Теория формальных языков и компиляторов
Часть 2. Конечно-автоматные распознаватели

Лекция 9. Автоматные распознаватели

В этом разделе будут рассмотрены конечно-автоматные распознаватели строк с языков, определенных не порождающими грамматиками, а конечными автоматами. Языки, определяемые этими автоматами, уже рассмотрены в классификации Хомского и называются автоматными. Такой подход к языкам с точки зрения конечно-автоматных распознавателей не противоречит порождающим грамматикам, а существенно дополняет теорию формальных языков. Далее будут даны определения детерминированных и недетерминированных автоматов.

9.1 Автоматные распознаватели и грамматики Хомского

С точки зрения теории формальных грамматик, для языков L(G[Z]), порождаемых автоматными грамматиками, можно построить **распознаватель (парсер)**. Такой распознаватель определяется самой порождаемой грамматикой G[Z], а точнее, правилами вывода автоматной грамматики вида $N \to tM$, причем $N, M \in V_N$, $t \in V_T$. Совокупность правил вывода можно представить в виде диаграммы состояний, или графа Г, как было рассмотрено в предыдущей лекции.

Если граф Г задан другим способом, то он уже не является порождающим от продукций P грамматики G[Z]. Такой граф называют **автоматом-распознавателем** A (рис. 9.1), а язык L(A) определяется так называемой конфигурацией автомата. Если входная цепочка α, поданная на вход автоматного распознавателя, переводит A последовательно из некоторого начального состояния s_0 в одно из заключительных F, то $\alpha \in L(A)$ – цепочка принадлежит заданному языку. Считается, что если $\alpha \in L(A)$, то

реакция автомата на эту цепочку – «да», в противном случае автомат отвечает – «нет».

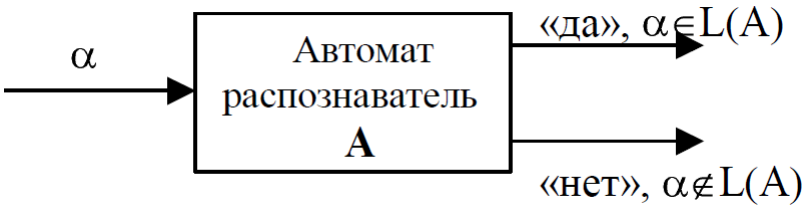


Рис. 9.1. Реакция автомата A на входную цепочку α

Определение 9.1. Распознаватель (recognizer) – это алгоритм, представленный в виде **конечного автомата**, который обрабатывая входные цепочки, принимает их («ДА») или отвергает («НЕТ»).

При классификации Хомского порождаемые соответствующей грамматикой языки являются того же типа, что и грамматика G[Z], которая их порождает. Связь порождающих грамматик из классификации Хомского и автоматов-распознавателей устанавливается через соответствующий язык.

С точки зрения конечных автоматов справедливы следующие утверждения:

- язык является автоматным, если он определяется (задается или распознается) **конечным автоматом** (возможно и недетерминированным);
- язык является контекстно-свободным, если он определяется (задается или распознается) **автоматом с магазинной (стековой) памятью**.

В рамках данных лекций ограничимся изучением конечных автоматов.

9.2 Конечные автоматы

Идеология конечного автомата (КА) как некоторого **алгоритмического устройства** состоит в следующем: имеется входная цепочка x#, которая посимвольно читается слева направо считывающим механизмом A. При каждом входном символе автомат **переходит** в одно из

своих состояний $s \in S$. Процесс чтения символов из цепочки x# не бесконечен, так как длина всегда ограничена конечным символом # входной цепочки. Переход из одного, текущего состояния в другое осуществляется под управлением **функции переходов** δ до тех пор, пока не будет обработан заключительный символ #. Когда функция переходов δ допускает несколько переходов из текущего состояния при входном символе c из α, тогда этот переход – любой из допустимых. В начале работы автомат всегда находится в начальном состоянии s_0.

Конфигурация конечного автомата определяется в виде (s, ω, n), где s – текущее состояние A, $s \in S$; ω – цепочка входных символов, $\omega \in \Sigma^+$ (эта цепочка принадлежит усеченной итерации некоторого алфавита Σ); n – положение указателя в цепочке ω, $n \in \{0, 1, 2, \ldots\}$, $n \leq |\omega|$. Дадим строгое математическое определение A.

Определение 9.2. Конечным автоматом-распознавателем A называется пятёрка объектов:

$$A = (S, \Sigma, s_0, \delta, F),$$

где S – конечное непустое множество состояний; Σ – входной алфавит автомата (конечное непустое множество входных символов); $s_0 \in S$ – начальное состояние A; δ: $S \times \Sigma \rightarrow S$ – функция переходов; $F \subseteq S$ – множество заключительных (конечных) состояний.

