МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение
высшего образования
«Курганский государственный университет»

Кафедра «Безопасность информационных и автоматизированных систем»

Теория автоматов и формальных языков

Методические указания
к выполнению практических работ для студентов
направлений подготовки 09.03.04, 09.04.04
«Программная инженерия»

Курган 2019

Кафедра: «Безопасность информационных и автоматизированных систем»
Дисциплина: «Теория автоматов и формальных языков» для студентов направлений подготовки 09.03.04, 09.04.04
Составил: канд. техн. наук, доцент В. А. Стукало

Утверждены на заседании кафедры протокол № 4 от « 24 » октября 2018 г.

Рекомендованы методическим советом университета « 20 » декабря 2017 г.

Оглавление

Практическая работа № 1

Распознавание типов формальных языков и грамматик

Цели работы:

- закрепить понятия «алфавит», «цепочка», «формальная грамматика», «формальный язык», «выводимость цепочек», «эквивалентная грамматика»;
- сформировать умения и навыки распознавания типов формальных языков и грамматик по классификации Хомского, построения эквивалентных грамматик.

Основы теории

Определение 1.1. Алфавитом V называется конечное множество символов.

Определение 1.2. Цепочкой α в алфавите V называется любая конечная последовательность символов этого алфавита.

Определение 1.3. Цепочка, которая не содержит ни одного символа, называется пустой цепочкой и обозначается ε.

Определение 1.4. Формальное определение цепочки символов в алфавите V:

1) ε − цепочка в алфавите V;
2) если α – цепочка в алфавите V и a − символ этого алфавита, то αa – цепочка в алфавите V;
3) β − цепочка в алфавите V тогда и только тогда, когда она является таковой в силу утверждений 1 и 2.

Определение 1.5. Длиной цепочки α называется число составляющих ее символов (обозначается $|\alpha|$).

Обозначим через V^* множество, содержащее все цепочки в алфавите V, включая пустую цепочку ε, а через V^+ − множество, содержащее все цепочки в алфавите V, исключая пустую цепочку ε.

Пример 1.1. Пусть $V = \{1, 0\}$, тогда $V^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$, а $V^+ = \{0, 1, 00, 01, 10, 11, 000, \ldots\}$.

Определение 1.6. Формальной грамматикой называется четверка вида:

$$G = (V_T, V_N, P, S),$$

где V_N − конечное множество нетерминальных символов грамматики (обычно прописные латинские буквы);

V_T − множество терминальных символов грамматики (обычно строчные латинские буквы, цифры, и т.п.), $V_T \cap V_N = \varnothing$;

P – множество правил вывода грамматики, являющееся конечным подмножеством множества $(V_T \cup V_N)^+ \times (V_T \cup V_N)^*$; элемент (α, β) множества P называется правилом вывода и записывается в виде $\alpha \to \beta$ (читается: «из цепочки α выводится цепочка β»);

S – начальный символ грамматики, $S \in V_N$.

Для записи правил вывода с одинаковыми левыми частями вида $\alpha \to \beta_1$, $\alpha \to \beta_2$, …, $\alpha \to \beta_n$ используется сокращенная форма записи $\alpha \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$.

Пример 1.2. Грамматика $G_1=(\{0, 1\}, \{A, S\}, P_1, S)$, где множество P_1 состоит из правил вида: 1) $S \to 0A1$; 2) $0A \to 00A1$; 3) $A \to \varepsilon$.

Определение 1.7. Цепочка $\beta \in (V_T \cup V_N)^*$ непосредственно выводима из цепочки $\alpha \in (V_T \cup V_N)^+$ в грамматике $G = (V_T, V_N, P, S)$ (обозначается: $\alpha \Rightarrow \beta$), если $\alpha = \xi_1 \gamma \xi_2$ и $\beta = \xi_1 \delta \xi_2$, где $\xi_1, \xi_2, \delta \in (V_T \cup V_N)^*$, $\gamma \in (V_T \cup V_N)^+$ и правило вывода $\gamma \to \delta$ содержится во множестве P.

Определение 1.8. Цепочка $\beta \in (V_T \cup V_N)^*$ выводима из цепочки $\alpha \in (V_T \cup V_N)^+$ в грамматике $G = (V_T, V_N, P, S)$ (обозначается $\alpha \Rightarrow^* \beta$), если существует последовательность цепочек $\gamma_0, \gamma_1, \ldots, \gamma_n$ ($n \geq 0$) такая, что $\alpha = \gamma_0 \Rightarrow \gamma_1 \Rightarrow \ldots \Rightarrow \gamma_n = \beta$.

Пример 1.3. В грамматике G_1 $S \Rightarrow^* 000111$, т.к. существует вывод

$$S \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000111.$$

Определение 1.9. Языком, порожденным грамматикой $G = (V_T, V_N, P, S)$, называется множество всех цепочек в алфавите V_T, которые выводимы из начального символа грамматики S с помощью правил множества P, т.е. множество

$$L(G) = \{\alpha \in V_T^* \mid S \Rightarrow^* \alpha\}.$$

Пример 1.4. Для грамматики G_1 $L(G_1)=\{0^n1^n \mid n>0\}$.

Определение 1.10. Цепочка $\alpha \in (V_T \cup V_N)^*$, для которой существует вывод $S \Rightarrow^* \alpha$, называется сентенциальной формой в грамматике $G = (V_T, V_N, P, S)$.

Определение 1.11. Грамматики G_1 и G_2 называются эквивалентными, если $L(G_1) = L(G_2)$.

Пример 1.5. Для грамматики G_1 эквивалентной будет грамматика $G_2 = (\{0, 1\}, \{S\}, P_2, S)$, где множество правил вывода P_2 содержит правила вида $S \to 0S1 \mid 01$.

Классификация грамматик по Хомскому

Тип 0. Грамматика $G = (V_T, V_N, P, S)$ называется грамматикой типа 0, если на ее правила вывода не наложено никаких ограничений, кроме тех, которые указаны в определении грамматики.

Тип 1. Грамматика $G = (V_T, V_N, P, S)$ называется контекстно-зависимой грамматикой (КЗ-грамматикой), если каждое правило вывода из множества P имеет вид $\alpha \to \beta$, где $\alpha \in (V_T \cup V_N)^+$, $\beta \in (V_T \cup V_N)^*$ и $|\alpha| \leq |\beta|$.

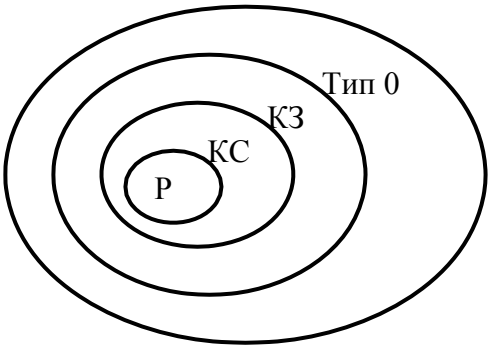
Тип 2. Грамматика $G = (V_T, V_N, P, S)$ называется контекстно-свободной (КС-грамматикой), если ее правила вывода имеют
вид $A \to \beta$, где $A \in V_N$ и $\beta \in V^*$.

Тип 3. Грамматика $G = (V_T, V_N, P, S)$ называется регулярной (Р-грамматикой), выровненной вправо, если ее правила вывода имеют
вид $A \to aB \mid a$, где $a \in V_T$; $A, B \in V_N$.

Грамматика $G = (V_T, V_N, P, S)$ называется регулярной (Р-грамматикой) выровненной влево, если ее правила вывода имеют
вид $A \to Ba \mid a$, где $a \in V_T$; $A, B \in V_N$.

Определение 1.12. Язык $L(G)$ называется языком типа k, если его можно описать грамматикой типа k, где k – максимально возможный номер типа грамматики.

Соотношение типов грамматик и языков представлено на рисунке 1.



Р – регулярная грамматика; КС – контекстно-свободная грамматика;
КЗ – контекстно-зависимая грамматика; Тип 0 – грамматика типа 0

Рисунок 1 – Соотношение типов формальных языков и грамматик

Пример 1.6. Примеры различных типов формальных языков и грамматик по классификации Хомского. Терминалы будем обозначать строчными символами, нетерминалы – прописными буквами, начальный символ грамматики – S.

а) язык типа 0 $L(G)=\{ a^2b^{n^2-1} \mid n \geq 1\}$ определяется грамматикой с правилами вывода:

1) $S \to aaCFD$; 2) $AD \to D$;
3) $F \to AFB \mid AB$; 4) $Cb \to bC$;
5) $AB \to bBA$; 6) $CB \to C$;
7) $Ab \to bA$; 8) $bCD \to \varepsilon$.

б) контекстно-зависимый язык $L(G)=\{a^nb^nc^n \mid n \geq 1\}$ определяется грамматикой с правилами вывода:

1) $S \to aSBC \mid abc$; 2) $bC \to bc$;
3) $CB \to BC$; 4) $cC \to cc$;

5) $BB \to bb$.

в) контекстно-свободный язык $L(G)=\{(ab)^n(cb)^n \mid n>0\}$ определяется грамматикой с правилами вывода:

1) $S \to aQb \mid accb$;
2) $Q \to cSc$.

г) регулярный язык $L(G)=\{\omega\perp \mid \omega\in\{a, b\}^+$, где нет двух рядом стоящих $a\}$ определяется грамматикой с правилами вывода:

1) $S \to A\perp \mid B\perp$;
2) $A \to a \mid Ba$;
3) $B \to b \mid Bb \mid Ab$.

Постановка задачи к практической работе № 1

При выполнении практической работы следует реализовать следующие действия:

1) составить несколько грамматик (по возможности разных типов), порождающих формальный язык, заданный в соответствии с вариантом;
2) определить тип формальных грамматик и языка по классификации Хомского;
3) привести примеры вывода нескольких цепочек в соответствии с правилами грамматик.

Варианты индивидуальных заданий представлены в таблице 1.

Таблица 1 − Варианты индивидуальных заданий к практической работе №1.

Вариант	Формальный язык
1	$L(G)=\{a^n b^m c^k \mid n, m, k>0\}$
2	$L(G)=\{(ab)^n(cb)^m \mid n, m\geq 0\}$
3	$L(G)=\{0^n(10)^m \mid n, m\geq 0\}$
4	$L(G)=\{wcwcw \mid w\in\{a, b\}^+\}$
5	$L(G)=\{c^{2n}d^n \mid n>0\}$
6	$L(G)=\{l+l-l \mid l\in\{a, b\}^+\}$
7	$L(G)=\{(10)^{n-1}(01)^{n+1} \mid n>0\}$
8	$L(G)=\{(ac)^n \mid n>0, a\in\{b, d\}, c\in\{+, -\}\}$
9	$L(G)=\{\perp(010)^n\perp \mid n>0\}$
10	$L(G)=\{a_1a_2\ldots a_n a_n\ldots a_2a_1 \mid a_i\in\{0, 1\}\}$
11	$L(G)=\{a_1a_2\ldots a_n a_1a_2\ldots a_n \mid a_i\in\{c, d\}\}$
12	$L(G)=\{ab.b \mid a_i\in\{+, -\}, b\in\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^+\}$

Практическая работа № 2

Построение конечного автомата по регулярной грамматике

Цели работы:

- закрепить понятия «регулярная грамматика», «недетерминированный и детерминированный конечный автомат»;
- сформировать умения и навыки построения конечного автомата по регулярной грамматике и преобразования недетерминированного конечного автомата к детерминированному конечному автомату.

Основы теории

Распознавателем для регулярной грамматики является конечный автомат (КА).

Определение 2.1. Детерминированным конечным автоматом (ДКА) называется пятерка объектов:

$$M = (Q, T, F, H, Z),$$

где Q − конечное множество состояний автомата;

T − конечное множество допустимых входных символов;

F − функция переходов, отображающая множество $Q \times T$ во множество Q;

H − конечное множество начальных состояний автомата;

Z − множество заключительных состояний автомата, $Z \subseteq Q$.

Определение 2.2. Недетерминированным конечным автоматом (НКА) называется конечный автомат, в котором в качестве функции переходов используется отображение $Q \times T$ во множество всех подмножеств множества состояний автомата $P(Q)$, т.е. функция переходов неоднозначна, так как текущей паре (q, t) соответствует множество очередных состояний автомата $q' \in P(Q)$.

Способы представления функции переходов

Командный способ. Каждую команду КА записывают в форме $F(q, t) = p$, где $q, p \in Q, t \in T$.

Табличный способ. Строки таблицы переходов соответствуют входным символам автомата $t \in T$, а столбцы – состояниям Q. Ячейки таблицы заполняются новыми состояниями, соответствующими значению функции $F(q, t)$.

Неопределенным значениям функции переходов соответствуют пустые ячейки таблицы.

Графический способ. Строится диаграмма состояний автомата – неупорядоченный ориентированный помеченный граф. Вершины графа помечены именами состояний автомата. Дуга ведет из состояния q в состоянии p и помечается списком всех символов $t \in T$, для которых $F(q, t) = p$. Вершина, соответствующая входному состоянию автомата, снабжается стрелкой. Заключительное состояние на графе обозначается двумя концентрическими окружностями.

Алгоритм 2.1. Построение КА по регулярной грамматике

Вход: регулярная грамматика $G = (V_T, V_N, P, S)$.

Выход: КА $M = (Q, T, F, H, Z)$**.**

Шаг 1. Пополнить грамматику правилом $A \to aN$, где $A \in V_N$, $a \in V_T$ и N – новый нетерминал, для каждого правила вида $A \to a$, если в грамматике нет соответствующего ему правила $A \to aB$, где $B \in V_N$.

Шаг 2. Начальный символ грамматики S принять за начальное состояние КА H. Из нетерминалов образовать множество состояний автомата $Q = V_N \cup \{N\}$, а из терминалов – множество символов входного алфавита $T = V_T$.

Шаг 3. Каждое правило $A \to aB$ преобразовать в функцию переходов $F(A, a) = B$, где $A, B \in V_N$, $a \in V_T$.

Шаг 4. Во множество заключительных состояний включить все вершины, помеченные символами $B \in V_N$ из правил вида $A \to aB$, для которых имеются соответствующие правила $A \to a$, где $A, B \in V_N$, $a \in V_T$.

Шаг 5. Если в грамматике имеется правило $S \to \varepsilon$, где S – начальный символ грамматики, то поместить S во множество заключительных состояний.

Шаг 6. Если получен НКА, то преобразовать его в ДКА.

Алгоритм 2.2. Преобразование НКА в ДКА

Вход: НКА $M = (Q, T, F, H, Z)$.

Выход: ДКА $M' = (Q', T, F', H, Z')$.

Шаг 1. Пометить первый столбец таблицы переходов M' ДКА начальным состоянием (множеством начальных состояний) НКА M.

Шаг 2. Заполняем очередной столбец таблицы переходов M', помеченный символами D. Для этого определяем те состояния M, которые могут быть достигнуты из каждого символа строки D при каждом входном символе x. Поместить каждое найденное множество R (в том числе $\varnothing$) в соответствующие позиции столбца D таблицы M', т.е.

$$F'(D, x) = \{s \mid s \in F(t, x) \text{ для некоторого } t \in D\}.$$

Шаг 3. Для каждого нового множества R (кроме $\varnothing$), полученного в столбце D таблицы переходов M', добавить новый столбец в таблицу, помеченный R.

Шаг 4. Если в таблице переходов КА M' есть столбец с незаполненными позициями, то перейти к шагу 2.

Шаг 5. Во множество Z' ДКА M' включить каждое множество, помечающее столбец таблицы переходов M' и содержащее $q \in Z$ НКА M.

Шаг 6. Составить таблицу новых обозначений множеств состояний и определить ДКА M' в этих обозначениях.

Пример 2.1. Дана регулярная грамматика $G = (\{a, b\}, \{S, A, B\}, P, S)$ с правилами P: $S \to aB \mid aA$; $B \to bB \mid a$; $A \to aA \mid b$. Построить по регулярной грамматике КА и преобразовать полученный автомат к детерминированному виду.

Решение задачи включает следующую последовательность действий.

1 Построим по регулярной грамматике КА.

1.1 Пополним грамматику правилами $A \to bN$ и $B \to aN$, где N - новый нетерминал.

1.2 Начальное состояние конечного автомата $H = S$. Множество состояний автомата $Q = V_N = \{S, A, B, N\}$, множество символов входного алфавита $T = V_T = \{a, b\}$.

1.3 Значения сформированной функции переходов даны в таблице 2.

Таблица 2 − Функция переходов автомата M

F	S	A	B	N
a	A, B	A	N	$\varnothing$
b	$\varnothing$	N	B	$\varnothing$

1.4 Множество заключительных состояний $Z = \{N\}$.

1.5 Для начального символа грамматики ε-правила отсутствуют.

Конечный автомат M − недетерминированный, граф НКА представлен на рисунке 2 (слева).

