Министерство образования и науки Украины
Севастопольский национальный технический университет

РЕГУЛЯРНЫЕ ГРАМАТИКИ И КОНЕЧНЫЕ АВТОМАТЫ

Методические указания
к выполнению лабораторной работы
для студентов, обучающихся по
направлению **6.050101**
“Компьютерные науки”
очной и заочной форм обучения

Севастополь
2014

УДК 510.5

Регулярных грамматики и конечные автоматы: методические указания / Сост. Е. Н. Заикина, Ю. В. Коваленко. — Севастополь: Изд-во СевНТУ, 2014 — 20 с.

Целью методических указаний является:

1) научиться производить построение детерминированных конечных автоматов (ДКА), допускающих определённые цепочки символов языка;

2) освоить приёмы описания конечных автоматов (КА) в виде графов, таблиц переходов и регулярных выражений;

3) научиться производить построение формальной автоматной граммати-ки, соответствующей конечному автомату и наоборот;

4) выполнять построения ДКА по недетерминированным конечным автоматам (НКА);

Методические указания составлены в соответствии с требованиями программы дисциплины «Теория алгоритмов и элементы математической логики» для студентов направления 6.050101 и утверждены на заседании кафедры Инфор-мационных систем, протокол № 1 от 29 августа 2014 года.

Допущено учебно-методическим центром СевНТУ в качестве методических указаний.

Рецензент: Скороход Б.А., доктор технических наук, проф. кафедры ТК

СОДЕРЖАНИЕ

1 ЦЕЛЬ РАБОТЫ

1) научиться производить построение детерминированных конечных автоматов (ДКА), допускающих определённые цепочки символов языка;

2) освоить приёмы описания конечных автоматов (КА) в виде графов, таблиц переходов и регулярных выражений;

3) научиться производить построение формальной автоматной грамматики, соответствующей конечному автомату и наоборот;

4) выполнять построения ДКА по недетерминированным конечным автоматам (НКА);

2 ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ

2.1 Грамматики

Алфавит (словарь) V – некоторое конечное непустое множество, его элементами являются ***символы***.

Цепочка над алфавитом (в алфавите) есть произвольная конечная последовательность символов.

Язык L – подмножество множества всевозможных цепочек, выделенное с помощью некоторого конечного множества правил.

Грамматика G – набор правил описания синтаксиса.

Порождающая грамматика – набор синтаксических правил, описывающий процедуру получения цепочек (выражений) языка.

Распознающая грамматика – набор синтаксических правил синтаксических, описывающих процедуру проверки корректности цепочек (выражений) языка.

Формальной грамматикой $G[S]$ (или просто грамматикой) называется [18] конечное непустое множество, задаваемое упорядоченной четвёркой

$$G[S] = (V_T, V_N, S, R),$$

в которой

V_T – словарь терминальных символов;

V_N – словарь нетерминальных символов;

S – нетерминальный символ, который должен появиться в левой части хотя бы одного правила (он называется аксиомой или помеченным символом);

R – конечное (счётное) множество правил грамматики (правил подстановки, продукций) вида

$$\alpha \to \beta$$

где $\alpha \in (V_N \cup V_T)^+$, $\beta \in (V_N \cup V_T)^*$, индексы означают “+” – отсутствуют пустые цепочки, “*” – есть пустые цепочки.

Объединение словарей терминалов и нетерминалов дает словарь грамматики $V = V_N \cup V_T$.

Заметим, что различные грамматики могут порождать одни те же языки.

Пример записи правил подстановки R грамматики $G[Z]$:

$\mathbf{Z} \to b\mathbf{A}|\mathbf{A}$

$\mathbf{A} \to a|a\mathbf{A}$

В записи обозначено: символ “$\rightarrow$” означает “определяется как”, “это есть” либо “заменяется на”; | – “или”, “(альтернатива)”, служит для сокращённой записи нескольких правил с одинаковыми правыми частями; $V_T = \{a, b\}$ – терминалы, словарь терминальных символов; $V_N = \{\text{Z, A}\}$ – нетерминалы, словарь нетерминальных символов; Z – аксиома.

Цепочки, которые порождаются данной грамматикой суть *a, aa, aaa, a…a* или *ba, baa, ba...a*. То есть, $L\{G[Z]\} = \{a^n, ba^n, n > 0\}$.

Сентенциальная форма – это строка, порожденная аксиомой грамматики (выведенная из аксиомы).

Существуют лево- и правосторонние формы вывода, отражающие порядок применения правил грамматики к нетерминальным символам, находящимся в цепочки, при порождении очередной цепочки цепочек. Например,

$$\mathbf{S} \Rightarrow b\mathbf{A} \Rightarrow ba\mathbf{A} \Rightarrow baa,$$

(иногда сокращённо записывают $\mathbf{S} \Rightarrow^* baa$). Причём каждая цепочка является сентенциальной формой.

Левосторонняя форма вывода от правосторонней отличается тем, что заменяется самый левый нетерминальный символ. Вывод, приведённый в примере в равной мере можно считать как левосторонним, так и правосторонним, поскольку в каждой цепочке нетерминальный символ будет единственным.

