Московский авиационный институт

(национальный исследовательский университет)

Институт «Компьютерные науки и прикладная математика»

Лабораторные работы

по курсу

«Системы программирования»

IV семестр

1. Спроектировать грамматику по паттерн-модели регулярного языка.
2. Преобразовать спроектированную грамматику в конечный автомат, составить диаграмму переходов КА и реализовать.
3. Определить свойства КА. Изучить алгоритм преобразования НДКА в ДКА.

4. Устранить из КС-грамматики бесполезные символы и ε–правила.
5. Устранить из КС-грамматики цепные правила и устранить левую рекурсию.
6. Определить форму КС-грамматики и сделать ее приведение.
7. Спроектировать МП-автомат для приведенной КС-грамматики.
8. Реализовать МП-автомат для приведенной КС-грамматики.

9. Для LL(k) анализатора построить управляющую таблицу M.
10. Аналитически написать правила вывода для цепочки LL(k) анализатора.
11. Реализовать управляющую таблицу M Для LL(k) анализатора.

12. Построить множество LR(0)-таблиц не содержащих ε–правила.
13. Для LR(k)-грамматики спроектировать матрицу oblow.
14. Определить функции перехода g(X).
15. Определить функцию переноса-свертки f(u).
16. Для функции перехода g(X) и функции переноса-свертки f(u) спроектировать управляющую таблицу.

Студент: Павлов И.Д.
Группа: М8О-207Б-21
Руководитель: Семенов А.С.

Москва, 2023

Практическая работа №1 (1-3 лаб.)

Лабораторные работы №1-2

Формулировка задания:

Спроектировать грамматику для двух заданных паттернов. Составить на основе разработанных регулярных грамматик конечные автоматы, распознающие эквивалентные им языки.

1. pattern = "192\.168\.1\.\d{1, 3}"

Автоматная грамматика:

$L(\text{pattern}) = L(\text{"192\.168\.1\.\d\{1, 3\}"}) = \{\text{"192.168.1.0"}\ ,\ldots,\ \text{"192.168.1.999"}\}$
$G(T, V, P, S_0) = G(\{0, \ldots, 9, .\}, \{S_0, A, \ldots, M\}, \{p_1, p_2, \ldots, p_{16}\}, S_0)$

Правила регулярной грамматики:

p1: $S_0 \to 1A$

p2: $A \to 9B$

p3: $B \to 2C$

p4: $C \to .D$

p5: $D \to 1E$

p6: $E \to 6F$

p7: $F \to 8G$

p8: $G \to .H$

p9: $H \to 1I$

p10: $I \to .J$

p11: $J \to 0K \mid 1K \mid \ldots \mid 9K$

p12: $K \to \varepsilon$

p13: $K \to 0L \mid 1L \mid \ldots \mid 9L$

p14: $L \to \varepsilon$

p15: $L \to 0M \mid 1M \mid \ldots \mid 9M$

p16: $M \to \varepsilon$

Пример цепочек:

$S_0 \Rightarrow^1 1A \Rightarrow^2 19B \Rightarrow^3 192C \Rightarrow^4 192.D \Rightarrow^5 192.1E \Rightarrow^6 192.16F \Rightarrow^7 192.168G \Rightarrow^8 192.168.H \Rightarrow^9 192.168.1I \Rightarrow^{10} 192.168.1.J \Rightarrow^{11} 192.168.1.1K \Rightarrow^{12} 192.168.1.1$

$S_0 \Rightarrow^1 1A \Rightarrow^2 19B \Rightarrow^3 192C \Rightarrow^4 192.D \Rightarrow^5 192.1E \Rightarrow^6 192.16F \Rightarrow^7 192.168G \Rightarrow^8 192.168.H \Rightarrow^9 192.168.1I \Rightarrow^{10} 192.168.1.J \Rightarrow^{11} 192.168.1.2K \Rightarrow^{13} 192.168.1.25L \Rightarrow^{15} 192.168.1.255M \Rightarrow^{16} 192.168.1.255$

Конечный автомат:

$L(KA) = L(G)$

$KA = (Q, \Sigma, \delta, S_0, F)$, где

$Q = \{S_0, A, \ldots, M, q_f\}$, $\Sigma = \{0, 1, \ldots, 9, .\}$, $S_0 = S_0$, $F = \{K, L, M\}$,

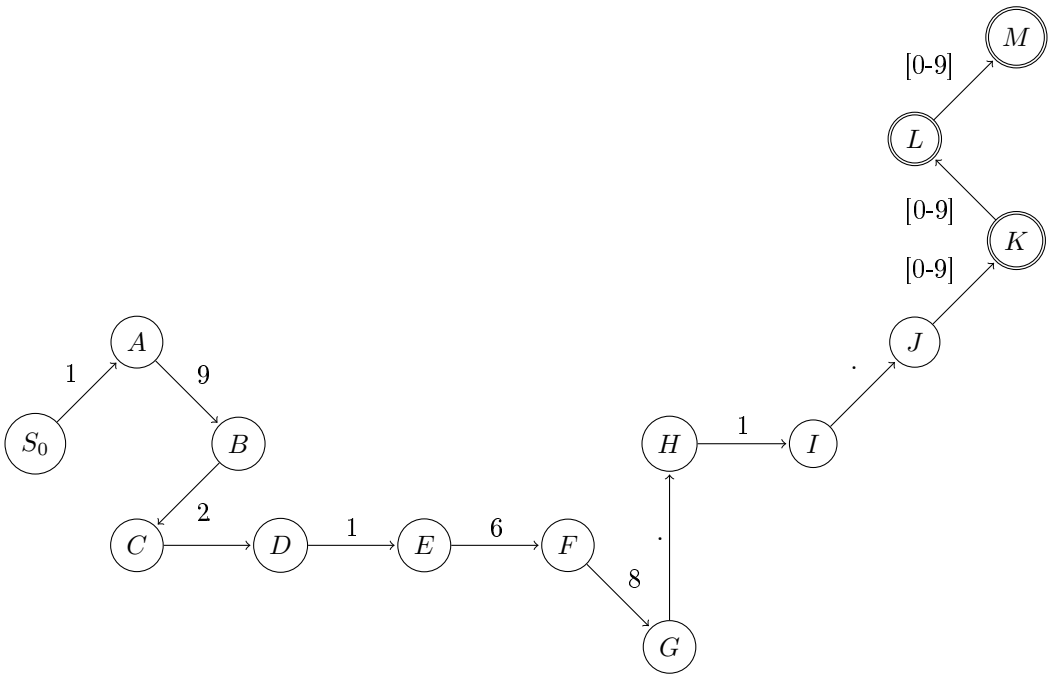
$\delta = \{$

1. $\delta(S_0, 1) = \{A\}$	2. $\delta(A, 9) = \{B\}$
3. $\delta(B, 2) = \{C\}$	4. $\delta(C, .) = \{D\}$
5. $\delta(D, 1) = \{E\}$	6. $\delta(E, 6) = \{F\}$
7. $\delta(F, 8) = \{G\}$	8. $\delta(G, .) = \{H\}$
9. $\delta(H, 1) = \{I\}$	10. $\delta(I, .) = \{J\}$
11. $\delta(J, 0) = \{K\}$	$\ldots$
19. $\delta(J, 9) = \{K\}$	20. $\delta(K, \varepsilon) = \{K\}$
21. $\delta(K, 0) = \{L\}$	$\ldots$
30. $\delta(K, 9) = \{L\}$	31. $\delta(L, \varepsilon) = \{L\}$
32. $\delta(L, 0) = \{M\}$	$\ldots$
41. $\delta(L, 9) = \{M\}$	42. $\delta(M, \varepsilon) = \{M\}$

$\}$

Примеры конфигурации КА:

1. $(S_0, 192.168.1.1) \vdash^1 (A, 92.168.1.1) \vdash^2 (B, 2.168.1.1) \vdash^3 (C, .168.1.1) \vdash^4 (D, 168.1.1) \vdash^5 (E, 68.1.1) \vdash^6 (F, 8.1.1) \vdash^7 (G, .1.1) \vdash^8 (H, 1.1) \vdash^9 (I, .1) \vdash^{10} (J, 1) \vdash^{12} (K, \varepsilon) \vdash^{20} (q_f, \varepsilon)$
2. $(S_0, 192.168.1.255) \vdash^1 (A, 92.168.1.255) \vdash^2 (B, 2.168.1.255) \vdash^3 (C, .168.1.255) \vdash^4 (D, 168.1.255) \vdash^5 (E, 68.1.255) \vdash^6 (F, 8.1.255) \vdash^7 (G, .1.255) \vdash^8 (H, 1.255) \vdash^9 (I, .255) \vdash^{10} (J, 255) \vdash^{13} (K, 55) \vdash^{26} (L, 5) \vdash^{37} (M, \varepsilon) \vdash^{42} (q_f, \varepsilon)$



Лемма о накачке:

$$\forall L \subseteq \sum^*(regular(L)) \Rightarrow$$
$$(\exists p \geq 1$$
$$(\forall w \in L((|w| \geq p) \Rightarrow$$
$$(\exists x, y, z \in \sum^*(w = xyz \Rightarrow$$
$$(|y| \geq 1 \wedge |xy| \leq p \wedge i \geq 0 \wedge (xy^i z \in L))$$
$$))))))$$

```
Введите строчку для проверки: 192.168.1.255
Проверить цепочку:
1 - на принадлежность регулярному языку
2 - на принадлежность кс-языку
3 - найти все повторения
Введите 1 или 2 или 3: 1

Для цепочки: 192.168.1.255
192.168.1.2 5^2
Цепочка принадлежит регулярному языку
Вывести все повторения? (1 - да, 0 - нет): 1

192.168.1.2 5^2
```

2. pattern = "(?i)(\W|^)(baloney|darn|drat|fooey|gosh\sdarnit|heck)(\W|$)"

Автоматная грамматика:

$L(\text{pattern}) = L(\text{"}(?i)(\backslash W|\wedge)(baloney|darn|drat|fooey|gosh\backslash sdarnit|heck)(\backslash W|\$)\text{"})=$
$= \{\text{"baloney", "Baloney" , ...}\}$
$G(T, V, P, S_0) = G(\{B, b, A, a, L, l, O, o, N, n, E, e, Y, y, D, d, R, r, N, n, T, t, F, f, G, g, S, s, H, h, \backslash s, I, i, C, c, \backslash W\},$
$\{S_0, A, \ldots, Z, AA, q_f\}, \{p_1, p_2, \ldots, p_{34}\}, S_0),$
где $\backslash W$ это все символы, не являющиеся буквой и цифрой; $\backslash s$ это пробел

Правила регулярной грамматики:

p1: $S_0 \to \backslash W S_0$

p2: $S_0 \to bA \mid BA$

p3: $S_0 \to dG \mid DG$

p4: $S_0 \to gO \mid GO$

p5: $S_0 \to fL \mid FL$

p6: $S_0 \to hY \mid HY$

p7: $A \to aB \mid AB$

p8: $B \to lC \mid LC$

p9: $C \to oD \mid OD$

p10: $D \to nE \mid NE$

p11: $E \to eF \mid EF$

p12: $F \to yq_f \mid Yq_f$

p13: $G \to aH \mid AH$

p14: $G \to rJ \mid RJ$

p15: $H \to rI \mid RI$

p16: $I \to nq_f \mid Nq_f$

p17: $J \to aK \mid AK$

p18: $K \to tq_f \mid Tq_f$

p19: $L \to oM \mid OM$

p20: $M \to oE \mid OE$

p21: $O \to oP \mid OP$

p22: $P \to sQ \mid SQ$

p23: $Q \to hR \mid HR$

p24: $R \to \backslash sS$

p25: $S \to dT \mid DT$

p26: $T \to aU \mid AU$

p27: $U \to rV \mid RV$

p28: $V \to nW \mid NW$

p29: $W \to iX \mid IX$

p30: $X \to tQ_f \mid Tq_f$

p31: $Y \to eZ \mid EZ$

p32: $Z \to cAA \mid CAA$

p33: $AA \to kq_f \mid KAG$

p34: $q_f \to \backslash Wq_f \mid \varepsilon$

Пример цепочек:

$$S_0 \Rightarrow^{1} !S_0 \Rightarrow^{2} !BA \Rightarrow^{7} !BaB \Rightarrow^{8} !BaLC \Rightarrow^{9} !BaLOD \Rightarrow^{10} !BaLOnE \Rightarrow^{11} !BaLOnEF \Rightarrow^{12} !BaLOnEy$$

$$S_0 \Rightarrow^{5} fL \Rightarrow^{19} fOM \Rightarrow^{20} fOoE \Rightarrow^{11} fOoEF \Rightarrow^{12} fOoEy$$

