Глава 1. Элементы теории формальных языков и грамматик

1.5. Контекстно-свободные грамматики

Формальное определение основных синтаксических понятий большинства языков программирования может быть учтено с помощью БНФ (РБНФ) или СД. При этом вид левой части каждой продукции может быть ограничен лишь единственным нетерминалом, т. е. синтаксис большинства языков программирования в своей основной части может быть определен с помощью КС-грамматик.

КС-грамматика – грамматика $G = (V_T, V_N, P, S)$, каждая продукция которой имеет вид $A \to \beta$, где $A \in V_N$, $\beta \in (V_T \cup V_N)^*$. Для краткости продукции вида $A \to \beta$ с нетерминалом A в левой части будем называть *A-продукциями*.

При соблюдении соглашений об обозначении терминалов и нетерминалов для задания грамматики достаточно определить множество продукций и указать начальный нетерминал. Если не оговорено особо, начальным нетерминалом будем считать нетерминал левой части первой продукции множества.

Язык, порожденный КС-грамматикой, называется *КС-языком*. Термин «КС» обусловлен тем, что любой символ $A \in V_N$ в сентенциальной форме грамматики G может быть раскрыт согласно продукции $A \to \beta$ независимо от того, какими строками он окружен внутри самой сентенциальной формы.

В КС-грамматике любая строка $x \in L(G)$ может быть выведена из начального символа S. Общепринятым методом представления такого вывода является *дерево разбора* (*дерево грамматического разбора*, *дерево вывода*).

Дерево разбора строится по порождению $S \Rightarrow A_1A_2...A_n \Rightarrow ... \Rightarrow T_1T_2...T_k$, где $A_i \in (V_T \cup V_N)$, $i = 1, 2, \ldots, n$; $T_j \in V_T$, $j = 1, 2, \ldots, k$, следующим образом. Из корня дерева, помеченного начальным нетерминалом S, отходит n ветвей по числу символов в строке, непосредственно порожденной начальным нетерминалом. Каждая из n ветвей заканчивается вершиной, помеченной символом A_i. Вершины метятся слева направо в порядке возрастания номера индекса. Если в процессе порождения к нетерминалу A_i применена продукция $A_i \rightarrow B_1B_2...B_t$, то вершина A_i становится корнем поддерева, из которого выходит t ветвей, каждая заканчивается вершиной, помеченной символом B_i, $i = 1, 2, \ldots, t$. Такое построение производится для всех вершин, помеченных нетерминальными символами. Построение дерева заканчивается, когда все листья оказываются помеченными терминалами T_1, T_2, ..., T_k. Совокупность этих меток при просмотре вершин слева направо образует терминальную строку (сентенцию) $T_1T_2...T_k$ (*крона дерева разбора*). Очевидно, что в полностью построенном дереве листья соответствуют терминалам, а внутренние вершины – нетерминалам.

Вывод строки $x \in L(G)$ называется *левосторонним* (*левым*), если каждая очередная сентенциальная форма получается из предыдущей заменой самого левого нетерминала.

Вывод строки $x \in L(G)$ называется *правосторонним* (*правым*), если каждая очередная сентенциальная форма получается из предыдущей заменой самого правого нетерминала.

Пример построения дерева разбора для грамматики G с продукциями

$S \to AB$

$A \to aA|a$

$B \to bB|b$

при выводе строки *aaabb* (в более краткой форме – a^3b^2) представлен на рис. 1.2. Это дерево соответствует левостороннему выводу:

$$S \Rightarrow AB \Rightarrow aAB \Rightarrow aaAB \Rightarrow aaaB \Rightarrow aaabB \Rightarrow aaabb.$$

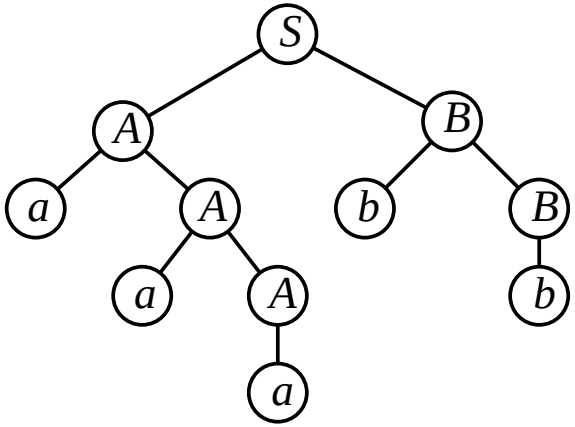

Рис. 1.2. Дерево разбора строки *aaabb*

Этому же дереву соответствует правосторонний вывод строки *aaabb*: $S \Rightarrow AB \Rightarrow AbB \Rightarrow Abb \Rightarrow aAbb \Rightarrow aaAbb \Rightarrow aaabb$.

