ГЛАВА 3

Лексический анализ

3.1 Назначение и необходимость фазы лексического анализа
3.2 Транслитератор
3.3 Регулярные языки и грамматики
3.4 Конечные автоматы
    3.4.1 Определение КА.
    3.4.2 Диаграмма переходов (состояний)
3.5 Матрица переходов КА
3.6 Детерминированный и недерминированный автомат
3.7 Лексический анализатор
3.8 Связь регулярных грамматик КА
    3.8.1 Построение КА на основе леволинейной грамматики
    3.8.2 Построение леволинейной грамматики на основе КА

3.1 Назначение и необходимость фазы лексического анализа

- Лексический анализ – первая фаза процесса трансляции, предназначенная для группировки символов входной цепочки в более крупные конструкции, называемые лексемами.
- Лексемы – минимальные несущие смысл объединения символов. С каждой лексемой связано два понятия:
  - класс лексемы, определяющий общее название для категории элементов, обладающих общими свойствами
  - значение лексемы, определяющее подстроку символов входной цепочки, соответствующих распознанному классу лексемы. В зависимости от класса, значение лексемы может быть преобразовано во внутреннее представление уже на этапе лексического анализа.

3.1 Назначение и необходимость фазы лексического анализа

Задачи, решаемые сканером (преимущества сканера):

- представление элементарных конструкций языка в более удобной внутренней форме
- уменьшение длины программы, за счет устранения из нее несущественных для дальнейшего анализа пробелов, комментариев, игнорируемых символов.
- один и тот же язык программирования может иметь различные внешние представления элементарных конструкций.

3.2 Транслитератор

- Устройство, осуществляющее сопоставление класс с каждым отдельным символом, называется **транслитератором**.
- Наиболее типичными классами символов являются:
  - буква;
  - цифра;
  - разделитель;
  - игнорируемый;
  - запрещенный;
  - прочие.

3.3 Регулярные языки и грамматики

Пример. Классы лексем:

- идентификаторы;
- служебные слова (множество идентификаторов);
- целые числа, вещественные, строки (литералы);
- однолитерные разделители ('+', '-', '(', ')'...);
- двухлитерные разделители ('//', '/*', '*/', ':=')
- комментарии.

3.3 Регулярные языки и грамматики

<идентификатор> ::= буква | <идентификатор> буква |
<идентификатор> цифра
<целое> ::= цифра | <целое> цифра
<разделитель> ::= +| - | / | (|) …
<разделитель> ::= <slash> / | <slash> * | <ast>/ | <colon>=
<ast> ::= *
<colon> ::= :

Регулярные грамматики

A ->Bγ | γ, где A, B ∈ VN, γ ∈ VT^*

A -> γB | γ, где A, B ∈ VN, γ ∈ VT^*

Автоматные грамматики

A -> Bt | t, где A, B ∈ VN, t ∈ VT

A -> tB | t, где A, B ∈ VN, t ∈ VT

3.3 Регулярные языки и грамматики

- Доказано, что класс регулярной и автоматной грамматики почти эквиваленты. Любую регулярную грамматику можно привести к автоматному виду.

- **Домашнее задание:**

Алгоритм преобразования регулярной грамматики к автоматному виду

ГЛАВА 3. Лексический анализ

3.4 КОНЕЧНЫЕ АВТОМАТЫ

3.4.1 Определение КА

Конечный автомат (КА) – это пятерка (Q, V, δ, q_0, F), где:

- Q – конечное множество состояний;
- V – конечное множество допустимых входных символов (алфавит);
- δ – отображение множества Q × VT -> Q, определяющее поведение автомата; отображение δ часто называют функцией переходов;
- $q_0 \in Q$ – начальное состояние;
- $F \in Q$ – непустое множество конечных состояний.