Конечный автомат:

$L(KA) = L(G)$

$KA = (Q, \Sigma, \delta, S_0, F)$, где

$Q = \{S_0, A, \ldots, Z, AA, q_f\}$, $\Sigma = \{B, b, A, a, L, l, O, o, N, n, E, e, Y, y, D, d, R, r, N, n, T, t, F, f, G, g, S, s, H, h, \backslash s, I, i, C, c, \backslash W\}$, $S_0 = S_0$, $F = q_f$,

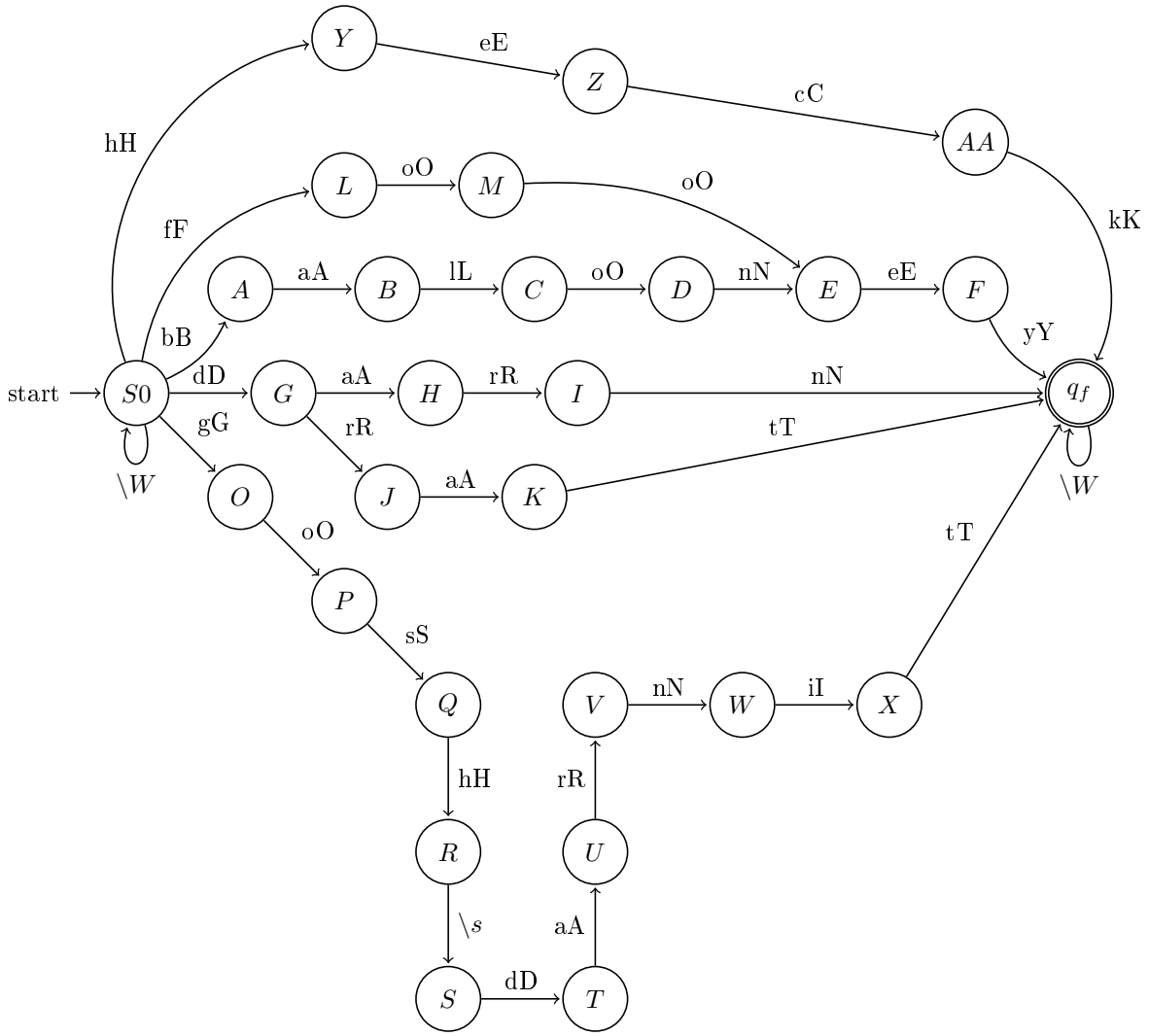
$\delta = \{$

1. $\delta(S_0, \backslash W) = \{S_0\}$
2. $\delta(S_0, b) = \{A\}$
3. $\delta(S_0, B) = \{A\}$
4. $\delta(S_0, d) = \{G\}$
5. $\delta(S_0, D) = \{G\}$
6. $\delta(S_0, g) = \{O\}$
7. $\delta(S_0, G) = \{O\}$
8. $\delta(S_0, f) = \{L\}$
9. $\delta(S_0, F) = \{L\}$
10. $\delta(S_0, h) = \{Y\}$
11. $\delta(S_0, H) = \{Y\}$
12. $\delta(S_0, H) = \{Y\}$
13. $\delta(A, a) = \{B\}$
14. $\delta(A, A) = \{B\}$
15. $\delta(B, l) = \{C\}$
16. $\delta(B, L) = \{C\}$
17. $\delta(C, o) = \{D\}$
18. $\delta(C, O) = \{D\}$
19. $\delta(D, n) = \{E\}$
20. $\delta(D, N) = \{E\}$
21. $\delta(E, e) = \{F\}$
22. $\delta(E, E) = \{F\}$
23. $\delta(F, y) = \{q_f\}$
24. $\delta(E, Y) = \{q_f\}$
25. $\delta(G, a) = \{H\}$
26. $\delta(G, A) = \{H\}$
27. $\delta(H, r) = \{I\}$
28. $\delta(H, R) = \{I\}$
29. $\delta(I, n) = \{q_f\}$
30. $\delta(I, n) = \{q_f\}$
31. $\delta(G, r) = \{J\}$
32. $\delta(G, R) = \{J\}$
33. $\delta(J, a) = \{K\}$
34. $\delta(J, A) = \{K\}$
35. $\delta(K, t) = \{q_f\}$
36. $\delta(K, T) = \{q_f\}$
37. $\delta(L, o) = \{M\}$
38. $\delta(L, O) = \{M\}$
39. $\delta(M, o) = \{E\}$
40. $\delta(M, O) = \{E\}$
41. $\delta(O, o) = \{P\}$
42. $\delta(O, o) = \{P\}$
43. $\delta(P, s) = \{Q\}$
44. $\delta(P, S) = \{Q\}$
45. $\delta(Q, h) = \{R\}$
46. $\delta(Q, H) = \{R\}$
47. $\delta(R, \backslash s) = \{S\}$
48. $\delta(S, d) = \{T\}$
49. $\delta(S, D) = \{T\}$
50. $\delta(T, a) = \{U\}$
51. $\delta(T, A) = \{U\}$
52. $\delta(U, r) = \{V\}$
53. $\delta(U, R) = \{V\}$
54. $\delta(V, n) = \{W\}$
55. $\delta(V, N) = \{W\}$
56. $\delta(W, i) = \{X\}$
57. $\delta(W, I) = \{X\}$
58. $\delta(X, t) = \{q_f\}$
59. $\delta(X, t) = \{q_f\}$
60. $\delta(Y, e) = \{Z\}$
61. $\delta(Y, E) = \{Z\}$
62. $\delta(Z, c) = \{AA\}$
63. $\delta(Z, C) = \{AA\}$
64. $\delta(AA, k) = \{q_f\}$
65. $\delta(AA, K) = \{q_f\}$
66. $\delta(q_f, \backslash W) = \{q_f\}$

}

Примеры конфигурации КА:

1. $(S_0, !BaLOnEy) \vdash^1 (S_0, BaLOnEy) \vdash^3 (A, aLOnEy) \vdash^{13} (B, LOnEy) \vdash^{16} (C, OnEy) \vdash^{18} (D, nEy) \vdash^{19} (E, Ey) \vdash^{22} (F, y) \vdash^{23} (q_f, \varepsilon)$



Лемма о накачке:

$$\forall L \subseteq \sum{}^*(regular(L)) \Rightarrow$$
$$(\exists p \geq 1$$
$$(\forall w \in L((|w| \geq p) \Rightarrow$$
$$(\exists x, y, z \in \sum{}^*(w = xyz \Rightarrow$$
$$(|y| \geq 1 \wedge |xy| \leq p \wedge i \geq 0 \wedge (xy^i z \in L))$$
$$))))))$$

```
Введите строчку для проверки: !!!BaLoNeY!!!
Проверить цепочку:
1 - на принадлежность регулярному языку
2 - на принадлежность кс-языку
3 - найти все повторения
Введите 1 или 2 или 3: 1

Для цепочки: !!!BaLoNeY!!!
 !^2 !BaLoNeY!!!
Цепочка принадлежит регулярному языку
Вывести все повторения? (1 - да, 0 - нет): 1

 !^2 !BaLoNeY!!!
!!!BaLoNeY !^3
 !^2 !BaLoNeY !^3
 !^2 !BaLoNeY !^2 !
```

Лабораторная работа №3

Формулировка задания:

Реализовать конечные автоматы, составленные в ЛР №2

```
FSAutomate[] automats = new FSAutomate[] {
  new FSAutomate(
  new List<Symbol>() { "S01", "A1", "B1", "C1", "D1", "E1", "F1", "G1",
    "H1", "I1", "J1", "K1", "L1", "M1" },
  new List<Symbol>() { "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "."  },
  new List<Symbol>() { "K1", "L1", "M1" },
  "S01"
  ),
  new FSAutomate(
  new List<Symbol>() { "S02", "A2", "B2", "C2", "D2", "E2", "F2", "G2", "H2", "I2",
       "J2",
    "K2", "L2", "M2", "N2", "O2", "P2", "Q2", "R2", "S2", "T2", "U2",
    "V2", "W2", "X2", "Y2", "Z2", "AA2", "qf2"},
  new List<Symbol>() { "B", "b", "A", "a", "L", "l", "O", "o", "N", "n", "E", "e",
    "Y", "y", "D", "d", "R", "r", "N", "n", "T", "t", "F", "f",
    "G", "g", "S", "s", "H", "h", " ", "I", "i", "C", "c", @"\W" },
  new List<Symbol>() { "qf2" },
  "S02"
  ),
};

string[] numbers = { "1", "2", "3", "4", "5", "6", "7", "8", "9", "0" };

automats[0].AddRule("S01", "1", "A1");
automats[0].AddRule("A1", "9", "B1");
automats[0].AddRule("B1", "2", "C1");
automats[0].AddRule("C1", ".", "D1");
automats[0].AddRule("D1", "1", "E1");
automats[0].AddRule("E1", "6", "F1");
automats[0].AddRule("F1", "8", "G1");
automats[0].AddRule("G1", ".", "H1");
```

```
31      automats[0].AddRule("H1", "1", "I1");
32      automats[0].AddRule("I1", ".", "J1");
33      foreach (string number in numbers)
34      {
35        automats[0].AddRule("J1", number, "K1");
36      }
37      foreach (string number in numbers)
38      {
39        automats[0].AddRule("K1", number, "L1");
40      }
41      foreach (string number in numbers)
42      {
43        automats[0].AddRule("L1", number, "M1");
44      }
45
46      automats[1].AddRule("S02", @"\W", "S02");
47      automats[1].AddRule("S02", "b", "A2");
48      automats[1].AddRule("S02", "B", "A2");
49      automats[1].AddRule("A2", "a", "B2");
50      automats[1].AddRule("A2", "A", "B2");
51      automats[1].AddRule("B2", "l", "C2");
52      automats[1].AddRule("B2", "L", "C2");
53      automats[1].AddRule("C2", "o", "D2");
54      automats[1].AddRule("C2", "O", "D2");
55      automats[1].AddRule("D2", "n", "E2");
56      automats[1].AddRule("D2", "N", "E2");
57      automats[1].AddRule("E2", "e", "F2");
58      automats[1].AddRule("E2", "E", "F2");
59      automats[1].AddRule("F2", "y", "qf2");
60      automats[1].AddRule("F2", "Y", "qf2");
61      automats[1].AddRule("S02", "d", "G2");
62      automats[1].AddRule("S02", "D", "G2");
63      automats[1].AddRule("G2", "a", "H2");
64      automats[1].AddRule("G2", "A", "H2");
65      automats[1].AddRule("H2", "r", "I2");
66      automats[1].AddRule("H2", "R", "I2");
67      automats[1].AddRule("I2", "n", "qf2");
68      automats[1].AddRule("I2", "N", "qf2");
69      automats[1].AddRule("G2", "r", "J2");
70      automats[1].AddRule("G2", "R", "J2");
71      automats[1].AddRule("J2", "a", "K2");
72      automats[1].AddRule("J2", "A", "K2");
73      automats[1].AddRule("K2", "t", "qf2");
74      automats[1].AddRule("K2", "T", "qf2");
75      automats[1].AddRule("S02", "f", "L2");
76      automats[1].AddRule("S02", "F", "L2");
77      automats[1].AddRule("L2", "o", "M2");
78      automats[1].AddRule("L2", "O", "M2");
79      automats[1].AddRule("M2", "o", "E2");
80      automats[1].AddRule("M2", "O", "E2");
81      automats[1].AddRule("S02", "g", "O2");
82      automats[1].AddRule("S02", "G", "O2");
83      automats[1].AddRule("O2", "o", "P2");
```

```
84      automats[1].AddRule("O2", "O", "P2");
85      automats[1].AddRule("P2", "s", "Q2");
86      automats[1].AddRule("P2", "S", "Q2");
87      automats[1].AddRule("Q2", "h", "R2");
88      automats[1].AddRule("Q2", "H", "R2");
89      automats[1].AddRule("R2", " ", "S2");
90      automats[1].AddRule("S2", "d", "T2");
91      automats[1].AddRule("S2", "D", "T2");
92      automats[1].AddRule("T2", "a", "U2");
93      automats[1].AddRule("T2", "A", "U2");
94      automats[1].AddRule("U2", "r", "V2");
95      automats[1].AddRule("U2", "R", "V2");
96      automats[1].AddRule("V2", "n", "W2");
97      automats[1].AddRule("V2", "N", "W2");
98      automats[1].AddRule("W2", "i", "X2");
99      automats[1].AddRule("W2", "I", "X2");
100     automats[1].AddRule("X2", "t", "qf2");
101     automats[1].AddRule("X2", "T", "qf2");
102     automats[1].AddRule("S02", "h", "Y2");
103     automats[1].AddRule("S02", "H", "Y2");
104     automats[1].AddRule("Y2", "e", "Z2");
105     automats[1].AddRule("Y2", "E", "Z2");
106     automats[1].AddRule("Z2", "c", "AA2");
107     automats[1].AddRule("Z2", "C", "AA2");
108     automats[1].AddRule("AA2", "k", "qf2");
109     automats[1].AddRule("AA2", "K", "qf2");
110     automats[1].AddRule("qf2", @"\W", "qf2");
111
112     var dka1 = new FSAutomate();
113     dka1.BuildDeltaDKAutomate(automats[0], false);
114     var dka2 = new FSAutomate();
115     dka2.BuildDeltaDKAutomate(automats[1], false);
116     var automats1 = new FSAutomate[] { dka1, dka2};
117     var exectionOrder = new FSAutomate[] { dka1, dka2 };
118     string[] names = { "KA1", "KA2" };
119
120     Console.WriteLine();
121
122     Console.WriteLine();
123     Console.WriteLine("Enter number of automate:");
124     int automateNumber = -1;
125     automateNumber = int.Parse(Console.ReadLine());
126     while (automateNumber != 1 && automateNumber != 2)
127     {
128       Console.WriteLine("Bad input, try again");
129       Console.WriteLine("Enter number of automate:");
130       automateNumber = int.Parse(Console.ReadLine());
131     }
132     Console.WriteLine("You chose automate number {0}", automateNumber);
133     --automateNumber;
134     if (automateNumber == 0)
135     {
136       Console.WriteLine("Enter line (\"192\\.168\\.1\\.\\d{1, 3}\", for ex.:
```

```
            192.168.1.1) to execute");
137         Console.WriteLine("or \"exit\" to exit:");
138       }
139       else if (automateNumber == 1)
140       {
141         Console.WriteLine("Enter line (\"(?i)(\\W|^)(baloney|darn|drat|fooey|gosh\\
              sdarnit|heck)(\\W|$)\", for ex.: drat) to execute");
142         Console.WriteLine("or \"exit\" to exit:");
143       }
144
145       string input;
146       while ((input = Console.ReadLine()) != "exit")
147       {
148         automats[automateNumber].Execute(input);
149         Console.WriteLine("Enter line to execute or \"exit\" to exit:");
150       }
```

Примечание: для автомата №2 был дописан код библиотеки RagLib, отвечающий за регулярное выражение $\backslash W$:

```
1     using System;
2     using System.Collections.Generic;
3     using System.Linq;
4     using System.Collections;
5     using RaGlib.Core;
6     using RaGlib.Automata;
7     using System.Text.RegularExpressions;
8
9     namespace RaGlib {
10      public class FSAutomate : Automate {
11        public FSAutomate(List<Symbol> Q, List<Symbol> Sigma, List<Symbol> F, string q0
              ) : base(Q, Sigma, F, q0) {}
12
13        public void AddRule(string state, List<Symbol> terms, string nextState)
14        {
15          foreach (var term in terms)
16          {
17            this.Delta.Add(new DeltaQSigma(state, term.ToString(), new List<Symbol> {
                  new Symbol(nextState) }));
18          }
19        }
20
21        public void AddRule(string state, string term, string nextState)
22        {
23          var regexOfRange = new Regex(@"\[.-.\]");
24          var regexSet = new Regex(@"\[.+\]");
25          //var regexNoAlph = new Regex(@"\W");
26          bool regexNoAlph = false;
27          if (term == @"\W")
28          {
29            regexNoAlph = true;
30          }
```

```
31          Match match1 = regexOfRange.Match(term);
32          Match match2 = regexSet.Match(term);
33          //Match match3 = regexNoAlph.Match(term);
34
35          if (match1.Success)
36          {
37            string res = match1.Value;
38            var inList = new List<Symbol>();
39            for (char i = res[1]; i <= res[3]; ++i) //[0-9] or [a-z]
40            {
41              var j = new Symbol();
42              j = i.ToString();
43              inList.Add(j);
44            }
45            this.AddRule(state, inList, nextState);
46          }
47          else if (match2.Success)
48          {
49            string res = match2.Value;
50            var inList = new List<Symbol>();
51            for (int i = 1; i < res.Length - 1; ++i) //[0123456789]
52            {
53              char s = res[i];
54              var j = new Symbol();
55              j = s.ToString();
56              inList.Add(j);
57            }
58            this.AddRule(state, inList, nextState);
59          }
60          else if (regexNoAlph)
61          {
62            var inList = new List<Symbol>();
63            for (int i = 0; i < 128; ++i)
64            {
65              char s = (char)i;
66              if (!Char.IsLetterOrDigit(s)) {
67                var j = new Symbol();
68                j = s.ToString();
69                inList.Add(j);
70              }
71            }
72            this.AddRule(state, inList, nextState);
73          }
74          else
75          {
76            this.Delta.Add(new DeltaQSigma(state, term, new List<Symbol> { new Symbol(
                nextState) }));
77          }
78        }
79
80        public FSAutomate() : base() {}
81        public void Execute(string chineSymbol) {
82          var currState = this.Q0;
```

```
83          int flag = 0;
84          int i = 0;
85          for (; i < chineSymbol.Length; i++) {
86            flag = 0;
87            foreach (var d in this.Delta) {
88              if (d.LHSQ == currState && d.LHSS == chineSymbol.Substring(i, 1))
89              {
90                currState = d.RHSQ[0].symbol;
91                flag = 1;
92                break;
93              }
94            }
95            if (flag == 0) break;
96          } // end for
97
98          Console.WriteLine("Length: " + chineSymbol.Length);
99          Console.WriteLine(" i :" + i.ToString());
100         Debug("curr", currState.symbol);
101         if (this.F.Contains(currState) && i == chineSymbol.Length)
102         Console.WriteLine("chineSymbol belongs to language");
103         else
104         Console.WriteLine("chineSymbol doesn't belong to language");
105       } // end Execute
106
107       public bool Execute_FSA(string chineSymbol) {  //
108         var currState = this.Q0;
109         int flag = 0;
110         int i = 0;
111         for (; i<chineSymbol.Length; i++) {
112           flag=0;
113           foreach (var d in this.Delta) { // var d
114             if (d.LHSQ == currState && d.LHSS==chineSymbol.Substring(i,1)) {
115               currState = d.RHSQ[0];
116               flag=1;
117               break;
118             }
119           }
120           if (flag==0) break;
121         } // end for
122
123         // Console.WriteLine("Length: " + chineSymbol.Length);
124         //Console.WriteLine(" i :" + i.ToString());
125         //Debug("curr", currState);
126         return (this.F.Contains(currState) && i==chineSymbol.Length);
127       } // end Execute_FSA
128
129     } // KAutomate
130
131   }
```

перевод из НДКА в ДКА

```
1     var dka11 = new FSAutomate();
2     Console.WriteLine("Choose the automat");
```

```
3       int n;
4       n = int.Parse(Console.ReadLine());
5       if (n == 1)
6       {
7         dka11.BuildDeltaDKAutomate(automats[0], false);
8         dka11.DebugAuto();
9         Console.WriteLine("Enter line to execute:");
10        dka11.Execute(Console.ReadLine());
11      } else if (n == 2)
12      {
13        dka11.BuildDeltaDKAutomate(automats[1], false);
14        dka11.DebugAuto();
15        Console.WriteLine("Enter line to execute:");
16        dka11.Execute(Console.ReadLine());
17      } else
18      {
19        Console.WriteLine("Wrong number of automate");
20      }
```

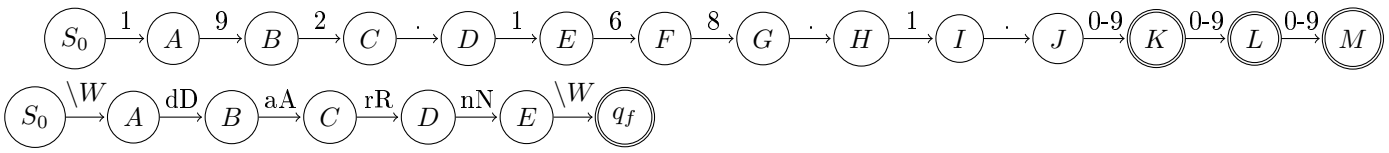
Пример работы программы:

```
Были построены автоматы:
KA1: "192\.168\.1\.\d{1, 3}"
KA2: "(?i)(\W|^)(baloney|darn|drat|fooey|gosh\sdarnit|heck)(\W|$)"

Enter number of automate:
1
You chose automate number 1
Enter line ("192\.168\.1\.\d{1, 3}", for ex.: 192.168.1.1) to execute
or "exit" to exit:
192.168.1.255
Length: 13
 i :13
curr: M1
chineSymbol belongs to language
Enter line to execute or "exit" to exit:
192.168.1.1
Length: 11
 i :11
curr: K1
chineSymbol belongs to language
```

