Глава 2. Лексический анализ

2.5. Регулярные грамматики

Регулярные (или автоматные) грамматики – это грамматики типа 3 в иерархии Хомского. Выделяют классы праволинейных и леволинейных регулярных грамматик.

Регулярная грамматика $G = (V_T, V_N, P, S)$ называется *праволинейной*, если все ее продукции имеют вид $A \to a$ или $A \to aB$, где $a \in V_T$, $A, B \in V_N$.

Регулярная грамматика $G = (V_T, V_N, P, S)$ называется *леволинейной*, 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 ε-продукция вида $S \to \varepsilon$ (S – начальный символ грамматики), при этом ни одна продукция грамматики не должна содержать нетерминал S в своей правой части. Следует отметить, что на практике при проектировании лексических анализаторов для формального описания шаблонов токенов обычно используют ε-свободные регулярные грамматики, поскольку лексемы не могут быть пустыми.

В связи с тем, что классы праволинейных и леволинейных регулярных грамматик эквивалентны, в дальнейшем будем рассматривать праволинейные регулярные грамматики.

Поскольку в процессе вывода некоторой сентенциальной формы производится замена нетерминала на правую часть соответствующей продукции, которая для ε-свободных регулярных грамматик содержит не более одного нетерминала, то любая сентенциальная форма регулярной грамматики либо содержит только терминалы, либо содержит один нетерминал, являющийся самым правым символом.

Язык, порождаемый регулярной грамматикой, является регулярным. Для любого регулярного языка существует порождающая его регулярная грамматика.

Если L_1 и L_2 – регулярные языки, то $L_1 \cup L_2$ и L_1L_2 тоже являются регулярными языками. Пусть $L_1 = \{a, bb\}$, $L_2 = \{\varepsilon, cc, d\}$. Тогда их объединение $L_1 \cup L_2 = \{\varepsilon, a, bb, cc, d\}$, конкатенация (сцепление) $L_1L_2 = \{a, bb, acc, bbcc, ad, bbd\}$. Эти утверждения – теоретическая база для объединения языков токенов в один регулярный язык символов языка программирования.

Пример регулярной грамматики для идентификатора в его классическом определении как последовательности букв и цифр, начинающейся с буквы:

$S \to lA$,

$A \to lA \mid dA \mid \perp$,

где терминалы l и d обозначают соответственно букву и цифру, нетерминал A – последовательность букв и/или цифр, терминал $\perp \notin \{l, d\}$ – маркер конца ввода лексемы идентификатора, т. е. любой символ кроме буквы и цифры.

Процесс вывода строки $lddld\perp$ (например, индентификатор $x21a8$) соответствует следующей схеме вывода:

$S \Rightarrow lA \Rightarrow ldA \Rightarrow lddA \Rightarrow lddlA \Rightarrow lddldA \Rightarrow lddld\perp$.

Существует полное соответствие между регулярными грамматиками и КА. Построение конечного автомата-распознавателя $M = (K, T, \delta, k_0, F)$ по заданной регулярной грамматике $G = (V_T, V_N, P, S)$ такое, что $L(G) = T(M)$, заключается в следующем:

1. $T = V_T$, т. е. входной алфавит КА совпадает с множеством терминалов регулярной грамматики.

2. $K = V_N \cup \{\#\}$, где # – специально выделенное конечное состояние, т. е. множество состояний КА есть множество нетерминалов, дополненное выделенным конечным состоянием.

3. $k_0 = S$, т. е. начальным состоянием КА является начальный нетерминал грамматики.

4. Если $S \to \varepsilon \in P$, то $F = \{S, \#\}$, в противном случае $F = \{\#\}$.

5. Продукции вида $A \to aB$ соответствует функция переходов $\delta(A, a) = B$; продукции вида $A \to a$ – функция переходов $\delta(A, a) = \#$.

В общем случае построенный по регулярной грамматике автомат будет недетерминированным. Поэтому на следующем этапе полученный НКА необходимо преобразовать в эквивалентный ДКА (см. разд. 2.5).

Пусть дана регулярная грамматика G с продукциями

$S \to d \mid dA$,

$A \to d \mid dA \mid pB$,

$B \to d \mid dB$.

