Московский авиационный институт

(национальный исследовательский институт)

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 KС-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 cпроектировать 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:* Орусский В.Р.

*Группа: М8О-206Б-21*

*Руководитель: Киндинова В. В.*

*Оценка:*

*Дата:*

**Москва 2023**

**Практическая работа №1 (1-3 лаб.)**

*Лабораторные работы №1-3*

**Формулировка задания**:

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

Спроектируем грамматику для заданного языка:

1. Нахождение внутри текста обособленного слова baloney без чувствительности к регистру

**pattern = “(?i)(\W|^)(baloney)(\W|$)”**

**Автоматная грамматика:**

L(pattern) = L(“(?i)(\W|^)(baloney)(\W|$)”) = {“baloney”, “Baloney”, “BAloney”, …}

*Правила регулярной грамматики:*

*Пример цепочек:*

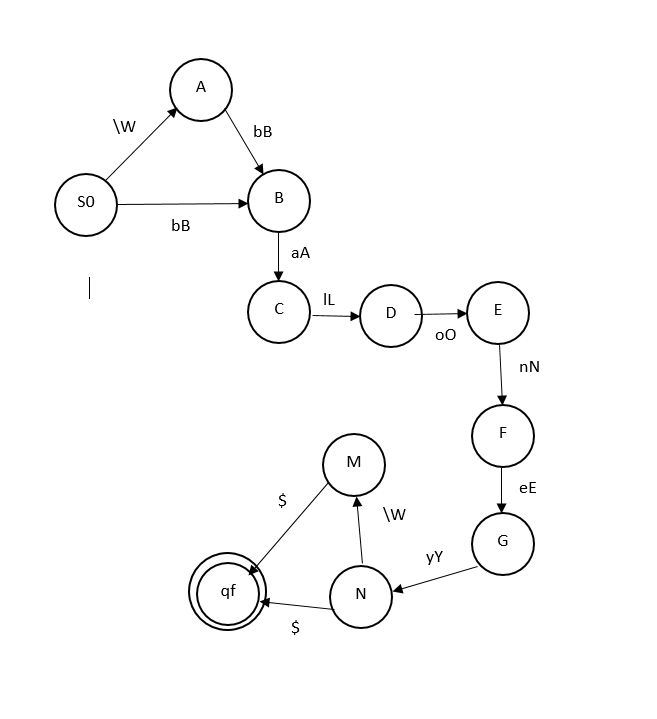
**Конечный автомат:**

𝐿(𝐾𝐴) = 𝐿(𝐺)

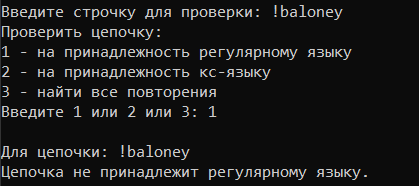
𝐾𝐴 = (𝑄, 𝛴, 𝛿, , 𝐹), где 𝑄 = , 𝛴 = {𝐵, 𝑏, 𝐴, 𝑎, 𝐿, 𝑙, 𝑂, 𝑜, 𝑁, 𝑛, 𝐸, 𝑒, 𝑌, 𝑦, ∖𝑊,$}, , 𝐹 = {}

}

*Примеры конфигурации КА:*



*Лемма о накачке:*



**2. pattern =  “(\W|^)po[#\-]{0,1}\s{0,1}\d{2}[\s-]{0,1}\d{4}(\W|$)”**

**Автоматная грамматика:**

L(pattern) = L(“(\W|^)po[#\-]{0,1}\s{0,1}\d{2}[\s-]{0,1}\d{4}(\W|$)”) = {“po#000000”, …, “!po- 99-9999!”, …}

*Правила регулярной грамматики:*

*Пример цепочек:*

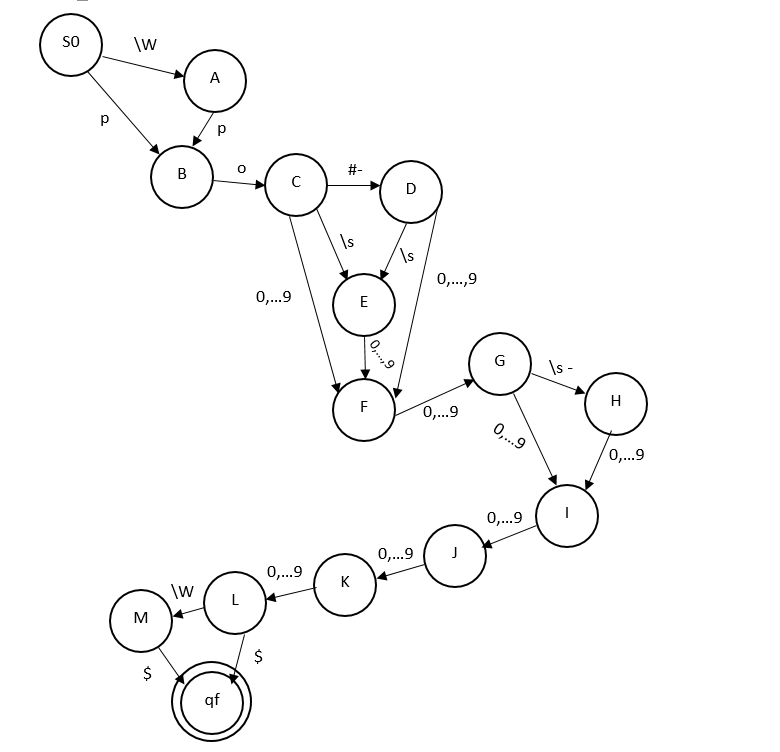
**Конечный автомат:**

𝐿(𝐾𝐴) = 𝐿(𝐺)