Процесс вывода удобно представлять в виде ориентированного графа, который называется ***синтаксическим деревом*** или ***деревом вывода***. Для рассмотренной сентенциальной формы оно имеет вид (рисунок 1):

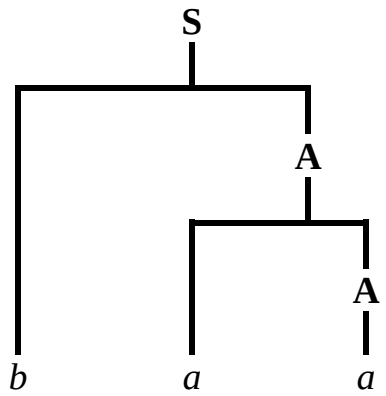

Рисунок 1 – Дерево вывода сентенциальной формы *baa*

Грамматики и языки классифицируются по Хомскому по виду продукций следующим образом:

1. Естественные языки

$$\alpha \rightarrow \beta, \text{ где } \alpha \in (V_N \cup V_T)^+, \ \beta \in (V_N \cup V_T)^*.$$

2. Контекстно-зависимые (контекстно-чувствительные) языки

$$\alpha\mathbf{U}\beta \rightarrow \alpha u\beta, \text{ где } \mathbf{U} \in V_N, u \in (V_N \cup V_T)^+, \alpha, \beta \in (V_N \cup V_T)^*.$$

3. Контекстно-свободные языки

$$\mathbf{U} \rightarrow u, \text{ где } \mathbf{U} \in V_N, u \in (V_N \cup V_T)^*.$$

4. Автоматные (регулярные) языки

$$\mathbf{U} \to t|t\mathbf{W}, \text{ (либо } \mathbf{U} \to \mathbf{W}t\text{) где } \mathbf{U}, \mathbf{W} \in V_N, t \in V_T.$$

Грамматика является ***однозначной***, если ***для каждой*** сентенциальной формы существует ***единственное*** синтаксическое дерево, т.е. любая сентенциальная форма выводится единственным образом. Отметим,что ***порядок*** применения правил ***не существенен***.

Пример 1. Определить, однозначны ли грамматики:

а) $\mathbf{S} \to \mathbf{AB}|a$, $\mathbf{A} \to b\mathbf{A}|a$, $\mathbf{B} \to c\mathbf{A}$

б) $\mathbf{S} \to \mathbf{S}b\mathbf{S}|\mathbf{S}c\mathbf{S}|a$

Для ответа на вопрос, поставленный в задаче, воспользуемся определением и построим деревья:

а)

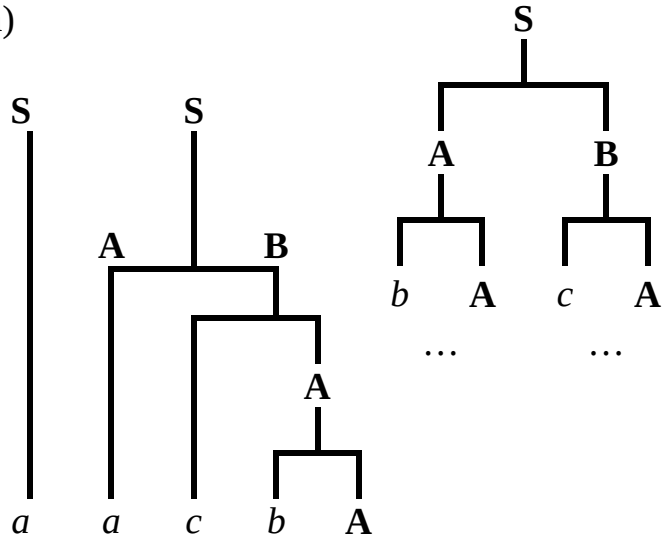

Рисунок 2.1 – Деревья выводов грамматики а)

б)

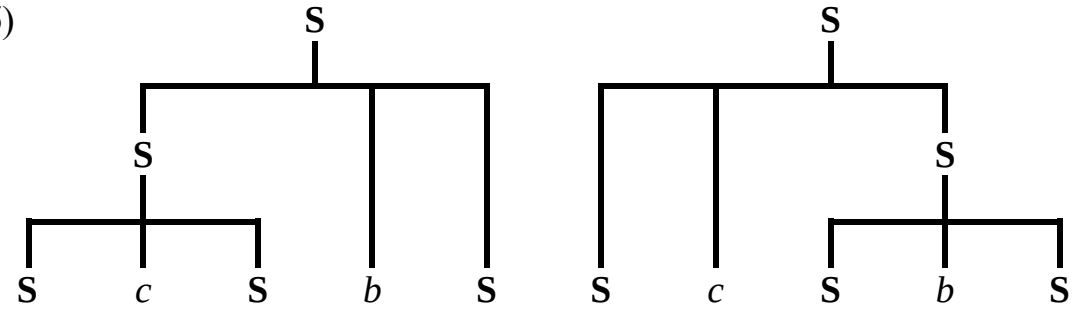

Рисунок 2.2 – Два дерева выводов сентенциальной формы **S***c***S***b***S**

Как видно из сопоставления деревьев, цепочке **S***c***S***b*S соответствуют два дерева. Следовательно, грамматика б) неоднозначна.