3.4.1 Определение КА

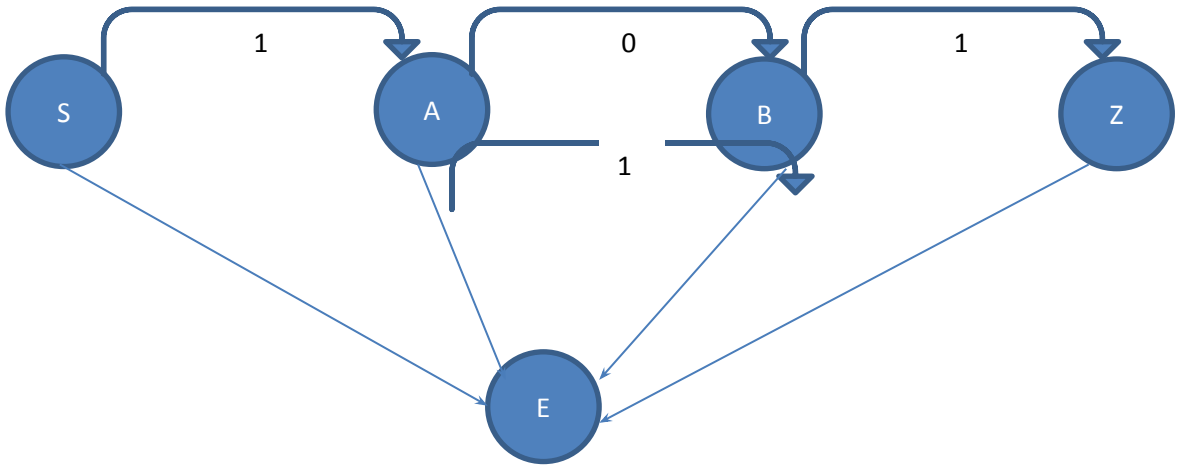
- Конечный автомат называется **полностью определенным**, если в каждом его состоянии определена функция перехода для каждого входного символа.

  Для a є V и q ∈ Q определена δ (a, q)= R, R <= Q
- Множество цепочек, допускаемых конечным автоматом, составляет определяемый им язык.
- Два конечных автомата называются **эквивалентными**, если они определяют один и тот же язык.

3.4.2 Диаграмма переходов (состояний)

- Граф переходов (дерево, диаграмма) – направленный помеченный граф с символами состояний конечного автомата в вершинах, в котором есть дуга (p,q) , помеченная символом a є V (p,q ∈ Q), если в КА определена функция δ (a,p) и q ∈ δ (a,p).

M ({S,A,B,Z}, {0,1}, δ, S, {Z})
δ(1,S) = {A}
δ(0,A) = {B}
δ(1,B)={A,Z}



3.5 Матрица переходов КА

- Каждая строка этой матрицы представляет состояние автомата, а каждый столбец соответствует возможному входному элементу.
- В ячейках матрицы указывается структура из трех полей:
  - состояние
  - функция которая выполняется при переходе из одного состояния в другое
  - сообщение об ошибке

3.5 Матрица переходов КА

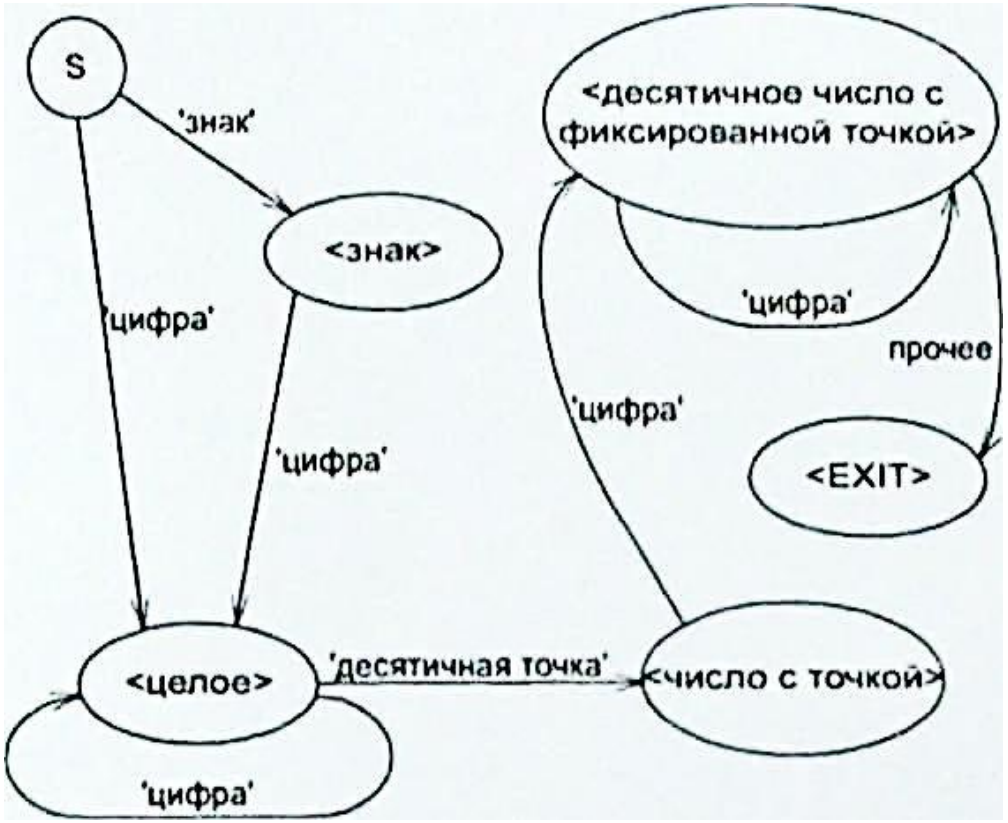
<десятичное целое с фиксированной точкой> ::=
<число с точкой> <цифра> |
< десятичное целое с
фиксированной точкой > <цифра>