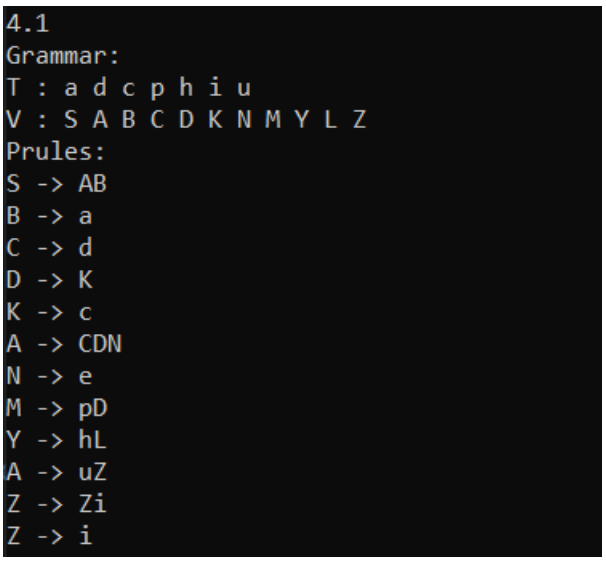
```
Были построены автоматы:
KA1: "192\.168\.1\.\d{1, 3}"
KA2: "(?i)(\W|^)(baloney|darn|drat|fooey|gosh\sdarnit|heck)(\W|$)"

Enter number of automate:
2
You chose automate number 2
Enter line ("(?i)(\W|^)(baloney|darn|drat|fooey|gosh\sdarnit|heck)(\W|$)", for ex.: drat) to execute
or "exit" to exit:
!!!BaLoneY!!@
Length: 13
 i :13
curr: qf2
chineSymbol belongs to language
Enter line to execute or "exit" to exit:
FooEy
Length: 5
 i :5
curr: qf2
chineSymbol belongs to language
```

```
Automate definition:
Q: S01  A1 B1 C1 D1 E1 F1 G1 H1 I1 J1 K1 L1 M1
Sigma: 0 1 2 3 4 5 6 7 8 9 .
Q0: S01
F: K1 L1 M1
DeltaList:
 delta(S01,1,-> (A1
 delta(A1,9,-> (B1
 delta(B1,2,-> (C1
 delta(C1,.,-> (D1
 delta(D1,1,-> (E1
 delta(E1,6,-> (F1
 delta(F1,8,-> (G1
 delta(G1,.,-> (H1
 delta(H1,1,-> (I1
 delta(I1,.,-> (J1
 delta(J1,0,-> (K1
 delta(K1,0,-> (L1
 delta(L1,0,-> (M1
 delta(L1,1,-> (M1
 delta(L1,2,-> (M1
 delta(L1,3,-> (M1
 delta(L1,4,-> (M1
 delta(L1,5,-> (M1
 delta(L1,6,-> (M1
 delta(L1,7,-> (M1
 delta(L1,8,-> (M1
 delta(L1,9,-> (M1
 delta(K1,1,-> (L1
 delta(K1,2,-> (L1
 delta(K1,3,-> (L1
 delta(K1,4,-> (L1
 delta(K1,5,-> (L1
 delta(K1,6,-> (L1
 delta(K1,7,-> (L1
 delta(K1,8,-> (L1
 delta(K1,9,-> (L1
 delta(J1,1,-> (K1
 delta(J1,2,-> (K1
 delta(J1,3,-> (K1
 delta(J1,4,-> (K1
 delta(J1,5,-> (K1
 delta(J1,6,-> (K1
 delta(J1,7,-> (K1
 delta(J1,8,-> (K1
 delta(J1,9,-> (K1
Enter line to execute:
192.168.1.1
Length: 11
 i :11
```

```
Automate definition:
Q: S02 A2  B2 C2 D2 E2 F2 qf2 G2 H2 I2 J2 K2 L2 M2 O2 P2 Q2 R2 S2 T2 U2 V2 W2 X2 Y2 Z2 AA2
Sigma: B b A a L l O o N n E e Y y D d R r N n T t F f G g S s H h   I i C c \W
Q0: S02
F: qf2
DeltaList:
 delta(S02,B,-> (A2
 delta(A2,A,-> (B2
 delta(B2,L,-> (C2
 delta(C2,O,-> (D2
 delta(D2,N,-> (E2
 delta(E2,E,-> (F2
 delta(F2,Y,-> (qf2
 delta(F2,y,-> (qf2
 delta(E2,e,-> (F2
 delta(D2,n,-> (E2
 delta(D2,N,-> (E2
 delta(D2,n,-> (E2
 delta(C2,o,-> (D2
 delta(B2,l,-> (C2
 delta(A2,a,-> (B2
 delta(S02,b,-> (A2
 delta(S02,D,-> (G2
 delta(G2,A,-> (H2
 delta(H2,R,-> (I2
 delta(I2,N,-> (qf2
 delta(I2,n,-> (qf2
 delta(I2,N,-> (qf2
 delta(I2,n,-> (qf2
 delta(H2,r,-> (I2
 delta(G2,a,-> (H2
 delta(G2,R,-> (J2
 delta(J2,A,-> (K2
 delta(K2,T,-> (qf2
 delta(K2,t,-> (qf2
 delta(J2,a,-> (K2
 delta(G2,r,-> (J2
 delta(S02,d,-> (G2
 delta(S02,F,-> (L2
 delta(L2,O,-> (M2
 delta(M2,O,-> (E2
 delta(M2,o,-> (E2
 delta(L2,o,-> (M2
 delta(S02,f,-> (L2
 delta(S02,G,-> (O2
 delta(O2,O,-> (P2
 delta(P2,S,-> (Q2
 delta(Q2,H,-> (R2
 delta(R2, ,-> (S2
 delta(S2,D,-> (T2
```


S_0 1 A 9 B 2 C . D 1 E 6 F 8 G . H 1 I . J 0-9 K 0-9 L 0-9 M

S_0 \W A dD B aA C rR D nN E \W q_f



Практическая работа №2 (4-8 лаб.)

Заданная грамматика:

G = {(a, d, c, ε, p, h, i, u), (S, A, B, C, D, K, N, M, Y, L, Z), ($S \to AB$), ($B \to a$), ($C \to d$), ($D \to K$), ($K \to c$), ($A \to CDN$), ($N \to \varepsilon$), ($M \to pD$), ($Y \to hL$), ($A \to uZ$), ($Z \to Zi$), ($Z \to i$), S}

Лабораторная работа №4

Формулировка задания:

Устранить из КС-грамматики бесполезные символы и ε–правила

Определение. Нетерминальный символ $A \in T \cup V$ называется бесполезным в КС-грамматике G, если он *непроизводящий* или *недостижимый*.

Рассмотрим алгоритмы и их применения:

1. Устранение непроизводящих символов.

Определение. Нетерминальный символ $A \in V$ называется производящим, если из него можно вывести терминальную цепочку, т.е. существует вывод $A \Rightarrow^+ \alpha$, где $\alpha \in T^+$. В противном случае символ называется непроизводящим.

1. Составить множество V_p нетерминалов, для которых найдется хотя бы одно правило, правая часть которого не содержит нетерминалов.

2. Если найдено такое правило, что нетерминалы, стоящие в его правой части уже занесены в V_p, то добавить в V_p нетерминал, стоящий в его левой части.

3. Если на шаге 2, V_p больше не пополняется, то V_p содержит все производящие нетерминалы грамматики, а все нетерминалы, не попавшие в него являются непроизводящими.

i = 0
$V_p^i = \emptyset$
do
    i = i + 1;
    foreach ($A \to \alpha \in P$, $\alpha \in (T \cup V_p^{i-1})^+$)
        if ($A \notin V_p^{i-1}$)
            $V_p^i = V_p^{i-1} \cup \{A\}$ //$V_p^i = \{A | (A \to \alpha) \in P, \alpha \in (T \cup V_p^{i-1})^+\}$
        end
    end foreach
while ($V_p^i \neq V_p^{i-1}$)

if ($S \notin V_p^i$) S - непроизводящий символ, то определить $S \in V_p^i$
    n = 0
    foreach ($A \in V_p^i$)
        if $|\alpha| \geq n, (A \to \alpha \in P)$ при $n = 0$ если есть $A \to \varepsilon$
            $n = |\alpha|$
            $S = A$
        end
    end foreach
end

1. $V_p^1 = \{A, B, C, K, M, N, Y, Z\}$ - для этих нетерминалов нашлось хотя бы одно правило, правая часть которых не содержит нетерминалов.
2. $V_p^2 = \{S, A, B, C, D, K, M, N, Z\}$ - если найдено такое правило, что все нетерминалы, стоящие в его правой части уже занесены в V_p, то добавить в V_p нетерминал, стоящий в его левой части.

V_p содержит все производящие нетерминалы грамматики, а все нетерминалы, не попавшие в него, являются непроизводящими ($V - V_p = Y, L$).
Добавляем в P' только правила, в правой части которых только производящие символы:
$P' = \{p_1, \ldots, p_{10}\}$

p_1 $S \to AB$

p_2 $A \to CDN$

p_3 $A \to uZ$

p_4 $B \to a$

p_5 $C \to d$

p_6 $D \to K$

p_7 $K \to c$

p_8 $M \to pD$

p_9 $Z \to Zi$

p_{10} $Z \to i$

p_{11} $N \to \varepsilon$

2. Устранение недостижимых символов.

VT_r - *множество достижимых символов* (нетерминальных и терминальных).
Утверждение. Если терминал A в левой части правила грамматик G является достижимым, то достижимы и все символы α правой части этого правила $A \to \alpha$. $VT_r = \{x | A \to \alpha, x \in \alpha, \alpha \subseteq (V \cup T)^*\}$

1. Построить множество VT_r^i - достижимых терминалов и не терминалов.

2. Построить P'', T', V'.
   $P'' = \emptyset$
   foreach $(A \to \{X_1, X_2, \ldots, X_n\}, A, X_1, X_2, \ldots, X_n \in VT_r^i \in P)$
   $\quad P'' := P'' \cup (A \to \{X_1, X_2, \ldots, X_n\})$
   end foreach

   $T' := T \cap VT_r^i$
   $V' := V \cap VT_r^i$

$VT_r^1 = \{S\}$
$VT_r^2 = \{S, A, B\}$
$VT_r^3 = \{S, A, C, D, N, u, Z, B, a\}$
$VT_r^4 = \{S, A, C, d, D, K, N, \varepsilon, u, Z, i, B, a\}$
$VT_r^5 = \{S, A, C, d, D, K, c, N, \varepsilon, u, Z, i, B, a\}$

Так, не входящие в VT_r символы $\{p\}$ недостижимы $P'' = \{p_1, \ldots, p_{10}\}$

p_1 $S \to AB$

p_2 $A \to CDN$

p_3 $A \to uZ$

p_4 $B \to a$

p_5 $C \to d$

p_6 $D \to K$

p_7 $K \to c$

p_8 $Z \to Zi$

p_9 $Z \to i$

p_{10} $N \to \varepsilon$

Получаем $G' = (\{\varepsilon, i, c, d, a, u\}, \{S, A, D, N, B, C, K, Z\}, P'', S)$

3. Устранить из КС-грамматики ε–правила.

КС-грамматика называется *неукорачивающей* КС-грамматикой (НКС-грамматикой, КС-грамматикой без ε-правил) при условии что P не содержит $S \to \varepsilon$ и S не встречается в правых частях остальных правил.
В грамматике с правилами вида $A \to \varepsilon$ длина выводимой цепочки при переходе из k-го шага к $(k-1)$-му уменьшается. Поэтому грамматики с правилами вида $A \to \varepsilon$ называются укорачивающими.
Восходящий синтаксический разбор в укорачивающих грамматиках сложнее по сравнению с разбором в неукорачивающих грамматиках, т.к. при редукции необходимо отыскать такой фрагмент входной цепочки, в которую можно вставить пустой символ.

Для произвольной КС-грамматики, порождающий язык без пустой цепочки, можно построить эквивалентную неукорачивающую КС-грамматику.

Построение множества V_ε укорачивающих (обнуляемых) нетерминалов:
Утверждение. Нетерминал A - укорачивающий (обнуляемый), если $A \Rightarrow^* \varepsilon$.