Легко провести аналогию между рассмотренным выше графом переходов автоматной грамматики Г(G[Z]) и конечным автоматом A. Действительно, V_N и S – два **семантически эквивалентных** множества, в первом случае это множество нетерминальных символов, которые в графе Г(G[Z]) становятся множеством состояний. Причем определено и начальное состояние на графе Г, это Z. V_T и Σ – два словаря терминальных символов. P и δ – объекты логического уровня, определенные соответственно графом и автоматом. Финальные состояния K и F для графа и автомата также эквивалентны.

Принципиальной разницей между Г(G[Z]) и конечным автоматом А является идентификация графа и автомата. Как уже отмечалось, граф Г(G[Z]) определен, если задана порождающая грамматика, по которой строится граф (N→tM). Конечный автомат А имеет только терминальный словарь и определен однозначно своей пятеркой (S, Σ, s_0, δ, F).

Определим **итерацию функции переходов** δ как отображение на множество состояний декартова произведения множеств состояний автомата S на итерацию словаря, т.е. δ*: S×Σ*→S. Это означает, что для каждой строки из Σ* (из итерации входного алфавита) можно определить состояние, в котором будет распознаватель после считывания этой строки.

Определение 9.3. Конечный автомат A = (S, Σ, s_0, δ, F) допускает входную цепочку α∈Σ*, если символы из α переводят А в одно из заключительных состояний F так, что δ*(s_0, α)∈F.

Теперь мы подошли к еще одному определению языка. Только в данном случае **язык L будет являться производным от конечного автомата-распознавателя А**.

Определение 9.4. Языком L(A) над словарём Σ называется множество всех цепочек α, которые допускает (принимает) конечный автомат А, т.е.

$$L(A) = \{\alpha \mid \alpha \in \Sigma^*, \delta^*(s_0, \alpha) \in F\}.$$

Это определение языка отличается от данного выше языка L(G[Z]), порождаемого грамматикой. Хотя в обоих случаях язык определяется множеством цепочек, допускаемых в одном случае порождающей грамматикой в другом – конечным автоматом. С точки зрения нового определения языка L(A) как множества допускающих конечным автоматом цепочек соответственно изменяется определение автоматного языка в отсутствие его порождающих автоматных грамматик.

Определение 9.5. Автоматным языком называют язык L(A). Иначе говоря, **язык относится к классу автоматных, если существует конечный автомат А, принимающий этот язык**.

Приведенное определение не противоречит уже данному определению L(G[Z]) для порождающих грамматик. Действительно, ранее было показано, что граф автоматной грамматики Г(G[Z]) легко превращается в КА, если положить: $s_0 = Z$; $\Sigma = V_T$, $F = \{K\}$, $S = V_N$. Тогда легко доказать, что множество продукций P эквивалентно множеству функций переходов δ, так как правила вида $N \to tM$, где $N, M \in V_N$, $t \in V_T$, можно представить как отображение декартова произведения множества состояний и нетерминального словаря на множество состояний, т.е. $P: V_N \times V_T \to V_N$.

Конечные автоматы представляются в виде графов, или диаграмм состояний, только вместо P используется {δ}, вместо V_T – Σ и т.д. Состояние автомата принято нумеровать или обозначать буквой, а не обозначать нетерминальными символами, в автомате нет этого понятия (но в дальнейшем мы не всегда будем следовать этому правилу). Начальное состояние нумеруется как 0, а заключительные состояния нумеруют n (n – целое) и для отличия либо рисуют более жирно, либо обводят двойным кружком, как в синтаксических диаграммах.

Рассмотрим взаимосвязь между автоматными грамматиками и конечными автоматами. Пусть имеется грамматика G[I] с правилами:

P: 1) I→aB 2) B→bC | aD 3) C→cD 4) D→b | cI

Граф Г(G[I]) этой грамматики представлен на рис. 9.2.

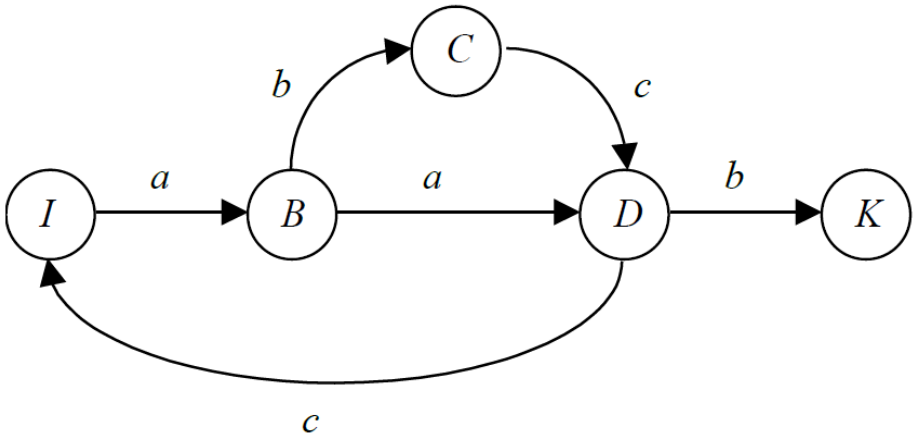


Рис. 9.2. Граф Г(G[I])