Возможны и другие варианты вывода одной и той же строки, не являющейся ни левосторонней, ни правосторонней. Например, строку a^3b^2 можно вывести следующим образом:

$S \Rightarrow AB \Rightarrow AbB \Rightarrow aAbB \Rightarrow aaAbB \Rightarrow aaAbb \Rightarrow aaabb$,

$S \Rightarrow AB \Rightarrow aAB \Rightarrow aAbB \Rightarrow aAbb \Rightarrow aaAbb \Rightarrow aaabb$,

$S \Rightarrow AB \Rightarrow aAB \Rightarrow aaAB \Rightarrow aaAbB \Rightarrow aaabB \Rightarrow aaabb$.

Все эти схемы вывода соответствуют одному и тому же дереву разбора. Отметим, что дереву разбора соответствует единственный левосторонний (правосторонний) вывод.

Если для любой строки $x \in L(G)$ все возможные схемы вывода соответствуют одному и тому же дереву разбора, то такая КС-грамматика называется *однозначной*. Если же различным схемам вывода соответствуют несовпадающие деревья разбора, то грамматика называется *неоднозначной*.

Поскольку каждому дереву разбора соответствует единственный левосторонний вывод, то можно переопределить неоднозначную грамматику следующим образом: КС-грамматика G называется неоднозначной, если существует такая строка $x \in L(G)$, которой можно поставить в соответствие две или более различные левосторонние схемы вывода.

Для примера рассмотрим неоднозначную грамматику с продукциями $S \to SaS \mid b$.

Строку *babab* можно получить двумя различными левосторонними схемами вывода:

$S \Rightarrow SaS \Rightarrow SaSaS \Rightarrow baSaS \Rightarrow babaS \Rightarrow babab$,

$S \Rightarrow SaS \Rightarrow baS \Rightarrow baSaS \Rightarrow babaS \Rightarrow babab$.

Этим схемам вывода соответствуют различные деревья разбора (рис. 1.3).

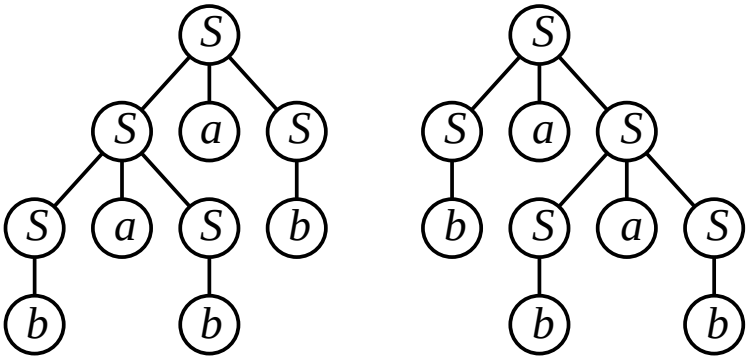

Рис. 1.3. Деревья разбора строки *babab*

Задача установления неоднозначности какой-либо грамматики алгоритмически неразрешима, т. е. не существует алгоритма, который принимал бы любую грамматику в качестве входа и определял бы, однозначна она или нет. В некоторые языки уже заложена неоднозначность. Это означает, что их нельзя генерировать с помощью однозначной грамматики. С другой стороны, некоторые неоднозначные грамматики можно преобразовать в однозначные, генерирующие тот же язык. Например, грамматика с продукциями $S \to Sab \mid b$ является однозначной и генерирует тот же язык, что и рассмотренная выше неоднозначная грамматика. Строка *babab* получается следующей левосторонней схемой вывода: $S \Rightarrow Sab \Rightarrow Sabab \Rightarrow babab$.

Рассмотрим еще один, ставший уже классическим, пример неоднозначной грамматики, определяющей условный оператор языка Паскаль:

$S \to$ **if** E **then** $S \mid$ **if** E **then** S **else** $S \mid$ **other**

Нетерминал E означает выражение, нетерминал S – оператор, терминал **other** – другие операторы.

Строку

if *E* **then** *S* **if** *E* **then** *S* **else** *S*

можно получить двумя левосторонними схемами вывода. Для конструкций подобного вида существует соглашение, что **else** относится к ближайшему незанятому **then**. Это правило устранения неоднозначности можно реализовать, изменив соответствующим образом грамматику, сделав ее однозначной.

Идея заключается в том, что между **then** и **else** допускается либо полный условный оператор **if-then-else**, либо любой оператор, не являющийся условным (такую синтаксическую конструкцию обозначим нетерминалом *A*). Остальные возможные конструкции обозначим нетерминалом *B*. В результате получится эквивалентная однозначная грамматика.

$S \to A \mid B$

$A \to$ **if** E **then** A **else** $A \mid$ **other**

$B \to$ **if** E **then** $S \mid$ **if** E **then** A **else** B

Грамматика получилась более сложной, менее естественной по сравнению с неоднозначной грамматикой.

При описании языка, если используется неоднозначная грамматика, всегда задаются правила разрешения неоднозначности. Эти правила реализуются либо построением эквивалентных однозначных грамматик (если это не слишком усложняет грамматику), либо непосредственно при разработке синтаксического анализатора, включая в него специальные механизмы реализации этих правил.