1. A - укорачивающий для правила $A \to \varepsilon$. (см. Шаг 1)
2. A - укорачивающий для правила $A \to B_1 B_2 \dots B_k$, если каждый нетерминал B_1 в правиле укорачивающий. (см. Шаг 2)

Вход: КС грамматика $G = (T, V, P, S)$.
Выход: $V_\varepsilon = \{A | A \Rightarrow^+, A \in V\}$
Шаг 1. Постриоить множество V_ε^i - укорачивающих нетерминалов для правил вида $A \to \varepsilon$.
$i = 0$
$V_\varepsilon^i = \emptyset$
foreach $(A \in V)$
    if $(A \in V) \in P$ // $A \to \varepsilon$ правило укорачивающееся
        $V_\varepsilon^i := V_\varepsilon^i \cup \{A\}$
    end
end foreach

$V_\varepsilon^0 = \{N\}$ - множество укорачивающих нетерминалов для правил вида $A \to \varepsilon$

Шаг 2. Построить множество V_ε^i - укорачивающих нетерминалов для правил вида $A \to \varepsilon$. $A \to B_1 B_2 \dots B_k$, если каждый нетерминал B_1 в правиле укорачивающий. $V_\varepsilon^i = V_\varepsilon^{i-1} \cup \{A | (A \to \alpha) \in P$ и $a \in (V_\varepsilon^{i-1})^+\}$. $A \in (V_\varepsilon^{i-1})^+$ - рассматриваются только правила, содержащие в правых частях только нетерминальные укорачивающие символы: $A \to \alpha$, например $A \to CDF$.

do
    $i = i + 1$
    $V_\varepsilon^{i-1} = V_\varepsilon^i$
    foreach $((A \to \alpha) \in P$ и $\alpha \in (V_\varepsilon^{i-1})^+)$
        $V_\varepsilon^i := V_\varepsilon^{i-1} \cup A$
    end
while $(V_\varepsilon^{i-1} \neq V_\varepsilon^i)$

Необходимо построить множество V_ε^1 укорачивающих нетерминалов для правил вида $A \to B_1 B_2 \dots B_k$, если каждый нетерминал B_1 в правиле укорачивающий. Однако в данном примере таких правил нет.
Итак, множество укорачивающих терминалов: $V_\varepsilon = \{N\}$

Алгоритм устранения ε-правил в КС-грамматике основан на использовании множества укорачивающих нетерминалов. Алгоритм преобразует КС-грамматику с ε-правилами в эквивалентную КС-грамматику. Пусть $X_i \in \{X|(T \cup V) \cup \{\varepsilon\}, 0 < i \leq k\}$. Построить из $\{X_1, X_2, \ldots, X_k\}$ множество цепочек, в которых либо присутствует, либо устранен каждый из нетерминалов V_ε.

Вход: $G' = (T, V_\varepsilon^i, P, S)$
Выход: $G'' = (T, V', P', S')$

$P' = 0$
foreach $(A \to X_1, X_2, \ldots, X_k \in P, X_1 \neq \varepsilon)$
  $V'_\varepsilon = \{[X_1, X_2, \ldots, X_k] \cup V_\varepsilon | X_i < X_m\}$ V_ε^i - упорядоченное множество
  if $(V'_\varepsilon = \emptyset)$
    $P' = P' \cup A \to X_1, X_2, \ldots, X_k$
  else
    $Y = \emptyset$

    foreach $a \in \{X_i \ldots X_m \subset V_\varepsilon'^+ \mid |\alpha| \leq |V'_\varepsilon|$ и $X_i < X_m\}$
      foreach $v \in a$
        foreach $(X_i \in \{X_1 X_2 \ldots X_k\}, A \neq X_i)$ правило $A \to A$ бессмысленно
          if $X_i = v$
            $Y_i = \varepsilon$
          end
          $Y = Y \cup Y_i$
        end foreach
        $P' = P' \cup (A \to Y_1, Y_2, \ldots, Y_k)$
      end foreach
    end foreach
  end
end foreach

if $(S \in V_\varepsilon)$
  $V' = V' \cup \{S'\}$
  $P' = P' \cup (S' \to S|\varepsilon)$
end
Положить $G'' = (T, V', P', S')$

$G = (\{a, d, c, p, h, i, u\}, \{S, A, B, C, D, K, Z\}, P', S)$; $P' = \{p_1, \ldots, p_9\}$

p_1 $S \to AB$

p_2 $B \to A$

p_3 $A \to CD$

p_4 $C \to d$

p_5 $D \to K$

p_6 $K \to c$

p_7 $A \to uZ$

p_8 $Z \to Zi$

p_9 $Z \to i$

```
        Delete e-rules:
Executing:
e-rules:
N -> e
NoShortNoTerms : N
V1: : S A B C D K N M Y L Z
        e-rules have benn deleted!
Prules:
S -> AB
B -> a
C -> d
D -> K
K -> c
A -> CDN
A -> CD
M -> pD
Y -> hL
A -> uZ
Z -> Zi
Z -> i
                Deleting unuseful symbols
Executing:
Vp : B C K D A S M Z
Vr : S A B a C D d K c u Z i
T1 : a d c u i
V1 : S A B C D K Z
        Unuseful symbols have been deleted
Prules:
S -> AB
B -> a
A -> CD
C -> d
D -> K
K -> c
A -> uZ
Z -> i
Z -> Zi
```

Лабораторная работа №5

Формулировка задания:

Устранить из КС-грамматики цепные правила и устранить левую рекурсию.

Алгоритм устранения цепных правил. Правило вида $A \to B$, где A и B - нетерминалы называется цепным.

Вход: КС грамматика $G = (T, V, P, S)$.

Выход: Эквивалентная КС грамматика $G' = (T, V, P', S)$ без цепных правил.

Шаг 1. Для каждого правила $X \to A \in P$ построить множество всех цепных символов V_X^i.

foreach $X \in V$ построить множество $V_X = \{A \mid X \Rightarrow^+ A\}$.

$i = 0$

$V_X^i = \{X\}$

do

i++

foreach ($X \in V_X^{i-1}$, $X \to A \in P, A \in V$)

$V_X^i = V_X^{i-1} \cup \{A\}$

end

while ($V'_X \neq V_{i,X}$)

end

Шаг 2. Для каждого цепного правила $A \to R \in P$, построить правила $A \to Y_\rho$ без цепных символов $Y \notin V_A^i$. Добавить в P' не цепные правила.

$P = \emptyset$

foreach ($A \to \alpha R \beta \in P$)

if ($\alpha = \varepsilon, \beta = \varepsilon, R \in V_A^i$)

foreach ($R \in V_A^i$) найти правую не цепную часть правила

if $R \to Y_\rho \in P, Y \notin V_A^i$,

$P' = P' \cup \{A \to Y_\rho\}$

end

end

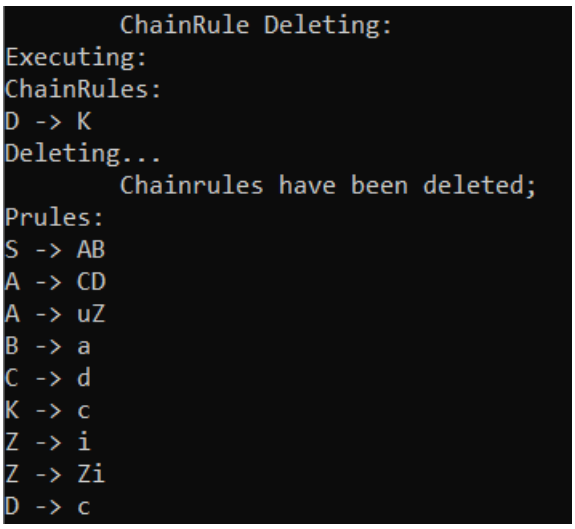
else

$P' = P' \cup \{A \to \alpha R \beta\}$

end

end.

Цепные правила: $D \to K, K \to c$. Заменим на $D \to c$



Устранить из грамматики левую рекурсию
Вход: КС грамматика $G = (T, U, P, S_0)$, без ε-правил (вида $A \to \varepsilon$).
Выход: Эквивалентная приведенная КС грамматика $G' = (T, V, P', S'_0)$.

1. Упорядочить нетерминалы V по возрастанию $V = [A_1, A_2, \ldots, A_n]$.

2. Преобразовать правила $A_i \to \alpha$ так, чтобы цепочка α начиналась либо с терминала, либо с такого A_j, что $j < i, i = 1$.

   foreach i or 1 to n

   foreach j or 1 to $i - 1$

3. Множество A_i правил — это $A_i \to A_i\alpha_1 | \ldots | A_i\alpha_n | \beta_1 | \ldots | \beta_m$, где ни одна цепочка β_j не начинается с A_k, если $k \leq i$. Заменим A_i-правила правилами:

   $$A_i \to \beta_1 | \ldots | \beta_p | \beta_1 A'_i | \ldots | \beta_\rho A'_i$$
   $$A'_i \to \alpha_1 | \ldots | \alpha_m | \alpha_1 A'_i | \ldots | \alpha_\rho A'_i$$

   где A_i и A_{i+1} — новые символы. Правые части всех A_i-правил начинаются теперь с терминала или с A_k для некоторого $k > i$.

4. Если $i = n$, то останов и получена грамматика G', иначе $mj = i, i = i + 1$.

5. Заменить каждое правило вида $A_i \to A_j\alpha$ правилами $A_i \to \beta_1\alpha | \ldots | \beta_m\alpha$, где $A_j \to \beta_1 | \ldots | \beta_m$ — все A_j-правила.
   Так как правая часть каждого A_j-правила начинается уже с терминала или с A_k для $k > j$, то и правая часть каждого A_i-правила будет обладать этим же свойством.

6. Если $j = i - 1$, перейти к шагу 2, иначе $j = j + 1$ и перейти к шагу 4.

Если S_0 присутствовал в языке исходной грамматики, добавим новый начальный символ S'_0 и правила S'_0
Левая рекурсия: $Z \to Zi$. Заменим правило $Z \to Zi$ на правила $Z \to iZ'$, $Z' \to i | iZ'$

```
Left Recursion delete.
Prules:
S -> AB
A -> CD
A -> uZ
B -> a
C -> d
D -> c
K -> c
Z -> i
Z -> iZ'
Z' -> i
Z' -> iZ'
```

```
Normal Grammatic:
T : a d c u i
V : S A B C D K Z Z'
Prules:
S -> AB
A -> CD
A -> uZ
B -> a
C -> d
D -> c
K -> c
Z -> i
Z -> iZ'
Z' -> i
Z' -> iZ'
Start symbol: S
```

Код:

```
var regGr = new Grammar(new List<Symbol>() { "a", "d", "c", "p", "h", "i", "u", "" },
new List<Symbol>() { "S", "A", "B", "C", "D", "K", "N", "M", "Y", "L", "Z" },
"S");

regGr.AddRule("S", new List<Symbol>() { "A", "B" });
regGr.AddRule("B", new List<Symbol>() { "a" });
regGr.AddRule("C", new List<Symbol>() { "d" });
regGr.AddRule("D", new List<Symbol>() { "K" });
regGr.AddRule("K", new List<Symbol>() { "c" });
regGr.AddRule("A", new List<Symbol>() { "C", "D", "N" });
regGr.AddRule("N", new List<Symbol>() { "" });
regGr.AddRule("M", new List<Symbol>() { "p", "D" });
regGr.AddRule("Y", new List<Symbol>() { "h", "L" });
regGr.AddRule("A", new List<Symbol>() { "u", "Z" });
regGr.AddRule("Z", new List<Symbol>() { "Z", "i" });
regGr.AddRule("Z", new List<Symbol>() { "i" });
Console.WriteLine("Grammar:");
regGr.Debug("T", regGr.T);
regGr.Debug("V", regGr.V);
regGr.DebugPrules();

Grammar G1 = regGr.EpsDelete();
```

```
23      G1.DebugPrules();
24
25      Grammar G2 = G1.unUsefulDelete();
26      G2.DebugPrules();
27
28      Grammar G3 = G2.ChainRuleDelete();
29      G3.DebugPrules();
30
31      Grammar G4 = G3.LeftRecursDelete_new6();
32      G4.DebugPrules();
33
34      Console.WriteLine("————————————————————————————————————————————————————");
35      Console.WriteLine("Normal Grammatic:");
36      G4.Debug("T", G4.T);
37      G4.Debug("V", G4.V);
38      G4.DebugPrules();
39      Console.Write("Start symbol: ");
40      Console.WriteLine(G4.S0 + "\n");
```

Лабораторная работа №6

Формулировка задания:

Определить форму КС-грамматики и сделать ее приведение.

Определение. КС грамматика $G = (T, V, P, S)$ называется грамматикой в нормальной форме Грейбах, если в ней нет ε-правил, т.е. правил вида $A \to \varepsilon$. Каждое правило из P отличное от $S \to \varepsilon$, имеет вид $A \to a\alpha$, где $a \in T$, $\alpha \in V^*$.