2.2 Конечные автоматы

Согласно теории, любой регулярной грамматике соответствует конечный автомат (КА), множество входных цепочек которого соответствует множеству цепочек порождаемых грамматикой.

Конечным автоматом (КА) формально называют совокупность следующих объектов (компонентов):

$$A = (Q, \Sigma, \delta, q_0, F), \text{ где}$$

Q – конечное множество неструктурируемых состояний автомата;

Σ – непустое множество (алфавит) входных символов (букв или литер);

δ – функция переходов КА, определяемая на декартовом произведении $Q \otimes \Sigma$;

$q_0 \in Q$ – начальное состояние КА;

$F \subset Q$ – множество заключительных состояний КА.

“Физический смысл” функции перехода состоит в ***определении очередного состояния*** КА в зависимости от ***символа на входе*** и ***текущего состояния*** автомата.

Конечных состояний может быть много, начальное – одно.

Существует система формальных правил, обеспечивающих соответствие КА и регулярной грамматики.

При этом едполагается представление конечного автомата в виде ориентированного графа, на котором особо обозначены начальные и конечные состояния.

По виду продукций различают праволинейную грамматику, в продукциях которой нетерминалы являются головными символоми в правых частях, и леволинейную грамматику, в правых частях правил которых нетерминальные символы находятся последними. По этой причине существуют два алгоритма преобразования.

Для праволинейной автоматной грамматики.

1. Каждый нетерминал представляется узлом или состоянием на диаграмме.
2. Каждому правилу вида $Q ::= t$, $t \in V_T$, $Q \in V_N$ соответствует дуга, направленная от начального состояния к состоянию Q, помеченная терминальным символом t.
3. Каждому правилу вида $Q ::= Rt$, $t \in V_T$, $Q, R \in V_N$ соответствует дуга, направленная от состояния R к состоянию Q, помеченная терминальным символом t.

Для леволинейной автоматной грамматики.

1. Каждый нетерминал представляется узлом или состоянием на диаграмме.
2. Каждому правилу вида $Q ::= t$, $t \in V_T$, $Q \in V_N$ соответствует дуга, направленная от состояния Q к конечному состоянию, помеченная терминальным символом t.
3. Каждому правилу вида $Q ::= tR$, $t \in V_T$, $Q, R \in V_N$ соответствует дуга, направленная от состояния Q к состоянию R, помеченная терминальным символом t.

Представление КА в виде графа привлекательно тем, что совмещает “в одном лице” все объекты, определяющие автомат.

Возможно описание КА в виде функции переходов, задаваемой как перечисления образов функции перехода $q_{i+1} = \delta(qi, a_i)$, где q_i, q_{i+1} – суть текущее и последующее состояния КА , $a_i \in V_T$ – текущий символ на входе, иногда эта функция оформляется в виде таблицы переходов.

Существуют две разновидности КА:

- детерминированный конечный автомат (ДКА);
- недетерминированный конечный автомат (НКА).

Для ДКА выполняется условие: из текущего состояния переход по конкретному символу входного алфавита может быть осуществлен только в одно из следующих состояний. Если условие не выполняется, имеем недетерминированный КА (НКА).

Различают ***частичный*** и ***полный*** КА.

Для полного КА все образы функций переходов полностью определяются на декартовом произведении $Q \otimes \Sigma$, в противном случае КА считается частичный.

Для неопределённых образов функции переходов применяется термин “образ функции перехода пуст”.

КА функционирует по следующим правилам.

1. Исходным состоянием КА является начальное состояние q_0.
2. Цепочка, составленная из литер алфавита Σ, по одной букве поступает на вход КА. Возврат к начальным литерам цепочки не возможен.
3. Литера s_j допускается КА, находящимся в состоянии q_i, если образ функции переходов не пуст: $\delta(q_i, s_j) \neq \{\}$. Иными словами, определён переход из текущего состояния КА в следующее.
4. Если литера входной цепочки не допускается, то цепочка отвергается, признаётся не принадлежащей входному языку КА, сам автомат остаётся в текущем состоянии.
5. Цепочка литер допускается КА, если после поступления её последнего символа автомат оказывается в одном из заключительных состояний $q_r \in F$. В противном случае цепочка отвергается.

Эти особенности работы конечного автомата используется в практике синтаксического анализа для проверки корректности цепочек.

Примеры описания конечного автомата.

1. В виде таблицы переходов

	Состояния				
Символы	0	1	2	3	4
a	4	1	3	4	1
b	3	2	-	1	2
c	2	4	-	2	3

$Q = \{0, 1, 2, 3, 4\}$; $F = \{2\}$; $q_0=0$; $\Sigma = \{a, b, c\}$; функция переходов δ задана таблицей.

Так как переходы из состояния 2 по символам *b* и *c* не определены, то КА является неполным (частичным).