<число с точкой> ::= <целое>
<десятичная точка>

<целое> ::= <знак> <целое> |
<цифра> | <целое> <цифра>

<десятичная точка> ::= .

<знак> ::= + | -



<цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

3.5 Матрица переходов КА

Состояние\Следующий элемент	<знак>	<десятичная точка>	<цифра>	<прочие>
1 Start	2, F1, M0	6, F0, M1	3, F2, M0	6, F0, M1
2 <знак>	6, F0, M2	6, F0, M3	3, F2, M0	6, F0, M4
3 <целое>	6, F0, M5	4, F4, M0	3, F2, M0	6, F0, M4
4 <число с точкой>	6, F0, M6	6, F0, M1	5, F3, M0	6, F0, M6
5 <дес. число с фикс. т.>	6, F0, M8	6, F0, M1	5, F3, M0	Exit
6 Error	Exit	Exit	Exit	Exit

Матрица переходов состояний для распознавания десятичных чисел с фиксированной точкой

3.5 Матрица переходов КА

Подпрограмма	Действие
Инициализация	num=0; sign = "+"; count = 1.0
F0	Exit
F1	sign = insign
F2	num = 10 * num + innum
F3	count = count / 10; num = num + innum * count
F4	

innum – значение следующей введенной цифры;
insign – значение знака во входной строке;
sign – значение знака распознаного числа;
num – значение распознаного числа;
count – количество цифр после точки.

3.5 Матрица переходов КА

Номер сообщения	Сообщение
M0	
M1	Ошибка. Строка не число: нет ни цифры ни знака
M2	Ошибка. Два последовательных знака
M3	Ошибка. Перед точкой нет ни одной цифры
M4	Ошибка. За знаком не следует цифра
M5	Ошибка. Отсутствует точка
M6	Ошибка. За точкой нет ни одной цифры
M7	Ошибка. Две десятичные точки в числе
M8	Ошибка. Знак в середине числа
Exit	All OK

3.6 Детерминированный и недерминированный автомат

- Конечный автомат (Q, V, δ, q_0, F) называется **детерминированным конечным автоматом** (ДКА), если в каждом из его состояний для любого входного символа функция переходов содержит не более одного состояний.

  Для $a \in V$, $q \in Q$, $r \in Q$, определена $\delta(a, q) = \{r\}$
- **В недетерминированном КА** $\delta(a,q) = \{r_1, r_2, ..., r_n\}$ – означает, что из состояния q по входному символу a можно осуществить переход в любое из состояний r_i, $i = 1, 2, ... ,n$.
- Доказано, что для любого КА можно построить ДКА.

Домашнее задание: Алгоритм преобразования произвольного КА к детерминированному виду.