Также полезно представлять грамматику в нормальной форме Хомского, что позволяет упростить рассмотрение свойств.

Определение. КС грамматика $G = (T, V, P, S)$ называется грамматикой в нормальной форме Хомского, если каждое правило из P имеет один из следующих видов:

1. $A \to BC$, где $A, B, C \in V$;
2. $A \to a$, где $a \in T$;
3. $S \to \varepsilon$, причем S не встречается в правых частях правил.

Приведенная грамматика не является грамматикой в нормальной форме Хомского, так как присутствует правило: $A \to uZ$. Приведенная грамматика является грамматикой в нормальной форме Грейбах, так как в ней нет – ε- правил, и каждое правило имеет вид $A \to a\alpha$, где $a \in T, \alpha \subset V^*$.

Лабораторная работа №7

Формулировка задания:

Спроектировать МП-автомат для приведенной КС-грамматики.

Приведенная грамматика:

$G = (T, V, P, S_0)$, где
$T = \{u, a, d, c, i\}, V = \{S, A, B, C, D, K, Z, Z'\}, S$

P:

p1: $S \to AB$

p2: $A \to CD$

p3: $A \to uZ$

p4: $B \to a$

p5: $C \to d$

p6: $D \to c$

p7: $Z \to i$

p8: $Z \to iZ'$

p9: $Z' \to i$

p10: $Z' \to iZ'$

Вывод цепочки:

$$S \Rightarrow^1 AB \Rightarrow^2 CDB \Rightarrow^3 dca$$

Определение. МП автомат — это семерка объектов $(Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$, где

Q — конечное множество состояний устройства управления;

Σ — конечный алфавит входных символов;

Γ — конечный алфавит магазинных символов;

δ — функция переходов, отображает множества $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma$ в конечных подмножествах множества $Q \times \Gamma^*$, то есть $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to 2^{Q \times \Gamma^*}$;

q_0 — начальное состояние, $q_0 \in Q$;

z_0 — начальный символ магазина, $z_0 \in \Gamma$;

F — множество заключительных состояний, $F \subseteq Q$.

Алгоритм. По КС-грамматике $G = (T, V, P, S)$ можно построить МП автомат $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ следующим образом:

1. Если $A \to \alpha$ — правило грамматики G, то $\delta(q, a, A) = (q, \beta) \mid \alpha \Rightarrow_G \beta a$ для всех $q \in Q$, $a \in \Sigma$.
2. Если $A \to a$ — правило грамматики G, то $\delta(q, a, A) = (q, \epsilon)$ для всех $q \in Q$, $a \in \Sigma$.

Здесь $Q = q$, $\Sigma = T$, $\Gamma = V \cup T$, $q_0 = q$, $Z_0 = S$ и $F = q$.

$L(\text{МП}) = L(G)$

$\text{МП} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$:

$Q = q, \Sigma = T, \Gamma = T \cup V, \delta = \delta, q_0 = q_0, z_0 = S_0, F = \{q\}$

$\text{МП} = (\{q\}, \{u, a, d, c, i\}, u, a, d, c, i, S, A, B, C, D, Z, Z', \delta, q, S, \{q\})$

δ :

1. $\delta(q, \varepsilon, S) = \{q, AB\}$
2. $\delta(q, \varepsilon, A) = \{q, CD\}$
3. $\delta(q, \varepsilon, B) = \{q, a\}$
4. $\delta(q, \varepsilon, C) = \{q, d\}$
5. $\delta(q, \varepsilon, Z') = \{q, i, Z'\}$
6. $\delta(q, \varepsilon, D) = \{q, c\}$
7. $\delta(q, \varepsilon, Z) = \{q, i, Z'\}$
8. $\delta(q, \varepsilon, Z') = \{q, i\}$
9. $\delta(q, a, a) = \{q, \varepsilon\}, \forall a \in \Sigma$

Последовательность тактов МП-автомата для цепочки dca:
$(q, dca, S) \vdash^1 (q, dca, AB) \vdash^2 (q, dca, CDB) \vdash^4 (q, dca, dDB) \vdash^9 (q, ca, DB) \vdash^6 (q, ca, cB) \vdash 9(q, a, B) \vdash^3 (q, \varepsilon, \varepsilon)$

Лабораторная работа №7

Формулировка задания:

Реализовать МП-автомат для приведенной КС-грамматики.

Код:

```
    case "7":
    { // Algorithm Grammar to PDA
      var CFGrammar = new Grammar(new List<Symbol>() { "a", "d", "c", "i", "u", "" },
      new List<Symbol>() { "S", "A", "B", "C", "D", "Z", "Z'" },
      "S");

      CFGrammar.AddRule("S", new List<Symbol>() { "A", "B" });
      CFGrammar.AddRule("A", new List<Symbol>() { "C", "D" });
      CFGrammar.AddRule("A", new List<Symbol>() { "u", "Z" });
      CFGrammar.AddRule("C", new List<Symbol>() { "d" });
      CFGrammar.AddRule("B", new List<Symbol>() { "a" });
      CFGrammar.AddRule("D", new List<Symbol>() { "c" });
      CFGrammar.AddRule("Z", new List<Symbol>() { "i", "Z'" });
      CFGrammar.AddRule("Z'", new List<Symbol>() { "i" });
      CFGrammar.AddRule("Z'", new List<Symbol>() { "i", "Z'" });

      Console.Write("Debug KC-Grammar ");
      CFGrammar.DebugPrules();

      var pda = new PDA(CFGrammar);
      pda.Debug();

      Console.WriteLine("\nEnter the line    :");
      Console.WriteLine(pda.Execute(Console.ReadLine()).ToString());
      break;
    }
```

```
Debug KC-Grammar Prules:
S -> AB
A -> CD
A -> uZ
C -> d
B -> a
D -> c
Z -> iZ'
Z' -> i
Z' -> iZ'
Delta rules:
 delta(q,ε,S) -> (q,A,B)
 delta(q,ε,A) -> (q,C,D)
 delta(q,ε,A) -> (q,u,Z)
 delta(q,ε,C) -> (q,d)
 delta(q,ε,B) -> (q,a)
 delta(q,ε,D) -> (q,c)
 delta(q,ε,Z) -> (q,i,Z')
 delta(q,ε,Z') -> (q,i)
 delta(q,ε,Z') -> (q,i,Z')
 delta(q,a,a) -> (q,ε)
 delta(q,d,d) -> (q,ε)
 delta(q,c,c) -> (q,ε)
 delta(q,i,i) -> (q,ε)
 delta(q,u,u) -> (q,ε)
 delta(q,e,e) -> (q,ε)
 delta(q0,ε,z0) -> (qf,ε)
```

```
Enter the line   :
dca
 step 1
 delta(q,ε,S) -> (q,A,B)
 step 1
 delta(q,ε,A) -> (q,C,D)
 step 1
 delta(q,ε,C) -> (q,d)
 step 1
 delta(q,d,d) -> (q,ε)
 step 2 d  d
 step 3 d
 step 1
 delta(q,ε,D) -> (q,c)
 step 1
 delta(q,c,c) -> (q,ε)
 step 2 c  c
 step 3 c
 step 1
 delta(q,ε,B) -> (q,a)
 step 1
 delta(q,a,a) -> (q,ε)
 step 2 a  a
 step 3 a
True
```

Практическая работа №3 (9-11 лаб.)

Лабораторная работа №9

Формулировка задания:

Для $LL(k)$ анализатора построить управляющую таблицу M.

Определение: КС-грамматика $G = (T, V, P, S)$ без ε-правил называется простой $LL(1)$ грамматикой (s-грамматикой, разделенной грамматикой), если для каждого $v \in V$ все его альтернативы начинаются различными терминальными символами. Единица в названии алгоритма означает, что при чтении анализируемой цепочки, находящейся на входной ленте, входная головка может заглядывать вперед на один символ.

$FIRST(A)$ – это множество первых терминальных символов, которыминачинаются цепочки, выводимые из нетерминала $A \in V$:

$FIRST(A) = \{a \in T | A \Rightarrow^+ a\beta, \beta \in (T \cup V)^*\}$

Обобщим определение множества $FIRST$ так, чтобы его можно былоприменить для правил произвольного вида. Множество $FIRST(\alpha)$ состоит измножества терминальных символов, которыми начинаются цепочки,выводимые из цепочки α.

$FIRST(\alpha) = \{a \in T | S \Rightarrow^+ \alpha \Rightarrow^+ a\beta$, где $\alpha \in (T \cup V)^+, \beta \in (T \cup V)^*\}$

$FOLLOW(A)$ – это множество следующих терминальных символов, которыемогут встретиться непосредственно справа от нетерминала в некоторойсентенциальной форме:

$FOLLOW(A) = \{a \in T | S \Rightarrow^* \alpha A\gamma$ и $a = FIRST(\gamma)\}$

магазин содержит цепочку $Xa \perp$, где Xa – цепочка магазинных символов (X - верхний символ магазина), а символ($\perp$) – специальный символ, называемый маркером дна магазина. Если верхним символом магазина является маркер дна, то магазин пуст. Выходная лентасодержит цепочку номеров правил π, представляющую собой текущее состояние левого разбора.

Исходная грамматика:

$G = (T, V, P, S_0)$, где

$T = \{u, a, d, c, i\}, V = \{S, A, B, C, D, K, Z, Z'\}, S$

P:

p1: $S \to AB$

p2: $A \to CD$

p3: $A \to uZ$

p4: $B \to a$

p5: $C \to d$

p6: $D \to c$

p7: $Z \to i$

p8: $Z \to iZ'$

p9: $Z' \to i$

p10: $Z' \to iZ'$

Сделаем приведение к $LL(1)$ для построения управляющей таблицы, проведя факторизацию.

Метод: для каждого нетерминала находим самый длинный префикс α,общий для двух или большего числа альтернатив. Если $\alpha \neq \varepsilon$, т.е. имеетсянетривиальный общий префикс, заменим все продукции

$A ß \alpha\beta 1|\alpha\beta 2|\ldots|\alpha\beta n|\gamma$, где γ представляет все альтернативы, не начинающиеся с α, продукциями
$A \to \alpha A'|\gamma$
$A' \to \beta 1|\beta 2|\ldots|\beta n$
Здесь A' — новый нетерминал. Выполняем это преобразование до тех пор,пока никакие две альтернативы нетерминала не будут иметь общий префикс.
$G_\varphi = (T, V, P', S)$, где
$T = \{u, a, d, c, i\}, V = \{S, A, B, C, D, K, T, W, Z, Z'\}, S$
P:

p1: $S \to AB$

p2: $A \to CD$

p3: $A \to uZ$

p4: $B \to a$

p5: $C \to d$

p6: $D \to c$

p7: $Z \to iT$

p8: $T \to \varepsilon$

p9: $T \to Z'$

p10: $Z' \to iW$

p11: $W \to Z'$

p12: $W \to \varepsilon$

Алгоритм построения управляющей таблицы M для LL(1)-грамматики
Вход: $LL(1)$-грамматика $G = (T, V, P, S)$
Выход: Управляющая таблица M для грамматики G. Таблица M определяется на множестве $(V \cup T \cup \{\perp\}) \times (T \cup \{\varepsilon\})$ по правилам:

1. Если $A \to \beta$ – правило вывода грамматики с номером i, то $M(, a) = (\beta, i)$ для всех $a \neq \varepsilon$, принадлежащих множеству $FIRST(\beta)$. Если $\varepsilon \in FIRST(\beta)$, то $M(, b) = (\beta, i)$ для всех $\beta \in FOLLOW(A)$.
2. $M(a, a) =$ ВЫБРОС $\forall a \in T$.
3. $M(\perp, \varepsilon) =$ ДОПУСК.
4. В остальных случаях $M(X, a) =$ ОШИБКА для $X(V \cup T \cup \{\perp\})$ и $a \in T \cup \{\varepsilon\}$

	a	d	c	i	u	ε
S		$(AB, 1)$				
A		$(DC, 2)$			$(uZ, 3)$	
B	$(a, 4)$					
C		$(d, 5)$				
D			$(c, 6)$			
Z				$(i, 7)$		
T					$(Z', 9)$	$(\varepsilon, 8)$
W				$(Z', 11)$		$(\varepsilon, 12)$
Z'				$(iW, 10)$		
a	ВЫБРОС					
d		ВЫБРОС				
c			ВЫБРОС			
i				ВЫБРОС		
u					ВЫБРОС	
$\perp$						ДОПУСК