2. В виде функции переходов

При этом δ задаётся явно как перечисление образов

$4 = \delta(0, a)$; $3 = \delta(0, b)$; $2 = \delta(0, c)$;

$1 = \delta(1, a)$; $2 = \delta(1, b)$; $4 = \delta(1, c)$;
$3 = \delta(2, a)$;
$4 = \delta(3, a)$; $1 = \delta(3, b)$; $2 = \delta(3, c)$;
$1 = \delta(4, a)$; $2 = \delta(4, b)$; $3 = \delta(4, c)$.

3. В виде ориентированного графа

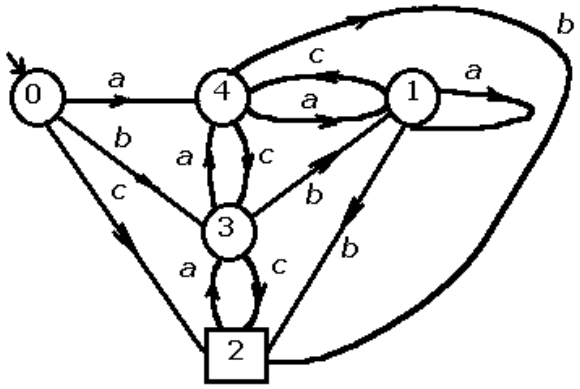

Рисунок 3 – Граф конечного автомата

Стрелкой на графе отмечено начальное состояние, а прямоугольником – конечное.

4. В виде регулярной грамматики

Установим следующие соответствия между номерами состояний и нетерминальными символами грамматики: 1 ~ W, 2 ~ Z, 3 ~ X, 4 ~ Y.

Воспользуемся правилами для построения праволинейной грамматики. Тогда продукции грамматики G[Z] следующие

$$\mathbf{Z} \to \mathbf{X}c|\mathbf{Y}b|\mathbf{W}b|c$$
$$\mathbf{Y} \to a|\mathbf{X}a|\mathbf{W}c$$
$$\mathbf{X} \to \mathbf{Y}c|\mathbf{R}a|b$$
$$\mathbf{W} \to \mathbf{W}a|\mathbf{X}b|\mathbf{Y}a$$

КА, построенный по формальной грамматике, может быть недетерминированным, но для реализации необходимо, чтобы автомат был детерминированным. По этой причине были разработан алгоритм построения ДКА по НКА.

2.2.1 Алгоритм построения ДКА, эквивалентного НКА

1. Выполняем построение таблицы переходов НКА.

2. Рассмотрим функцию переходов: если значение функции не содержится во множестве состояний автомата, то дополним это множество данной функцией.

3. Для вновь введенного состояния строим функцию переходов; анализируем функцию переходов в состояниях, определяющих данное состояние.

Определяем выходной сигнал, как дизъюнкцию выходных сигналов, входящих в данное обобщенное состояние состояний.

4. Повторяем п.п. 1 и 3 до тех пор, пока множество состояний автомата не будет изменено.

5. Если получившийся КА является частичным, то доопределим функцию перехода, вводя фиктивное состояние, соответствующее ошибке.

Замечание. Последний пункт можно выполнить как до начала применения алгоритма, так и по получении ДКА.

Пример 1. Построить ДКА для регулярной грамматики:

$$\mathbf{S} \to \mathbf{T}x|\mathbf{V}y$$
$$\mathbf{T} \to \mathbf{T}x|x$$
$$\mathbf{V} \to \mathbf{V}y|y$$

Решение.

1. Грамматике соответствует граф КА, показанный на рисунке 4 Функция перехода его однозначна для образов $\mathbf{T}= \delta(\mathbf{S}, x)$ и $\mathbf{V} = \delta(\mathbf{S}, y)$, неоднозначна для пар $\mathbf{Z} = \delta(\mathbf{T}, x)$, $\mathbf{T} = \delta(\mathbf{T}, x)$ и $Z = \delta(\mathbf{V}, y)$, $\mathbf{V} = \delta(\mathbf{V}, y)$, а тако же не определена для образов $?= \delta(\mathbf{T}, y)$ и $?= \delta(\mathbf{V}, y)$.

Таким образом, заданной в условии задачи грамматике соответствует частичный НКА, так как образы функций перехода определены и не полностью, и неоднозначно.

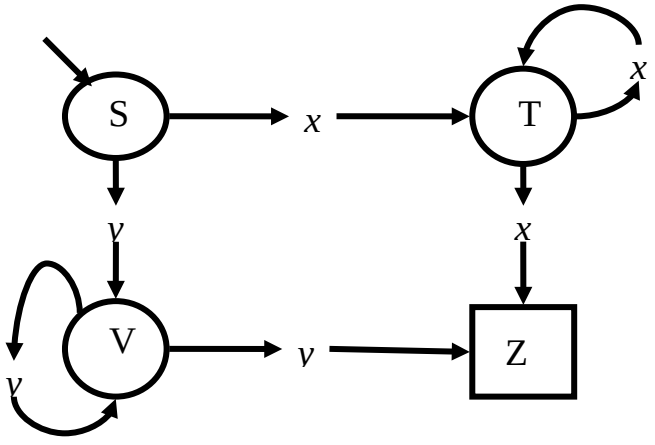

Рисунок 4– Граф переходов частичного КА

2. Таблица управления КА, соответствующая функции переходов, имеет вид:

	S	**T**	**V**	**Z**

x	**T**	**{Z, T}**	—	—
y	**V**	—	**{Z, V}**	—

3. Введем фиктивное состояние E, с его помощью частичный КА будет преобразован к полному КА

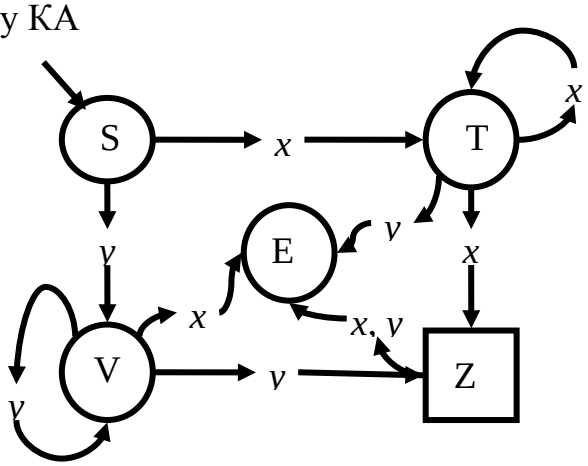

Рисунок 5 – Граф переходов полного КА

Для этого случая построим таблицу переходов

	S	**T**	**V**	**Z**	**{Z, T}**	**{Z, V}**	**E**
x	**T**	**{Z, T}**	**E**	**E**	**{Z, T}**	**E**	**E**
y	**V**	**E**	**{Z, V}**	**E**	**E**	{Z, T}	**E**

Так как в состояние Z нет ни одного перехода, то его можно удалить из множества внутренних состояний автомата.

Окончательно имеем:

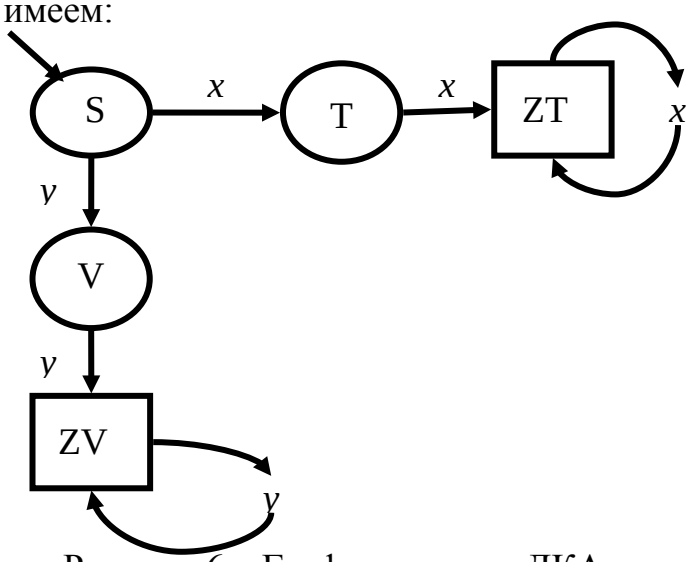

Рисунок 6 – Граф переходов ДКА

Пример 2. Пусть множество строк входного языка *L* определено на алфавите {0, 1} таким образом, что справа от символа 0 всегда находится символ 1. Определить ДКА, которым могут быть приняты все строки языка *L*, записать правила грамматики, их порождающей.

Решение.

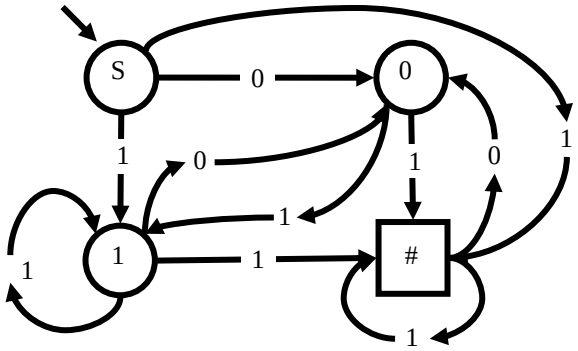

Рисунок 7 – Граф переходов исходного НКА

По условию задачи построим граф конечного автомата (рисунок 7).

Таблица переходов выглядит таким образом:

	S	0	1	#	{1, #}
0	0	–	0	0	0
1	{1, #}	{1, #}	{1, #}	#	{1, #}

Так как состояния 1 и # не достигаются, то они удаляются из множества внутренних состояний автомата. Окончательно получим

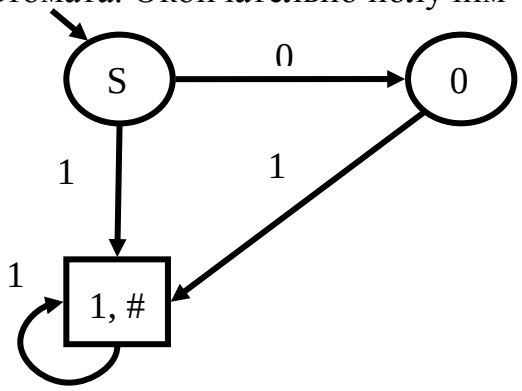

Рисунок 8– Граф переходов ДКА

Таблицу переходов, соответствующую этому графу, читателю, в качестве упражнения, предлагается построить самостоятельно

Введя обозначения Z={1, #}, A={0}, сконструируем, по правилам, изложенным выше, следующую праволинейную грамматику G[Z]

Z → 1|**Z**1|**A**1

A → 0|**Z**0

Леволинейная грамматика G[S] имеет вид

S → 0A|1**Z**|1

A → 1**Z**|1.