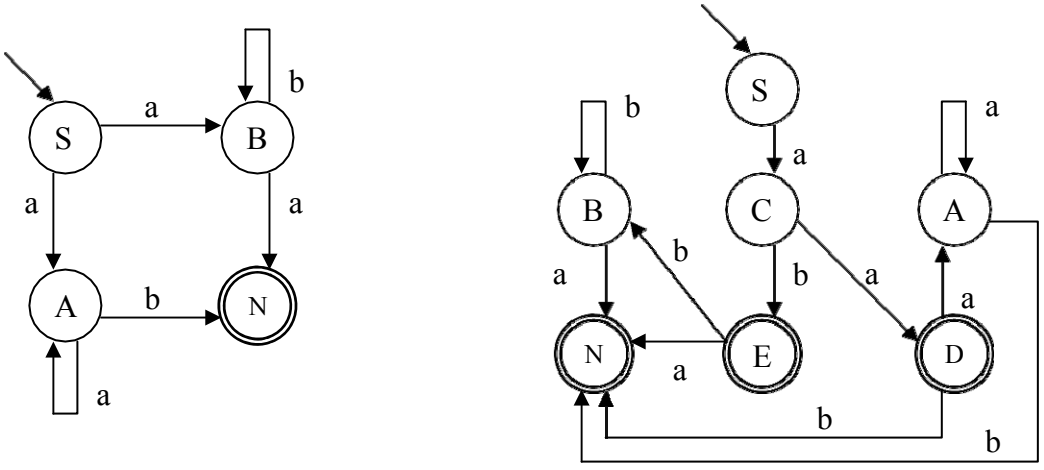


Рисунок 2 − Граф НКА (слева) и ДКА (справа) для Р-грамматики

2 Построим по НКА M ДКА M'.

2.1 Строим таблицу переходов для ДКА M' (таблица 3).

2.2 Во множество заключительных состояний автомата M' включим элементы $Z' = \{(A, N), (B, N), N\}$.

2.3 Введем следующие новые обозначения состояний автомата M': $(A, B)=C$, $(A, N)=D$, $(B, N)=E$.

Таблица 3 − Построение функции переходов для ДКА M'

Шаг	1	2	3	4	5	6	7
F	S	A, B	A, N	B, N	A	N	B
a	A, B	A, N	A	N	A	$\varnothing$	N
b	$\varnothing$	B, N	N	B	N	$\varnothing$	B

2.4 Искомый ДКА определяется следующей пятеркой объектов: $Q' = \{S, A, B, C, D, E, N\}$, $T = \{a, b\}$, функция переходов задана таблицами 3 и 4, $H = \{S\}$, $Z' = \{N, D, E\}$.

Граф полученного ДКА представлен на рисунке 2 (справа).

Таблица 4 − Функция переходов для ДКА M'

F'	S	A	B	C	D	E	N
a	C	A	N	D	A	N	$\varnothing$
b	$\varnothing$	N	B	E	N	B	$\varnothing$

Постановка задачи к практической работе № 2

При выполнении практической работы реализовать следующие действия:

1) выполнить проверку заданной грамматики на принадлежность к классу регулярных грамматик;

2) построить по заданной регулярной грамматике конечный автомат;

3) преобразовать недетерминированный конечный автомат к детерминированному конечному автомату;

4) построить графы получившихся конечных автоматов.

Варианты индивидуального задания представлены в таблице 5.

Таблица 5 − Варианты индивидуального задания к практической работе № 2

Вариант	Регулярная грамматика
1	G=($\{S, C, D\}$, $\{0, 1\}$, P, S), где P: 1) $S \to 1C \mid 0D$; 2) $C \to 0D \mid 0S \mid 1$; 3) $D \to 1C \mid 1S \mid 0$.
2	G=($\{S, A, B, C\}$, $\{a, b, c\}$, P, S), где P: 1) $S \to aA \mid bB \mid aC$; 2) $A \to bA \mid bB \mid c$; 3) $B \to aA \mid cC \mid b$; 4) $C \to bB \mid bC \mid a$.
3	G=($\{K, L, M, N\}$, $\{a, b, +, -, \perp\}$, P, K), где P: 1) $K \to aL \mid bM$; 2) $L \to -N \mid -M$; 3) $M \to +N$; 4) $N \to aL \mid bM \mid \perp$.
4	G=($\{X, Y, Z, W, V\}$, $\{0, 1, \sim, \#, \&\}$, P, X), где P: 1) $X \to 0Y \mid 1Z \mid \varepsilon$; 2) $Y \to 0Z \mid \sim W \mid \#$; 3) $Z \to 1Y \mid 1W \mid 0V$; 4) $W \to 0W \mid 1W \mid \#$; 5) $V \to \&Z$.

5	$G=(\{K, L, M, N, Q, P, R, S\}, \{0, 1, *, \$, /\}, V, K)$, где V: 1) $K\to 1L \mid 0N$; 2) $L\to 0M \mid 0P \mid /Q$; 3) $N\to 1R \mid 1M \mid *S$; 4) $Q\to 1P$; 5) $P\to *L \mid \$$; 6) $M\to \$$; 7) $S\to 0R$; 8) $R\to /N \mid \$$.
6	$G=(\{E, A, B, C, D\}, \{0, 1, a, b, c\}, P, E)$, где P: 1) $E\to 0A \mid \varepsilon$; 2) $A\to aB \mid aD$; 3) $B\to bB \mid 1C \mid c$; 4) $D\to aD \mid 0C \mid c$.
7	$G=(\{X, Y, Z, V, W\}, \{0, 1, x, y, z\}, P, X)$, где P: 1) $X\to yY \mid zZ$; 2) $Y\to 1V$; 3) $Z\to 0W \mid 0Y$; 4) $V\to xZ \mid xW \mid 1$; 5) $W\to 1Y \mid 0$.
8	$G=(\{S, A, B, C, D\}, \{a, b, c, d, \perp\}, P, S)$, где P: 1) $S\to aA \mid bB$; 2) $A\to cC \mid \perp$; 3) $C\to cC \mid cA$; 4) $B\to dD \mid \perp$; 5) $D\to dD \mid dB$.
9	$G=(\{K, L, M, N, P\}, \{0, 1, \&, \%, a, b\}, C, K)$, где C: 1) $K\to 1M \mid \varepsilon$; 2) $M\to 0L \mid \&N \mid \&P$; 3) $L\to 1L \mid 0L \mid \%P$; 4) $N\to aN \mid bN \mid \%P$; 5) $P\to 1P \mid aP \mid 0$.
10	$G=(\{I, J, K, M, N\}, \{0, 1, \sim, !\}, P, I)$, где P: 1) $I\to 0J \mid 1K \mid 0M$; 2) $J\to \sim K \mid 0M$; 3) $K\to \sim M \mid 0J \mid 0N$; 4) $M\to 1K \mid !$; 5) $N\to 0I \mid 1I \mid !$.
11	$G=(\{S, A, B, C, D, E\}, \{a, b, c, d, e, \$, \perp\}, P, S)$, где P: 1) $S\to aA \mid bB \mid cC$; 2) $A\to dD$; 3) $B\to \#D \mid \$E$; 4) $D\to dD \mid dB \mid \perp$; 5) $C\to cE$; 6) $E\to eE \mid eB \mid \perp$.
12	$G=(\{X, Y, Z, V\}, \{(,), y, z, v\}, P, X)$, где P: 1) $X\to (Y \mid \varepsilon$; 2) $Y\to yY \mid zY \mid zZ$; 3) $Z\to zZ \mid vZ \mid vV$; 4) $V\to vV \mid)$.

Практическая работа № 3

Минимизация конечных автоматов

Цели работы:

- закрепить понятия «недостижимые состояния автомата», «эквивалентные состояния автомата», «минимальный конечный автомат»;
- сформировать умения и навыки минимизации детерминированного конечного автомата.

Основы теории

Конечный автомат может содержать лишние состояния двух типов: недостижимые и эквивалентные состояния.

Определение 3.1. Два различных состояния q и q' в конечном автомате $M = (Q, T, F, H, Z)$ называются n-эквивалентными, $n \in N \cup \{0\}$, если, находясь в одном их этих состояний и получив на вход любую цепочку символов ω: $\omega \in V_T^*$, $|\omega| \le n$, автомат может перейти в одно и то же множество конечных состояний.

Определение 3.2. Состояние q КА называется недостижимым, если к нему нет пути из начального состояния автомата.

Определение 3.3. КА, не содержащий недостижимых и эквивалентных состояний, называется приведенным или минимальным КА.

Алгоритм 3.1. Устранение недостижимых состояний КА

Вход: КА $M = (Q, T, F, H, Z)$.

Выход: КА $M' = (Q', T, F', H, Z')$.

Шаг 1. Поместить начальное состояние КА в список достижимых состояний $Q_д$, т.е. $Q_д{}^0 = H$.

Шаг 2. Для новых элементов списка достижимых состояний пополнить список группой их состояний-приемников, отсутствующих в нем, т.е.

$$Q_д{}^i = Q_д{}^{i-1} \cup \{p \mid \forall q \in Q_д{}^{i-1} \; \exists F(q, t) = p\}.$$

Шаг 3. Повторить шаг 2, пока список достижимых состояний не перестанет меняться. То есть, если $Q_д{}^i \ne Q_д{}^{i-1}$, то i:=i+1, иначе $Q_д = Q_д{}^i$.

Шаг 4. Исключить из множества Q состояний КА все состояния, отсутствующие в списке $Q_д$ достижимых состояний, т.е. $Q' = Q \cap Q_д$.

Шаг 5. Исключить недостижимые заключительные состояния и пары функции переходов, содержащие недостижимые состояния, т.е. $Z' = Z \cap Q_д$,

$$F' = F - \{F(q, t) = p \mid q \in (Q - Q_д)\}.$$

Пример 3.1. Устранить недостижимые состояния КА $M = (Q, T, F, H, Z)$, где $Q = \{A, B, C, D, E, F, G\}$, $T = \{a, b\}$, $H = \{A\}$, $Z = \{D, E\}$ и функция переходов задана таблицей 6.

Граф исходного КА *M* представлен на рисунке 3.

Таблица 6 − Функция переходов конечного автомата *M*.

F	*A*	*B*	*C*	*D*	*E*	*F*	*G*
a	*B*			*C*	*B*	*D*	*F*
b	*C*	*D*	*E*	*E*	*D*	*G*	*E*

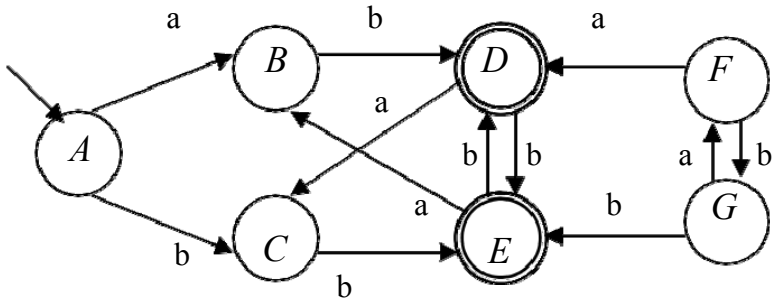


Рисунок 3 − Граф исходного конечного автомата *M*

Последовательность устранения недостижимых состояний КА имеет вид:

$Q_0 = \{A\}$;
$Q_1 = \{A, B, C\}$;
$Q_2 = \{A, B, C, D, E\}$;
$Q_3 = \{A, B, C, D, E\}$; т.к. $Q_2 = Q_3$, то $Q_д = \{A, B, C, D, E\}$.
$Q_н = \{F, G\}$; $Q' = \{A, B, C, D, E\}$; $Z' = \{D, E\}$.

Функция переходов автомата *M ′* представлена в таблице 7.

Таблица 7 − Функция переходов автомата *M ′*

F	*A*	*B*	*C*	*D*	*E*
a	*B*			*C*	*B*
b	*C*	*D*	*E*	*E*	*D*

Граф КА *M ′* после устранения недостижимых состояний представлен на рисунке 4.

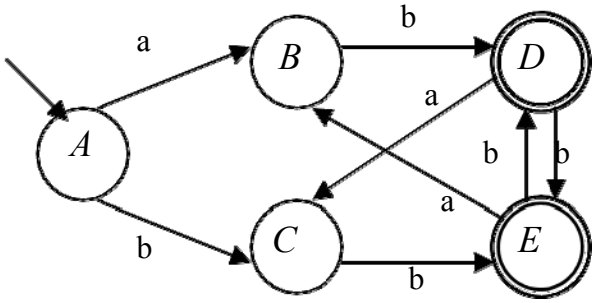


Рисунок 4 − Граф КА *M ′* после устранения недостижимых состояний

Алгоритм 3.2. Объединение эквивалентных состояний КА

Вход: КА $M' = (Q', T, F', H, Z')$ без недостижимых состояний.

Выход: минимальный КА

$$M' = (Q', T, F', H, Z').$$

Шаг 1. На первом шаге строим нулевое разбиение $R(0)$, состоящее из двух классов эквивалентности: заключительные состояния КА - Z и не заключительные − Q-Z.

Шаг 2. На очередном шаге построения разбиения $R(n)$ в классы эквивалентности включить те состояния, которые по одинаковым входным символам переходят в n-1 эквивалентные состояния, т.е.

$$R(n) = \{r_i(n) : \{q_{ij} \in Q : \forall t \in T \;\; F(q_{ij}, t) \subseteq r_j(n-1)\} \; \forall i, \; j \in N\}.$$

Шаг 3. До тех пор, пока $R(n) \neq R(n\text{-}1)$, полагаем $n := n+1$ и идем к шагу 2.

Шаг 4. Переобозначить оставшиеся неразбитые группы состояний и включить их в таблицу новых обозначений состояний автомата.

Шаг 5. Определить эквивалентный КА M' в новых обозначениях.

Пример 3.2. Минимизировать конечный автомат из примера 3.1.

Последовательность построения разбиений будет иметь вид:

$R(0) = \{\{A, B, C\}, \{D, E\}\}$, $n = 0$;
$R(1) = \{\{A\}, \{B, C\}, \{D, E\}\}$, $n = 1$;
$R(2) = \{\{A\}, \{B, C\}, \{D, E\}\}$, $n=2$.
Т.к. $R(1) = R(2)$, то искомое разбиение построено.

Переобозначим оставшиеся неразбитые группы состояний:
$X=\{B, C\}$, $Y=\{D, E\}$.

Получим минимальный автомат M', где $Q' = \{A, X, Y\}$, $Z' = \{Y\}$.

Функция переходов автомата M' представлена в таблице 8.

Таблица 8 − Функция переходов автомата M'

F'	A	X	Y
a	X		X
b	X	Y	Y

Граф переходов конечного автомата после его минимизации показан на рисунке 5.

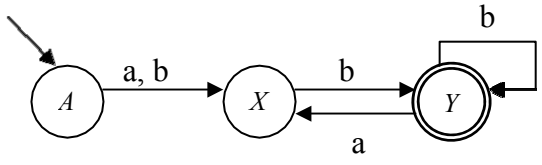


Рисунок 5 − Граф минимального КА M'

Постановка задачи к практической работе № 3

При выполнении практической работы реализовать следующие действия:

1) устранить недостижимые состояния конечного автомата;
2) исключить эквивалентные состояния конечного автомата;
3) построить граф минимального конечного автомата;
4) разработать серию контрольных примеров и провести тестирование КА.

Варианты индивидуальных заданий к практической работе № 3 представлены на рисунке 6.