3.7 Лексический анализатор

- Лексический анализатор – программа, которая используя набор сканеров, преобразует исходный текст программы во внутреннее представление и помещает определённую информацию в специальные таблицы.
- Типы лексем (сканеров): идентификаторы, служебные слова, литералы (или константы) и разделители.
- Каждой лексеме сопоставляется пара:

  (тип лексемы, указатель на инф. о ней)

3.7 Лексический анализатор

- <prog> ::= PROGRAM <prog_name> VAR <dec_list> BEGIN <stmt_ list> END.
- <prog_name> ::= id
- <dec_list> ::= <dec>; | <dec_list>; <dec>
- <dec> ::= <id_list>:<type>
- <type> ::= INTEGER
- <id_list> ::= <id> | <id_list>, <id>
- <stmt_list> ::= <stmt> | <stmt_list>; <stmt>
- <stmt> ::= <assign> | <read> | <write> | <for>
- <assign> ::= <id>:=<exp>
- <exp> <term> | <exp>+<term> | <exp>-<term>
- <term> ::= <factor> | <term>*<factor> | <term> DIV <factor>
- <factor> ::= <id> | <int> | (<exp>)
- <read> ::= READ(<id_list>)
- <write>::= WRITE(<id_ list>)
- <for> ::= FOR <index_ exp> DO <body>
- <index_exp> ::= <id>:=<exp> TO <exp>
- <body> ::= <stmt> | BEGIN <stmt_list> END
- <id> ::= <id_name>
- <id name> ::= <letter> | <id_name> <letter> | <id_name><digit>
- <int> ::= <digit> | <int><digit>
- <letter> ::= A | B | C|... | Z
- <digit>::= 0 | 1 | 2 |...| 9

3.7 Лексический анализатор

```
PROGRAM STATS
VAR
    SUM, SUMSQ, I, VALUE, MEAN, VARIANCE: INTEGER;
BEGIN
    SUM := 0;
    SUMSQ := 0;
    FOR I 1 TO 100 DO
    BEGIN
        READ( VALUE );
        SUM := SUM + VALUE;
        SUMSQ := SUMSQ + VALUE * VALUE;
    END;
    MEAN := SUM DIV 100;
    VARIANCE := SUMSQ DIV 100 - MEAN * MEAN;
    WRITE( MEAN, VARIANCE);
END.
```

3.7 Лексический анализатор

Таблица служебных слов TW 1

N	Терминальный символ	Признак разделителя
1	PROGRAM	-
2	BEGIN	-
3	VAR	-
4	INTEGER	-
5	END	-
6	FOR	-
7	TO	-
8	DO	-

Таблица разделителей TD 2

N	Терминальный символ	Признак разделителя
1	;	+
2	:	+
3	:=	

3.7 Лексический анализатор

Таблица литералов (констант) TNUM 3

N	Лите-рал	Ад-рес	Тип	Другая информа-ция
1	0		I	Значение
2	0		I	во
3	1		I	внутреннем
4	100		I	представлении

Таблица идентификаторов TID 4

N	Символи-ческое имя	Ад-рес	Тип	Другая информа-ция
1	STATS		I	
2	SUM		I	
3	SUMSQ		I	
4	I		I	

3.7 Лексический анализатор

Переменные:

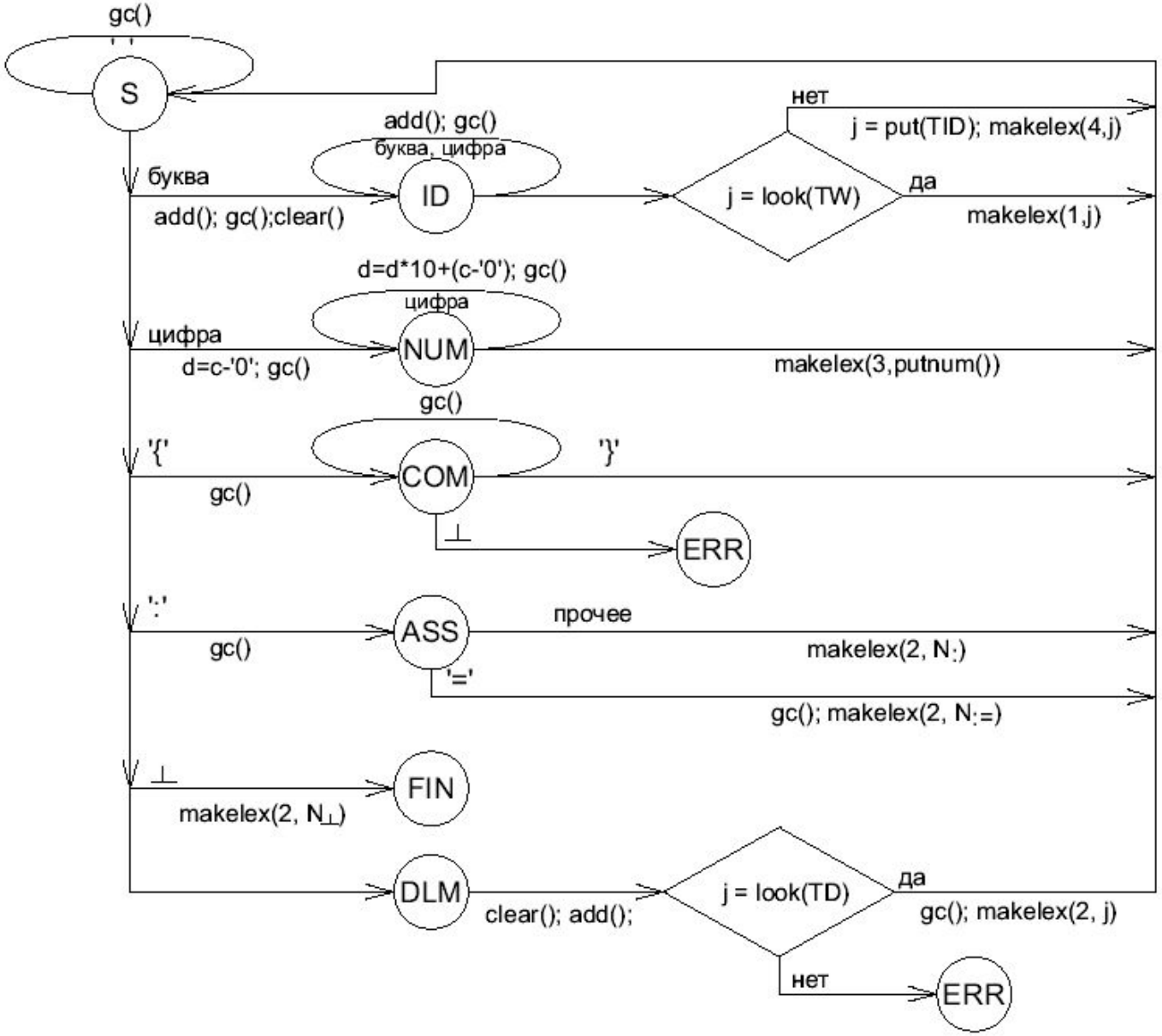
- buf – буфер для накопления символов лексем;
- c – очередной входной символ;
- d – переменная для формирования числового значения константы;
- TW – таблица служебных слов ;
- TD – таблица ограничителей;
- TID – таблица идентификаторов;
- TNUM – таблица констант;
- tabl – имя типа таблиц;
- ptable – указатель на tabl.

3.7 Лексический анализатор

Функции:

- void clear (void) – очистить буффер *buf,*
- void add (void) – добавление символа c в конец буфера *buf;*
- int look (ptabl T) – поиск в таблице T лексемы из буфера *buf;* результат – номер строки таблицы с информацией о лексеме либо 0, если лексемы нет в таблице;
- int putl (ptabl T) – запись в таблицу *T* лексемы из *buf;* результат – номер строки с информацией о лексеме;
- int puntnum(void) – запись в *TNUM* константы из d если ее там не было; результат – номер строки в таблице *TNUM* с информацией о константе-лексеме;
- void makelex (int k, int i) – формирование и вывод внутреннего представления лексемы; к – номер класса, i – номер в классе;
- void gc (void) – чтение из входного потока очередного символа и занесение его в c

3.7 Лексический анализатор



3.7 Лексический анализатор

- КА пользуясь набором сканера формирует набор лексем.

<1,1> <4,1> <1,3> <4,2>

PROGRAM STATS VAR SUM

<2,2> <1,4> <2,1> <1,2> ...

: INTEGER ; BEGIN ...