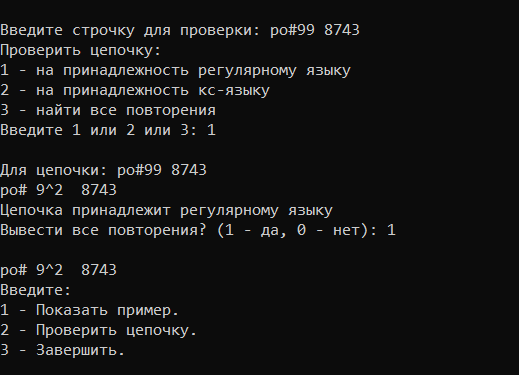
𝐾𝐴 = (𝑄, 𝛴, 𝛿, , 𝐹), где 𝑄 = , 𝛴 = { }, , 𝐹 = {L,M}

}

*Примеры конфигурации КА:*



*Лемма о накачке:*



*Лабораторная работа №3*

**Формулировка задания:**

Реализовать конечные автоматы, составленные в ЛР №2

**Код программы:**

automats[0].AddRule("S01", "!", "A1");

automats[0].AddRule("S01", "B", "B1");

automats[0].AddRule("S01", "b", "B1");

automats[0].AddRule("A1", "B", "B1");

automats[0].AddRule("A1", "b", "B1");

automats[0].AddRule("B1", "A", "C1");

automats[0].AddRule("B1", "a", "C1");

automats[0].AddRule("C1", "L", "D1");

automats[0].AddRule("C1", "l", "D1");

automats[0].AddRule("D1", "O", "E1");

automats[0].AddRule("D1", "o", "E1");

automats[0].AddRule("E1", "N", "F1");

automats[0].AddRule("E1", "n", "F1");

automats[0].AddRule("F1", "E", "G1");

automats[0].AddRule("F1", "e", "G1");

automats[0].AddRule("G1", "Y", "N1");

automats[0].AddRule("G1", "y", "N1");

automats[0].AddRule("N1", "!", "M1");

automats[0].AddRule("N1", "$", "qf1");

automats[0].AddRule("M1", "$", "qf1");

foreach (string curNumber in numbers)

{

automats[1].AddRule("C2", curNumber, "F2");

automats[1].AddRule("D2", curNumber, "F2");

automats[1].AddRule("E2", curNumber, "F2");

automats[1].AddRule("F2", curNumber, "G2");

automats[1].AddRule("G2", curNumber, "I2");

automats[1].AddRule("H2", curNumber, "I2");

automats[1].AddRule("I2", curNumber, "J2");

automats[1].AddRule("J2", curNumber, "K2");

automats[1].AddRule("K2", curNumber, "L2");

}

automats[1].AddRule("S02", "!", "A2");

automats[1].AddRule("S02", "p", "B2");

automats[1].AddRule("A2", "p", "B2");

automats[1].AddRule("B2", "o", "C2");

automats[1].AddRule("C2", "#", "D2");

automats[1].AddRule("C2", "-", "D2");

automats[1].AddRule("C2", " ", "E2");

automats[1].AddRule("D2", " ", "E2");

automats[1].AddRule("G2", " ", "H2");

automats[1].AddRule("G2", "-", "H2");

automats[1].AddRule("G2", "-", "H2");

automats[1].AddRule("L2", "!", "M2");

automats[1].AddRule("L2", "$", "qf2");

automats[1].AddRule("M2", "$", "qf2");

var dka1 = new FSAutomate();

dka1.BuildDeltaDKAutomate(automats[0], false);

var dka2 = new FSAutomate();

dka2.BuildDeltaDKAutomate(automats[1], false);

var automats1 = new FSAutomate[] { dka1, dka2};

var exectionOrder = new FSAutomate[] { dka1, dka2 };

string[] names = { "KA1", "KA2" };

Console.WriteLine();

Console.WriteLine("Были построены автоматы:");

Console.WriteLine("КА1: \"(?i)(\\W|^)(baloney)(\\W|$)\"");

Console.WriteLine("КА2: \"(\\W|^)po[#\\-]{0,1}\\s{0,1}\\d{2}[\\s-]{0,1}\\d{4}(\\W|$)\"");

Console.WriteLine();

Console.WriteLine("Enter number of automate:");

int automateNumber = -1;

automateNumber = int.Parse(Console.ReadLine());

while (automateNumber != 1 && automateNumber != 2)

{

Console.WriteLine("Bad input, try again");

Console.WriteLine("Enter number of automate:");

automateNumber = int.Parse(Console.ReadLine());

}

Console.WriteLine("You chose automate number {0}", automateNumber);

--automateNumber;

if (automateNumber == 0)

{

Console.WriteLine("Enter line (\"(?i)(\\W|^)(baloney)(\\W|$)\", for ex.: baloney) to execute");

Console.WriteLine("or \"exit\" to exit:");

}

else if (automateNumber == 1)

