Глава 1. Элементы теории формальных языков и грамматик

1.3. Формальные грамматики

Формальные грамматики предназначены для описания формальных языков и имеют большое сходство с БНФ. Однако их нельзя отождествлять, поскольку формальные грамматики имеют более общий характер.

Грамматика определяется как четверка $G = (V_T, V_N, P, S)$, где V_T – конечное множество *терминальных* символов (*терминалов*), т. е. символов, принадлежащих собственно описываемому формальному языку; V_N – конечное множество *нетерминальных* символов (*нетерминалов*), т. е. символов, принадлежащих метаязыку (необходимо отметить, что у V_T и V_N нет общих символов, т. е. $V_T \cap V_N = \varnothing$); P – конечное множество *продукций* (*правил вывода*, *порождающих правил*) вида $\alpha \to \beta$, где α – левая часть продукции, это такая строка, что $\alpha \in (V_T \cup V_N)^+$, а β – правая часть, такая строка, что $\beta \in (V_T \cup V_N)^*$; $S \in V_N$ – *начальный символ* (*аксиома*) грамматики.

Примем следующие соглашения об обозначениях:

1. Терминалы будем представлять строчными буквами и спецсимволами из словаря языка (символы цифр, операций, пунктуации и т. п.). Терминалы, являющиеся словами-символами языка, – выделять жирным шрифтом. Примеры терминалов: *a*, +, **begin**, **while**.

2. Нетерминалы будем представлять прописными буквами. Если обозначение нетерминала представляет собой многосимвольное слово – заключать его в угловые скобки (при этом необязательно использовать прописные буквы). Примеры нетерминалов: *A*, <Оператор>, <Последовательность операторов>.

Пусть дана грамматика $G = (\{a, b\}, \{S\}, P, S)$, где P представляет множество следующих продукций: $S \to aSb$, $S \to \varepsilon$, или в более краткой форме записи: $S \to aSb \mid \varepsilon$. Продукции с одинаковыми левыми частями будем называть *альтернативными*.

Чтобы вывести предложение этого языка, поступают следующим образом. Начинают с начального символа S и заменяют его на aSb или ε. Если S опять появится в полученной строке, его опять можно заменить с помощью одного из этих правил, и т. д. Полученная таким образом любая строка, не содержащая S, является предложением этого языка. Последовательность таких шагов называется *выводом* (*схемой вывода*, *порождением*) строки (предложения) и обычно записывается как

$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$.

Выводимая строка формально определяется следующим образом. Пусть $G = (V_T, V_N, P, S)$ – грамматика и пусть $\gamma_1\alpha\gamma_2 \in (V_T \cup V_N)^+$ – строка терминальных и нетерминальных символов длиной ≥ 1. Если $\alpha \to \beta$ – продукция из P, то подстрока α в строке может быть заменена строкой β, и в результате получится $\gamma_1\beta\gamma_2$. Это записывается

$$\gamma_1\alpha\gamma_2 \Rightarrow \gamma_1\beta\gamma_2,$$

при этом говорят, что строка $\gamma_1\alpha\gamma_2$ *генерирует* строку $\gamma_1\beta\gamma_2$ или что строка $\gamma_1\beta\gamma_2$ *выводится* из строки $\gamma_1\alpha\gamma_2$.

Если $\alpha_1, \alpha_2, ..., \alpha_n \in (V_T \cup V_N)^*$, и $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_{n-1} \Rightarrow \alpha_n$ $(n \geq 1)$, то обычно пишут сокращенно $\alpha_1 \overset{+}{\Rightarrow} \alpha_n$, и при этом говорят, что строка α_n выводится из строки α_1 за один или более шагов. Аналогично $\alpha_1 \overset{*}{\Rightarrow} \alpha_n$ означает, что строку α_n можно вывести из строки α_1 с помощью нуля или более применений правил грамматики.

Если строка $\alpha \in (V_T \cup V_N)^*$ такая, что $S \overset{*}{\Rightarrow} \alpha$, то строку α называют *сентенциальной формой* грамматики *G*. *Сентенцией* грамматики *G* называют произвольную сентенциальную форму из V_T^*, т. е. произвольную строку терминальных символов, которая может быть выведена из начального символа *S*. Тогда множество всех сентенций грамматики *G* называется языком, порожденным грамматикой *G*, и обозначается *L*(*G*).

Таким образом, $L(G) = \{x \in V_T^* \mid S \overset{*}{\Rightarrow} x\}$.

Если две различные грамматики G и G' порождают один и тот же язык, т. е. $L(G) = L(G')$, то грамматики G и G' *эквивалентны*. Например, грамматики $G_1 = (\{a, b\}, \{S\}, P_1, S)$ и $G_2 = (\{a, b\}, \{S, A, B\}, P_2, S)$ с продукциями

P_1: $S \to aSb \mid ab$ $\qquad$ P_2: $S \to aA \mid aB$

$A \to Sb$

$B \to b$

эквивалентны, поскольку порождают один и тот же язык $L(G_1) = L(G_2) = \{a^n b^n \mid n \geq 1\}$.

В последующем изложении для краткости вместо полной четверки $G = (V_T, V_N, P, S)$ грамматику будем представлять только множеством продукций P, считая, что начальный символ грамматики обязательно стоит в левой части первой продукции.

1.4. Классификация грамматик и языков

Одна из стандартных классификаций грамматик – *иерархия Хомского*, основанная на наложении определенных ограничений на вид продукций грамматики.

Тип 0 – грамматика общего вида

Любая грамматика определенного выше вида называется *грамматикой типа 0*, или *грамматикой общего вида*. На вид продукций не наложено никаких ограничений. Не имеют практического применения из-за своей сложности.