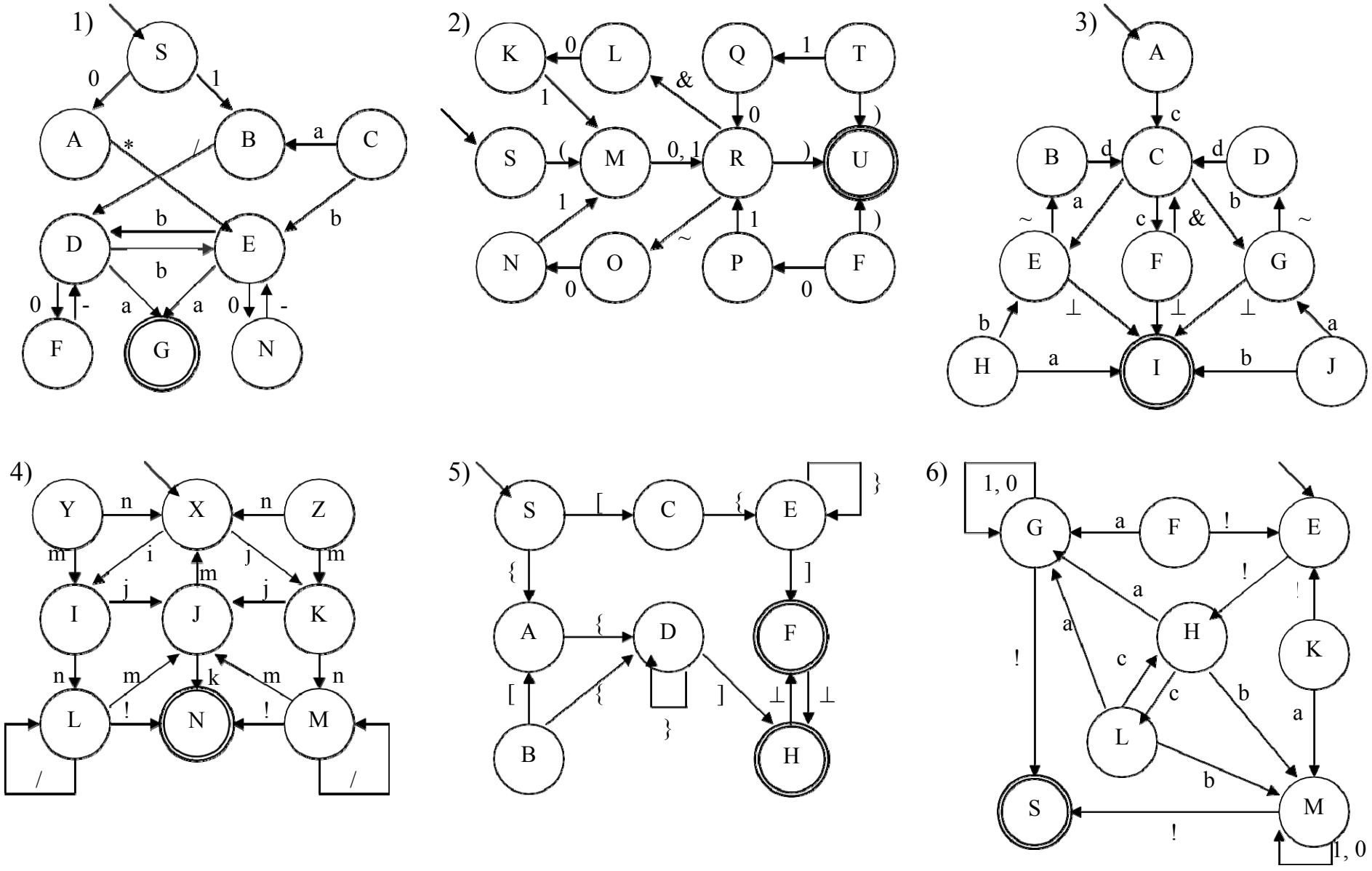


Рисунок 6 − Варианты индивидуальных заданий к практической работе № 3

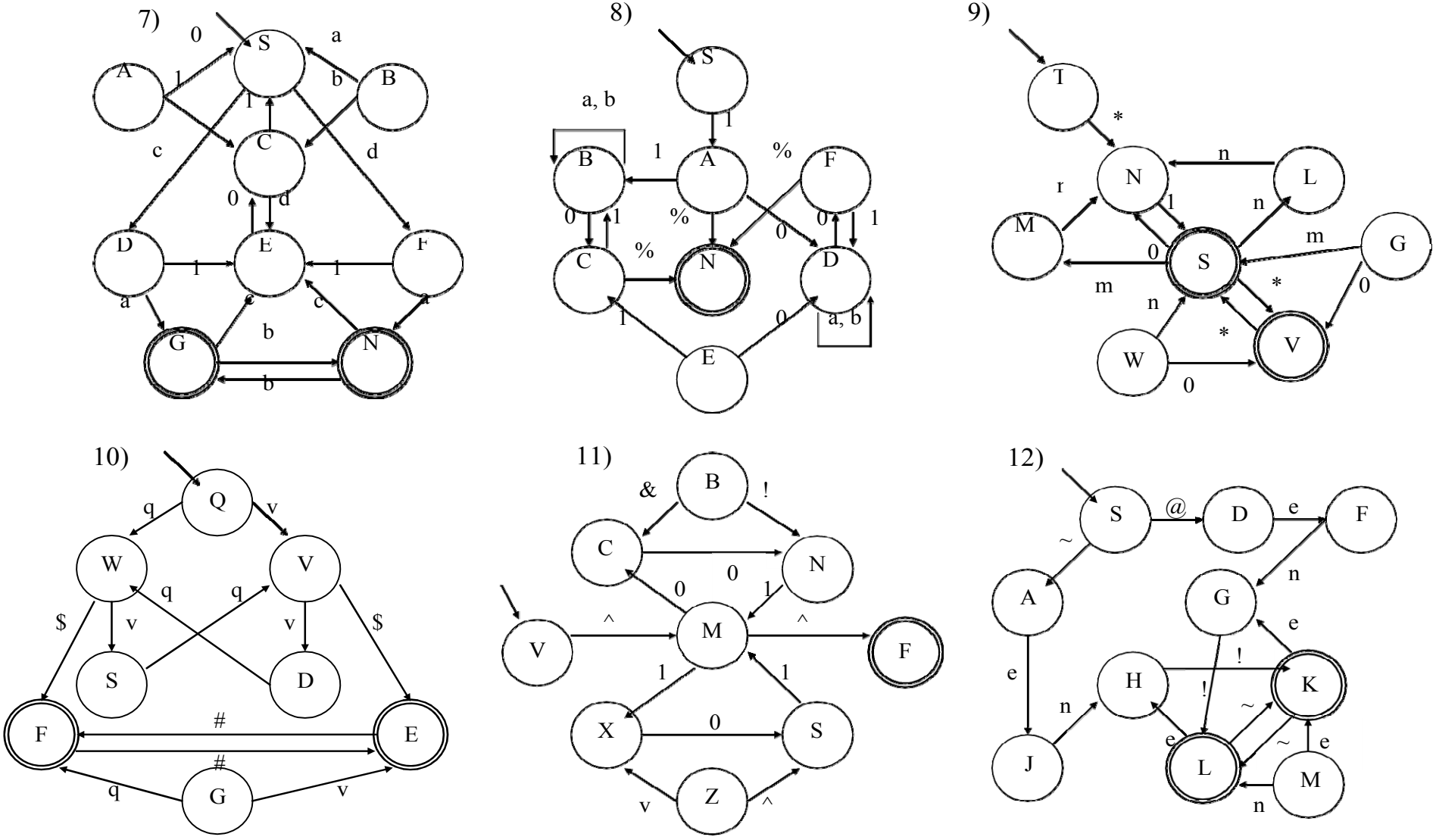


Рисунок 6 − Варианты индивидуальных заданий к практической работе № 3 (продолжение)

Практическая работа № 4

Эквивалентные преобразования контекстно-свободных грамматик

Цели работы:

- закрепить понятия «эквивалентные грамматики», «приведенная КС-грамматика»;

- сформировать умения и навыки эквивалентных преобразований контекстно-свободных грамматик.

Основы теории

Определение 4.1. КС-грамматика называется приведенной, если она не имеет циклов, ε-правил и бесполезных символов.

Рассмотрим основные алгоритмы приведения КС-грамматик.

Перед всеми другими исследованиями и преобразованиями КС-грамматик выполняется проверка существования языка грамматики.

Алгоритм 4.1. Проверка существования языка грамматики

Вход: КС-грамматика $G=(VT,VN,P,S)$.

Выход: заключение о существовании или отсутствии языка грамматики.

Определим множество нетерминалов, порождающих терминальные строки $N=\{Z \mid Z\in V_N, Z\Rightarrow^* x, x\in V_T^*\}$

Шаг 1. Положить $N_0=\varnothing$.

Шаг 2. Вычислить $N_i=N_{i-1}\cup\{A \mid (A\to\alpha)\in P\, и\, \alpha\in(N_{i-1}\cup V_T)\}$

Шаг 3. Если $N_i\neq N_i-1$, то положить $i=i+1$ и перейти к пункту 2, иначе считать $N=Ni$.

Если $S\in N$, то выдать сообщение о том, что язык грамматики существует, иначе сообщить об отсутствии языка.

Пример 4.1. Дана грамматика $G=(\{0,1\},\{S, A, B\}, P, S)$, где множество правил P: 1) $S\to AB$; 2) $A\to 0A$; 3) $A\to 0$; 4) $B\to 1$. Построим последовательность приближений множества N:

$$N_0 = Щ$$
$$N_1=\{A,B\}$$
$$N_2=\{S,A,B\}$$
$$N_3=\{S,A,B\}$$

Т.к. $N_2=N_3$, то $N=\{S, A, B\}$, следовательно, язык грамматики существует, потому что начальный символ $S\in N$.

Определение 4.2. Бесполезными символами грамматики называют:

а) нетерминалы, не порождающие терминальных строк, т.е. множество символов $\{X \mid X\in V_N, \neg\exists(X\Rightarrow^* x), x\in V_T^*\}$;

б) недостижимые нетерминалы, порождающие терминальные строки, т.е. множество символов

$\{X \mid X\in V_N, \neg\exists(S\Rightarrow^*\alpha X\beta), \exists(X\Rightarrow^* x)\alpha,\beta\in V^*; x\in V_T^*\}$;

в) недостижимые терминалы, т.е. множество символов

$\{X \mid X \in V_T, \neg\exists(S \Rightarrow^* \alpha X \beta); \alpha, \beta \in V^*\}$;

Алгоритм 4.2. Устранение нетерминалов, не порождающих терминальных строк

Вход: КС-грамматика $G=(VT,VN,P,S)$.

Выход: КС-грамматика $G' = (VT,V_N' , P', S)$, такая, что $L(G') = L(G)$ и для всех $Z \in V_N'$ существуют выводы $Z \Rightarrow^* x$, где $x \in V_T^*$.

Шаг 1. Определить множество не терминалов, порождающих терминальные строки, с помощью алгоритма 4.1.

Шаг 2. Вычислить $V_N' = V_N \cap N$, $N_Б = V_N - V_N'$,$P' = P - P_Б$, где $P_Б \subseteq P$ − это множество правил, содержащих бесполезные не терминалы $X \in N_Б$.

Пример 4.2. Дана грамматика $G=(\{a, b, c\},\{S, A, B, C\}, P, S)$ с правилами P: 1) $S \to ab$; 2) $S \to AC$; 3) $A \to AB$; 4) $B \to b$; 5) $C \to cb$.

Преобразуем ее в эквивалентную грамматику G' по алгоритму 4.2:

$N_0 = Ø$

$N_1 = \{S, B, C\}$

$N_2 = \{S, B, C\}$

Т.к. $N_1=N_2$, то $N = \{S, B, C\}$. После удаления бесполезных нетерминалов и правил вывода получим грамматику $G' = (\{a, b, c\},\{S, B, C\}, P', S)$ с правилами P': 1) $S \to ab$; 2) $B \to b$; 5) $C \to cb$.

Алгоритм 4.3. Устранение недостижимых символов

Вход: КС-грамматика $G=(VT,VN,P,S)$.

Выход: КС-грамматика $G' = (V_T' ,V_N' , P', S)$, такая, что $L(G') = L(G)$ и для всех $Z \in V'$ существует вывод $S \Rightarrow^* \alpha Z \beta,\ где \alpha, \beta \in (V')^*$.

Определим множество достижимых символов Z грамматики G, т.е. множество

$$W = \{Z \mid Z \in V, \exists(S \Rightarrow^* \alpha Z \beta); \alpha, \beta \in V^*\}$$

Шаг 1. Положить $W_0 = S$.

Шаг 2. Вычислить очередное приближение следующим образом:

$W_i = W_{i-1} \cup \{X \mid X \in V, (A \to \alpha X \beta) \in P, A \in W_{i-1}; \alpha, \beta \in V^*\}$

Шаг 3. Если $W_i \neq W_i - 1$, то положить $i:=i+1$ и перейти к шагу 2, иначе считать $W=Wi$.

Шаг 4. Вычислить $V_N' = V_N \cap W, V_T' = V_T \cap W, V_Б = V - W$, $P' = P - P_Б$, где $P_Б \subseteq P$ − это множество правил, содержащих недостижимые символы $X \in V_Б$.

Пример 4.3. Дана грамматика $G=(\{a, b, c\},\{S, B, C\}, P, S)$ с правилами P': 1) $S\to ab$; 2) $B\to b$; 5) $C\to cb$.

Преобразуем ее в эквивалентную грамматику G' по алгоритму 4.3:

$W_0 = \{S\};$

$W_1 = \{S, a, b\};$

$W_2 = \{S, a, b\}.$

Т.к. $W_1=W_2$, то$W=\{S, a, b\}$.

Множество недостижимых символов $V_Б = \{B,C, c\}$. Тогда после удаления недостижимых символов получим грамматику $G' = (\{a, b\},\{S\}, P, S)$ с правилом P': $S\to ab$.

Алгоритм 4.4. Устранение ε-правил

Вход: КС-грамматика $G=(VT,VN, P, S)$.

Выход: Эквивалентная КС-грамматика $G' = (V_T,V_N' , P', S')$ без ε-правил для всех нетерминальных символов, кроме начального, который не должен встречаться в правых частях правил грамматики.

Шаг 1. В исходной грамматике G найти ε-порождающие нетерминальные символы $A\in VN$, такие, что $A\Rightarrow^*\varepsilon$.

1.1 Положить $N_0 = \{A \mid (A\to\varepsilon)\in P\}$.

1.2 Вычислить $N_i= Ni-1 \cup \{B \mid (B\to\alpha)\in P, \alpha\in N_i^{*}-1\}$.

1.3 Если $N_i\neq N_i-1$, то положить $i:=i+1$ и перейти к пункту 1.2, иначе считать $N= Ni$.

Шаг 2. Из множества P правил исходной грамматики G перенести во множество P' все правила за исключением ε-правил, т.е. $P' = P - \{(A \to \varepsilon) \in P$ для всех $A\in VN\}$.

Шаг 3. Пополнить множество P' правилами, которые получаются из каждого правила этого множества путем исключения всевозможных комбинаций ε-порождающих нетерминалов в правой части. Полученные при этом ε-правила во множество P' не включать.

Шаг 4. Если $S\in N$, то $P' = P\cup\{S' \to\varepsilon, S' \to S\}, V_N' = V_N\cup S'$, где $V\cap\{S'\} = \emptyset$; иначе $V_N' = VN, S' = S$.

Пример 4.4. Дана грамматика $G=(\{0,1\},\{S, A, B\}, P, S)$ с правилами P:

1) $S\to AB$; 2) $A\to 0A|\varepsilon$; 3) $B\to 1B|\varepsilon$. Преобразуем ее в эквивалентную грамматику по алгоритму 4.4.

Шаг 1.

$N_0 = \{A, B\};$

$N_1 = \{S, A, B\};$

$N_2 = \{S, A, B\}.$

Т.к. $N_1 = N_2$, то искомое множество построено и $N = \{S, A, B\}$.

Шаг 2, 3. Множество P': 1) $S\to AB|\ A|\ B$; 2) $A\to 0A|0$; 3) $B\to 1B|1$.

Шаг 4. Т.к. $S\in N$, то введем новый нетерминал C и пополним множество P' правилом вида $C\to S|\varepsilon$. Результирующая грамматика будет иметь вид: $G' = (\{0,1\},\{S, A, B,C\}, P, C)$ с правилами P': 1) $C\to S|\varepsilon$; 2) $S\to AB|\ A|B$; 3) $A\to 0A|0$; 4) $B\to 1B|1$.

Алгоритм 4.5. Устранение цепных правил

Вход: КС-грамматика $G= (V_T,V_N, P, S)$.

Выход: Эквивалентная КС-грамматика $G' = (V_T,V_N', P', S')$ без цепных правил, т.е. правил вида $A\to B$, где $A, B\in V_N$.

Шаг 1. Для каждого нетерминала A вычислить множество выводимых из него не терминалов, т.е. множество $N^A=\{B|\ A\Rightarrow^* B$, где $B\in V_N\}$.

1.1 Положить $N_0^A = \{A\}$

1.2 Вычислить $N_i^A= N_i^A{}_{-1} \cup \{C|(B\to C)\in P, B\in N_i^A{}_{-1}, C\in V_N\}$.

1.3 Если $N_i^A\neq N_i^A{}_{-1}$, то положить $i:=i+1$ и перейти к пункту 1.2, иначе считать $N_A= N_{iA}$.

Шаг 2. Построить множество P' так: если $(B\to\alpha)\in P$ не является цепным правилом $(\alpha\notin V_N)$, то включить в P' правило $A\to\alpha$ для каждого A, такого, что $B\in N^A$.

Пример 4.5. Грамматика $G= (\{+, n\},\{L, M, N\}, P, L)$ с правилами P:

1) $L\to M$; 2) $M\to N$; 3) $N\to N+|\ n$. Преобразуем ее в эквивалентную грамматику G' по алгоритму 4.5.

Шаг 1.

$N_0^L = \{L\};$

$N_1^L = \{L,M\};$

$N_2^L = \{L,M,N\};$

$N_3^L = \{L,M,N\}.$

Т.к. $N_2^L= N_3^L$, то $N^L=\{L, M, N\}$.

$N_0^M = \{M\};$

$N_1^M = \{M,N\};$

$N_2^M = \{M,N\};$

Т.к. $N_1^M= N_2^M$, то $N^M=\{M, N\}$.

$N_0^N = \{N\};$

$N_1^N = \{N\};$

Т.к. $N_1{}^N = N_0{}^N$, то $N^N = \{\}.N$

Шаг 2. Преобразовав правила вывода грамматики, получим грамматику $G' = (\{+, n\}, \{L, M, N\}, P', L)$ с правилами P':

1) $L \to N+ \mid n$; 2) $M \to N+ \mid n$; 3) $N \to N+ \mid n$.

Алгоритм 4.6. Устранение левой факторизации правил

Вход: КС-грамматика $G = (V_T, V_N, P, S)$.

Выход: Эквивалентная КС-грамматика $G' = (V_T, V_N', P', S')$ без одинаковых префиксов в правых частях правил, определяющих нетерминалы.

Шаг 1. Записать все правила для нетерминала X, имеющие одинаковые префиксы $\alpha \in V^*$, в виде одного правила с альтернативами:

$X \to \alpha\beta_1 \mid \alpha\beta_2 \mid ... \mid \alpha\beta_n; \beta_1, \beta_2, ..., \beta_n \in V^*$

Шаг 2. Вынести за скобки влево префикс α в каждой строке-альтернативе: $X \to \alpha(\beta_1 \mid \beta_2 \mid ... \mid \beta_n)$.

Шаг 3. Обозначить новым нетерминалом Y выражение, оставшееся в скобках: $X \to \alpha Y, Y \to \beta_1 \mid \beta_2 \mid ... \mid \beta_n$.