Пустые клетки в таблице означают ОШИБКУ.
Аналитичекое представление для таблицы :

Правило грамматики	Множество	Значение M
$p_1 : S \to AB$	$FIRST(S) = \{d\}$	$M(S, d) = AB, 1$
$p_2 : A \to CD$	$FIRST(A) = \{d\}$	$M(A, d) = CD, 2$
$p_3 : A \to uZ$	$FIRST(A) = \{u\}$	$M(A, u) = uZ, 3$
$p_4 : B \to a$	$FIRST(B) = \{a\}$	$M(B, a) = a, 4$
$p_5 : C \to d$	$FIRST(C) = \{d\}$	$M(C, d) = d, 5$
$p_6 : D \to c$	$FIRST(D) = \{c\}$	$M(D, c) = c, 6$
$p_7 : Z \to iT$	$FIRST(Z) = \{i\}$	$M(Z, i) = i, 7$
$p_8 : T \to \varepsilon$	$FIRST(T) = \{\varepsilon\}$	$M(T, \varepsilon) = \varepsilon, 8$
$p_9 : T \to Z'$	$FIRST(T) = \{i\}$	$M(T, i) = Z', 9$
$p_{10} : Z' \to iW$	$FIRST(Z') = \{i\}$	$M(Z', i) = iW, 10$
$p_{11} : W \to Z'$	$FIRST(W) = \{i\}$	$M(W, i) = Z', 11$
$p_{12} : W \to \varepsilon$	$FIRST(W) = \{\varepsilon\}$	$M(W, \varepsilon) = \varepsilon, 12$

Лабораторная работа №10

Формулировка задания:

Аналитически написать правила вывода для цепочки $LL(k)$ анализатора.

Шаг 1. Алгоритм находится в начальной конфигурации $((dca), S_0 \perp, \varepsilon)$, где $S_0 = S$
Значение управляющей таблицы $M(S, a) = (AB, 1)$, при этом выполняются следующие действия:

- Заменить верхний символ магазина R цепочкой V.
- Не сдвигать читающую головку.
- На выходную ленту поместить номер использованного правила 1.

Шаг 2. Получаем следующие конфигурации:

Текущая конфигурация	Значение M
$(dca, S\perp, \varepsilon)\vdash$	$M(S,d) = AB, 1$
$(dca, AB\perp, 1)\vdash$	$M(A,d) = CD, 2$
$(dca, CDB\perp, 12)\vdash$	$M(C,d) = d, 5$
$(dca, dDB\perp, 125)\vdash$	$M(d,d) =$ ВЫБРОС
$(ca, DB\perp, 125)\vdash$	$M(D,c) = (C, 6)$
$(ca, cB\perp, 1256)\vdash$	$M(c,c) =$ ВЫБРОС
$(a, B\perp, 1256)\vdash$	$M(B,a) = (a, 4)$
$(a, a\perp, 12564)\vdash$	$M(a,a) =$ ВЫБРОС
$(\varepsilon, \perp, 12564)\vdash$	$M(\perp, \varepsilon) =$ ДОПУСК

Лабораторная работа №11

Формулировка задания:

Реализовать управляющую таблицу M Для $LL(k)$ анализатора.

Код:

```
    case "9.1":
   {
     var LL = new Grammar(new List<Symbol>() { "a", "d", "c", "i", "u", "" },
     new List<Symbol>() { "S", "A", "B", "C", "D", "Z", "Z'" },
     "S");

     LL.AddRule("S", new List<Symbol>() { "A", "B" });
     LL.AddRule("A", new List<Symbol>() { "C", "D" });
     LL.AddRule("A", new List<Symbol>() { "u", "Z" });
     LL.AddRule("C", new List<Symbol>() { "d" });
     LL.AddRule("B", new List<Symbol>() { "a" });
     LL.AddRule("D", new List<Symbol>() { "c" });
     LL.AddRule("Z", new List<Symbol>() { "i", "Z'" });
     LL.AddRule("Z'", new List<Symbol>() { "i" });
     LL.AddRule("Z'", new List<Symbol>() { "i", "Z'" });

     var parser = new LLParser(LL);
     string stringChain = Console.ReadLine();

     var chain = new List<Symbol> { };
     foreach (var x in stringChain)
     chain.Add(new Symbol(x.ToString()));
     if (parser.Parse(chain))
     {
       Console.WriteLine(parser.OutputConfigure);
     }
     else
     {
     }
     break;
   }
```

```
9.1
Создадим таблицу. Сначала создадим по столбцу для каждого из этих терминалов:
a, d, c, i, u, ,
Также создаем строку для Эпсилон
Рассмотрим нетерминал S
   Первый символ правила S -> AB - u
   Это правило заносим в таблицу на пересечении строки нетерминала S и столбца терминала u

   Первый символ правила S -> AB - d
   Это правило заносим в таблицу на пересечении строки нетерминала S и столбца терминала d

Рассмотрим нетерминал A
   Первый символ правила A -> CD - d
   Это правило заносим в таблицу на пересечении строки нетерминала A и столбца терминала d

   Первый символ правила A -> uZ - u
   Это правило заносим в таблицу на пересечении строки нетерминала A и столбца терминала u

Рассмотрим нетерминал B
   Первый символ правила B -> a - a
   Это правило заносим в таблицу на пересечении строки нетерминала B и столбца терминала a

Рассмотрим нетерминал C
   Первый символ правила C -> d - d
   Это правило заносим в таблицу на пересечении строки нетерминала C и столбца терминала d

Рассмотрим нетерминал D
   Первый символ правила D -> c - c
   Это правило заносим в таблицу на пересечении строки нетерминала D и столбца терминала c

Рассмотрим нетерминал Z
   Первый символ правила Z -> iZ' - i
   Это правило заносим в таблицу на пересечении строки нетерминала Z и столбца терминала i

Рассмотрим нетерминал Z'
   Первый символ правила Z' -> i - i
   Это правило заносим в таблицу на пересечении строки нетерминала Z' и столбца терминала i

   Первый символ правила Z' -> iZ' - i
   Это правило заносим в таблицу на пересечении строки нетерминала Z' и столбца терминала i
```

```
Пример вводимых строк: (i+i), (i+*i)
Введите строку:
dca
Допуск. Цепочка символов = L(G).
12465
```

Практическая работа №4 (12-16 лаб.)

Формулировка задания:

Построить множество LR(0)-таблиц не содержащих ε–правила. Построить управляющую таблицу M для LR(k)-грамматики, написать правило вывода выделенной строки. Описать работу алгоритма LR(k) анализатора. Построить LR(k) анализатор на основе грамматического вхождения.

Синтаксический LR-анализатор анализирует входную цепочку слева направо (L), и строит правый (R) вывод грамматики. Грамматики, для которых можно построить детерминированный восходящий анализатор, называются LR(k)-грамматиками (входная цепочка читается слева (Left) направо, выходом анализатора является правый (Right) разбор, k – число символов входной цепочки, на которое можно "заглянуть" вперёд для выделения основы).

Для определения LR(k)-грамматики используются:

1. Множество $FIRST_k(\gamma)$, состоящее из префиксов длины K терминальных цепочек, выводимых из γ. Если из y выводятся терминальные цепочки, длина которых меньше K, то эти цепочки также включаются в множество $FIRST_k(\gamma)$. Формально: $FIRST_k(\gamma) = \{x | \gamma \Rightarrow_i^*$ и$|x| = k$ или $\gamma \Rightarrow_i^* x$ и $|x| < k\}$

2. Пополненной грамматикой, полученной из КС-грамматики $G = (T, V, P, S)$, называется грамматики $G' = (V \in \{S'\}, T, P = \{S' \to S\}, S')$. Если правила грамматики G' пронумерованы числами $1, 2, \ldots,$ то, будем считать, что $S' \to S$ — нулевое правило грамматики G', а нумерация остальных правил такая же, как в грамматики G.

Определение. КС-грамматика $G = (T, V, P, S)$ называется EB(k)- грамматикой для k > 0, если существуют два правых вывода для пополненной грамматики $G' = (T, V', P', S')$ полученной из $G : A \to \beta(ab\_B\_c), B \to \sigma, \sigma = By$

$S' \Rightarrow'_r \alpha Aw \Rightarrow_r \alpha\beta w$

$S' \Rightarrow'_r \gamma\beta x \Rightarrow_r \gamma\sigma y,$

для которых $FIRST_k(w) = FIRST_k(y)$ следует, что $\alpha Ay = \gamma Bx$.

Поскольку $y\sigma x = ay$, то $|y\sigma x| = |\alpha\beta y|$, и, учитывая, что $|y\sigma| < |\beta|$, заключаем: $|| > ||$, те. $= z$ при некотором z $V^+, |z| > 0$. Заметим, что цепочка z является префиксом цепочки x, а не ее окончанием, так как именно цепочка y является окончанием всей сентенциональной формы $a\beta y$.

Условие $y\sigma x = \alpha\beta y$, можно переписать как $\gamma\sigma zy$, и потому $\gamma\sigma z = \alpha\beta$.

Если $\alpha\beta w$ и $\alpha\beta y$ — правовыводимые цепочки пополненной грамматики G', и $FIRST_k(w) = FIRST_k(y)$, $A \to$ — последнее правило, использованное в правом выводе цепочки $\alpha\beta w$, то правило $\to$ должно использоваться также в правом разборе при свертке $\alpha\beta y$ к αAy.

Определение. Основа – кодируемая цепочка символов в верхней части магазина. Магазинный алфавит построен таким образом, что для каждого магазинного символа (за исключением S_0 и $\perp$), кодируемая им цепочка является префиксом правой части некоторого правила грамматики.

Для любой $LR(k)$-грамматики $G = (T, V, P, S)$ можно построить детерминированный анализатор с правым разбором входной цепочки. Анализатор состоит из магазина, входной ленты, выходной ленты и управляющего устройства (пара функций f и g).

Определение. Грамматическое вхождение - это символы полного словаря грамматики, снабженные двумя индексами. Первый индекс i задает номер правила грамматики, в правую часть которого входит данный символ, а второй индекс j - номер позиции символа в этой правой части.

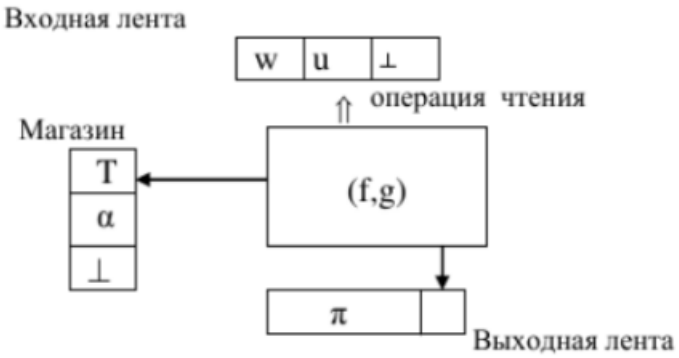


Рис. 1.18. LR(k)-анализатор

Существует два способа построения LR(k) анализаторов:

1. На основе активных префиксов (построения расширенного магазинного алфавита) и отношения OBLOW;
2. На основе LR(0)-ситуаций и функций CLOSURE и GOTO.

Построим двумя способами LR(k) анализатор для заданной грамматики:

$G = (T, V, P, S_0)$, где

$T = \{u, a, d, c, i\}, V = \{S, A, B, C, D, Z, Y\}, S$

P:

p1: $S \to AB$

p2: $A \to CD$

p3: $A \to uZ$

p4: $B \to a$

p5: $C \to d$

p6: $D \to c$

p7: $Z \to i$

p8: $Z \to iY$

p9: $Y \to i$

p10: $Y \to iY$

	S_0	A_1	B_1	C_2	D_2	u_3	Z_3	a_4	d_5	c_6	i_7	i_8	Y_8	i_9	i_{10}	Y_{10}
$\perp$	1	1		1		1			1							
S_0																
A_1			1					1								
B_1																
C_2					1					1						
D_2																
u_3							1									
Z_3																
a_4																
d_5																
c_6																
i_7																
i_8													1	1	1	
i_9																
i_{10}																
Y_{10}														1	1	1

Шаг 4. Алгоритм перенос-свертка. Выполнение алгоритма описывается в терминах конфигураций, представляющих собой тройки вида $(\perp\ \alpha T, ax, \pi)$, где αT — цепочка магазинных символов (T-верхний символ магазина), ax — необработанная часть входной цепочки, π выход (строка из номеров правил), построенный к настоящему моменту времени.