Эта грамматика описывает синтаксис чисел целого типа (например, *ddd*) и вещественного типов с фиксированной точкой (например, *ddpddd*), где терминал *d* обозначает цифру, терминал *p* – точку.

Определим функцию переходов δ автомата:

$\delta(S, d) = \{A, \#\}$; $\delta(B, d) = \{B, \#\}$;

$\delta(S, p) = \varnothing$; $\delta(B, p) = \varnothing$;

$\delta(A, d) = \{A, \#\}$; $\delta(\#, d) = \varnothing$;

$\delta(A, p) = \{B\}$; $\delta(\#, p) = \varnothing$.

Граф переходов полученного НКА показан на рис. 2.8, эквивалентный ему ДКА – на рис. 2.9.

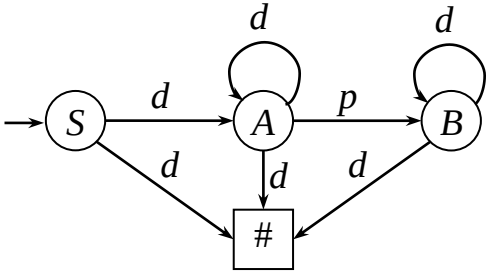


Рис. 2.8. Граф переходов НКА, построенный по регулярной грамматике

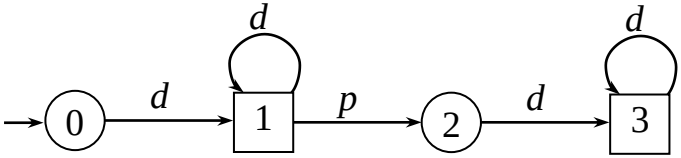


Рис. 2.9. Граф переходов ДКА, эквивалентного НКА

При построении автомата не учтен следующий момент. После распознавания лексического класса принятой лексемы сканер передает синтаксическому анализатору сформированный токен, т. е. автомат-распознаватель должен остановиться, сформировать токен и перейти к распознаванию очередной лексемы с начального состояния. Это означает, что из конечного состояния автомата не должно быть переходов в какое-либо другое состояние, а переход в конечное состояние должен быть выполнен после завершения распознавания лексемы.

Для распознавания лексемы сканеру может потребоваться чтение дополнительных символов входного потока, следующих за текущим символом (опережающее чтение). Пусть автомат на рис. 2.9 находится в состоянии 1. Если очередным входным символом является символ d или p, то продолжается распознавание лексемы. Если же очередным входным символом будет любой другой символ $\perp_1 \notin \{d, p\}$, то это означает, что завершено распознавание лексемы как числа целого типа. Аналогично, в состоянии 3 чтение любого символа $\perp_2 \neq d$ говорит о том, что лексема является числом вещественного типа. Символы $\perp_1$ и $\perp_2$ можно трактовать как маркеры конца ввода лексемы.

Таким образом, для распознавания лексемы как числа требуется чтение дополнительного символа входного потока, этот символ следует вернуть во входной поток (символ может являться началом другой лексемы). Действия, связанные с определением токена и возврата лишних прочитанных символов во входной поток, легко можно реализовать в конечных состояниях соответствующих автоматов-распознавателей.

Это требует следующей модификации автомата. Если конечное состояние имеет переход в другое состояние, его надо сделать обычным внутренним состоянием и добавить из него переход в новое конечное состояние по маркеру конца ввода. Рекомендуется конечные состояния нумеровать отрицательными целыми числами, чтобы легко отличать конечные состояния от других состояний автомата. Модификация автомата на рис. 2.9 представлена на рис. 2.10.