Пример 3. Построить КА, который будет принимать любое английское слово, начинающееся на un и заканчивающееся на d. Написать на языке программирования C++ программу подсчета числа таких слов в произвольном текстовом файле.

Решение.

1. Построим граф проверяющий слова. Его вид показан на рисунке 9.
На дугах графа нанесены обозначения: *d*, *n*, u - символы латинского алфавита, соответствующие написанию; *б* – любая буква латинского языка, не совпадающая с буквами {*d*, *n*, *u*}; *“ ”*– пробел между словами.
Состояние ”ошибка” соответствует словам, не удовлетворяющим условию.

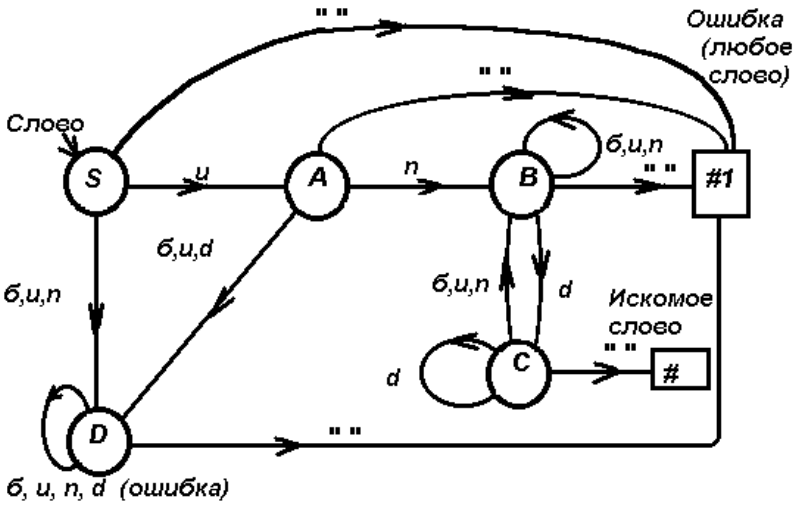

Рисунок 9– Граф переходов ДКА, проверяющего слова

2. Построим по этому графу таблицу переходов.

	S	***A***	***B***	***C***	***D***	***#1***	***#***
u	A	D	B	B	D	-	-
n	D	B	B	B	D	-	-
d	D	D	C	C	D	-	-
“_“	#1	#1	#1	#	#1	-	-
б	D	D	B	B	D	-	-

Имея в виду дальнейшую программную реализацию на языке C++, введём следующую нумерацию состояний:

$$S \Rightarrow 0,\ A \Rightarrow 1,\ B \Rightarrow 2,\ C \Rightarrow 3,\ D \Rightarrow 4,\ \#1 \Rightarrow 5,\ \# \Rightarrow 6.$$

Нумерация выполнена для программной реализации, это позволит сэкономить количество символов в операторе инициализации таблицы переходов конечного автомата.
3. Текст программы

```
main()
{
char c;                    /* текущая литера */
```

```
int i, n;
file * text;
int mp[5][6] =   {{1, 4, 2, 2, 4}, //управляющая таблица
                 {4, 2, 2, 2, 4},
                 {4, 4, 3, 3, 4},
                 {5, 5, 5, 6, 5},
                 {4, 4, 2, 2, 4}};
text = fopen('xx.txt', 'r'); // исходный текст в файле
i=0; n=0;        /* i - номер состояния КА, n - число пра-
вильных слов */
while ((c = getc(text))! = EOF)
    {
    switch(c)
        {
        case 'u': i = mp[0][i]; break;
        case 'n': i = mp[1][i]; break;
        case 'd': i = mp[2][i]; break;
        case '  ': i = mp[3][i]; break;
        default:  i = mp[4][i];break;
        }
    if (i ==6) {i = 0; n++;}
    if (i ==5) {i = 0;}
    }
printf("количество символов =%d", n);
fclose(text);
}
```

Язык, анализируемый конечным автоматом, может быть описан грамматиками класса 3 в праволинейной или леволинейной формах [5], которые могут быть построены согласно формальной системе правил соответствия конечного автомата и формальной грамматики. Для праволинейной грамматики [6]: а) каждый нетерминал грамматики соответствует состоянию КА; б) каждому правилу вида $\mathbf{W}\rightarrow v$ соответствует дуга, направленная из начального к состоянию **W**, помеченная символом *v*; в) каждому правилу вида $\mathbf{W}\rightarrow\mathbf{R}v$ соответствует дуга из состояния **R** к состоянию **W**, помеченная символом *v*; где $\mathbf{W},\mathbf{R}\in V_H$, $v\in V_T$. Для построения леволинейной грамматики меняются позиции символов в правилах грамматики и направление «входящий – исходящий» для дуг.