Шаг 4. Пополнить множество нетерминалов новым нетерминалом Y и заменить правила, подвергшиеся факторизации, новыми правилами для X и Y

Шаг 5. Повторить шаги 1-4 для всех нетерминалов грамматики, для которых это возможно и необходимо.

Пример 4.6. Дана грамматика $G = (\{k, l, m, n\}, \{S\}, P, S)$ с правилами P:

1) $S \to kSl$; 2) $S \to kSm$; 3) $S \to n$. Преобразуем ее в эквивалентную грамматику G' по алгоритму 4.6:

Шаг 1. $S \to kSl \mid kSm \mid n$ Шаг 2. $S \to kS(l \mid m) \mid n$.

Шаг 2. Пополнив множество нетерминалов новым нетерминалом C и заменив правила, подвергшиеся факторизации, получим грамматику
$G' = (\{k,l,m,n\}, \{S,C\}, P', S)$ с правилами P': 1) $S \to kSC$; 2) $S \to n$; 3) $C \to l$; 4) $C \to m$.

Алгоритм 4.7. Устранение прямой левой рекурсии

Вход: КС-грамматика $G = (V_T, V_N, P, S)$.

Выход: Эквивалентная КС-грамматика $G' = (V_T, V_N', P', S')$ без прямой левой рекурсии, т.е. без правил вида $A \to A\alpha, A \in V_N, \alpha \in V^*$

Шаг 1. Вывести из грамматики все правила для рекурсивного нетерминала X:

$X \to X\alpha_1 \mid X\alpha_2 \mid ... \mid X\alpha_m (X \in V_N; \alpha_1, \alpha_2, ..., \alpha_m \in V^*)$

$X \to \beta_1 \mid \beta_2 \mid ... \mid \beta_n (\beta_1, \beta_2, ..., \beta_n \in V^*)$

Шаг 2. Внести новый нетерминал Y так, чтобы он описывал любой «хвост» строки, порождаемой рекурсивным нетерминалом X:

$Y \to \alpha_1 Y \mid \alpha_2 Y \mid ... \mid \alpha_m Y$

$Y \to \alpha_1 \mid \alpha_2 \mid ... \mid \alpha_m$

Шаг 3. Заменить в рекурсивном правиле для X правую часть, используя новый нетерминал и все нерекурсивные правила для X так, чтобы генерируемый язык не изменился:

$X \to \beta_1 Y \mid \beta_2 Y \mid ... \mid \beta_n Y$

$X \to \beta_1 \mid \beta_2 \mid ... \mid \beta_n$

$Y \to \alpha_1 Y \mid \alpha_2 Y \mid ... \mid \alpha_m Y$

$Y \to \alpha_1 \mid \alpha_2 \mid ... \mid \alpha_m$

Шаг 4. Пополнить множество нетерминалов грамматики новым нетерминалом Y. Пополнить множество правил грамматики правилами, полученными на шаге 3.

Шаг 5. Повторить действия шагов 1-4 для всех рекурсивных нетерминалов грамматики, после чего полученные множества нетерминалов и правил принять в качестве V_N' и P'.

Пример 4.7. Дана грамматика G= ({a, b, c, d, z},{S, A, B, C}, P, S) с правилами P: 1) $S \to Aa$; 2) $A \to Bb$; 3) $B \to Cc \mid d$; 4) $C \to Ccbz \mid dbz$. После устранения прямой левой рекурсии получим эквивалентную грамматику $G' =$ ({a, b, c, d, z},{S, A, B, C, Z}, P', S) с правилами P':

1) $S \to Aa$; 2) $A \to Bb$; 3) $B \to Cc \mid d$; 4) $C \to dbzZ \mid dbz$; 5) $Z \to cbzZ \mid cbz$.

Постановка задачи к практической работе № 4

При выполнении практической работы следует реализовать следующие действия:

1 проверить грамматику на принадлежность к классу КС-грамматик;

2 проверить существование языка КС-грамматики;

3 выполнить эквивалентные преобразования грамматики, направленные на удаление:

а) бесполезных символов;

б) недостижимых символов;

в) ε-правил;

г) цепных правил;

д) левой факторизации правил;

е) прямой левой рекурсии.

Варианты индивидуальных заданий представлены в таблице 9

Таблица 9 − Варианты индивидуальных заданий к практическим работам № 4 и 5

Вариант	Контекстно-свободная грамматика
1	G=({S, A, B, D, E},{a, b, c, e}, P, S), где P: 1) $S \to AB \mid \varepsilon$; 2) $A \to Aa \mid S \mid a$; 3) $B \to bD \mid bS \mid b$; 4) $D \to ccD$; 5) $E \to eE \mid e$

2	$G=(\{E, T, F, G, H\}, \{+, -, *, /, n, m, h\}, P, E)$, где P: 1) $E \to T \mid E+T \mid E-T \mid \varepsilon$; 2) $T \to F \mid F*T \mid F/T \mid \varepsilon$; 3) $F \to G \mid Fn \mid n$; 4) $G \to Gm$; 5) $H \to Hh \mid h$
3	$G=(\{S, R, T, X, Y\}, \{a, b, p, g, y\}, P, S)$, где P: 1) $S \to R \mid T$; 2) $R \to pX \mid paR \mid paT \mid \varepsilon$; 3) $T \to Tg \mid g$; 4) $X \to aXb$; 5) $Y \to aYa \mid y$
4	$G=(\{Q, A, B, C, D\}, \{a, b, c, d\}, P, Q)$, где P: 1) $Q \to acA \mid acB \mid \varepsilon$; 2) $B \to A \mid Cb \mid \varepsilon$; 3) $A \to Aa \mid Ab \mid a$; 4) $C \to dCc$; 5) $D \to dc$
5	$G=(\{R, T, F, G, K\}, \{m, i, j, k, \wedge, \sim, \perp\}, P, R)$, где P: 1) $R \to R\sim T\perp \mid R\wedge T\perp \mid \varepsilon$; 2) $T \to F \mid Fi \mid Fj \mid Gk \mid \varepsilon$; 3) $G \to GkG$; 4) $K \to Ki \mid Km \mid m$
6	$G=(\{S, X, Y, Z, K\}, \{x, y, z, k, \#, \$\}, P, S)$, где P: 1) $S \to X \mid Y \mid Z$; 2) $X \to x\#X \mid x\#Y \mid \varepsilon$; 3) $Y \to Yy\$ \mid Yz\$ \mid \$ \mid \varepsilon$; 4) $Z \to Zz\$$; 5) $K \to Kk\$ \mid k\$$
7	$G=(\{S, L, M, P, N\}, \{n, m, l, p, @, \perp\}, V, S)$, где V: 1) $S \to @nL \mid @mM \mid P$; 2) $L \to M \mid Ll\perp \mid Lm\perp \mid \varepsilon$; 3) $M \to L \mid Mm \mid mm$; 4) $N \to pN@ \mid @$; 5) $P \to nmP$
8	$G=(\{X, Y, Z, K, L\}, \{a, b, l, =, <, >, \wedge, \vee, \neg\}, V, X)$, где V: 1) $X \to Y \mid Y=Y \mid Y<Y \mid Y>Y \mid K$; 2) $Y \to Y\wedge Z \mid Y\vee Z \mid \varepsilon$; 3) $Z \to \neg a \mid \neg b \mid \varepsilon$; 4) $K \to \neg K$; 5) $L \to l \mid a \mid b$
9	$G=(\{Q, A, B, C, D\}, \{0, 1, -\}, P, Q)$, где P: 1) $Q \to 01A \mid 01B \mid A$; 2) $A \to 0B1 \mid B \mid 1 \mid \varepsilon$; 3) $B \to BA0 \mid B1 \mid C \mid \varepsilon$; 4) $C \to 0C11$; 5) $D \to -D1 \mid -0 \mid -1$
10	$G=(\{R, T, U, W, V\}, \{0, 1, +, -, *, /\}, P, R)$, где P: 1) $R \to T1T \mid T1U \mid W \mid \varepsilon$; 2) $T \to U \mid T01 \mid T10 \mid \varepsilon$; 3) $U \to +U \mid +0 \mid +1$; 4) $W \to W-W \mid W+W$; 5) $V \to *0 \mid /1$
11	$G=(\{S, R, T, F, E\}, \{a, b, k, \{, [, \},], \perp\}, P, S)$, где P: 1) $S \to \{R \mid [R$; 2) $R \to Ra\} \mid Ra] \mid a \mid T \mid F \mid \varepsilon$; 3) $F \to \{F\} \mid bb$; 4) $T \to [T]$; 5) $E \to k\perp$
12	$G=(\{Y, K, M, L, S\}, \{a, b, *, /, \wedge\}, P, Y)$, где P: 1) $Y \to KS \mid KM$; 2) $K \to K* \mid K/ \mid S$; 3) $S \to Sa/ \mid Sb/ \mid \varepsilon$; 4) $M \to *M*$; 5) $L \to L\wedge \mid \wedge a$

Практическая работа № 5

Построение автомата с магазинной памятью по контекстно-свободной грамматике

Цели работы:

- закрепить понятия «автомат с магазинной памятью (МП-автомат)», «расширенный МП-автомат», «конфигурация МП-автомата»; «строка и язык, допускаемые МП-автоматом»;

- сформировать умения и навыки построения МП-автомата и расширенного МП-автомата по КС-грамматике, разбора входной строки с помощью МП-автомата.

Основы теории

КС-языки можно распознавать с помощью автомата с магазинной памятью (МП-автомата).

Определение 5.1. МП-автомат можно представить в виде семерки:

$$M = (Q, T, N, F, q_0, N_0, Z),$$

где Q – конечное множество состояний автомата;

T – конечный входной алфавит;

N – конечный магазинный алфавит;

F – магазинная функция, отображающая множество $(Q \times (T \cup \{\varepsilon\}) \times N)$ во множество всех подмножеств множества $Q \times N^*$, т.е.

$F : (Q \times (T \cup \{\varepsilon\}) \times N) \to P(Q \times N^*)$;

q_0 – начальное состояние автомата, $q_0 \notin Q$;

N_0 – начальный символ магазина, $N_0 \notin T$;

Z – множество заключительных состояний автомата, $Z \subseteq Q$.

Определение 5.2. Конфигурацией МП-автомата называется тройка вида: $(q, \omega, \alpha) \notin (Q \times T^* \times N^*)$

где q − текущее состояние автомата, $q \notin Q$;

ω − часть входной строки, первый символ которой находится под входной головкой, $\omega \in T^*$;

α − содержимое магазина $\alpha \in N^*$.

Общая схема МП-автомата представлена на рисунке 7.

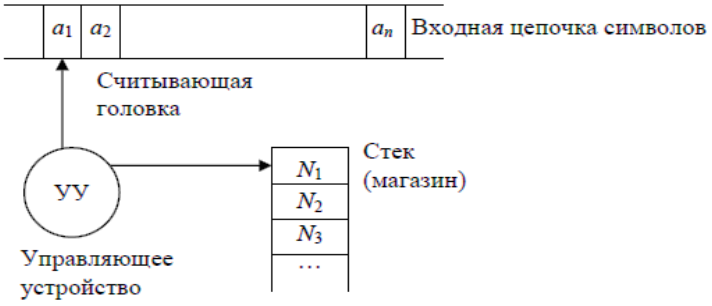


Рисунок 7 − Схема МП-автомата

Алгоритм 5.1. Функционирование МП-автомата

Начальной конфигурацией МП-автомата является конфигурация (q_0,N_0).

Шаг работы МП-автомата будем представлять в виде отношения непосредственного следования конфигураций (обозначается «|=») и отношения достижимости конфигураций (обозначается «|=*»). Если одним из значений магазинной функции $F(q \in Q, t \in (T \cup \{e\}), S \in N)$ является $(q' \in Q, \gamma \in N^*)$, то записывается $(q, t\omega, S\alpha) \models (q', \omega, \lambda\alpha)$. При этом возможны следующие варианты.

1) случай $t \in T$. Автомат находится в текущем состоянии q, читает входной символ t, имеет в вершине стека символ S. Он переходит в очередное состояние q', сдвигает входную головку на ячейку вправо и заменяет верхний символ S строкой магазинных символов. Вариант $\gamma = \varepsilon$ означает, что S удаляется из стека.

2) случай $t = \varepsilon$. Отличается от первого случая тем, что входной символ t просто не принимается во внимание, и входная головка не сдвигается. Такой шаг работы МП-автомата называется шагом, который может выполняться даже после завершения чтения входной строки.

Заключительной конфигурацией МП-автомата является конфигурация (q, ε, α), где $q \in Z$.

Определение 5.3. МП-автомат допускает входную строку ω, если существует путь по конфигурациям $(q_0, w, N_0) \models^* (q, \varepsilon, \alpha)$ для некоторых $q \in Z$ и $a \in N^*$.

Определение 5.4. Язык L, распознаваемый (принимаемый) МП-автоматом M, определяется как множество вида $L(M) = \{\omega \mid \omega \in T^* \ и \ (q_0, \omega, N_0) \models^* (q, \varepsilon, \alpha)$ для некоторых $q \in Z$ и $a \in N^*$.

Определение 5.5. МП-автомат с магазинной функцией $F : (Q \times (T \cup \{\varepsilon\}) \times N^*) \to P(Q \times N^*)$ называется расширенным МП-автоматом, т.е. автоматом, который может заменять цепочку символов конечной длины в верхушке стека на другую цепочку символов конечной длины.

Существуют КС-языки, МП-автоматы и расширенные МП-автоматы, определяющие один и тот же язык.

Алгоритм 5.2. Построение МП-автомата по КС-грамматике

Построим МП-автомат, выполняющий левосторонний разбор. Данный автомат обладает только одним состоянием и принимает входную строку опустошением магазина. Стек используется для размещения текущей сентенции, первоначально это начальный символ грамматики. Очередная сентенция получается заменой верхнего нетерминала стека.

Вход: КС-грамматика $G = (V_T, V_N, P, S)$.

Выход: МП-автомат $M = (Q, T, N, F, q_0, n_0, Z)$ такой, что $L(M) = L(G)$.

Шаг 1. Положить $Q = \{q\}, q_0 = q, Z = \varnothing, N = V_T \cup V_N, T = V_T, N_0 = S$.

Шаг 2. Для каждого правила вида $(A \to \beta) \in P$, где $\beta \in V^*$, сформировать магазинную функцию вида $F(q, \varepsilon, A) = (q, \beta)$. Эти функции предписывают заме-

щать нетерминал в вершине стека по правилу грамматики.

Шаг 3. Для каждого $t \in V_T$ сформировать магазинную функцию вида $F(q,t,t) = (q,\varepsilon)$, которая выталкивает из стека символ, совпадающий с входным, и перемещает читающую головку. Эти функции обеспечивают опустошение стека.

Пример 5.1. Дана КС-грамматика:

$$G\big(\{+,\ (,\ ),a\},\ \{S,A\},\ \{S \to S+A \mid A, A \to (S) \mid a\},\ \{S\}\big).$$

Последовательность построения МП-автомата будет иметь вид.

1) $Q = \{q\}, q_0 = q, T = \{+,\ (,\ ),a\}, N = \{+,\ (,\ ),a,S,A\}, N_0 = S, Z = \varnothing.$

2) $F(q,\varepsilon,S) = (q_0, S+A), F(q,\varepsilon,S) = (q,A), F(q,\varepsilon,A) = (q,(S));$

$F(q,\ \varepsilon, A) = (q,a).$

3) $F(q,t,t) = (q,\varepsilon)$ для каждого $t \in \{+,\ (,\ ),a\}.$

Распознавание строки (a) построенным МП-автоматом представлено в таблице 10. Полученный МП-автомат является недетерминированным.

Таблица 10 − Распознавание МП-автоматом строки (a)

Номер конфигурации	Текущее состояние	Входная строка	Содержимое магазина
1	q	(a)	S
2	q	(a)	A
3	q	(a)	(S)
4	q	a)	S)
5	q	a)	A)
6	q	a)	a)
7	q	)	)
8	q	ε	ε

Алгоритм 5.3. Построение расширенного МП-автомата по КС-грамматике

Построим МП-автомат, выполняющий правосторонний разбор. Данный автомат имеет единственное текущее состояние и одно заключительное состояние, в котором стек пуст. Стек содержит левую часть текущей сентенции. Первоначально в стек помещается специальный магазинный символ, маркер пустого стека #. На каждом шаге автомат по правилу грамматики замещает нетерминалом строку верхних символов стека или дописывает в вершину входной символ.

Вход: КС-грамматика $G = (V_T, V_N, P, S)$.

Выход: расширенный МП-автомат $M = (Q,T,N,F,q_0,n_0,Z)$ такой, что $L(M) = L(G)$.

Шаг 1. Положить $Q=\{q,r\}, q_0=q, Z=r, N=V_T\cup V_N\cup\{\#\}, T=V_T, N_0=\#.$

Шаг 2. Для каждого правила вида $(A\to\beta)\in P$, где $\beta\in V^*$, сформировать магазинную функцию вида $F(q,\varepsilon,\beta)=(q,A)$, предписывающую заменять правую часть правила в вершине стека нетерминалом из левой части, независимо от текущего символа входной строки.

Шаг 3. Для каждого терминала $t\in T$ сформировать магазинную функцию вида $F(q,t,\varepsilon)=(q,t)$, которая помещает символ входной строки в вершину стека, если там нет правой части правила, и перемещает читающую головку.