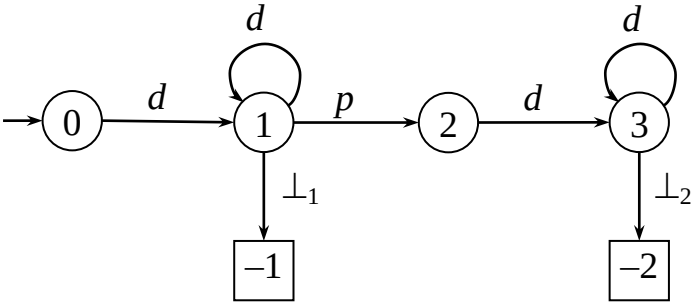


Рис. 2.10. Модифицированный ДКА

Другой путь построения автомата – включить необходимые маркеры конца ввода лексемы в исходный синтаксис лексического класса, построить соответствующую регулярную грамматику и синтезировать автомат. Например, для описания синтаксиса числа можно построить регулярную грамматику

$S \to dA$,

$A \to dA \mid \perp_1 \mid pB$,

$B \to dC$,

$C \to dC \mid \perp_2$.

Результатом построения автомата-распознавателя по этой грамматике (с последующей его детерминизацией) будет тот же автомат, что и на рис. 2.10.

Можно также, наоборот, по детерминированному автомату $M = (K, T, \delta, k_0, F)$ построить регулярную грамматику $G = (V_T, V_N, P, S)$:

1. Множество терминалов V_T грамматики совпадает с входным алфавитом T автомата, т. е. $V_T = T$.

2. Множеству нетерминалов V_N соответствует множество K состояний автомата, т. е. $V_N = K$. Если состояния автомата помечены их номерами, то лучше предварительно переобозначить их прописными латинскими буквами, чтобы получить множество нетерминалов в соответствии с принятым соглашением об их обозначениях.

3. Начальное состояние k_0 автомата становится начальным нетерминалом S грамматики.

4. В множество P включаются все продукции вида $k_i \to ak_j$, если имеется функция переходов $\delta(k_i, a) = k_j$, а также все продукции вида $k_i \to a$, где $\delta(k_i, a) = k_j$ и $k_j \in F$.

В общем случае в полученной грамматике могут появиться непроизводящие нетерминалы (т. е. нетерминалы, не порождающие ни одной терминальной строки). Такие нетерминалы соответствуют состояниям автомата, из которых нет переходов в другие состояния. Продукции с такими нетерминалами можно безболезненно удалить из грамматики.

Построим регулярную грамматику по автомату на рис. 2.10. Обозначим состояния латинскими буквами:

Номер состояния	0	1	2	3	–1	–2
Обозначение	S	A	B	C	D	E

По функции переходов получим продукции:

$\delta(S, d) = A \quad \Rightarrow S \to dA$

$\delta(A, d) = A \quad \Rightarrow A \to dA$

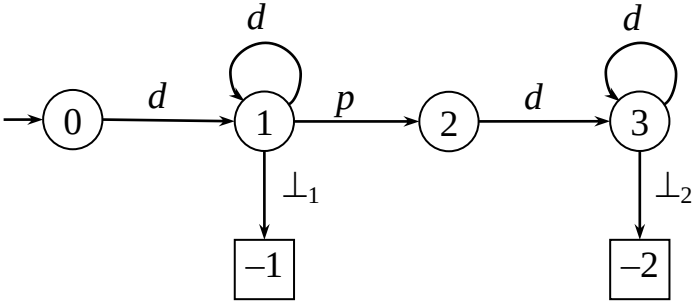
$\delta(A, p) = B \quad \Rightarrow A \to pB$

$\delta(A, \perp_1) = D \quad \Rightarrow A \to \perp_1 D \mid \perp_1$

$\delta(B, d) = C \quad \Rightarrow B \to dC$

$\delta(C, d) = C \quad \Rightarrow C \to dC$

$\delta(C, \perp_2) = E \quad \Rightarrow C \to \perp_2 E \mid \perp_2$



Продукции $A \to \perp_1$ и $C \to \perp_2$ добавлены, поскольку состояния D и E являются конечными. В полученной грамматике нетерминалы D и E непроизводящие, поэтому эти нетерминалы и содержащие их продукции $A \to \perp_1 D$ и $C \to \perp_2 E$ можно удалить из грамматики. В результате получим регулярную грамматику

$S \to dA,$

$A \to dA \mid \perp_1 \mid pB,$

$B \to dC,$

$C \to dC \mid \perp_2.$