Грамматики, язык которых есть цепочки латинских литер с правильными головой ***un*** и хвостом ***d***, построены для выделенной пунктиром части рисунка 2.4 – графа конечного автомата.

Праволинейная грамматика	Леволинейная грамматика

Аксиома грамматики: C	Аксиома грамматики: **S**
Тип разбора, осуществляемого конечным автоматом: ***восходящий*.**	Тип разбора, осуществляемого конечным автоматом: *нисходящий.*
C→**C***d*|**B***d*	**S**→*u***A**
B→**A***n*|**C***u*|**C**σ|**C***n*	**A**→*n***B**
A→*u***.**	**B**→*d***C**|*d*
	C→*d***C**|*u***B**|*n***B**|σ**B.**

3 ХОД РАБОТЫ

1. Изучить теоретический раздел.
2. Согласно варианту задания, выполнить практический расчёт, показать промежуточные результаты.
3. Написать программу на языке C++ , реализующую алгоритм поставленной задачи.
4. Подготовить ответы на контрольные вопросы.
5. Сделать выводы в развёрнутом виде.

4 ВАРИАНТЫ ЗАДАНИЙ

1. Построить КА, восстановить грамматику, составить регулярные выражения, соответствующие цепочкам:
1.1 оператор GOTO и GO TO, между слогами которого может быть произвольное число пробелов;
1.2 над алфавитом {*0,1*}, в которых число единиц чётное, а нулей нечётное;
1.3 над алфавитом {*0,1*}, в которых между единицами чётное число нулей;
1.4 над алфавитом {*0,1*}, в которых за каждым вхождением пары 11 следует 0;
1.5 над алфавитом {*0,1*}, в которых каждый третий символ единица;
1.6 над алфавитом {*0,1*}, в которых имеется хотя бы одна единица;
1.7 над алфавитом {*0,1*}, в которых допускаются две цепочки 01 и 01000;
1.8 над алфавитом {*0,1*}, которые начинаются с нуля и оканчивающаяся единицей;
1.9 над алфавитом {*0,1*}, которые содержат всего три единицы;
1.10 над алфавитом {*0,1*}, в которых перед и после каждой единицы стоят нули;

2. По графам конечных автоматов, представленным на рисунке 9, построить соответствующие грамматики и регулярные выражения, описывающее языки.

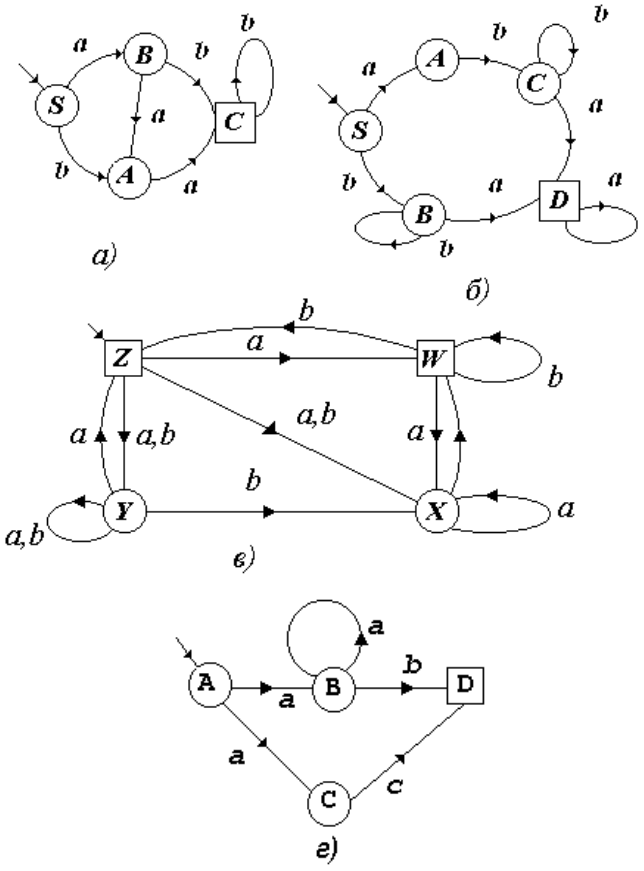

Рисунок 9 – Граф конечного автомата

3. Найти множество цепочек, распознаваемых конечным автоматом

3.1

Сигнал		не доп.	не доп.	доп.
Состояние		A	B	C
Вход	0	B	B	C
	1	A	C	C

3.2

Сигнал		не доп.	Доп.	не доп.
Состояние		A	B	C
Вход	0	B	C	C
	1	C	B	C

3.3

Сигнал		не доп.	доп.	доп.	не доп.
Состояние		A	B	C	D
Вход	0	B	D	C	D
	1	C	B	D	D

3.4

Сигнал		не доп.	не доп.	доп.	не доп.
Состояние		A	B	C	D
Вход	0	B	D	C	D
	1	A	C	D	D

4. Опишите множество цепочек, распознаваемых недетерминированным конечным автоматом.