Шаг 4. Предусмотреть магазинную функцию для перевода автомата в заключительное состояние $F(q,\varepsilon,\#S)=(r,\varepsilon)$.

Пример 5.2. Для грамматики из примера 5.1 построить расширенный МП-автомат. Последовательность построения МП-автомата будет иметь вид.

$1) Q=\{q,r\}, q_0=q, T=\{+,\ (,\ ),a\}, N=\{+,\ (,\ ),a,S,A\}, N_0=\#, Z=r.$

$2) F(q,\varepsilon,S+A)=(q,S), F(q,\varepsilon,A)=(q,S), F(q,\varepsilon,(S))=(q,A), F(q,\varepsilon,a)=(q,A).$

$3) F(q,t,\varepsilon)=(q,t)$ *для каждого* $t\in\{+,\ (,\ ),a\}.$

$4) F(q,\varepsilon,\ \#S)=(r,\varepsilon).$

Распознавание строки (a) расширенным МП-автоматом представлено в таблице 11. Полученный МП-автомат является детерминированным.

Таблица 11 − Распознавание расширенным МП-автоматом строки (a)

Номер конфигурации	Текущее состояние	Входная строка	Содержимое магазина
1	q	(a)	#
2	q	a)	#(
3	q	)	#(a
4	q	)	#(A
5	q	)	#(S
6	q	ε	#(S)
7	q	ε	#A
8	q	ε	#S
9	r	ε	ε

Постановка задачи к практической работе № 5

При выполнении практической работы следует реализовать следующие действия:

а) проверить грамматику на принадлежность к классу КС-грамматик;

б) построить МП-автомат по КС-грамматике;

в) построить расширенный МП-автомат по КС-грамматике;

г) продемонстрировать разбор некоторой входной строки с помощью по-

строенных автоматов для случая:

- входная строка принадлежит языку исходной КС-грамматики и допускается МП-автоматом;
- входная строка не принадлежит языку исходной КС-грамматики и не допускается МП-автоматом.

Индивидуальные варианты заданий представлены в таблице 9.

Практическая работа № 6

Функционирование распознавателя для *LL*(1)-грамматик

Цели работы:

- закрепить понятие «*LL*(*k*)-грамматика», необходимые и достаточные условия *LL*(*k*)-грамматики;

- сформировать умения и навыки построения множеств *FIRST*(*k*, α) и *FOLLOW*(*k*, α), распознавателя для *LL*(1)-грамматик.

Основы теории

Определение 6.1. КС-грамматика обладает свойством *LL*(*k*) для некоторого $k>0$, если на каждом шаге вывода для однозначного выбора очередной альтернативы МП-автомату достаточно знать символ на верхушке стека и рассмотреть первые *k* символов от текущего положения считывающей головки во входной строке.

Определение 6.2. КС-грамматика называется *LL*(*k*)-грамматикой, если она обладает свойством *LL*(*k*) для некоторого $k>0$.

В основе распознавателя *LL*(*k*)-грамматик лежит левосторонний разбор строки языка. Исходной сентенциальной формой является начальный символ грамматики, а целевой – заданная строка языка. На каждом шаге разбора правило грамматики применяется к самому левому нетерминалу сентенции. Данный процесс соответствует построению дерева разбора цепочки сверху вниз (от корня к листьям). Отсюда и произошла аббревиатура *LL*(*k*): первая «*L*» (от слова «*left*») означает левосторонний ввод исходной цепочки символов, вторая «*L*» − левосторонний вывод в процессе работы распознавателя.

Определение 6.3. Для построения распознавателей для *LL*(*k*)-грамматик используются два множества:

- *FIRST*(*k*, α) – множество терминальных цепочек, выводимых из цепочки $\alpha\in(V_T\cup V_N)^*$, укороченных до *k* символов;
- *FOLLOW*(*k*, *A*) – множество укороченных до *k* символов терминальных цепочек, которые могут следовать непосредственно за $A\in V_N$ в цепочках вывода.

Формально эти множества можно определить следующим образом:

- $FIRST(k, \alpha) = \{\omega\in V_T^* \mid \exists$ вывод $\alpha \Rightarrow^* \omega$ и $|\omega| \le k$ или $\exists$ вывод $\alpha \Rightarrow^* \omega x$ и $|\omega| = k$; $x, \alpha \in (V_T\cup V_N)^*, k > 0\}$;

- $FOLLOW(k, A) = \{\omega\in V_T^* \mid \exists$ *вывод* $S\Rightarrow^*\alpha A\gamma$ и $\omega\in FIRST(k, \gamma)$; $\alpha,\gamma\in V^*$, $A\in V_N, k>0\}$.

Теорема 6.1. Необходимое и достаточное условие *LL*(1)-грамматики

Для того чтобы грамматика $G(V_N, V_T, P, S)$ была *LL*(1)-грамматикой, необходимо и достаточно, чтобы для каждого символа $A\in V_N$, у которого в грамматике существует более одного правила вида $A\to\alpha_1 \mid \alpha_2 \mid\ldots\mid \alpha_n$, выполнялось требование:

$$FIRST(1, a_i FOLLOW(1, A)) \cap FIRST(1, a_j FOLLOW(1, A)) = \varnothing,$$

$\forall i \neq j,\ 0 < i \le n,\ 0 < j \le n.$

Т.е. если для символа A отсутствует правило вида $A \to \varepsilon$, то все множества $FIRST(1, \alpha_1)$, $FIRST(1, \alpha_2)$,…, $FIRST(1, \alpha_n)$ должны попарно не пересекаться, если же присутствует правило $A \to \varepsilon$, то они не должны также пересекаться с множеством $FOLLOW(1, A)$.

Для построения распознавателей для $LL(1)$-грамматик необходимо построить множества $FIRST(1, x)$ и $FOLLOW(1, A)$. Причем если строка x будет начинаться с терминального символа a, то $FIRST(1, x)=a$, и если она будет начинаться с нетерминального символа A, то $FIRST(1, x)=FIRST(1, A)$. Следовательно, достаточно рассмотреть алгоритмы построения множеств $FIRST(1, A)$ и $FOLLOW(1, A)$ для каждого нетерминального символа A.

Алгоритм 6.1. Построение множества $FIRST(1, A)$

Для выполнения алгоритма необходимо предварительно преобразовать исходную грамматику G в грамматику G', не содержащую ε-правил (см. лабораторную работу № 4). Алгоритм построения множества $FIRST(1, A)$ использует грамматику G'.

Шаг 1. Первоначально внести во множество первых символов для каждого нетерминального символа A все символы, стоящие в начале правых частей правил для этого не терминала, т.е.

$$\forall A \in V_N FIRST(1, A) = \{X \mid A \to X\alpha \in P,\ X \in (V_T \cup V_N),\ \alpha \in (V_T \cup V_N)^*\}.$$

Шаг 2. Для всех $A \in V_N$ положить:

$$FIRST_{i+1}(1, A) = FIRST_i(1, A) \cup FIRST_i(1, B),\ \forall B \in (FIRST(1, A) \cap V_N).$$

Шаг 3. Если существует $A \in V_N$, такой что $FIRST_{i+1}(1, A) \neq FIRST_i(1, A)$, то присвоить $i=i+1$ и вернуться к шагу 2, иначе перейти к шагу 4.

Шаг 4. Исключить из построенных множеств все нетерминальные символы, т.е. $\forall A \in V_N FIRST(1, A) = FIRST_i(1, A)_{\setminus N}$.

Алгоритм 6.2. Построение множества $FOLLOW(1, A)$

Алгоритм основан на использовании правил вывода грамматики G.

Шаг 1. Первоначально внести во множество последующих символов для каждого нетерминального символа A все символы, которые в правых частях правил вывода встречаются непосредственно за символом A, т.е.

$$\forall A \in V_N FOLLOW_0(1, A) = \{X \mid \exists B \to \alpha A X \beta \in P,\ B \in V_N,\ X \in (V_T \cup V_N), \alpha, \beta \in (V_T \cup V_N)^*\}.$$

Шаг 2. Внести пустую строку во множество $FOLLOW(1, S)$, т.е.

$$FOLLOW(1, S) = FOLLOW(1, S) \cup \{\varepsilon\}.$$

Шаг 3. Для всех $A \in V_N$ вычислить:

$FOLLOW'_i(1,A)=FOLLOW_i(1,A)\cup FIRST(1,B),\forall B\in(FOLLOW_i(1,A)\cap V_N)$.

Шаг 4. Для всех $A\in V_N$ положить:

$$FOLLOW''_i(1, A)=FOLLOW'_i(1, A)\cup FOLLOW'_i(1, B),\forall B\in(FOLLOW'_i(1, A)\cap V_N),$$

если ∃ правило $B\to\varepsilon$.

Шаг 5. Для всех $A\in V_N$ определить:

$FOLLOW_{i+1}(1, A) = FOLLOW''_i(1, A)\cup FOLLOW''_i(1, B)$, для всех нетерминальных символов $B\in V_N$, имеющих правило вида $B\to\alpha A$, $\alpha\in(V_T\cup V_N)^*$.

Шаг 6. Если существует $A\in V_N$ такой, что $FOLLOW_{i+1}(1,A)\neq FOLLOW_i(1,A)$, то положить $i:=i+1$ и вернуться к шагу 3, иначе перейти к шагу 7.

Шаг 7. Исключить из построенных множеств все нетерминальные символы, т.е. $\forall A\in V_N\ FOLLOW(1, A) = FOLLOW_i(1, A)_{\setminus N}$.

Алгоритм 6.3. Функционирование распознавателя цепочек для $LL(1)$-грамматик

Шаг 1. Помещаем в стек начальный символ грамматики S, а во входной буфер − исходную цепочку символов.

Шаг 2. До тех пор, пока в стеке и во входном буфере останется только пустая строка ε либо будет обнаружена ошибка в алгоритме разбора, выполняем одно из следующих действий:

- если на верхушке стека находится нетерминальный символ A и очередной символ входной строки символ a, то выполняем операцию «свертка» по правилу $A\to x$ при условии, что $a\in FIRST(1, x)$, т.е. извлекаем из стека символ A и заносим в стек строку x, не меняя содержимого входного буфера;
- если на верхушке стека находится нетерминальный символ A и очередной символ входной строки символ a, то выполняем операцию «свертка» по правилу $A\to\varepsilon$ при условии, что $a\in FOLLOW(1, A)$, т.е. извлекаем из стека символ A и заносим в стек строку ε, не меняя содержимого входного буфера;
- если на верхушке стека находится терминальный символ a, совпадающий с очередным символом входной строки, то выполняем операцию «выброс», т.е. удаляем из стека и входного буфера данный терминальный символ;
- если содержимое стека и входного буфера пусто, то исходная строка прочитана полностью, и разбор завершен удачно;
- если ни одно из данных условий не выполнено, то цепочка не принадлежит заданному языку, и алгоритм завершает свою работу с ошибкой.

Пример 6.1. Дана грамматика $G(\{S, T, R\}, \{+, -, (,), a, b\}, P, S)$ с правилами P: 1) $S\to TR$; 2) $R\to\varepsilon \mid +TR \mid - TR$; 3) $T\to(S) \mid a \mid b$. Построить распознаватель для строки $(a+(b-a))$ языка грамматики G.

Этап 1. Преобразуем грамматику Gв грамматику G', не содержащую ε-правил:

$N_0 = \{R\}$;

$N_1 = \{R\}$, т.к. $N_0 = N_1$, то во множество P' войдут правила:

1) $S \to TR|T$; 2) $R \to +TR \mid +T \mid -TR|-T$;

3) $T \to (S) \mid a \mid b$.

Этап 2. Построение множеств $FIRST(1, A)$ для каждого не терминала A представлено в таблице 12.

Таблица 12 − Построение множеств $FIRST(1, A)$

$FIRST_i(1, A)$	0	1	2	$FIRST(1, A)$
S	T	T, (, a, b	T, (, a, b	(, a, b
R	+, -	+, -	+, -	+, -
T	(, a, b	(, a, b	(, a, b	(, a, b

Этап 3. Построение множеств $FOLLOW(1, A)$ для каждого не терминала A представлено в таблице 13.

Таблица 13 − Построение множеств $FOLLOW(1, A)$

Шаг	Не терминалы	$FOLLOW_i(1, A)$	$FOLLOW_i'(1, A)$	$FOLLOW_i''(1, A)$
0	S	)	), ε	), ε
	R	∅	∅	∅
	T	R	R, +, -	R, +, -
1	S	), ε	), ε	), ε
	R	), ε	), ε	), ε
	T	R, +, -	R, +, -	R, +, -,), ε
2	S	), ε	), ε	), ε
	R	), ε	), ε	), ε
	T	R, +, -,), ε	R, +, -,), ε	R, +, -,), ε
$FOLLOW(1, S)$		), ε		
$FOLLOW(1, R)$		), ε		
$FOLLOW(1, T)$		+, -,), ε		

Этап 4. Множества $FIRST(1, A)$ и $FOLLOW(1, A)$ для каждого нетерминала A сведены в таблицу 14.

Таблица 14 − Множества $FIRST(1, A)$ и $FOLLOW(1, A)$

A	$FIRST(1, A)$	$FOLLOW(1, A)$
S	(, a, b	), ε
R	+, -	), ε
T	(, a, b	+, -,), ε

Грамматика G является $LL(1)$-грамматикой, т.к. для каждого нетерминала

A, имеющего альтернативные выводы, множества $FIRST(1, A)$ попарно не пересекаются, а для нетерминала R они также не пересекаются со множеством $FOLLOW(1, R)$.

Шаг 5. Разбор строки $(a+(b-a))$ для грамматики G показан в таблице 15.

Таблица 15 − Разбор строки $(a+(b-a))$ для грамматики G

Стек	Входной буфер	Действие
S	$(a+(b-a))$	Свертка $S\to TR$, т.к. $(\in FIRST(1, TR)$
TR	$(a+(b-a))$	Свертка $T\to(S)$, т.к. $(\in FIRST(1, (S))$
$(S)R$	$(a+(b-a))$	Выброс
$S)R$	$a+(b-a))$	Свертка $S\to TR$, т.к. $a\in FIRST(1, TR)$
$TR)R$	$a+(b-a))$	Свертка $T\to a$, т.к. $a\in FIRST(1, a)$
$aR)R$	$a+(b-a))$	Выброс
$R)R$	$+(b-a))$	Свертка $R\to+TR$, т.к. $+\in FIRST(1, TR)$
$+TR)R$	$+(b-a))$	Выброс
$TR)R$	$(b-a))$	Свертка $T\to(S)$, т.к. $(\in FIRST(1, (S))$
$(S)R)R$	$(b-a))$	Выброс
$S)R)R$	$b-a))$	Свертка $S\to TR$, т.к. $b\in FIRST(1, TR)$
$TR)R)R$	$b-a))$	Свертка $T\to b$, т.к. $b\in FIRST(1, b)$
$bR)R)R$	$b-a))$	Выброс
$R)R)R$	$-a))$	Свертка $R\to-TR$, т.к. $-\in FIRST(1, -TR)$
$-TR)R)R$	$-a))$	Выброс
$TR)R)R$	$a))$	Свертка $T\to a$, т.к. $a\in FIRST(1, a)$
$aR)R)R$	$a))$	Выброс
$R)R)R$	$))$	Свертка $R\to\varepsilon$, т.к. $) \in FOLLOW(1, R)$
$)R)R$	$))$	Выброс
$R)R$	$)$	Свертка $R\to\varepsilon$, т.к. $) \in FOLLOW(1, R)$
$)R$	$)$	Выброс
R	ε	Свертка $R\to\varepsilon$, т.к. $\varepsilon\in FOLLOW(1, R)$
ε	ε	Строка принята полностью

Шаг 6. Получили следующую цепочку вывода:

$S\Rightarrow TR\Rightarrow(S)R\Rightarrow(TR)R\Rightarrow(aR)R\Rightarrow(a+TR)R\Rightarrow(a+(S)R)R\Rightarrow(a+(TR)R)R\Rightarrow$
$\Rightarrow(a+(bR)R)R\Rightarrow(a+(b-TR)R)R\Rightarrow(a+(b-aR)R)R\Rightarrow(a+(b-a)R)R\Rightarrow(a+(b-a))R$
$\Rightarrow(a+(b-a))$.

Нисходящее дерево разбора цепочки представлено на рисунке 8.

Постановка задачи к практической работе № 6

При выполнении практической работы следует реализовать следующие действия:

1 построить множества *FIRST*(1, *A*) и *FOLLOW*(1, *A*) для каждого нетерминального символа грамматики;

2 проверить необходимое и достаточное условия *LL*(1) для КС-грамматики;

3 проиллюстрировать функционирование распознавателя для *LL*(1)-грамматик, составив набор контрольных примеров для случаев:

а) введенная КС-грамматика не является *LL*(1)-грамматикой;

б) исходная КС-грамматика является *LL*(1)-грамматикой, но входная строка не принадлежит языку грамматики;

в) заданная КС-грамматика является *LL*(1)-грамматикой, и введенная строка принадлежит языку грамматики.

Разбор цепочек показать с помощью таблицы, строки вывода и дерева вывода.

Вариантами индивидуальных заданий к лабораторной работе № 6 являются выходные данные лабораторной работы № 4.