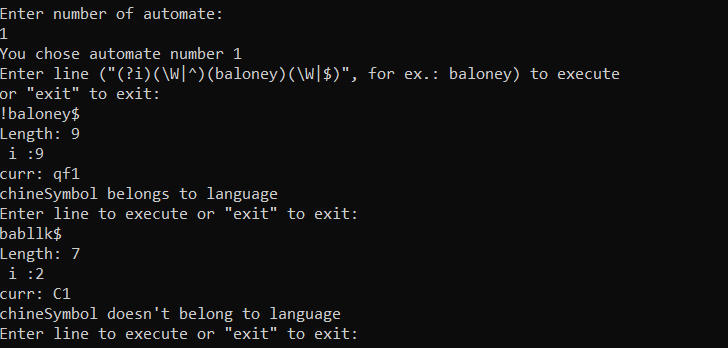
{

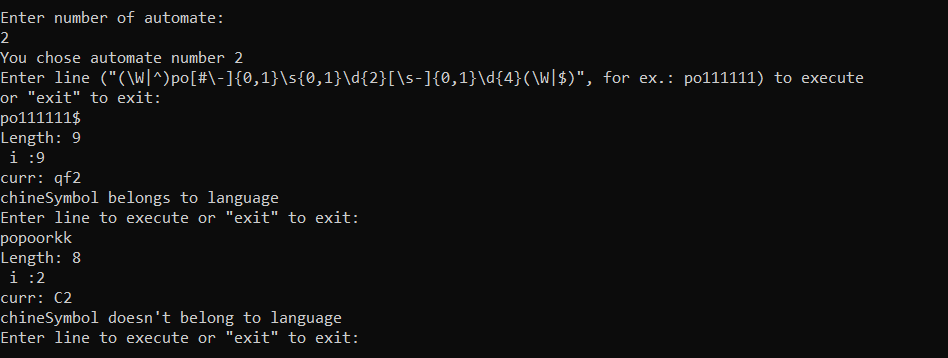
Console.WriteLine("Enter line (\"(\\W|^)po[#\\-]{0,1}\\s{0,1}\\d{2}[\\s-]{0,1}\\d{4}(\\W|$)\", for ex.: po111111) to execute");

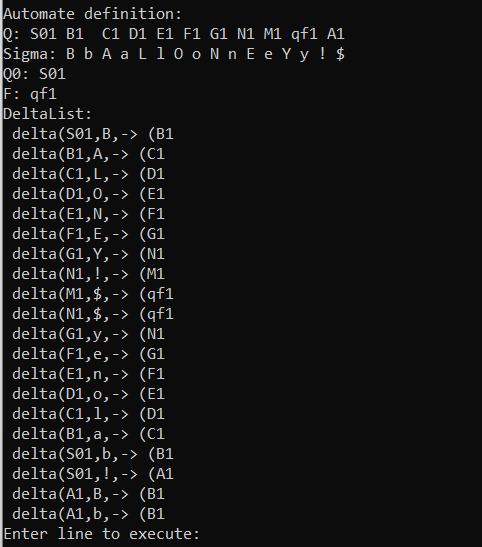
Console.WriteLine("or \"exit\" to exit:");

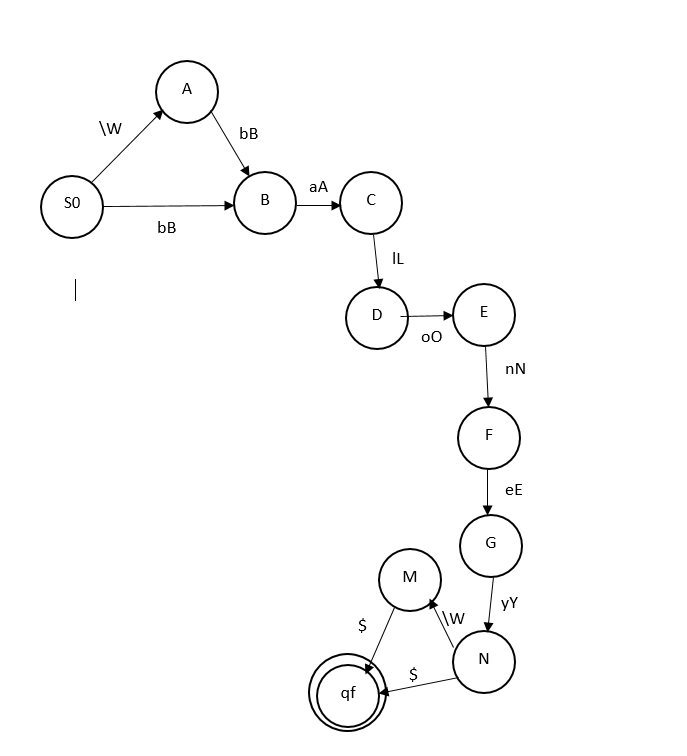
}

**Пример работы программы:**









**Практическая работа №2 (лабораторные 4-8)**

*Лабораторная работа №4:*

*Заданная грамматика:*

***G = {( a, b, c, d, f, g, epsilon, p, q, m, z ),( S, A, B, C, D, J, E, N, Y, I ),( A → BCD, B → cD, C → dfB, D → J, J → g, S → aAbE, E → epsilon, N → bCp, Y → q, S → zI, I → Im, I → m),S}***

***Лабораторная работа №4: Устранить из КС-грамматики бесполезные символы и ε–правила***

Исходная грамматика:

**G** = (**T**, **V**, **P**, **S0**), где

**T** = {a,b,c,d,f,g,p,q,m,z,};

**V** = {S,A,B,C,D,J,E,N,Y,I};

**P** = {A → BCD, B → cD, C → dfB, D → J, J → g, S → aAbE, E → epsilon, N → bCp, Y → q, S → zI, I → Im, I → m};

**S0** = K.

p1: A → BCD

p2: B → cD

p3: C → dfB

p4: D → J

p5: J → g

p6: S → aAbE

p7: E →

p8: N → bCp

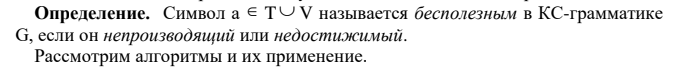
p9: Y → q

p10: S → zI

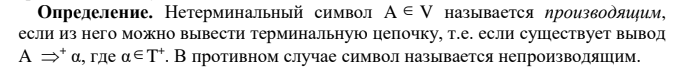
p11: I → Im

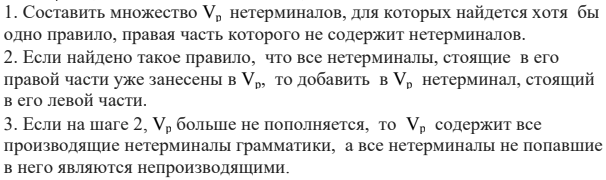
p12: I → m

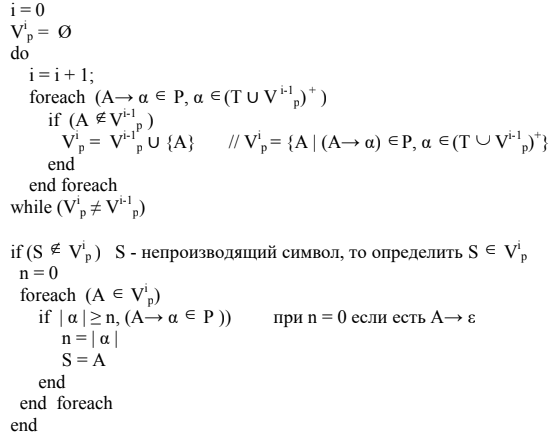
**Устранить из КС-грамматики бесполезные символы.**