Таблица состоит из двух подтаблиц — функции действия и функции переходов. Входным символам с ленты соответствуют столбцы таблицы, символам магазина — строки.

Управление алгоритмом осуществляется двумя функциями, задаваемых в таблице:

1. По верхнему символу магазина и входному символу, определяется функция действия: ПЕРЕНОС или СВЕРТКА;
2. ПЕРЕНОС определяет значение функции перехода, равное магазинному символу, который нужно втолкнуть в магазин;
3. СВЕРТКА(1), однозначно определяет правило 1 для свертки.

Шаг 1. Пример кодирования активных префиксов. Для каждого магазинного символа (за исключением S_0 и $\perp$), каждая кодируемая цепочка является *префиксом* правой части некоторого правила грамматики. Например, магазинный символ &, находящийся на вершине означает что:

1. В верхушке магазина находится терминальный символ грамматики &
2. В верхней части магазина находится кодируемая этим символом цепочка символов $F\&$, то есть префикс основы - правой части правила 1.

Индекс каждого символа соответствует номеру правила, префикс правой части которого кодируется этим символом. И наоборот, каждый непустой префикс правой части правила кодируется некоторым магазинным символом. Например, правая часть правила 3 имеет два непустых префикса: $*$ и L, которые соответственно кодируются символами $*_3$ и L_3.

Символ переносится в магазин только в том случае, если он кодирует цепочку, «совместимую» с цепочкой, которая будет находиться в магазине после переноса.

Цепочка, кодируемая данным магазинным символом, совместима с цепочкой в магазине, если она является суффиксом магазинной цепочки после переноса данного символа.

Символ грамматикти	Магазинный символ	Кодируемая цепочка	Операции
$S_0 = S$	S_0	S	Д
A	A_1	A	П
B	B_1	AB	С1
C	C_2	C	П
D	D_2	CD	С2
Z	Z_3	uZ	С3
Y	Y_8	iY	С8
	Y_{10}	iY	С10
u	u_3	u	П
a	a_4	a	С4
d	d_5	d	С5
c	c_6	c	С6
i	i_7	i	С7
	i_8	i	П
	i_9	i	С9
	i_{10}	i	П

Построим матрицу функции $g(x)$:

	u	a	d	c	i	S	A	B	C	D	Z	Y
$\perp_0 0$	$u_3 1$		$d_5 1$			$S_0 1$	$A_1 1$		$C_2 1$			
$u_3 1$					$i_7 1 i_8 1$						$Z_3 2$	
$d_5 1$												
$S_0 1$												
$A_1 1$		$a_4 1$						$B_1 2$				
$C_2 1$				$c_6 1$						$D_2 2$		
$a_4 1$												
$B_1 2$												
$c_6 1$												
$D_2 2$												
$i_7 1 i_8 1$					$i_9 1 i_{10} 1$							
$Z_3 2$												
$Y_8 2$												
$Y_{10} 2$												

Построим матрицу функций действий $f(x)$:

	u	a	d	c	i	$\perp$
$\perp_0 0$	П		П			
$u_3 1$					П	
$d_5 1$				С5		
$S_0 1$						
$A_1 1$		П				
$C_2 1$				П		
$a_4 1$						С4
$B_1 2$						С1
$c_6 1$		С6				
$D_2 2$		С2				
$i_7 1 i_8 1$		С7			П	
$Z_3 2$		С3				
$Y_8 2$		С8				
$Y_{10} 2$		С10				

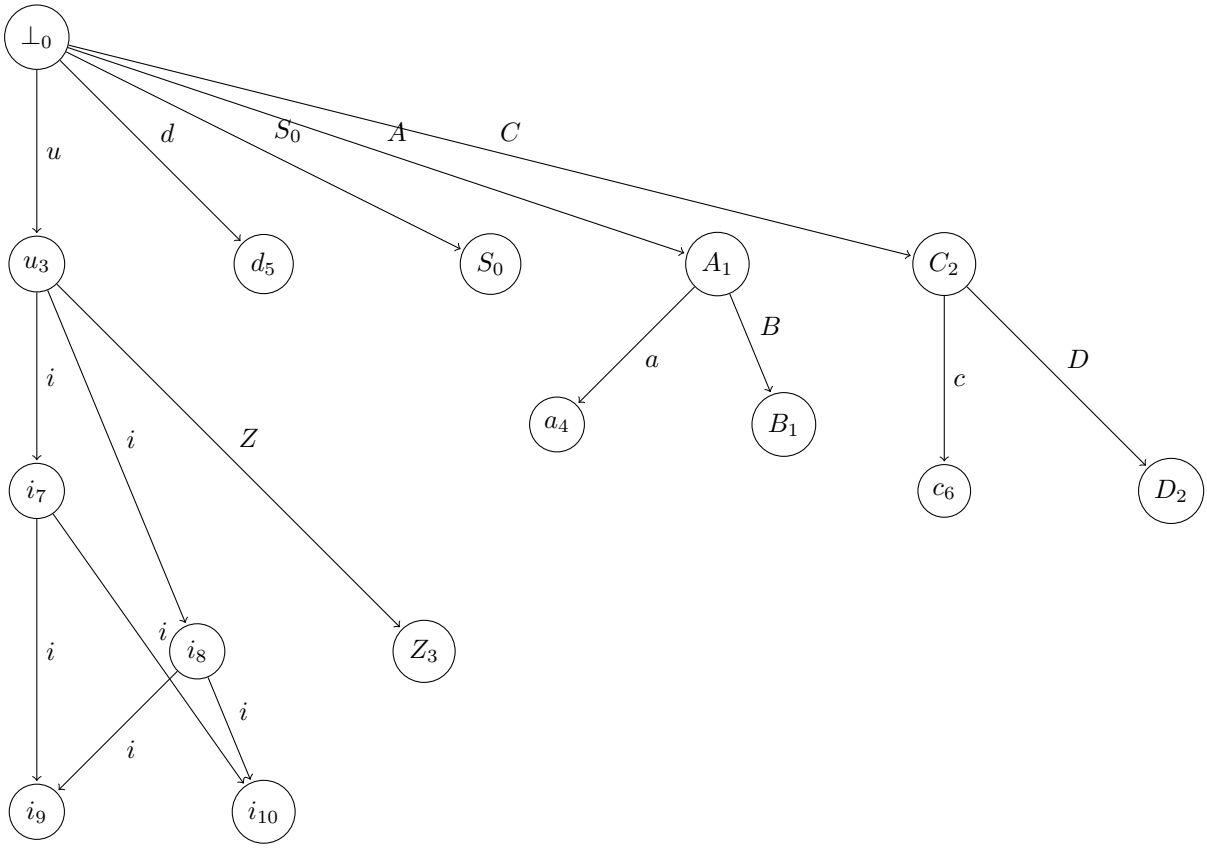
Исходный код:

```
public void Example_LR1()
{
  grammar.AddRule("S", new List<Symbol> { new Symbol("A"), new Symbol("B") });
  Console.WriteLine("S AB");
  grammar.AddRule("A", new List<Symbol> { new Symbol("C"), new Symbol("D") });
  Console.WriteLine("A CD");
  grammar.AddRule("A", new List<Symbol> { new Symbol("u"), new Symbol("Z") });
  Console.WriteLine("A uZ");
  grammar.AddRule("B", new List<Symbol> { new Symbol("a") });
  Console.WriteLine("B a");
  grammar.AddRule("C", new List<Symbol> { new Symbol("d")});
  Console.WriteLine("C d");
  grammar.AddRule("D", new List<Symbol> { new Symbol("c") });
  Console.WriteLine("D c");
  grammar.AddRule("Z", new List<Symbol> { new Symbol("i") });
  Console.WriteLine("Z i");
  grammar.AddRule("Z", new List<Symbol> { new Symbol("i"), new Symbol("Y") });
  Console.WriteLine("Z iY");
  grammar.AddRule("Y", new List<Symbol> { new Symbol("i") });
  Console.WriteLine("Y i");
  grammar.AddRule("Y", new List<Symbol> { new Symbol("i"), new Symbol("Y") });
  Console.WriteLine("Y iY");
  grammar.T.AddRange(new List<Symbol> { new Symbol("u"), new Symbol("a"), new
      Symbol("d"), new Symbol("c"), new Symbol("i") });
  grammar.V.AddRange(new List<Symbol> { new Symbol("S"), new Symbol("A"), new
      Symbol("B"), new Symbol("C"), new Symbol("D"),
    new Symbol("Z"), new Symbol("Y") });
}
```

Пример разбора цепочек:

$(\perp, dca\ \perp, \varepsilon) \vdash^{\Pi} (\perp\ d_5, ca\ \perp, \varepsilon) \vdash^{C5} (\perp\ C_2, ca\ \perp, 5) \vdash^{\Pi} (\perp\ C_2 c, a\ \perp, 5) \vdash^{C6} (\perp\ C_2 D_2, a\ \perp, 56) \vdash^{C2} (\perp\ C_2 A_1, a\ \perp, 562) \vdash^{\Pi} (\perp\ C_2 a, \perp, 562) \vdash^{C4} (\perp\ C_2 B_1, \perp, 5624) \vdash^{C1} (\perp\ S_0, \perp, 56241) \vdash$ Д ДОПУСК

Диаграмма:


$\perp_0$
u
d
S_0
A
C
u_3
d_5
S_0
A_1
C_2
i
i
Z
B
a
c
D
a_4
B_1
c_6
D_2
i_7
i
i
i_8
Z_3
i
i
i
i_9
i_{10}

```
Исходная
КС - грамматика :
Алфавит нетерминальных символов: SABCDZY
Алфавит терминальных символов: uadci
Правила:
(1) S -> AB
(2) A -> CD
(3) A -> uZ
(4) B -> a
(5) C -> d
(6) D -> c
(7) Z -> i
(8) Z -> iY
(9) Y -> i
(10) Y -> iY
Начальный нетерминал: S

Для продолжения нажмите <Enter>
        Delete e-rules:
Executing:
e-rules:
NoShortNoTerms :
V1: : S A B C D Z Y
        e-rules have benn deleted!

Пополненная грамматика:
КС - грамматика :
Алфавит нетерминальных символов: SABCDZYП
Алфавит терминальных символов: uadci$
Правила:
(0) П -> S
(1) S -> AB
(2) A -> CD
(3) A -> uZ
(4) B -> a
(5) C -> d
(6) D -> c
(7) Z -> i
(8) Z -> iY
(9) Y -> i
(10) Y -> iY
Начальный нетерминал: S
```

```
Полученная матрица отношения OBLOW:
     S01 A11 B12 C21 D22 u31 Z32 a41 d51 c61 i71 i81 Y82 i91i101Y102
   ^   1   1       1       1           1
 S01
 A11           1                   1
 B12
 C21                   1                   1
 D22
 u31                           1              1   1
 Z32
 a41
 d51
 c61
 i71
 i81                                              1   1   1
 Y82
 i91
i101                                                  1   1   1
Y102

Для продолжения нажмите <Enter>

Полученная матрица для функции переходов g(X):
            u       a       d       c       i       S       A       B       C       D       Z
     ^00    u31             d51                     S01     A11             C21
     u31                                    i71i81                                          Z32
     d51
     S01
     A11            a41                                                     B12
     C21                            c61                                             D22
     a41
     B12
     c61
     D22
  i71i81                                    i91i101
     Z32
     Y82
    Y102

Для продолжения нажмите <Enter>
```

```
Введите цепочку: uia
Входная цепочка uia$ распознана.
Результат правого вывода: 7 3 4 1
Вы хотите продолжить (да/нет)? - y

Введите цепочку: dca
Входная цепочка dca$ распознана.
Результат правого вывода: 5 6 2 4 1
Вы хотите продолжить (да/нет)? -
```