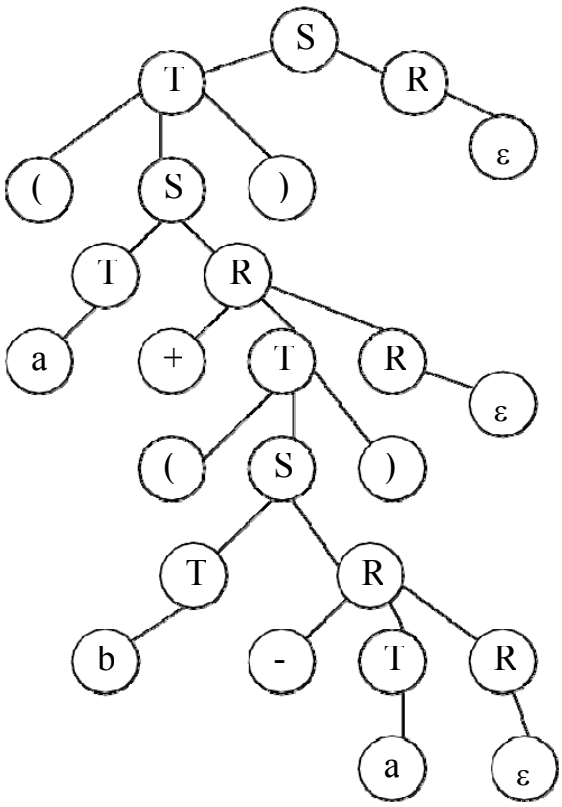


Рисунок 8 − Дерево вывода для цепочки (*a*+(*b*-*a*)) в грамматике *G*

Практическая работа № 7

Функционирование распознавателя для грамматик простого предшествования

Цели работы:

- закрепить понятие «грамматика простого предшествования»;
- сформировать умения и навыки построения множеств $L(A)$ и $R(A)$, матрицы предшествования символов грамматики и распознавателя для грамматик простого предшествования методом «сдвиг-свертка».

Основы теории

Определение 7.1. Приведенная КС-грамматика $G(V_N,V_T,P,S)$ называется грамматикой простого предшествования, если выполняются следующие условия.

1) для каждой упорядоченной пары терминальных и нетерминальных символов выполняется не более чем одно из трех отношений предшествования:

а) $B_i =\cdot B_j$ ($\forall B_i, B_j \in V$), если и только если существует правило $A\to xB_iB_jy \in P$, где $x, y \in V^*$;

б) $B_i<\cdot B_j$ ($\forall B_i, B_j \in V$), если и только если существует правило $A\to xB_iDy \in P$ и вывод $D\Rightarrow^* B_jz$, где $A, D \in V_N$, $x, y, z \in V^*$;

в) $B_i\cdot>B_j$ ($\forall B_i, B_j \in V$), если и только если существует правило $A\to xCB_jy$ и вывод $C\Rightarrow^* zB_i$ или существует правило $A\to xCDy \in P$ и вывод $C\Rightarrow^* zB_i$ и $D\Rightarrow^* B_jw$, где A, C, $D\in V_N$, $x, y, z, w\in V^*$.

2) различные правила в грамматике имеют разные правые части.

Определение 7.2. Отношения $=\cdot$, $<\cdot$, $\cdot>$ называют отношениями простого предшествования для символов грамматики.

В основе распознавателя для грамматик простого предшествования лежит правосторонний разбор строки языка. Исходной сентенциальной формой является заданная строка языка, а целевой – начальный символ грамматики. На каждом шаге разбора в исходной цепочке символов пытаются выделить подцепочку, совпадающую с правой частью некоторого правила вывода грамматики, и заменить ее не терминалом, стоящим в левой части этого правила. Данная операция называется сверткой к нетерминалу, а заменяемая подстрока – основой сентенции. Описанный процесс разбора соответствует построению дерева вывода цепочки снизу вверх (от листьев к корню).

Метод предшествования основан на том факте, что отношения между двумя соседними символами распознаваемой строки соответствуют трем следующим вариантам:

- $B_i =\cdot B_{i+1}$, если символы B_i и B_{i+1} принадлежат основе;
- $B_i<\cdot B_{i+1}$, если B_{i+1} – крайний левый символ некоторой основы;
- $B_i\cdot>B_{i+1}$, если B_i – крайний правый символ некоторой основы.

Алгоритм 7.1. Поиск основы сентенции грамматики

Если грамматика является грамматикой простого предшествования, то для поиска основы каждой ее сентенции надо просматривать элементы сентенции слева направо и найти самую левую пару символов x_j и x_{j+1}, такую что $x_j \cdot > x_{j+1}$. Окончанием основы сентенции будет x_j. Далее просматривать элементы сентенции справа налево, начиная с символа x_j до тех пор, пока не будет найдена самая правая пара символов x_{i-1} и x_i, такая что $x_{i-1} <\cdot x_i$. Заголовком основы будет символ x_i. Таким образом, будет найдена основа сентенции, имеющая вид $x_i x_{i+1} \ldots x_{j-1} x_j$. Схема поиска основы сентенции грамматики представлена на рисунке 9.

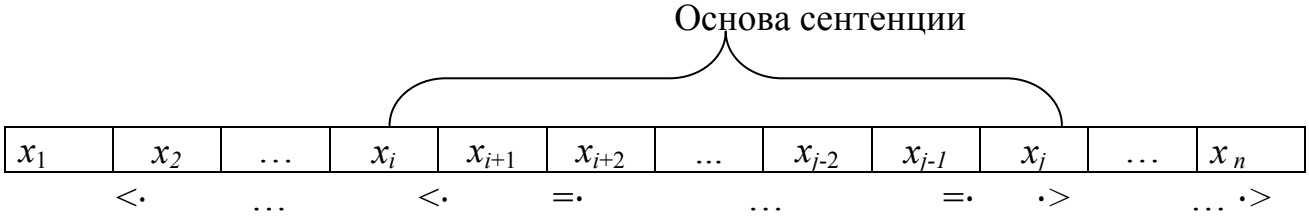


Рисунок 9 − Схема поиска основы сентенции грамматики

На основе отношений предшествования строят матрицу предшествования грамматики. Строки и столбцы матрицы предшествования помечаются символами грамматики. Пустые клетки матрицы указывают на то, что данные символы не связаны отношением предшествования.

Определение 7.3. Построение матрицы предшествования основано на двух вспомогательных множествах, определяемых следующим образом:

- $L(A) = \{X \mid \exists A \Rightarrow^* Xz\}$, $A \in V_N$, $X \in V$, $z \in V^*$ − множество крайних левых символов относительно нетерминального символа A;
- $R(A) = \{X \mid \exists A \Rightarrow^* zX\}$, $A \in V_N$, $X \in V$, $z \in V^*$ − множество крайних правых символов относительно нетерминального символа A.

Определение 7.4. Отношения предшествования можно определить с помощью введенных множеств следующим образом:

- $B_i = \cdot B_j$ ($\forall B_i$, $B_j \in V$), если и только если существует правило $A \to xB_iB_jy \in P$, где $A \in V_N$, x, $y \in V^*$;
- $B_i <\cdot B_j$ ($\forall B_i$, $B_j \in V$), если и только если существует правило $A \to xB_iDy \in P$ и $B_j \in L(D)$, где A, $D \in V_N$, x, $y \in V^*$;
- $B_i \cdot > B_j$ ($\forall B_{i,}$ $B_j \in V$), если и только если существует правило $A \to xCB_jy$ и $B_i \in R(C)$ или существует правило $A \to xCDy \in P$ и $B_i \in R(C)$, $B_j \in L(D)$, где A, C, $D \in V_N$, x, $y \in V^*$.

Матрицу предшествования дополняют символами $\perp_н$ и $\perp_к$ (начало и конец цепочки). Для них определены следующие отношения предшествования:

- $\perp_н <\cdot X$, $\forall X \in V$, если $X \in L(S)$;
- $\perp_к \cdot > X$, $\forall X \in V$, если $X \in R(S)$.

Алгоритм 7.2. Построение множеств $L(A)$ и $R(A)$

Шаг 1. Для каждого нетерминального символа A ищем все правила, содержание A в левой части. Во множество $L(A)$ включаем самый левый символ из правой части правил, а во множество $R(A)$ – самый крайний правый символ из правой части, т.е.

$$\forall A \in V_N: L_0(A) = \{X \mid A \to Xy, X \in V, y \in V^*\}, R_0(A) = \{X \mid A \to yX, X \in V, y \in V^*\}.$$

Шаг 2. Для каждого нетерминального символа A: если множество $L(A)$ содержит нетерминальные символы грамматики A', A'', …, то множество $L(A)$ надо дополнить символами, входящими в соответствующие множества $L(A')$, $L(A'')$ и т.д., и не входящими в $L(A)$. Аналогичную операцию выполнить для множеств $R(A)$, т.е.

$$\forall A \in V_N: L_i(A) = L_{i-1}(A) \cup L_{i-1}(B), \forall B \in (L_{i-1}(A) \cap V_N),$$
$$R_i(A) = R_{i-1}(A) \cup R_{i-1}(B), \forall B \in (R_{i-1}(A) \cap V_N).$$

Шаг 3. Если на предыдущем шаге хотя бы одно множество $L(A)$ или $R(A)$ для некоторого символа грамматики изменилось, то вернуться к шагу 2, иначе построение закончено. Т.е. если существует $A \in V_N$: $R_i(A) \neq R_{i-1}(A)$ или $L_i(A) \neq L_{i-1}(A)$, то положить $i := i+1$ и вернуться к шагу 2, иначе построение закончено и $R(A) = R_i(A)$ и $L(A) = L_i(A)$.

Алгоритм 7.3. Функционирование распознавателя для грамматик простого предшествования

Шаг 1. Поместить в верхушку стека символ $\perp_н$, считывающую головку – в начало входной цепочки символов.

Шаг 2. До тех пор, пока не будет обнаружена ошибка либо успешно завершен алгоритм разбора, сравниваем отношение простого предшествования символа на верхушке стека и очередного символа входной строки. При этом возможны следующие ситуации:

- если самый верхний символ стека имеет меньшее или равное предшествование, чем очередной символ входной строки, то производим операцию «сдвиг» (перенос текущего символа из входной цепочки в стек и перемещение считывающей головки на один символ вправо);

- если самый верхний символ стека имеет большее предшествование, чем очередной символ входной строки, то выполняем операцию «свертка». Для этого находим на верхушке стека «основу» сентенции, т.е. все символы, имеющие равное предшествование, или один символ на верхушке стека. Символы основы удаляем из стека, выбираем правило вывода грамматики, имеющее правую часть, совпадающую с основой, и помещаем в стек левую часть выбранного правила. Если такого правила вывода найти не удалось, то выдается сообщение об ошибке, и разбор завершен неудачно;

- если не установлено ни одно отношение предшествования между текущим символом входной цепочки и самым верхним символом в стеке, то алгоритм прерывается сообщением об ошибке;

- если в стеке остаются символы $\perp_н S$, а во входном буфере только символ $\perp_к$, то входная строка прочитана полностью, и алгоритм разбора завершен успешно.

Пример 7.1. Дана грамматика $G(\{a, (,)\}, \{S,R\},P,S)$, с правилами P: 1) $S\rightarrow(R \mid a$; 2) $R\rightarrow Sa)$. Построить распознаватель для строки $(((aa)a)a)\perp_к$.

Шаг 1. Построим множества крайних левых и крайних правых символов $L(A)$ и $R(A)$ относительно всех нетерминальных символов грамматики (таблица 16).

Таблица 16 − Построение множеств $L(A)$ и $R(A)$ для грамматики G

Шаг	$L_i(A)$	$R_i(A)$
0	$L_0(S)=\{(, a\}$ $L_0(R)=\{S\}$	$R_0(S)=\{R, a\}$ $R_0(R)=\{)\}$
1	$L_1(S)=\{(, a\}$ $L_1(R)=\{S, (, a\}$	$R_1(S)=\{R, a,)\}$ $R_1(R)=\{)\}$
2	$L_2(S)=\{(, a\}$ $L_2(R)=\{S, (, a\}$	$R_2(S)=\{R, a,)\}$ $R_2(R)=\{)\}$
Результат	$L(S)=\{(, a\}$ $L(R)=\{S, (, a\}$	$R(S)=\{R, a,)\}$ $R(R)=\{)\}$

Шаг 2. На основе построенных множеств и правил вывода грамматики составим матрицу предшествования символов (таблица 17).

Поясним заполнение матрицы предшествования. В правиле грамматики $S\rightarrow(R$ символ (стоит слева от нетерминального символа R. Во множестве $L(R)$ входят символы S, (, a. Ставим знак $<\cdot$ в клетках матрицы, соответствующих этим символам, в строке для символа (.

В правиле грамматики $R\rightarrow Sa)$ символ a стоит справа от нетерминального символа S. Во множество $R(S)$ входят символы R, a,). Ставим знак $>$ в клетках матрицы, соответствующих этим символам, в столбце для символа a.

В строке символа $\perp_н$ ставим знак $<\cdot$в клетках символов, входящих во множество $L(S)$, т.е. символов (, a. В столбце символа $\perp_к$ ставим знак $\cdot>$ в клетках, входящих во множество $R(S)$, т.е. символов R, a,).

В клетках, соответствующих строке символа S и столбцу символа a, ставим знак $=\cdot$, т.к. существует правило $R\rightarrow Sa)$, в котором эти символы стоят рядом. По тем же соображениям ставим знак $=\cdot$ в клетках строки a и столбца), а также строки (и столбца R.

Таблица 17 − Матрица предшествования символов грамматики

Символы	S	R	a	(	)	$\perp_к$
S			$=\cdot$			
R			$\cdot>$			$\cdot>$
a			$\cdot>$		$=\cdot$	$\cdot>$
(	$<\cdot$	$=\cdot$	$<\cdot$	$<\cdot$		
)			$\cdot>$			$\cdot>$
$\perp_н$			$<\cdot$	$<\cdot$		

Шаг 3. Функционирование распознавателя для цепочки (((aa)a)a) показано в таблице 18.

Таблица 18 − Алгоритм работы распознавателя цепочки (((*aa*)*a*)*a*)

Шаг	Стек	Входной буфер	Действие
1	$\perp_н$	$(((aa)a)a)\perp_к$	Сдвиг
2	$\perp_н($	$((aa)a)a)\perp_к$	Сдвиг
3	$\perp_н(($	$(aa)a)a)\perp_к$	Сдвиг
4	$\perp_н((($	$aa)a)a)\perp_к$	Сдвиг
5	$\perp_н(((a$	$a)a)a)\perp_к$	Свертка $S\to a$
6	$\perp_н(((S$	$a)a)a)\perp_к$	Сдвиг
7	$\perp_н(((Sa$	$)a)a)\perp_к$	Сдвиг
8	$\perp_н(((Sa)$	$a)a)\perp_к$	Свертка $R\to Sa)$
9	$\perp_н(((R$	$a)a)\perp_к$	Свертка $S\to(R$
10	$\perp_н((S$	$a)a)\perp_к$	Сдвиг
11	$\perp_н((Sa$	$)a)\perp_к$	Сдвиг
12	$\perp_н((Sa)$	$a)\perp_к$	Свертка $R\to Sa)$
13	$\perp_н((R$	$a)\perp_к$	Свертка $S\to(R$
14	$\perp_н(S$	$a)\perp_к$	Сдвиг
15	$\perp_н(Sa$	$)\perp_к$	Сдвиг
16	$\perp_н(Sa)$	$\perp_к$	Свертка $R\to Sa)$
17	$\perp_н(R$	$\perp_к$	Свертка $S\to(R$
18	$\perp_н S$	$\perp_к$	Строка принята

Шаг 4. Получили следующую цепочку вывода:
$S\Rightarrow(R\Rightarrow(Sa)\Rightarrow((Ra)\Rightarrow((Sa)a)\Rightarrow(((Ra)a)\Rightarrow(((Sa)a)a)\Rightarrow(((aa)a)a)$.

Восходящее дерево вывода цепочки представлено на рисунке 10.

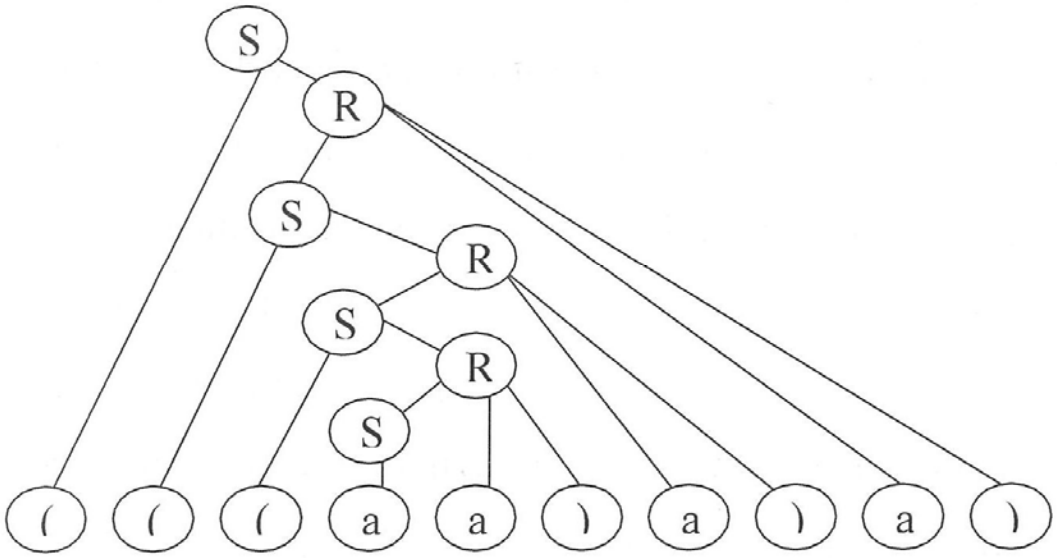


Рисунок 10 − Дерево вывода для цепочки (((*aa*)*a*)*a*) в грамматике *G*

Постановка задачи к практической работе № 7

При выполнении практической работы следует реализовать следующие действия:

1) построить множества $L(A)$ и $R(A)$ для каждого нетерминального символа грамматики;

2) сформировать матрицы простого предшествования для данной грамматики;

3) проверить условия простого предшествования для данной грамматики;

4) проиллюстрировать функционирование распознавателя для грамматик простого предшествования, составив набор контрольных примеров для случаев:

а) введенная грамматика не является грамматикой простого предшествования;

б) исходная грамматика является грамматикой простого предшествования, но анализируемая строка не принадлежит языку грамматики;

в) заданная грамматика является грамматикой простого предшествования, и входная строка принадлежит языку грамматики.