**1)Устранение непроизводящих символов.**



****

****

1. = {J,E,Y,I} – для этих нетерминалов нашлось хотя бы одно правило, правая часть которого не содержит нетерминалов

2. = {J,E,Y,I,D} – если найдено такое правило, что все нетерминалы, стоящие в его правой части уже занесены в Vp, то добавить в Vp нетерминал, стоящий в его левой части

3. = {J,E,Y,I,D,B}

4. = {J,E,Y,I,D,B,C}

5. = {J,E,Y,I,D,B,C,A}

6. = {J,E,Y,I,D,B,C,A,S}

7. = {J,E,Y,I,D,B,C,A,S,N}

8. Vp = {J,E,Y,I,D,B,C,A,S,N} содержит все производящие нетерминалы грамматики, а все нетерминалы, не попавшие в него, являются непроизводящими (V - Vp = {}).



Запишем P' (те правила, в правой части которых только производящие символы):

P' = {p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12}

p1: A → BCD

p2: B → cD

p3: C → dfB

p4: D → J

p5: J → g

p6: S → aAbE

p7: E →

p8: N → bCp

p9: Y → q

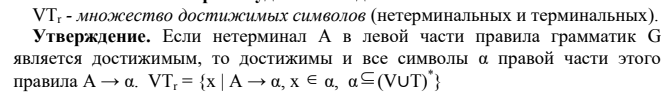
p10: S → zI

p11: I → Im

p12: I → m

G1 = ({a,b,c,d,f,g,p,q,m,z,}, {S,A,B,C,D,J,E,N,Y,I}, P', S)

**2)Устранение недостижимых символов (VTr – множество недостижимых символов):**

****

****

= {S}

= {S, a, A, b, E, z, I}

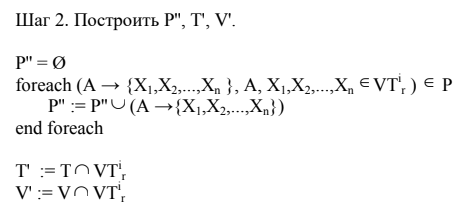
= {S, a, A, b, E, z, I, B, C, D, , m}

= {S, a, A, b, E, z, I, B, C, D, , m, c, d, f, J}

= {S, a, A, b, E, z, I, B, C, D, , m, c, d, f, J, g}

Так, не входящие в VTr символы {p, q, N, Y} недостижимы

**Добавим в P' только правила, состоящие из достижимых символов**

****

P" = {p1, p2, p3, p4, p5, p6, p7, p8, p9, p10}:

p1: A → BCD

p2: B → cD

p3: C → dfB

p4: D → J

p5: J → g

p6: S → aAbE

p7: E →

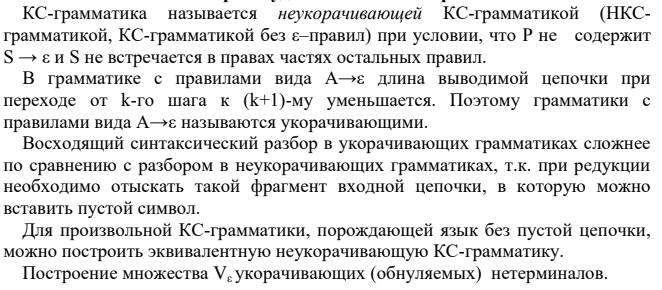
p8: S → zI

p9: I → Im

p10: I → m

Получаем G' = ({a,b,c,d,f,g,m,z,}, {S,A,B,C,D,J,E,I}, P", S)

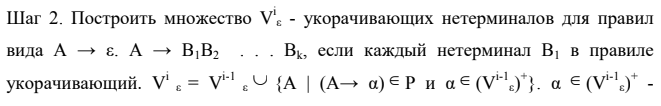
**Устранить из КС-грамматики ε–правила.**

****

*Построим множество укорачивающих нетерминалов согласно алгоритму:*

**

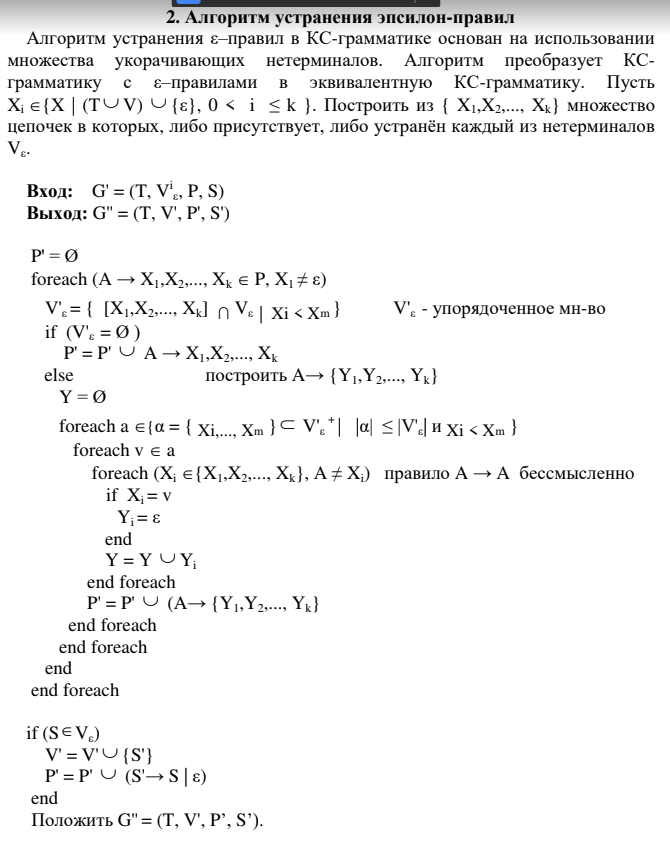
= {E} (множество укорачивающих нетерминалов для правил вида A → ε)





