не имеет требуемого вида).

Пример 1.64. Грамматика $S \to T$, $U \to abba$ праволинейная.

Пример 1.65. Грамматика $S \to aS$, $S \to bS$, $S \to aaaT$, $S \to aabaT$, $S \to abaaT$, $S \to aabbaT$, $S \to ababaT$, $S \to abbaaT$, $T \to aT$, $T \to bT$, $T \to \varepsilon$ праволинейная.

Пример 1.66. Грамматика $S \to \varepsilon$, $S \to aaaS$, $S \to abbS$, $S \to babS$, $S \to aabT$, $T \to abaT$, $T \to baaT$, $T \to bbbT$, $T \to bbaS$ праволинейная. Обобщённый вариант языка, порождаемого этой грамматикой, используется в доказательстве разрешимости арифметики Пресбургера [Sip, с. 207–208].

Определение 1.67. Правила вида $\alpha \to \varepsilon$ называются *ε-правилами*.

Лемма 1.68. *Каждая праволинейная грамматика является линейной. Каждая линейная грамматика является контекстно-свободной. Каждая контекстно-свободная грамматика без ε-правил является контекстной грамматикой.*

Определение 1.69. Классы грамматик типа 0, 1, 2 и 3 образуют *иерархию Хомского* (Chomsky hierarchy).

Определение 1.70. Язык называется *контекстным языком (контекстно-свободным языком, линейным языком, праволинейным языком)*, если он порождается некоторой контекстной грамматикой (соответственно контекстно-свободной грамматикой, линейной грамматикой, праволинейной грамматикой). Контекстно-свободные языки называются также *алгебраическими языками*.

Пример 1.71. Пусть дан произвольный алфавит $\Sigma = \{a_1, \ldots a_n\}$. Тогда язык Σ^* является праволинейным, так как он порождается грамматикой $S \to \varepsilon$, $S \to a_1 S$, $\ldots$, $S \to a_n S$.

Правила обычно записывают в виде $\boldsymbol{\alpha \to \beta}$ или $\boldsymbol{\alpha ::= \beta}$, что означает **$\alpha$ *порождает* (*состоит из*) β**.

Язык, порождаемый грамматикой, - *это множество цепочек, которые состоят только из терминалов и выводятся, начиная с одного, особо выделенного, нетерминала* **S**, *называемого* ***начальным символом*** *или* ***аксиомой*** *грамматики. Среди множества правил грамматики должно присутствовать хотя бы одно правило* $\boldsymbol{S \to \beta}$.

Рассмотрим ряд грамматик и обсудим алгоритм порождения, применяемый для вывода цепочек языка.

Пример 1.1.

<предложение>::=<подлежащее><группа сказуемого>
<подлежащее>::=мать | отец
<группа сказуемого>::=<сказуемое><дополнение>
<сказуемое>::=любит | обожает | боготворит
<дополнение>::=сына | дочь □

Если имеется множество правил, то ими можно воспользоваться для того, чтобы ***вывести*** или ***породить*** цепочку (предложение) по следующей схеме. Начнем с начального символа грамматики - <предложение>, найдем правило, в котором <предложение> слева от ::= , и подставим вместо <предложение> цепочку, которая расположена справа от ::=, то есть

<предложение> ⇒ <подлежащее> <группа сказуемого>.

Таким образом, мы заменяем синтаксическое понятие на одну из цепочек, из которых оно может состоять. Повторим процесс. Возьмем один из нетерминалов в цепочке <подлежащее> <группа сказуемого>, например <подлежащее>; найдем правило, где <подлежащее> находится слева от ::=, и заменим <подлежащее> в исходной цепочке на соответствующую цепочку, которая находится справа от ::=. Это дает

<подлежащее> < группа сказуемого > ⇒ мать < группа сказуемого > .

Символ "⇒" означает, что один символ слева от ⇒ в соответствии с правилом грамматики заменяется цепочкой, находящейся справа от ⇒. Полный вывод одного предложения будет таким:

<предложение> ⇒ <подлежащее> <группа сказуемого>
⇒ мать < группа сказуемого >
⇒ мать <сказуемое> <дополнение>
⇒ мать любит <дополнение>
⇒ мать любит сына.

Этот вывод предложения запишем сокращенно, используя новый символ $\Rightarrow^+$<предложение> $\Rightarrow^+$ мать любит сына.

На каждом шаге можно заменить ***любой*** нетерминал. В приведенном выше выводе всегда заменялся самый левый из них.