3.7 Лексический анализатор

```
F0={ gc() }
F1={ clear(); add(); gc()}
F3={ add(); gc() }
F4={ j==look(TW) – ?
        Да: makelex(1,j);
        Нет: j=put(TID);
          makelex(4,j); }
```

```
F5={ makelex(3, putnum()) }
F7={ makelex(2, look(TD)) }
F8={ add(),gc(), makelex(2,look(TD)) }
F9={ clear(), add()
        j==look(TD) – ?
        Да: gc(); makelex(2,j);
        Нет:  c==’\n’ -?
                Да:gc();
                Нет:State=7;  }
```

3.7 Лексический анализатор

		буква	цифра	‘ ‘;’\n’	‘{‘	‘}‘	‘:’	‘=’	⊥	Прочее
0	S	1, F0	2, F1	3, F1	3,F0	7	4, F1	6	5	6
1	ID	1, F4	7	1, F3	0, F4	0, F4	0, F4	0, F4	0, F4	0, F4
2	NUM	0, F5	2, F3	0, F5	0, F5	0, F5	0, F5	0, F5	0, F5	0, F5
3	COM	3, F0	3, F0	3, F0	3, F0	0, F0	3, F0	3, F0	7	3, F0
4	ASS	0, F7	0, F7	0, F7	0, F7	0, F7	0, F7	0, F8	0, F7	0, F7
5	FIN	—	—	—	—	—	—	—	—	—
6	DLM	0, F9	0, F9	0, F9	0, F9	0, F9	0, F9	0, F9	0, F9	0, F9
7	ERR	—	—	—	—	—	—	—	—	—

3.8 Связь регулярных грамматик КА

- На основании имеющейся регулярной грамматики можно построить эквивалентный ей КА, и наоборот на основе КА можно построить эквивалентную ему грамматику.
- Поскольку КА являются распознавателями регулярных языков, таким образом, построив по регулярной грамматике КА, решаем задачу разбора.

3.8.1 Построение КА на основе леволинейной грамматики

Необходимо построить КА $M(Q, V, \delta, q_0, F)$, по леволинейной грамматике $G(VT, VN, P, S)$.

1) Множество входных символов строится на основании терминальных символов $V=VT$

2) Множество состояний автомата строится на основании множества нетерминальных символов. Каждому нетерминалу ставится в соответствие состояние автомата и добавляется еще одно начальное состояние.

$$Q = VN \cup \{H\}$$

3.8.1 Построение КА на основе леволинейной грамматики

3) Просматриваем все множество правил грамматики G.

- Если встречается правило вида A -> t, A єVN, t є VT, то функцию перехода добавляем состояние A. A є δ(H,t).
- Если A -> Bt, A,B є VN, t є VT, то A є δ(B,t).

4) Начальное состояние автомата q_0=H.

5) Множество конечных состояний включает одно состояние, соответствующее начальному символу грамматики S.

F={S}

3.8.1 Построение КА на основе леволинейной грамматики

Построить автомат $M(Q, V, \delta, q_0, F)$ на основе имеющейся автоматной грамматики:

$G = (\{0,1\}, \{S,A,B\}, P, S)$

P: S -> A0 | B1

A -> S1|1

B -> S0|0

$L(G) = \{ (10 \mid 01)^n, n>0\}$

- S => A0 => 10
- S => A0 => S10 => B110 => 0110
- S => B1 => 01
- S => B1 => S01 =>A001 => 1001

3.8.1 Построение КА на основе леволинейной грамматики

Шаг 1. V = {0, 1}

Шаг 2. Q = {S, A, B, H}

Шаг 3. δ(A,0) = {S}

δ(B,1) = {S}

δ(S,1) = {A}

δ(H,1) = {A}

δ(S,0) = {B}

δ(H,0) = {B}

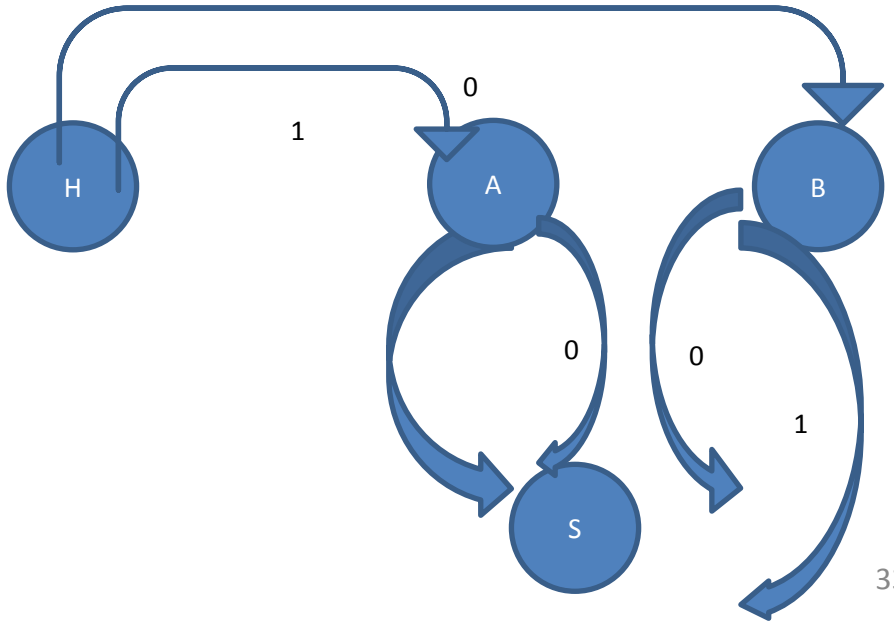
Шаг 4. q_0 = H

Шаг 5. F= {S}

S -> A0 | B1

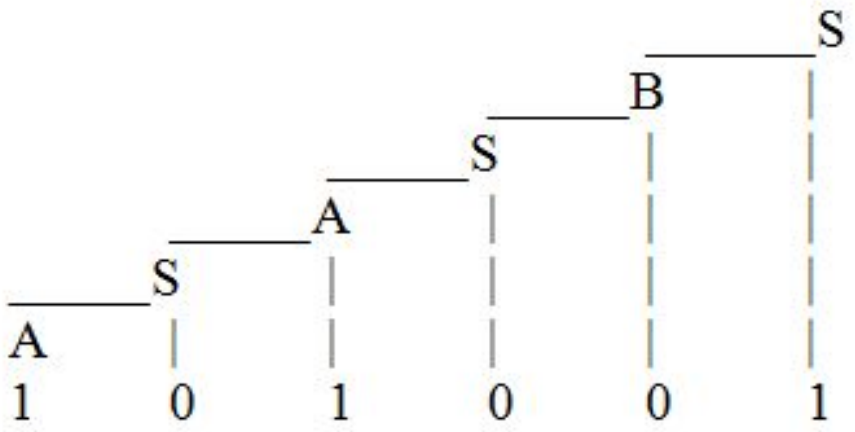
A -> S1|1

B -> S0|0



3.8.1 Построение КА на основе леволинейной грамматики

Шаг	Сос-тоя-ние	Остаток входной цепочки
1	H	101001
2	A	01001
3	S	1001
4	A	001
5	S	01
6	B	1
7	S	-



3.8.2 Построение леволинейной грамматики на основе КА

1) Множество терминальных символов строится на основании входных символов

$VT=V$

2) Множество нетерминалов грамматики G строится на основании множества состояний Q автомата за исключением начального состояния автомата

$VN = Q / \{q_0\}$

3.8.2 Построение леволинейной грамматики на основе КА

3) Просматриваем функции переходов автомата M для всех возможных состояний из Q для всех входных символов из V.

Если $\delta(A,t)=\{B_1,B_2, \dots B_n\}$, A,B є Q, t є V, $1<i<=n$,

то для всех состояний B_i выполняем следующее:

- Если $A= q_0$, то множество Bi -> t
- Если $A\neq q_0$, то множество Bi ->A t

4)

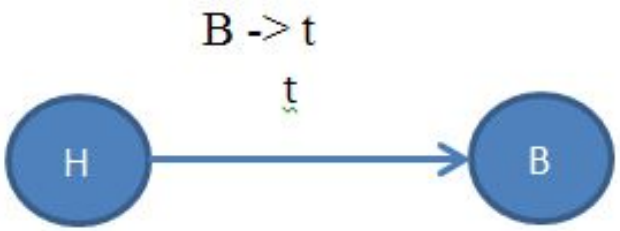
- Если множество конечных состояний F автомата M содержит только одно состояние F={f0}, то начальным символом грамматики принимается нетерминал, соответствующий этому состоянию.
- Если множество заключительных состояний $F=\{f_1,f_2,\dots,f_n\}$,