Необходимо построить множество V1ɛ укорачивающих нетерминалов для правилвида A → B1B2...Bk, если каждый нетерминал B1 в правилеукорачивающий. Однако в данном примере таких правил нет.

Итак, множество укорачивающих терминалов: = {E}.



G' = ({a,b,c,d,f,g,m,z,}, {S,A,B,C,D,J,E,I}, P", S)

P’ = {p1, p2, p3, p4, p5, p6, p7, p8, p9}:

p1: A → BCD

p2: B → cD

p3: C → dfB

p4: D → J

p5: J → g

p6: S → aAb

p7: S → zI

p8: I → Im

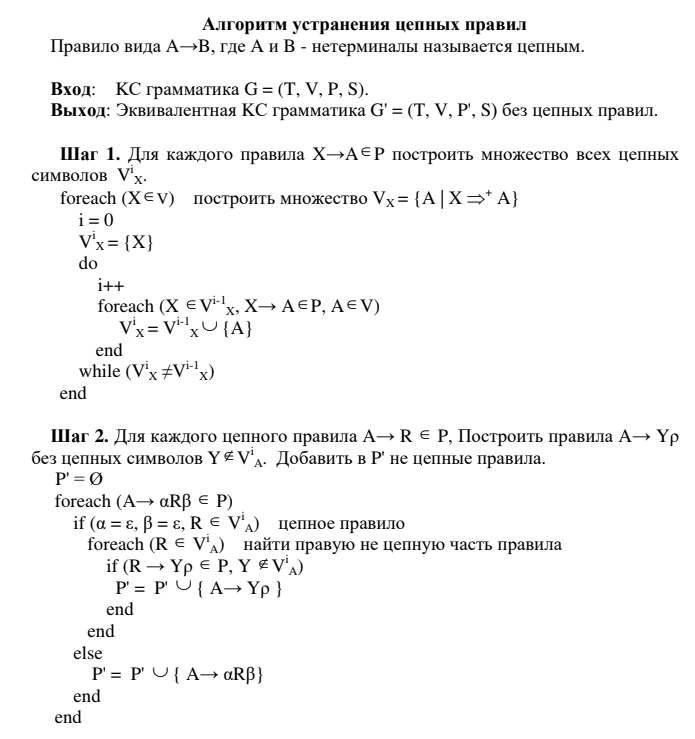
p9: I → m

G" = ({a,b,c,d,f,g,m,z}, {S,A,B,C,D,J,I}, P’, S).

В грамматике G” отсутствуют бесполезные символы и -правила.

***Лабораторная работа №5: Устранить из KС-грамматики цепные правила и левую рекурсию***

**1)Устранить из KС-грамматики цепные правила.**

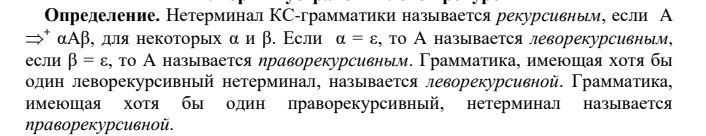


Цепные правила: D → J

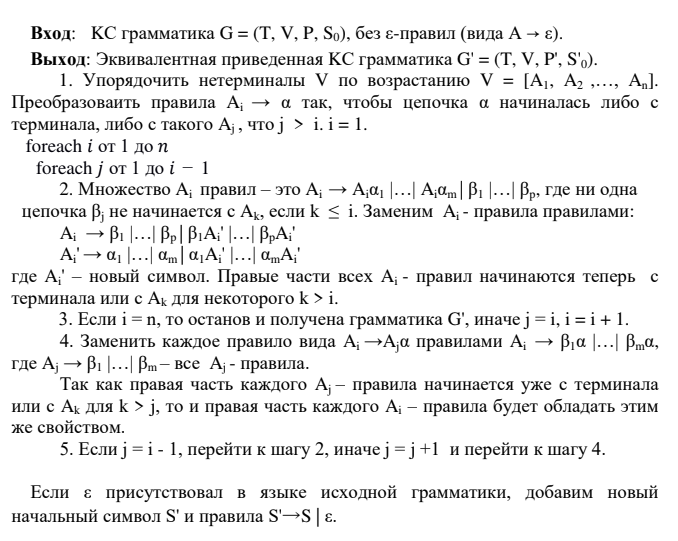
= {J}

Правила D → J, J → g “заменим” на D → g

**2)Устранить из KС-грамматики левую рекурсию**



**Алгоритм Устранения непосрественной левой рекурсии**



G’’’ = ({a,b,c,d,f,g,m,z}, {S,A,B,C,D,I}, P, S).

P = {p1, p2, p3, p4, p5, p6, p7, p8, p9}

p1: A → BCD

p2: B → cD

p3: C → dfB

p4: D → g

p5: S → aAb

p6: S → zI

p7: I → Im

p8: I → m

**ШАГ 1:** Пусть = S, = A, = B, = C, = D, = I. V = [, , , , , ].

**ШАГ 2:** Для i = 1, 2, … , 5 нет правил вида → α. Для i = 6 преобразуем правило p6: I → Im. Заменим I-правило правилом → m, и добавим в грамматику правило для нового нетерминала → m | m.