Тип 1 – контекстно-зависимые

К этому типу относят контекстно-зависимые и неукорачивающие грамматики.

Грамматика называется *контекстно-зависимой*, если каждая продукция имеет вид $\alpha A\beta \to \alpha\gamma\beta$, где $\alpha, \beta \in (V_T \cup V_N)^*$, $\gamma \in (V_T \cup V_N)^+$, $A \in V_N$.

Грамматика называется *неукорачивающей*, если для всех продукций вида $\alpha \to \beta$ выполняется ограничение $|\alpha| \le |\beta|$, где $|\alpha|$ и $|\beta|$ – соответствующие длины строк. В таких продукциях правая часть не короче левой.

Эти классы грамматик эквивалентны. В виде исключения в них допускается продукция вида $S \to \varepsilon$ (правая часть короче левой), где S – начальный символ грамматики, который не встречается в правых частях других продукций.

Эти грамматики могут использоваться при анализе текстов на естественных языках, однако при построении компиляторов практически не используются из-за своей сложности.

Пример грамматики типа 1:

$S \to aSBC \mid aBC$

$CB \to BC$

$aB \to ab$

$bB \to bb$

$bC \to bc$

$cC \to cc$

Данная грамматика порождает язык $\{a^n b^n c^n \mid n \geq 1\}$.

Пример вывода строки *aabbcc*:

$S \Rightarrow aSBC \Rightarrow aaBCBC \Rightarrow aabCBC \Rightarrow aabBCC \Rightarrow aabbCC$

$\Rightarrow aabbcC \Rightarrow aabbcc$.

Тип 2 – контекстно-свободные

Грамматика называется *грамматикой типа 2*, или *контекстно-свободной*, если каждая продукция имеет вид $A \to \beta$, где $A \in V_N$, $\beta \in (V_T \cup V_N)^*$. В КС-грамматике все левые части продукций состоят из одного нетерминального символа.

Продукцию вида $A \to \varepsilon$ (правая часть – пустая строка) будем называть ε-*продукцией*.

КС-грамматики широко используются для описания синтаксиса языков программирования.

Тип 3 – регулярные

Выделяют классы праволинейных и леволинейных регулярных грамматик.

Грамматика называется *праволинейной регулярной*, если все ее продукции имеют вид $A \to a$ или $A \to aB$, где $a \in V_T$, $A, B \in V_N$.

Грамматика называется *леволинейной регулярной*, если все ее продукции имеют вид $A \to a$ или $A \to Ba$, где $a \in V_T$, $A, B \in V_N$.

Если пустая строка принадлежит языку, допускается единственная продукция вида $S \to \varepsilon$ (S – начальный символ грамматики), при этом ни одна продукция грамматики не должна содержать нетерминал S в своей правой части.

Эти классы грамматик называют также *автоматными*, поскольку существуют простые методы синтеза конечных автоматов по заданным автоматным грамматикам.

Регулярные (автоматные) грамматики являются частным случаем линейных грамматик.

Грамматика называется *линейной*, если все ее продукции (за исключением ε-продукций) имеют вид $A \to \alpha$ или $A \to \alpha B \beta$, где $\alpha, \beta \in V_T^*$, $A, B \in V_N$, т. е. в правой части продукции может содержаться не более одного нетерминала. Если во всех продукциях вида $A \to \alpha B \beta$ имеет место $\alpha = \varepsilon$, грамматика называется *леволинейной*, если же $\beta = \varepsilon$ – *праволинейной*. Классы леволинейных и праволинейных грамматик эквивалентны и описывают регулярные языки, они эквивалентны регулярным автоматным грамматикам (поэтому в ряде работ их относят к регулярным грамматикам). Любую праволинейную (леволинейную) грамматику можно преобразовать в эквивалентную регулярную автоматную грамматику.

Например, продукцию вида $A \to abcB$ можно заменить на продукции $A \to aC$, $C \to bD$, $D \to cB$ (обозначив подстроку bcB через нетерминал C, подстроку cB через нетерминал D) и получить эквивалентную автоматную грамматику.

В связи с тем, что классы праволинейных и леволинейных регулярных грамматик эквивалентны, в дальнейшем будем подразумевать под регулярными грамматиками праволинейные регулярные автоматные грамматики.

Следует отметить, что продукции регулярной грамматики должны быть либо только леволинейными, либо только праволинейными. Их совместное использование в общем случае выводит грамматику из класса регулярных. Например, грамматика

$S \to aA \mid aB$

$A \to Sb$

$B \to b$

описывает контекстно-свободный язык $\{a^n b^n \mid n \geq 1\}$ и не является регулярной.

Очевидно, что эта иерархия – включающая, т. е. регулярные грамматики являются контекстно-свободными, а они в свою очередь – контекстно-зависимыми и т. д.

Иерархии грамматик соответствует иерархия языков. Однако при этом необходимо учитывать следующий факт. Например, если язык генерируется посредством контекстно-зависимой грамматики, то это не обязательно означает, что язык только контекстно-зависимый и не может быть КС, поскольку, если язык контекстно-свободный, то всегда можно определить эквивалентную контекстно-свободную грамматику.

Распознавателем языка типа 0 является машина Тьюринга. Распознаватель контекстно-зависимых языков – линейно ограниченный автомат, представляющий собой машину Тьюринга, в которой лента не бесконечна, а ограничена длиной входного слова.

Наибольшее практическое применение находят регулярные (на этапе лексического анализа) и контекстно-свободные (на этапе синтаксического анализа) грамматики, которые позволяют специфицировать большинство конструкций современных языков программирования и использовать в качестве распознавателей строк языка достаточно простые средства. В частности, распознавателем регулярного языка является конечный автомат, а распознавателем контекстно-свободного языка – магазинный автомат (МП-автомат).