Вывод, на каждом шаге которого заменяется самый левый нетерминал сентенциальной формы, называется ***левым*** (***левосторонним***) ***выводом***. Существует и часто используется также ***правый*** (***правосторонний***) ***вывод***, который получается, если в сентенциальной форме заменять всегда самый правый нетерминал.

Обратите внимание на то, что предложенная грамматика используется для описания многих предложений. Девять правил грамматики, если считать каждую альтернативу за отдельное правило, а так оно и есть, определяют двенадцать предложений (цепочек) языка:

мать любит сына	*мать обожает сына*	*мать боготворит сына*
мать любит дочь	*мать обожает дочь*	*мать боготворит дочь*
отец любит сына	*отец обожает сына*	*отец боготворит сына*
отец любит дочь	*отец обожает дочь*	*отец боготворит дочь*

Одно из назначений грамматики как раз и состоит в том, чтобы описывать ***все*** цепочки языка с помощью приемлемого числа правил. Это особенно важно, если учесть, что количество предложений в языке, чаще всего, бесконечно.

Рассмотрим еще один пример полезной грамматики

Пример 1.2. Грамматики целого числа без знака содержат следующие 13 правил

(1) <число>::=<чс>	$S \to A$
(2) <чс>::=<цифра><чс>	$A \to AB$
(3) <чс>::=<цифра>	$A \to B$
(4) <цифра>::=0	$B \to 0$
(5) <цифра>::=1	$B \to 1$
(6) <цифра>::=2	$B \to 2$
(7) <цифра>::=3	$B \to 3$
(8) <цифра>::=4	$B \to 4$
(9) <цифра>::=5	$B \to 5$
(10) <цифра>::=6	$B \to 6$
(11) <цифра>::=7	$B \to 7$
(12) <цифра>::=8	$B \to 8$
(13) <цифра>::=9	$B \to 9$.

Пусть ***G*** *- грамматика. Будем говорить, что цепочка* α ***непосредственно порождает*** *цепочку* β, *и обозначим* $\alpha \Rightarrow \beta$, *если для некоторых цепочек* φ *и* ψ *можно написать* $\alpha = \varphi U \psi$, $\beta = \varphi\gamma\psi$, *где* $U ::= \gamma$ *правило грамматики* ***G***. *Будем также говорить, что* β ***непосредственно выводима*** *из* α *или что* β ***непосредственно приводится (редуцируется, сворачивается)*** *к* α.

Цепочки φ и ψ, конечно, могут быть пустыми. Следовательно, для любого правила $A \to \alpha$ грамматики G имеет место $A \Rightarrow \alpha$. На рис. 1.1 даны некоторые примеры непосредственных выводов для грамматики G(<число>) из примера 1.2 и обозначений предыдущего определения.

Будем говорить, что α ***порождает*** β *или* β ***приводится*** *к* α *и записывать* $\alpha \Rightarrow^+ \beta$, *если существует последовательность непосредственных выводов* $\alpha = \gamma_0 \Rightarrow \gamma_1 \Rightarrow \gamma_2 \Rightarrow \ldots \Rightarrow \gamma_n = \beta$, *где* $n > 0$. *Эта последовательность называется* ***выводом длины n***. *Будем писать* $\alpha \Rightarrow^* \beta$, *если* $\alpha \Rightarrow^+ \beta$ *или* $\alpha = \beta$.

α		β	Номера правил	φ	ψ
<число>	⇒	<чс>	1	ε	ε
<чс>	⇒	<цифра> <чс>	2	ε	ε
<цифра> <чс>	⇒	2 <чс>	6	ε	<чс>
2 <чс>	⇒	2 <цифра>	3	2	ε
2 <цифра>	⇒	25	9	2	ε

Рисунок 1.1. Вывод для грамматики целых чисел без знака

Если просмотреть все строки на рисунке 1.1, то мы получим <число> ⇒ <чс> ⇒ <цифра> <чс> ⇒ 2 <чс> ⇒ 2 <цифра> ⇒ 25.

Таким образом, <число> ⇒⁺ 22 и длина вывода равна 5. (Если длина вывода известна можно, записывать в явном виде <число> $\Rightarrow^5$ 22)

Заметим, что пока в цепочке есть хотя бы один нетерминал, из нее можно вывести новую цепочку, однако если нетерминальные символы отсутствуют, то вывод завершен. Неслучайно "*терминалом*" (*terminal* - заключительный, конечный) называют символ, который не встречается в левой части ни одного из