**ШАГ 3:** i = n = 6. Алгоритм завершен.

Получили правила новой грамматики , не содержащей левой рекурсии:

A → BCD, B → cD, C → dfB, D → g, S → aAb, S → zI, → m, → m | m.

*Лабораторная работа №6*

**Формулировка задания:**

Определить форму КС-грамматики и сделать ее приведение

Грамматика, была получена из грамматики G применением следующих алгоритмов:

1. Устранение бесполезных символов: непроизводящих и недостижимых;

2. Преобразование в грамматику без ε–правил;

3. Устранение цепных правил;

4. Устранение левой рекурсии.

Таким образом, грамматика находится в приведённой форме.

Gпр = (T, V, P, S0).

T = { a,b,c,d,f,g,m,z };

V = { S,A,B,C,D,I,I1 };

P = { p1, p2, p3, p4, p5, p6, p7, p8};

p1: A → BCD

p2: B → cD

p3: C → dfB

p4: D → g

p5: S → aAb

p6: S → zI

p7: I → mI1

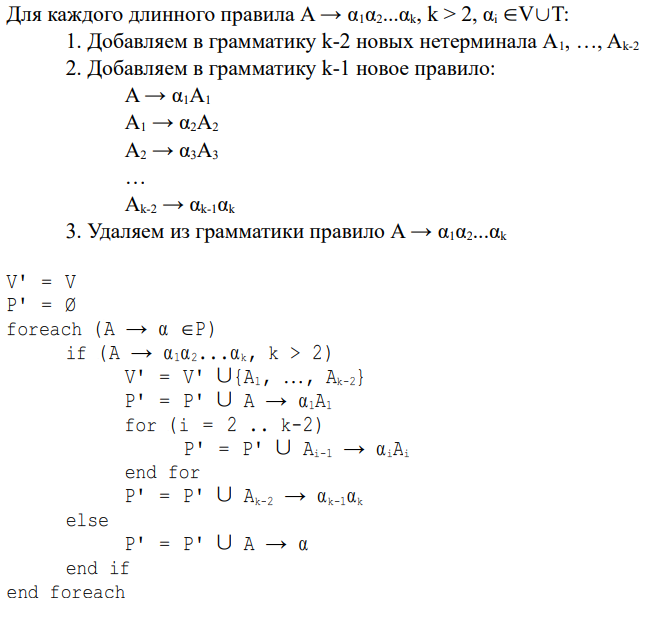
p8: I1 → mI1 | m

S0 = S.

Приведём грамматику к нормальной форме Хомского по алгоритму.

**Алгоритм приведения грамматики к нормальной форме Хомского**

1. **Удаление длинных правил**



P = { p1: S → aAb

p2: S → zI

p3: A → BCD

p4: B → cD

p5: C → dfB

p6: D → g

p7: I → mI1

p8: I1 → mI1 | m

}

Правило S → aAb:

1. Добавим новый нетерминал S1.

2. Добавим новые правила: S → aS1 S1 → Ab

3. Удалим из грамматики правило S → aAb

Правило A → BCD:

1. Добавим новый нетерминал A1.

2. Добавим новые правила: A → BA1 A1 → CD

3. Удалим из грамматики правило A → BCD

Правило C → dfB:

1. Добавим новый нетерминал C1.

2. Добавим новые правила: C → dC1 C1 → fB

3. Удалим из грамматики правило C → dfB

Таким образом, получили новую грамматику G = ({a,b,c,d,f,g,m,z}, {S,S1,A,A1,B,C,C1,D,I,I1}, {S → aS1, S → zI, S1 → Ab, A → BA1, A1 → CD, B → cD, C → dC1, C1 → fB, D → g, I → mI1, I1 → mI1 | m },S)

1. **Удаление бесполезных символов**

В G нет бесполезных символов

1. **Удаление -правил**

В G нет -правил

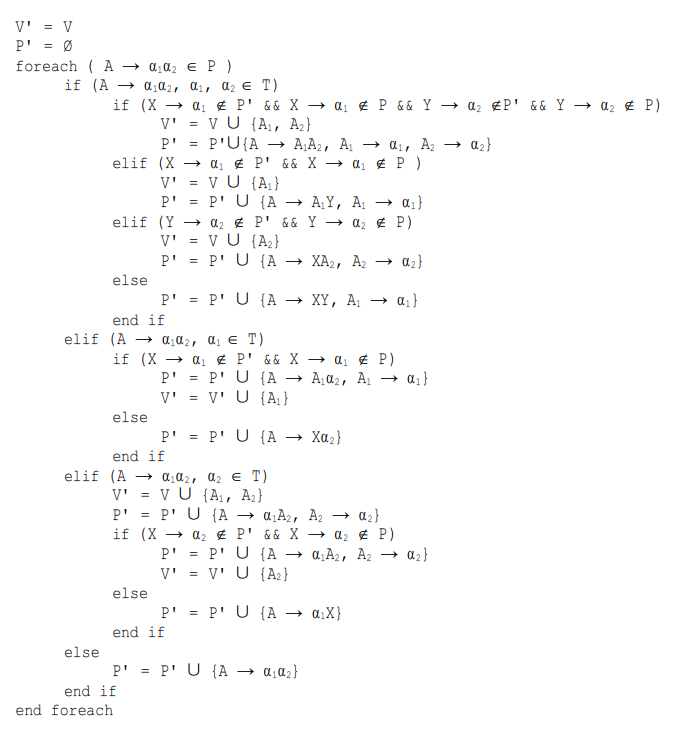
1. Если начальный нетерминал встречается в правых частях правил, нужно убрать правила такого вида. Заведем новый стартовый символ S' и добавим правило S' → S, а также S' → вместо S → , если оно было.