Язык L(G[I]) порождает множество цепочек {aab, aacaab,..., aac (aab)*, abcb, abccab,...}.

Ранее было показано, что если в Г(G[I]) нетерминалы принять множеством состояний S = {I, B, C, D, K}, то такой граф становится эквивалентным автоматным преобразователем (конечным автоматом), т.е. функция переходов δ(s, c), ∀s∈S, c∈Σ определена продукциями множества правил вывода P, причем $\Sigma = V_T$, $S = V_N$.

Запишем полностью **математическое задание** конечного автомата:

$$A = \{S, \Sigma, \delta, s_0, F\},$$

где S = {I, B, C, D, K}, Σ = {a, b, c}, F = {K}. Функция переходов δ записывается следующим образом:

δ(I, a) = {B};

δ(B, a) = {D};

δ(B, b) = {C};

δ(C, c) = {D};

δ(D, b) = {K};

δ(D, c) = {I}.

В этом случае легко убедиться в эквивалентности графа автоматной грамматики и конечного автомата. Отметим, что эквивалентный конечный автомат в общем случае является недетерминированным ввиду наличия в автоматной грамматике правила вывода B→ε.

Таким образом, установлена связь между графами Г(G[I]), синтаксическими диаграммами и конечными автоматами. При этом еще раз отметим, что G[I] должна быть обязательно автоматной, тогда прямой и обратный переходы от Г(G[I]) или $A = \{S, \Sigma, \delta, s_0, F\}$ к синтаксическим диаграммам определены однозначно и механизмы задания (определения или порождения) языка L эквивалентны.

Определение 9.6. КА называется **полностью определённым**, если в каждом его состоянии существует функция перехода над словарем Σ, т.е.

$$\forall c \in \Sigma,\ \forall s \in S\ \exists \delta(s, c) = R \mid R \subseteq S.$$

Рассмотрим КА с позиций данных определений. Пусть

$$A_1 = (\{I, M, B, F\}, \{a, b\}, \delta, I, \{F\}).$$

В этой записи все элементы, определяющие конфигурацию конечного автомата, записаны в одну строку (кроме значений функции переходов).

Зададим функцию переходов δ:

$$\delta(I, b) = B;\ \delta(B, a) = M;\ \delta(M, b) = \{B, F\}.$$

Граф состояний для КА представлен на рис. 9.3.

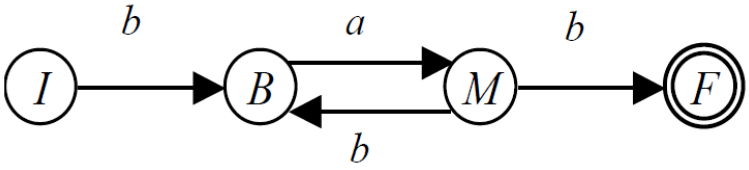


Рис. 9.3. Граф Г(G[I])

Как и раньше, в порождающих грамматиках автоматного типа данный автомат-распознаватель A_1 принимает язык $L(A_1)$, все цепочки которого образуются «путешествием» из начального состояния I в конечные F. Таким образом, $L(A_1)$ = {bab, babab, bababab…}. Или легко установить закономерность. Действительно, префиксом (или «головой») цепочек α является символ b, $b \in \Sigma$. Суффиксом строк является подстрока ab. Между префиксом и суффиксом цепочек любое число раз (от нуля до бесконечности) может повторяться подстрока ab. Поэтому можно записать:

$$L(A_1) = \{b(ab)^*ab\}.$$

Приведем еще одно определение, трактующее порождение языка L конечным автоматом A.

Определение 9.7. Слово $\omega = a_1 \ldots a_k$ над алфавитом допускается конечным автоматом $A = (S, \Sigma, \delta, s_0, F)$, если существует последовательность состояний $s_1, \ldots, s_n$ такая, что

$$\forall i, j : 1 \le i \le n,\ 1 \le j \le k,\ \exists \delta(s_i, a_j) = s_{i+1},\ s_1 = s_0,\ s_n \in F.$$

Это определение следует понимать, как существование функции переходов δ, которая под управлением входных символов a_j (j = 1, …, n) последовательно переводит автомат из начального состояния s_0 в конечное – $s_n \in F$.