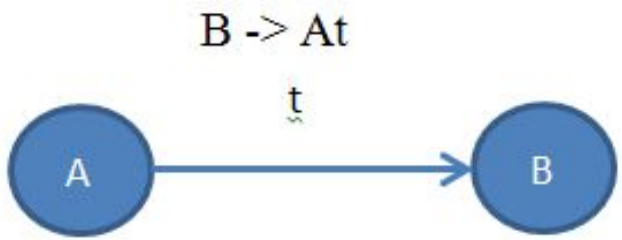
  то $S \to F_1 \mid F_2 \mid \dots \mid F_n$

VN = VN U {S}

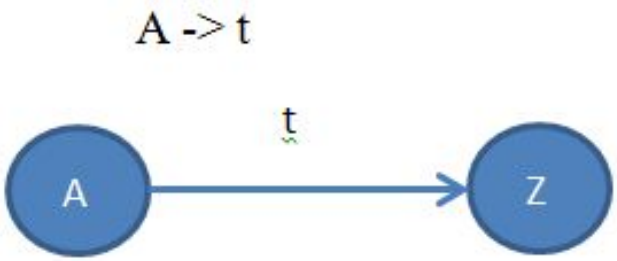
3.8.2 Построение леволинейной грамматики на основе КА

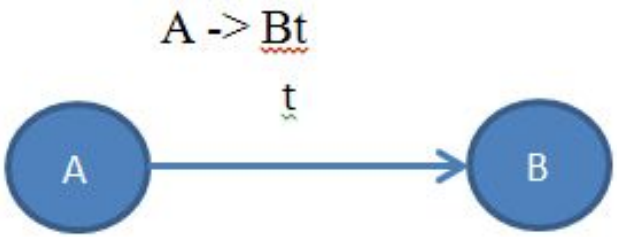
Для леволинейной грамматики



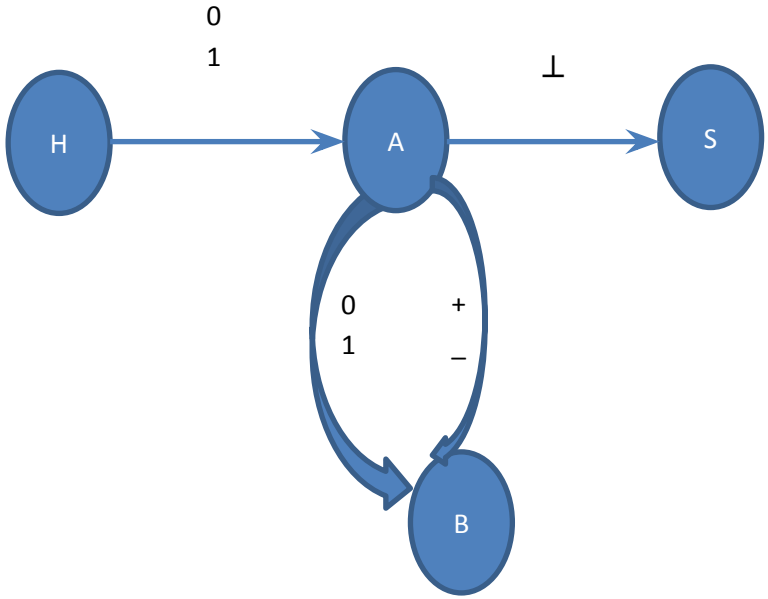


Для праволинейной грамматики





3.8.2 Построение леволинейной грамматики на основе КА



Шаг 1. VT = V= {0,1,+,-, ⊥}

Шаг 2. VN = {S, A, B}

Шаг 3.

P: S -> A ⊥

A -> A0 | A1 | B0 | B1 | 0 | 1

B -> A+ | A-

Шаг 4. S = S

V= {0,1,+,-, ⊥}

Q = {S, A, B, H}

q_0 = H

F= {S}

δ(B,1) = {A}
δ(B,0) = {A}
δ(A,+) = {B}
δ(A,-) = {B}

δ(A,0) = {A}
δ(A,1) = {A}
δ(H,0) = {A}
δ(H,1) = {A}

δ(A, ⊥) = {S}