В G нет начального нетерминала в правых частях правил и правил вида S→.

1. **Удаление цепных правил**

В грамматике нет цепных-правил.

1. Удаление ситуаций, в которых встречаются несколько терминалов



**P=**{S → aS1, S → zI, S1 → Ab, A → BA1, A1 → CD, B → cD, C → dC1, C1 → fB, D → g, I → mI1, I1 → mI1 | m}

1. S → aS1 => S → S2S1, S2 → a
2. S → zI => S → S3I, S3 → z
3. S1 → Ab => S1 → AA2, A2 → b
4. B → cD => B → B1D, B1 → c
5. C → dC1 => C → C2C1, C2 → d
6. C1 → fB => C1 → C3B, C3 → f
7. I → mI1 => I → I2I1, I2 → m
8. I1 → mI1 | m => I1 → I2I1 | m

Таким образом, Gпр. = (T, V, P, S0), где

T = { a,b,c,d,f,g,m,z },

V = {S,S1,S2,S3,A,A1,A2,B,B1,C,C1,C2,C3,D,I,I1,I2},

P **=** { S → S2S1, S → S3I, S1 → AA2, S2 → a, S3 → z, , A → BA1, A1 → CD, A2 → b, B → B1D, B1 → c, C → C2C1, C2 → d, C1 → C3B, C3 → f, D → g, I → I2I1, I1 → I2I1 | m, I2 → m }

S0 = S

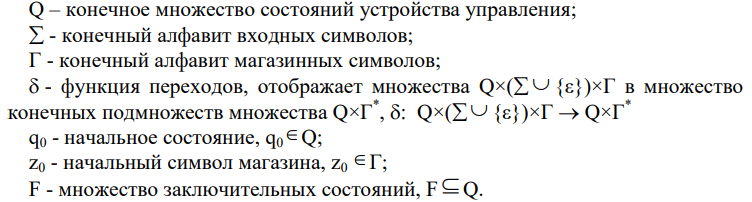
*Лабораторная работа №7*

**Формулировка задания:**

Спроектировать МП-автомат для приведённой КС-грамматики.

*Автоматы с магазинной памятью* (*МП-автоматы*) представляют собой модель распознавателей для языков, задаваемых КС-грамматиками. МП-автоматы имеют вспомогательную память, называемую магазином. В магазин можно поместить неограниченное количество символов. В каждый момент времени доступен только верхний символ магазина. Верхний символом магазина будем считать самый левый символ цепочки.

Так, МП = (Q, Σ, Г, δ, q0, z0, F), где



Приведённая грамматика:

G = (**T**, **V**, **P**, **S**), где

**T** = { a,b,c,d,f,g,m,z }, **V** = { S,S1,S2,S3,A,A1,A2,B,B1,C,C1,C2,C3,D,I,I1,I2}, **S0** = S

P **=** { S → S2S1, S → S3I, S1 → AA2, S2 → a, S3 → z, , A → BA1, A1 → CD, A2 → b, B → B1D, B1 → c, C → C2C1, C2 → d, C1 → C3B, C3 → f, D → g, I → I2I1, I1 → I2I1 | m, I2 → m }

P **= {**

p1: S → S2S1

p2: S → S3I

p3: S1 → AA2

p4: S2 → a

p5: S3 → z

p6: A → BA1

p7: A1 → CD

p8: A2 → b

p9: B → B1D

p10: B1 → c

p11: C → C2C1

p12: C1 → C3B

p13: C2 → d

p14: C3 → f

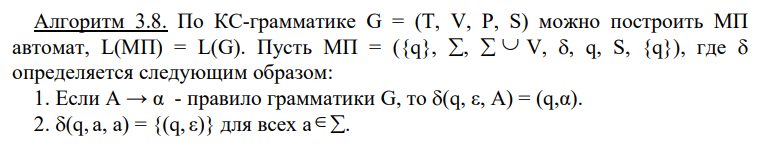
p15: D → g

p16: I → I2I1

p17: I1 → I2I1 | m

p18: I2 → m

}



L(МП) = L(G)

MП = (**Q**, **Σ**, **Г**, **δ**, **q0**, **z0**, **F**):

**Q** = {q}, **Σ** = **T**, **Г** = **T** U **V**, **δ** = **δ**, **q0** = **q**, **z0** = **S**, **F** = {q}

МП = ({q}, { a,b,c,d,f,g,m,z }, { a,b,c,d,f,g,m,z, S,S1,S2,S3,A,A1,A2,B,B1,C,C1,C2,C3,D,I,I1,I2}, **δ**, q, S, {q})

**Определение:** Конфигурацией МП-автомата называется тройка (q, ω, z) ∈ Q×Σ\*×Г\*, где q – текущее состояние управляющего устройства;

ω – необработанная часть входной цепочки (первый символ цепочки ω

находится под входной головкой; если ω = ε, то считается, что вся входная

цепочка прочитана);

z – содержимое магазина (самый левый символ цепочки z считается верхним

символом магазина; если z = ε, то магазин считается пустым).