4.1

Сигнал		не доп.	не доп.	доп.
Состояние		A	B	C
Вход	0	A,B	-	C
	1	-	B,C	-

4.2

Сигнал		не доп.	не доп.	доп.
Состояние		A	B	C
Вход	0	B	-	-
	1	-	C,A	-

5. Построить детерминированный конечный автомат по недетерминированному конечному автомату.

5.1

Сигнал		не доп.	не доп.	не доп.	доп.
Состояние		A	B	C	D
Вход	*a*	B	C,D	B	-
	c	-	B	-	-

5.2

Сигнал		не доп.	не доп.	доп.
Состояние		1	2	3
Вход	*a*	2,3	-	1
	b	2	1	3

6. Для заданных ниже конечных автоматов A_1 и A_2 построить детерминированный конечный автомат, воспринимающий цепочки вида: **а)** $S_1 \cup S_2$, **б)** $S_1 \cap S_2$, где S_i, $i=1,2$ - множество цепочек, допускаемых i –ым конечным автоматом.

		Конечный автомат A_1				Конечный автомат A_2			
Сигнал		доп.	доп.	не доп.		Не доп.	не доп.	доп.	не доп.
Состояние		A	B	C		A	B	C	D
Вход	0	C	B	C		D	C	D	D
	1	B	C	C		B	C	D	D

7. Построить КА, распознающий цепочки, порождаемые право- и леволинейными грамматиками

7.1 $\mathbf{S} \to \mathbf{A}a|\mathbf{B}b$
$\mathbf{A} \to \mathbf{A}a|a$
$\mathbf{B} \to \mathbf{B}b|b.$

7.2 $\langle \mathbf{Z} \rangle \to \langle \mathbf{cod} \rangle b$
$\langle \mathbf{cod} \rangle \to \langle \mathbf{cod} \rangle 1|\langle \mathbf{cod} \rangle 0$
$\langle \mathbf{cod} \rangle \to 1|0.$

7.3 $\mathbf{Z} \to \mathbf{A}0$
$\mathbf{A} \to \mathbf{A}0|\mathbf{Z}1|0.$

7.4 $\mathbf{A} \to a\mathbf{B}|a\mathbf{D}|a$
$\mathbf{B} \to a\mathbf{B}|b|b\mathbf{D}$
$\mathbf{D} \to d\mathbf{D}.$

7.5 $\mathbf{S} \to x\mathbf{V}|y\mathbf{T}$
$\mathbf{V} \to x\mathbf{V}|b\mathbf{B}|b$
$\mathbf{T} \to y\mathbf{T}|b\mathbf{B}|b$
$\mathbf{B} \to b\mathbf{B}|b.$

7.6 $\mathbf{A} \to 1\mathbf{A}|1\mathbf{B}|1$
$\mathbf{B} \to 0\mathbf{B}|1\mathbf{B}|1.$

7.7 $\mathbf{A} \to 0\mathbf{A}|1\mathbf{A}|1\mathbf{B}$
$\mathbf{B} \to 0\mathbf{C}|1\mathbf{C}$
$\mathbf{C} \to 0\mathbf{C}|1\mathbf{C}|1|0.$

7.8 $\mathbf{S} \to 0\mathbf{A}|1\mathbf{S}|\varepsilon$
$\mathbf{A} \to 0\mathbf{B}|1\mathbf{A}$
$\mathbf{B} \to 0\mathbf{S}|1\mathbf{B}.$

5 СОДЕРЖАНИЕ ОТЧЁТА

1. Цель работы.
2. Вариант задания.
3. Расчётная часть задания.
4. Текст программы, реализующей расчеты по соответствующему варианту.
5. Развернутый вывод по работе.

6 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое алфавит, цепочка, язык, грамматика G, порождающая грамматика, распознающая грамматика, формальной грамматика?
2. Что такое, сентенциальная форма, лево- и правосторонняя грамматики?
3. Что называется конечным автоматом?
4. Что называется детерминированным конечным автоматом (ДКА) и недетерминированным конечным автоматом (НКА)?
5. Каким образом функционирует конечный автомат?
6. Каким образом описываются конечные автоматы.?
7. Алгоритм построения ДКА.

7 БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Ахо А. Компиляторы: принципы, технологии, инструменты / А. Ахо, Дж. Ульман. – М.: Мир, 1978. – 619 с.
2. Вайнгартен Ф. Трансляция языков программирования / Ф. Вайнгартен. – М.: Мир, 1977. – 190 с.
3. Грис Д. Конструирование компиляторов для ЦВМ / Д. Грис. – М.: Мир, 1975. - 544 с.
4. Рейуорд-Смит В.Дж. Теория формальных языков. Вводный курс / В.Дж. Рейуорд-Смит. – М.: Мир, 1986. – 128 с.
5. Оллонгрен А. Определение языков программирования интерпретирующими автоматами / А. Оллонгрен. – М.: Мир, 1972. - 288 с.