Используя это определение и предполагая, что формальный язык представляется множеством слов (цепочек) ω, т.е. L(A) = {ω}, ω∈Σ*, можно дать следующее определение языка.

Определение 9.8. Язык L(A) = {ω}, ω∈Σ* распознается конечным автоматом тогда и только тогда, когда каждое слово языка над алфавитом допускается этим конечным автоматом.

Если A_1 принимает $L(A_1)$ = {b(ab)*ab}, то все другие цепочки β≠α, β∈Σ*, Σ = {a, b} должны переводить конечный автомат в состояние ошибки (ERROR или E). Отметим, что вновь введенное состояние E не является заключительным, так как в противном случае запрещенные цепочки β будут также принадлежать языку $L(A_1)$. Таким образом, на состоянии E замыкаются все неопределенные переходы из множества состояний S = {I, M, B, F}.

Поэтому в полностью определенном конечном автомате нужно записать все значения функции переходов:

$$\delta(I, a) = E;\ \delta(I, b) = B;$$

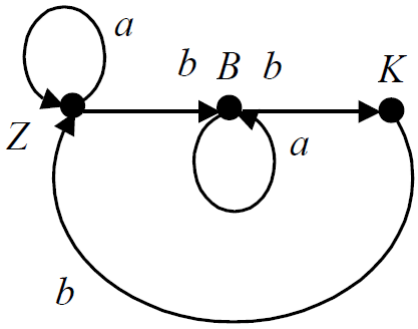
$$\delta(B, a) = M;\ \delta(B, b) = E;$$

$$\delta(M, a) = E;\ \delta(M, b) = \{B, F\}.$$

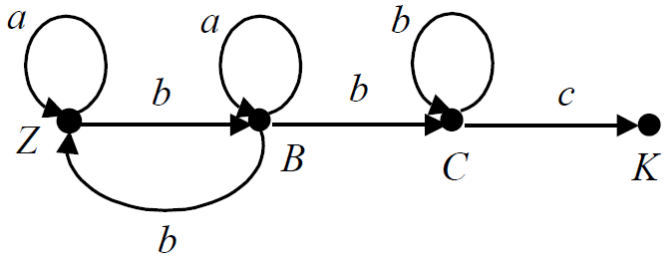
Упражнения

В №№ 1 – 4 записать математическое задание конечного автомата A = (S, Σ, s_0, δ, F), соответствующее графу Г(G[Z]). После этого запишите соответствующий полностью определенный конечный автомат.

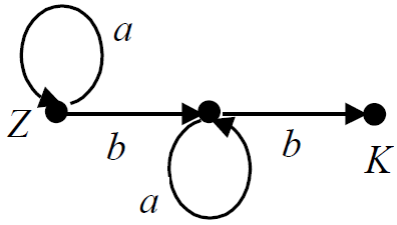
1.



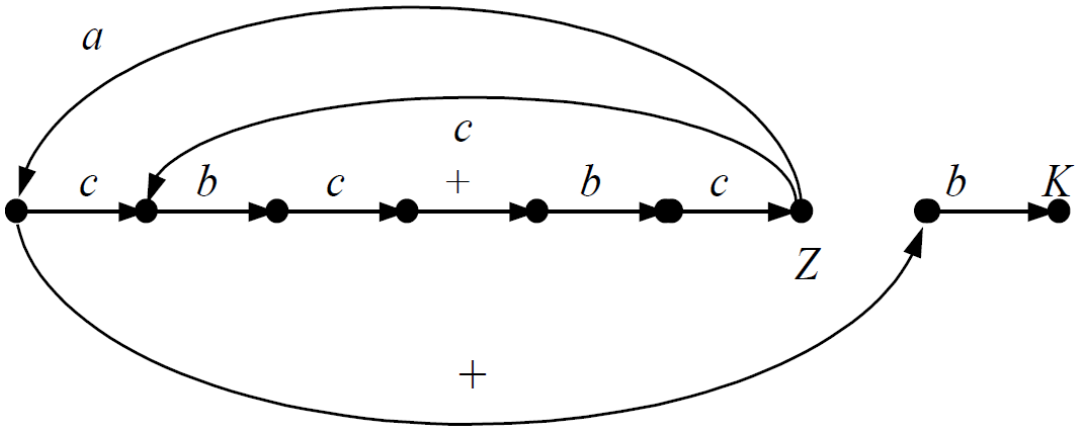
2.



3.



4.



Список использованных источников

1. Шорников Ю.В. Теория и практика языковых процессоров.