Разбор цепочек представить в виде таблицы, строки вывода и дерева вывода.

Вариантами индивидуального задания к лабораторной работе № 7 являются выходные данные лабораторной работы № 4.

Практическая работа № 8

Построение машины Тьюринга

Цели работы:

- закрепить знания о машине Тьюринга (МТ);
- сформировать умения и навыки построения МТ в различных видах.

Основы теории

Машина Тьюринга представляет собой автомат, имеющий бесконечную в обе стороны ленту, считывающую головку и управляющее устройство. Управляющее устройство может находиться в одном из состояний, образующих конечное множество $Q = \{q_0, q_1, ..., q_n\}$. Множество Q называют внутренним алфавитом машины Тьюринга.

Принципиальное отличие машины Тьюринга от вычислительных машин состоит в том, что ее запоминающее устройство представляет собой бесконечную ленту, из-за которой невозможна ее физическая реализация. Лента разделена на ячейки, в каждой из которых может быть записан один из символов конечного алфавита $A = \{a_0, a_1, . . . ,a_m\}$, который называют входным алфавитом машины Тьюринга. Во время функционирования машины Тьюринга может быть заполнено конечное число ячеек. Считывающая головка в каждый момент времени обозревает ячейку ленты, в зависимости от символа в этой ячейке и состояния управляющего устройства записывает в ячейку новый символ или оставляет его без изменения, сдвигается на ячейку влево или вправо или остается на месте. При этом управляющее устройство переходит в новое состояние или остается в старом. Среди состояний управляющего устройства выделены начальное состояние q_0 и заключительное состояние q_z.

Таким образом, за один такт работы машина Тьюринга может считать символ, записать вместо него новый или оставить его без изменения и сдвинуть головку на одну ячейку влево или вправо или оставить ее на месте.

Формальное определение машины Тьюринга

Машиной Тьюринга называется семерка вида:

$T = (Q, A, \delta, p_0, p_z, a_0, a_1)$, где

Q – конечное множество состояний, в которых может находиться управляющее устройство;

A – входной алфавит;

δ – функция переходов, $\delta = Q \cdot A \rightarrow Q \cdot A \cdot S$, где

$S = \{R, L, E\}$ − направления сдвига;

p_0– начальное состояние, $p_0 \in Q$;

p_z– заключительное состояние, $p_z \in Q$;

a_0 – символ для обозначения пустой ячейки, $a_0 \in A$;

a_1 – специальный символ-разделитель цепочек на ленте, $a_1 \in A$.

Командой машины Тьюринга называется элемент функции переходов $qa \rightarrow pbr$, где $q, p \in Q$; $a, b \in A$; $r \in S$. Каждая команда описывает один такт работы

машины Тьюринга.

Конфигурация машины Тьюринга представляется следующим образом: t = <CqaB>, где

C − цепочка, находящаяся слева от считывающей головки;

q − состояние машины Тьюринга;

a − символ, находящийся под головкой машины Тьюринга;

B − цепочка, находящаяся справа от головки машины Тьюринга.

Конфигурация <CqaB> непосредственно переходит в конфигурацию $<C_nq_na_nB_n>$, если новая конфигурация получена в результате применения одной команды к исходной конфигурации.

Обозначим непосредственный переход из одной конфигурации в другую следующим образом: $<CqaB>\Rightarrow<C_nq_na_nB_n>$.

Конфигурация, содержащая начальное состояние, называется начальной, а содержащая заключительное состояние − заключительной. Если цепочка C в конфигурации пустая, то начальная и заключительная конфигурации называются стандартными.

Машина Тьюринга перерабатывает цепочку x в цепочку y, если, действуя из начальной конфигурации и имея на ленте цепочку x, машина Тьюринга переходит в заключительную конфигурацию, имея на ленте цепочку y. Если начальная и заключительная конфигурации являются стандартными, то процесс переработки x в y является правильной переработкой.

Способы представления машины Тьюринга

Существует три способа представления машины Тьюринга: совокупностью команд, в виде графа, в виде таблицы соответствия.

Представление машины Тьюринга совокупностью команд

Совокупность всех команд, которые может выполнять машина, называется ее программой.

Машина Тьюринга считается заданной, если заданы ее внешний и внутренний алфавиты, программа, начальная конфигурация и указано, какие из символов обозначают пустую ячейку и заключительное состояние.

Чтобы записать совокупность команд, нужно воспользоваться следующими правилами:

1) начальному шагу алгоритма ставится в соответствие q_0 – начальное состояние;

2) циклы реализуются так, что последнее действие цикла должно соответствовать переходу в то состояние, которое соответствует началу цикла;

3) соседним шагам алгоритма соответствует переход в смежные состояния, которые соответствуют этим пунктам;

4) последний шаг алгоритма вызывает переход в заключительное состояние.

В качестве примера рассмотрим совокупность команд машины Тьюринга, которая инвертирует входную цепочку, записанную с использованием нулей и

единиц.

Пусть алфавит машины Тьюринга задан множеством A={0, 1, ε}, где символ ε соответствует пустой ячейке, а число состояний устройства управления задано в виде множества Q = {q_0, q_1,q_z}.

Если, например, начальная конфигурация имеет вид $q_0$110011, то конечная конфигурация после завершения операции инвертирования должна иметь вид q_z001100. Для решения задачи машиной будет порождена следующая последовательность команд:

$q_0 1 \rightarrow q_0 0R$, $q_1 0 \rightarrow q_1 0L$,
$q_0 0 \rightarrow q_0 1R$, $q_1 1 \rightarrow q_1 1L$,
$q_0 \varepsilon \rightarrow q_1 \varepsilon L$, $q_1 \varepsilon \rightarrow q_z \varepsilon R$.

В стандартной начальной конфигурации головка стоит над первым символом слева, и устройство управления находится в начальном состоянии. На следующем такте машина Тьюринга, не меняя своего состояния, заменяет символ 0 на 1 или 1 на 0 и сдвигается вправо на один символ. После просмотра всей цепочки под головкой оказывается символ, указывающий на пустую ячейку. В этом случае машина Тьюринга переходит в новое состояние и сдвигается влево на один символ. На последующих тактах управляющее устройство не меняет своего состояния, оставляет без изменения символ под головкой и перемещается влево до тех пор, пока не встретит пустую ячейку. Встретив пустую ячейку, машина Тьюринга переходит в заключительное состояние и перемещается вправо на один символ, переходя в стандартную заключительную конфигурацию.

Представление машины Тьюринга графом

При представлении машины Тьюринга посредством графа необходимо каждому состоянию поставить в соответствие вершину графа, а каждой команде – помеченную дугу.

Машина Тьюринга из рассмотренного примера инвертирования цепочки, состоящей из символов 0 и 1, будет представлена в виде графа на рисунке 11.

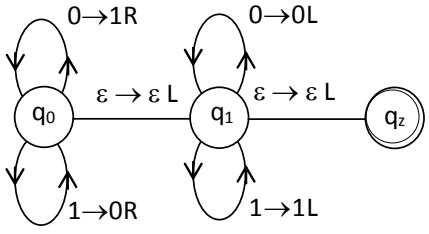


Рисунок 11 − Инвертирование цепочки

Конечная вершина графа на рисунке отмечена двумя концентрическими кружками. Если на очередном такте работы машины Тьюринга символ на ленте не изменяется, то в правой части команды его можно не писать (ε на ребре в состоянии q_z).

Представление машины Тьюринга таблицей соответствия

При представлении машины Тьюринга данным способом составляется таблица, в которой каждому состоянию соответствует строка, а каждому символу из входного алфавита − столбец. В клетках таблицы на пересечении строки и столбца будет находиться действие (или правая часть команды).

Таблица соответствия для задания машины Тьюринга из рассмотренного примера представлена в таблице 19.

Таблица 19 − Таблица соответствия для задания машины Тьюринга

	0	1	ε
q_0	q_0 1R	Q_0 0R	q_1ε L
q_1	q_1 0L	Q_1 1L	q_zεL

Инвертирование входной цепочки можно выполнить программой машины Тьюринга, приведенной в таблице соответствия. Эта программа включает шесть команд.

Операции над машинами Тьюринга

1 *Композиция машин Тьюринга.* Пусть машины T_1 и T_2 имеют программы P_1 и P_2. Предположим, что внутренние алфавиты этих машин не пересекаются; пусть q_{z1} − заключительное состояние машины T_1, а q_{02} − начальное состояние машины T_2. Заменим всюду в программе P_1 заключительное состояние q_{z1} на начальное состояние q_{02} машины T_2 и полученную программу объединим с программой P_2. Новая программа P определяет машину T, называемую композицией машин T_1 и T_2 по паре состояний (q_{z1}, q_{02}). Композиция машин может быть обозначена $T_1 \cdot T_2$ или T_1T_2. Более подробно композиция машин записывается следующим образом:

$T = T(T_1, T_2, (q_{z1}, q_{02}))$, где

$T_1 = (Q_1, A_1, \delta_1, p_{01}, p_{z1}, a_{01}, a_{11})$,
$T_2 = (Q_2, A_2, \delta_2, p_{02}, p_{z2}, a_{02}, a_{12})$.

Пусть $a_{01} = a_{02} = a_0$ и $a_{11} = a_{12} = a_1$. Внешний алфавит композиции T_1T_2 является объединением внешних алфавитов машин T_1 и T_2:

$$T = (Q_1 \cup Q_2 \setminus \{p_{z1}\}, A_1 \cup A_2, \left|\begin{matrix}(\delta_1, \cup \delta_2), p_{01}, p_{z2}, a_0, a_1).\\ p_{z1}\end{matrix}\right.$$

Операция композиции, выполняемая над алгоритмами, позволяет получать новые, более сложные алгоритмы из ранее известных простых алгоритмов.

2 *Итерация машины Тьюринга.* Эта операция применима только к одной машине. Пусть q_z − заключительное состояние машины T, а q_n − какое-либо состояние машины T, не являющееся заключительным. Заменим всюду в программе P машины T состояние q_z на q_n . Полученная программа определяет но-

вую машину $T'(q_z, q_n)$, которая называется итерацией машины T по паре состояний (q_z, q_n). Если машина Тьюринга имеет одно заключительное состояние, то после выполнения итерации получается машина, не имеющая заключительного состояния.

3 *Разветвление машин Тьюринга*. Пусть машины Тьюринга T_1, T_2 и T_3 задаются программами P_1, P_2 и P_3, соответственно. Считаем, что внутренние алфавиты этих машин попарно не пересекаются. Пусть q_{z11} и q_{z12} − какие-либо различные заключительные состояния машины T_1. Заменим всюду в программе P_1 состояние q_{z11} начальным состоянием q_{02} машины T_2, а состояние q_{z12} начальным состоянием q_{03} машины T_3. Затем новую программу объединим с программами P_2 и P_3. Получим программу P, задающую машину Тьюринга и обозначаемую:

$$T = T(T_1, (q_{z11}, q_{02}), T_2, (q_{z12}, q_{03}), T_3).$$

Машина T называется разветвлением машин T_2 и T_3, управляемым машиной T_1.

Примеры построения машины Тьюринга

Пример 8.1. Построить машину Тьюринга, которая правильно вычисляет функцию $f(x) = x+1$ по правилам двоичного сложения.

Решение. Исходя из формулировки задачи, требующей вычислить функцию по правилам сложения в двоичной системе сложения, выберем входной алфавит:

$$A = \{0, 1, \varepsilon\}.$$

Таблица 20 − Представление машины Тьюринга в виде таблицы

	0	1	ε
q_0 q_0	q_0 0R	q_0 1R	q_1ε L
q_1 q_1	q_2 1L	q_1 0L	q_2 1L
q_2 q_2	q_2 0L	q_2 1L	q_zε R

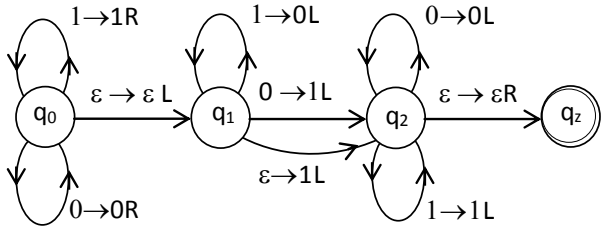


Рисунок 12 − Представление машины Тьюринга в виде графа

Запишем программу построенной машины Тьюринга для случая, когда входная цепочка на ленте равна двоичному числу 111. Справа от каждой команды приведем представление входной цепочки на ленте до выполнения дан-

ной команды. Символ, который находится под головкой, будем помечать подчеркиванием.

1) $q_0 1 \to q_0 1R$	$\varepsilon\underline{1}11\varepsilon$	6) $q_1 1 \to q_1 0L$	$\varepsilon 1\underline{1}0\varepsilon$
2) $q_0 1 \to q_0 1R$	$\varepsilon 1\underline{1}1\varepsilon$	7) $q_1 1 \to q_1 0L$	$\varepsilon\underline{1}00\varepsilon$
3) $q_0 1 \to q_0 1R$	$\varepsilon 11\underline{1}\varepsilon$	8) $q_1\varepsilon \to q_2 1L$	$\underline{\varepsilon}000\varepsilon$
4) $q_0\varepsilon \to q_1\varepsilon L$	$\varepsilon 111\underline{\varepsilon}$	9) $q_2\varepsilon \to q_z\varepsilon R$	$\underline{\varepsilon}1000\varepsilon$
5) $q_1 1 \to q_1 0L$	$\varepsilon 11\underline{1}\varepsilon$		

Из примера видно, что машина Тьюринга из стандартной начальной конфигурации, имея на ленте аргумент 111, выполнив совокупность команд 1-9, перешла в стандартное заключительное состояние, имея на ленте результат 1000. Действительно:

$111_2 + 1_2 = 1000_2$.

Пример 8.2. Построить машину Тьюринга, выполняющую операцию (x div 2) и имеющую входной алфавит A = {0, 1, ε}.

Таблица соответствия данной машины Тьюринга будет иметь следующий вид:

	0	1	ε		0	1	ε
q_0	q_0 0R	q_0 1R	$q_1\varepsilon$ L	q_2	q_3 1L	q_2 0L	q_3 1L
q_1	$q_3\varepsilon$ L	$q_2\varepsilon$ L		q_3	q_2 0L	q_2 1L	$q_z\varepsilon$ R

Операция (x div 2) реализована сдвигом цепочки вправо на 1 разряд.

Пример 8.3. Построить машину Тьюринга, которая выполняет копирование заданного аргумента. Выберем входной алфавит A={0,1,ε}. Представим данную машину таблицей соответствия:

	0	1	*	ε
q_0	$q_1$1L	$q_1$0R	q_0*L	$q_z\varepsilon$R
q_1		$q_1$1R	q_2*R	q_2*R
q_2		$q_2$1R		$q_3$1L
q_3	$q_0$0R	$q_3$1L	q_3*L	

Запишем программу построенной машины для заданной входной цепочки на ленте, равной двоичному числу 111. Справа от каждой команды приведем представление входной цепочки на ленте до выполнения данной команды. Символ, который находится под головкой, будем помечать подчеркиванием.

1) $q_0 1 \to q_1 0R$	$\varepsilon\underline{1}11\varepsilon$
2) $q_1 1 \to q_1 1R$	$\varepsilon 0\underline{1}1\varepsilon$
3) $q_1 1 \to q_1 1R$	$\varepsilon 01\underline{1}\varepsilon$
17) $q_3 1 \to q_3 1L$	$\varepsilon 00\underline{1}*11\varepsilon$
18) $q_3 0 \to q_0 0R$	$\varepsilon 0\underline{0}1*11\varepsilon$
19) $q_0 1 \to q_1 0R$	$\varepsilon 00\underline{1}*11\varepsilon$

4) $q_1\varepsilon \rightarrow q_2$*R ε011$\underline{\varepsilon}$
5) $q_2\varepsilon \rightarrow q_3$1L ε011*$\underline{\varepsilon}$
6) q_3* → q_3*L ε011$\underline{*}$1ε
7) $q_3$1 → $q_3$1L ε01$\underline{1}$*1ε
8) $q_3$1 → $q_3$1L ε0$\underline{1}$1*1ε
9) $q_3$0 → $q_0$0R ε$\underline{0}$11*1ε
10) $q_0$1 → $q_1$0R ε0$\underline{1}$1*1ε
11) $q_1$1 → $q_1$1R ε00$\underline{1}$*1ε
12) q_1* → q_2*R ε001$\underline{*}$1ε
13) $q_2$1 → $q_2$1R ε001*$\underline{1}$ε
14) $q_2\varepsilon \rightarrow q_3$1L ε001*1$\underline{\varepsilon}$
15) $q_3$1→$q_3$1L ε001*$\underline{1}$1ε
16) q_3*→q_3*L ε001$\underline{*}$11ε
20) q_1* → q_2*R ε000$\underline{*}$11ε
21) $q_2$1 → $q_2$1R ε000*$\underline{1}$1ε
22) $q_2$1 → $q_2$1R ε000*1$\underline{1}$ε
23) $q_2\varepsilon \rightarrow q_3$1L ε000*11$\underline{\varepsilon}$
24) $q_3$1 → $q_3$1L ε000*1$\underline{1}$1ε
25) $q_3$1 → $q_3$1L ε000*$\underline{1}$11ε
26) q_3* → q_3*L ε000$\underline{*}$111ε
27) $q_3$0→ $q_0$0R ε00$\underline{0}$*111ε
28) q_0* → q_0*L ε000$\underline{*}$111ε
29) $q_0$0→ $q_0$1L ε00$\underline{0}$*111ε
30) $q_0$0→$q_0$1L ε0$\underline{0}$1*111ε
31) $q_0$0→$q_0$1L ε$\underline{0}$11*111ε
32) $q_0\varepsilon \rightarrow q_z$1R $\underline{\varepsilon}$111*111ε

Из примера видно, что машина Тьюринга из стандартной начальной конфигурации, имея на ленте аргумент 111 и выполнив совокупность команд 1-32, перешла в стандартное заключительное состояние, имея на ленте результат $\underline{\varepsilon}$111*111ε.

Постановка задачи к практической работе № 8

1 Построить машину Тьюринга, выполняющую операцию конкатенации двух цепочек, заданных во входном алфавите A = {0, 1, *, ε}.

2 Построить машину Тьюринга, выполняющую операцию копирования входной цепочки, заданной в алфавите A = {1, *, ε}, где символ * используется в качестве разделителя двух цепочек.

3 Построить машину Тьюринга в алфавите A={0, 1}, которая, начав работу с последней единицы массива из единиц, сдвигает его на одну ячейку влево, не изменяя остального содержимого ленты. Головка останавливается на первой единице перенесенного массива.

4 По заданной совокупности команд машины Тьюринга T и начальной конфигурации K найти заключительную конфигурацию.

4.1 $q_0$1→$q_0$1R, $q_1$0 →q_z1E,
$q_0$0→$q_1$1R, $q_1$1 →$q_1$1L,
а) K=1101$q_0$01; б) K = 101$q_0$010.

4.2 $q_0$0→$q_0$1L, $q_1$1 →$q_0$0R,
$q_0$1→$q_1$1R, $q_1$0 →q_z0L,
а) K =1$q_1$0111; б) K = 1$q_1$1111.

4.3 $q_0$0→$q_1$0L, $q_1$0 →$q_1$1L,
$q_0$1→$q_0$0R, $q_1$1 →q_z0R,
а) K =1000$q_1$01; б) K =11$q_1$11101.

5 Построить машину Тьюринга, которая во входной цепочке, заданной в алфавите A={0, 1, ε}, переставляет единицы и нули так, чтобы все единицы были в начале, а нули в конце цепочки.

6 Построить композицию $T_1 \cdot T_2$ машин Тьюринга T_1 и T_2 по паре состояний (q_{1z}, q_{20}) и найти результат применения композиции $T_1 \cdot T_2$ к слову D.

6.1 Машины T_1 и T_2 заданы таблицами соответствия:

T_1:

	q10	q11	q12
0	q_{1z}0L	q_{12}0R	q_{10}0R
1	q_{11}1R	q_{12}1R	q_{10}0R

T_2:

	q20	q21
0	q_{21}1L	q_{2z}0R
1	q_{21}1L	q_{20}0L

а) D=111100111011; б) D =11010111.

6.2 Машины T_1 и T_2 заданы таблицами соответствия:

T_1

	q10	q11	q12
0	q_{11}0R	q_{12}0R	q_{1z}1L
1	q_{10}1R	q_{10}1R	

T_2

	q20	q21	q22
0	q_{21}0L	q_{22}0L	q_{2z}0R
1	q_{20}1L	q_{21}1L	q_{22}1L

а) D=11000101001; б) D =10100111110.

7 Найти результат применения итерации машины T по паре состояний (q_{z1}, q_i) к слову D. Заключительными состояниями машины T являются q_{z1} и q_{z2}. На ленте первоначально записаны нули, а в начальной конфигурации головка указывает на левый символ входной цепочки, состоящей из единиц.

7.1 Машина Тьюринга задана таблицей соответствия: T:

	q_0	q_1	q_2	q_3	q_4
0	q_{z1}0E	$q_3$0E	$q_4$0E	$q_4$1R	q_{z2}1L
1	$q_1$0R	$q_2$0R	$q_0$0R		

а) i = 1, $D=1^{3k}$, k≥ 1; б) i = 1, $D = 1^{3k+2}$, k ≥1.

7.2 Машина Тьюринга задана таблицей соответствия: T:

	q_0	q_1	q_2	q_3	q_4	q_5
0	$q_{z2}0R$	$q_{z2}0R$	$q_3 0R$	$q_4 1L$	$q_5 0L$	$q_{z1}0R$
1	$q_1 0R$	$q_2 0R$	$q_2 1R$	$q_1 0E$	$q_4 1L$	$q_5 1L$

а) i = 3, D = 1^{2k+1}, k≥1; б) i = 1, D = 1^{3k+2}, k≥1.

Практическая работа № 9

Автоматы Мили и Мура

Цели работы:

- закрепить понятия теории абстрактных автоматов, «автомат Мили», «автомат Мура»;
- сформировать умения и навыки построения автоматов в различных видах.

Основы теории

Автомат − это алгоритм, определяющий некоторое множество и, возможно, преобразующий его в другое множество.

Существует два типа автоматов:

1) распознаватели − автоматы без выхода, которые распознают, принадлежит ли входная цепочка заданному множеству L;
2) преобразователи − автоматы с выходом, которые преобразуют входную цепочку x в цепочку y при условии, что $x \in L$.

Абстрактный автомат – это математическая модель, описывающая дискретное устройство совокупностью входных, выходных сигналов и состояний.

Формальное определение автомата

Неинициальный автомат - это пятерка вида $A = (Q, X, Y, \delta, \gamma)$, где

Q − множество состояний (алфавит состояний);
X − входной алфавит;
Y − выходной алфавит;
δ − функция переходов, задающая отображение $Q \cdot X \rightarrow Q$;
γ − функция выходов, задающая отображение $Q \cdot X \rightarrow Y$.

Автомат реализует некоторое отображение множества слов входного алфавита X во множество слов выходного алфавита Y.

Автомат называется конечным, если множество его внутренних состояний и множество значений входных сигналов есть конечные множества.

Автомат называется детерминированным, если его функция переходов δ однозначна.

Автомат называется синхронным, если интервал временной дискретизации постоянен, в противном случае автомат называется асинхронным.

Функционирование автомата можно задать множеством команд вида

$qx \rightarrow py$, где q и $p \in Q$, $x \in X$, $y \in Y$.

Пусть на некотором такте t_i управляющее устройство находится в состоянии q, а из входной ленты читается символ x. Если во множестве команд есть команда $qx \rightarrow py$, то на такте t_i на выходную ленту записывается символ y, а к следующему такту t_{i+1} управляющее устройство перейдет в состояние p, т.е.:

$y(t) = \gamma(q(t), x(t))$,
$q(t+1) = \delta(q(t), x(t))$, где $t = 0, 1, 2, ...$

Если же команда qx→py отсутствует, то автомат оказывается блокированным и не реагирует на символ, принятый в момент t_i, а также перестает воспринимать символы в последующие моменты времени.

В соответствии с определением неинициального автомата в начальный момент состояние автомата может быть произвольным.

Если зафиксировано некоторое начальное состояние, то такой автомат называют инициальным, т.е. $q(0) = q_0$.

Инициальный автомат − это шестерка вида $A = (Q, X, Y, \delta, \gamma, q_0)$, где

Q − множество состояний (алфавит состояний);

X – входной алфавит;

Y − выходной алфавит;

δ − функция переходов (отображение $Q \cdot X \to Q$);

γ − функция выходов (отображение $Q \cdot X \to У$);

q_0 – начальное состояние.

В зависимости от способа формирования выходного сигнала в синхронных автоматах различают:

1 Автомат первого рода (автомат Мили);

2 Автомат второго рода (автомат Мура).

Автоматы Мили и Мура отличаются лишь формированием функции выходов.

Автомат Мили

Автомат Мили − это пятерка вида $M = (Q, X, Y, \delta, \gamma)$, где

Q − множество состояний автомата;

X − входной алфавит;

Y − выходной алфавит;

δ − функция переходов - $q(t+1) = \delta(q(t), x(t))$;

γ − функция выходов - $y(t) = \gamma(q(t), x(t))$, где $t = 0, 1, 2, \ldots$

Как и любой другой автомат, автомат Мили можно представить в виде таблицы или графа. В графе переходов автомата Мили на дугах указываются через символ «/» входные и выходные символы. Таблица переходов состоит из двух частей: в левой части записываются значения функции выходов, в правой части − значения функции переходов.

Пример. Построим конечный преобразователь, который распознает арифметические выражения, порождаемые грамматикой

$$S \to a+S \mid a-S \mid +S \mid -S \mid a,$$

и устраняет из этих выражений избыточные унарные операции. Например, выражение –a+–a–+–a он переведет в –a–a+a. Во входном языке символ «a» представляет идентификатор, и перед идентификатором допускается произвольная последовательность знаков унарных операции + и –. Заметим, что входной язык является регулярным множеством.

Пусть M = (Q, X, Y, δ, γ), где

1 Q = {q_0, q_1, q_2, q_3, q_4},
2 X = {a,+, –},
3 Y = X.

Графическое представление автомата Мили показано на рисунке 13, где «/e» означает отсутствие выходного сигнала.

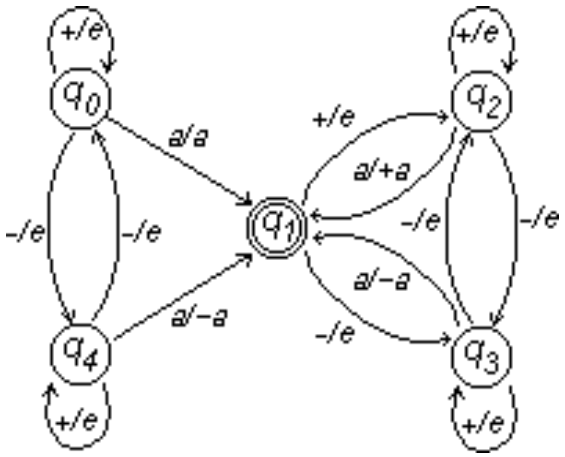


Рисунок 13 − Графическое представление преобразователя M.

Преобразователь M начинает работу в состоянии q_0 и, чередуя состояния q_0 и q_4 на входном символе «–», определяет, четное или нечетное число знаков «–» предшествует первому символу «a». Когда появляется «a», преобразователь M переходит в состояние q_1, допуская вход, и выдает «a» или «–a» в зависимости от того, четно или нечетно число появившихся минусов. Для следующих символов «a» он подсчитывает, четно или нечетно число предшествующих минусов, с помощью состояний q_2 и q_3. Единственное различие между парами q_2, q_3 и q_0, q_4 состоит в том, что если символу «a» предшествует четное число минусов, то первая из них выдает «+a», а не только «a».

Таблица переходов преобразователя выглядит следующим образом:

	Y			**K**		
K\X	**a**	+	–	**a**	+	–
q_0	a	ε	ε	q_1	q_0	q_4
q_1	-	ε	ε	-	q_2	q_3
q_2	+a	ε	ε	q_1	q_2	q_3
q_3	–a	ε	ε	q_1	q_3	q_2
q_4	–a	ε	ε	q_1	q_4	q_0

Автомат Мура

Автомат Мура − это также пятерка вида $U = (Q, X, Y, \delta, \gamma_1)$, где:
Q, X, Y, δ соответствуют определению автомата Мили;
γ_1 − функция выходов − $y(t) = \gamma_1(q(t))$,

$$y(t+1) = \gamma_1(q(t+1)) = \gamma_1(\delta(q(t), x(t))), \text{ где } t = 0, 1, 2, ...$$

При представлении автомата Мура графом дуги помечаются символами входного алфавита, а каждая вершина графа − состоянием и символом выходного алфавита.

При формальном сравнении определений автоматов Мили и Мура может показаться, что автомат Мура может быть задан как входонезависимый автомат Мили, т.е. такой автомат Мили, выходная функция которого удовлетворяет следующим условиям: $\forall a \in X, \forall b \in X, \forall q \in Q_1, (\gamma(q, a) = \gamma(q, b))$. Однако это не соответствует способу функционирования автоматов Мура в соответствии с введенным определением.

В автомате Мура реализована иная временная связь между переходами из одного состояния в другое и выходом, по сравнению с автоматом Мили, у которого выход, соответствующий некоторому входу и определенному состоянию, порождается во время перехода автомата в следующее состояние. Автомат Мили реагирует на входной сигнал без задержек, в пределах того же временного такта. У автомата Мура сначала порождается переход в следующее состояние, а потом выход, причем выход определяется только состоянием автомата.

Равносильность автоматов Мили и Мура

Реакцией автомата называется последовательность выходных сигналов автомата, полученная под воздействием некоторой последовательности входных сигналов, т.е. реакция − это выходное слова автомата на конкретное входное слово.

Равносильность автоматов Мили и Мура заключается в том, что множество реакций этих автоматов совпадает, только лишь с той разницей, что в автомате Мура выходной сигнал выдается не на переходе автомата из одного состояния в другое, как в автомате Мили, а с запаздыванием на один такт (после того, как автомат перешел в новое состояние).

Два автомата называются эквивалентными если:
- их входной и выходной алфавиты совпадают;
- их реакции из исходного состояния на любое входной слово совпадают.

Существует теорема: для любого автомата Мура существует эквивалентный автомат Мили и наоборот.

Постановка задачи к практической работе № 9

1 Построить конечный преобразователь, моделирующий работу светофора. Рассмотреть различные алгоритмы переключения светофора.

2 Построить автомат Мили, который читает текст, написанный на рус-

ском языке. Автомат должен считать все слова, начинающиеся с символа «б» и оканчивающиеся символом «т», т.е. такие, как «бит», «байт», «батут» и т.д. При построении автомата все буквы кроме «б» и «т» можно обозначить каким-либо символом, например, «|».

3 Построить автомат Мили, который читает программу, написанную на языке программирования высокого уровня. Автомат должен считать все целые константы.

Список литературы

1 Ахо А., Сети Р., Ульман Д. Компиляторы: принципы, технологии и инструменты пер. с англ. – Москва : Изд. дом «Вильямс», 2001. –768 с.

2 Бильгаева Н. Ц. Теория алгоритмов, формальных языков, грамматик и автоматов : учебное пособие. Улан-Удэ : Изд-во ВСГТУ, 2000. – 51 с.

3 Вайнгартен Ф. Трансляция языков программирования / под ред. В.В. Мартынюка − Москва : Мир, 1977. – 192 с.

4 Вильямс А. Системное программирование в Windows 2000 для профессионалов. – Санкт-Петербург : Питер, 2001. – 624 с.

5 Волкова И. А., Руденко Т. В. Формальные языки и грамматики. Элементы теории трансляции. – Москва : Диалог-МГУ, 1999. – 62 с.

6 Гордеев А. В., Молчанов А. Ю. Системное программное обеспечение. – Санкт-Петербург : Питер, 2001. – 736 с.

7 Грис Д. Конструирование компиляторов для цифровых вычислительных машин пер. с англ. – Москва : Мир, 1975. – 544 с.

8 Жильцова Л. П., Смирнова Т. Г. Основы теории автоматов и формальных языков в примерах и задачах. Часть 1: Учебно-методическое пособие. – Нижний Новгород : Нижегородский госуниверситет, 2010. – 65 с.

9 Ишакова Е. Н. Теория формальных языков, грамматик и автоматов: Методические указания к лабораторному практикуму. – Оренбург : ГОУ ОГУ, 2005. – 54 с.

10 Компаниец Р. И., Маньков Е. В., Филатов Н. Е. Системное программирование. Основы построения трансляторов. – Санкт-Петербург.: Корона принт, 2000. – 256 с.

11 Льюис Ф., Розенкранц Д., Стирнз Р. Теоретические основы проектирования компиляторов. – Москва : Мир, 1979. – 654 с.

Стукало Виктор Александрович

ТЕОРИЯ АВТОМАТОВ И ФОРМАЛЬНЫХ ЯЗЫКОВ

Методические указания
к выполнению практических работ для студентов
направлений подготовки 09.03.04, 09.04.04
«Программная инженерия»

Редактор Н.М. Быкова

Подписано к печати 12.03.19 Формат 60×84 1/16 Бумага 65 г/м^2
Печать цифровая Усл. печ. л. 3,6 Уч.- изд. л. 3,6
Заказ 51 Тираж 15 Не для продажи

Библиотечно-издательский центр КГУ.
640020, г. Курган, ул. Советская, 63/4.
Курганский государственный университет