Функция перехода **δ** определяется согласно алгоритму:



|  |  |
| --- | --- |
| p1: S → S2S1  p2: S → S3I  p3: S1 → AA2  p4: S2 → a  p5: S3 → z  p6: A → BA1  p7: A1 → CD  p8: A2 → b  p9: B → B1D  p10: B1 → c  p11: C → C2C1  p12: C1 → C3B  p13: C2 → d  p14: C3 → f  p15: D → g  p16: I → I2I1  p17: I1 → I2I1 | m  p18: I2 → m | 1: δ(q, ε, S) = {(q, S2S1), (q, S3I)}  2: δ(q, ε, S1) = (q, AA2)  3: δ(q, ε, S2) = (q, a)  4: δ(q, ε, S3) = (q, z)  5: δ(q, ε, A) = (q, BA1)  6: δ(q, ε, A1) = (q, CD)  7: δ(q, ε, A2) = (q, b)  8: δ(q, ε, B) = (q, B1D)  9: δ(q, ε, B1) = (q, c)  10: δ(q, ε, C) = (q, C2C1)  11: δ(q, ε, C1) = (q, C3B)  12: δ(q, ε, C2) = (q, d)  13: δ(q, ε, C3) = (q, f)  14: δ(q, ε, D) = (q, g)  15: δ(q, ε, I) = (q, I2I1)  16: δ(q, ε, I1) = {(q, I2I1),(q, m)}  17: δ(q, ε, I2) = (q, m)  18: δ(q, a, a) = (q, ε) для всех a Σ |

Последовательность тактов МП-автомата acgdfcggb:

(q0, acgdfcggb, S) ⸠1 (q, acgdfcggb, S2S1) ⸠3 (q, acgdfcggb, aS1) ⸠18 ((q, cgdfcggb, S1) ⸠2 (q, cgdfcggb, AA2) ⸠5 (q, cgdfcggb, BA1A2) ⸠8 (q, cgdfcggb, B1DA1A2) ⸠9 (q, cgdfcggb, cDA1A2) ⸠18 (q, bdfcggb, DA1A2) ⸠14 (q, gdfcggb, gA1A2) ⸠18 (q, dfcggb, A1A2) ⸠6 (q, dfcggb, CDA2) ⸠10 (q, dfcggb, C2C1DA2) ⸠12 (q, dfcggb, dC1DA2) ⸠17 (q, fcggb, C1DA2) ⸠11 (q, fcggb, C3BDA2) ⸠13 (q, fcggb, fBDA2) ⸠18 (q, cggb, DA2) ⸠8 (q, cggb, B1DDA2) ⸠9 (q, cggb, cDDA2) ⸠18 (q, ggb, DDA2) ⸠14 (q,ggb, gDA2) ⸠18 (q, gb, DA2) ⸠14 (q, gb, gA2) ⸠18 (q, b, A2) ⸠7 (q, b, b) ⸠18 (q, ε, ε)

МП-автоматы – детерменированные автоматы. МП-автомат относится к нисходящим синтаксическим анализаторам, так как строится сверху вниз.

*Лабораторная работа №8*

**Формулировка задания:**

Реализовать спроектированный МП-автомат для приведённой КС-грамматики.

**Код программы:**

var cfgr = new Grammar(new List<Symbol>() { "a", "b", "c", "d", "f", "g", "m", "z" },

new List<Symbol>() { "S", "S1", "S2", "S3", "A", "A1", "A2", "B", "B1", "C", "C1", "C2", "C3", "D", "I", "I1", "I2" },

"S");

cfgr.AddRule("S", new List<Symbol>() { "S2", "S1" });

cfgr.AddRule("S", new List<Symbol>() { "S3", "I", });

cfgr.AddRule("S1", new List<Symbol>() { "A", "A2" });

cfgr.AddRule("S2", new List<Symbol>() { "a"});

cfgr.AddRule("S3", new List<Symbol>() { "z"});

cfgr.AddRule("A", new List<Symbol>() { "B", "A1" });

cfgr.AddRule("A1", new List<Symbol>() { "C", "D" });

cfgr.AddRule("A2", new List<Symbol>() { "b"});

cfgr.AddRule("B", new List<Symbol>() { "B1", "D" });

cfgr.AddRule("B1", new List<Symbol>() { "c"});

cfgr.AddRule("C", new List<Symbol>() { "C2", "C1" });

cfgr.AddRule("C1", new List<Symbol>() { "C3", "B" });

cfgr.AddRule("C2", new List<Symbol>() { "d"});

cfgr.AddRule("C3", new List<Symbol>() { "f"});

cfgr.AddRule("D", new List<Symbol>() { "g"});

cfgr.AddRule("I", new List<Symbol>() { "I2","I1" });

cfgr.AddRule("I1", new List<Symbol>() { "I2", "I1" });

cfgr.AddRule("I1", new List<Symbol>() { "m"});

cfgr.AddRule("I2", new List<Symbol>() { "m"});

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

cfgr.DebugPrules();

var pda = new PDA(new List<Symbol>() { "q0", "q"},

new List<Symbol>() { "a", "b", "c", "d", "f", "g", "m", "z" },

new List<Symbol>() { "a", "b", "c", "d", "f", "g", "m", "z", "S", "S1", "S2", "S3", "A", "A1", "A2", "B", "B1", "C", "C1", "C2", "C3", "D", "I", "I1", "I2" },

"q0",

"S",

new List<Symbol>() { "q" });

pda.addDeltaRule("q", "a", "a", new List<Symbol>() { "q" }, new List<Symbol>() { "ε" });

pda.addDeltaRule("q", "b", "b", new List<Symbol>() { "q" }, new List<Symbol>() { "ε" });

pda.addDeltaRule("q", "c", "c", new List<Symbol>() { "q" }, new List<Symbol>() { "ε" });

pda.addDeltaRule("q", "d", "d", new List<Symbol>() { "q" }, new List<Symbol>() { "ε" });

pda.addDeltaRule("q", "f", "f", new List<Symbol>() { "q" }, new List<Symbol>() { "ε" });

pda.addDeltaRule("q", "g", "g", new List<Symbol>() { "q" }, new List<Symbol>() { "ε" });

pda.addDeltaRule("q", "m", "m", new List<Symbol>() { "q" }, new List<Symbol>() { "ε" });

pda.addDeltaRule("q", "z", "z", new List<Symbol>() { "q" }, new List<Symbol>() { "ε" });

pda.addDeltaRule("q0", "ε", "S", new List<Symbol>() { "q" }, new List<Symbol>() { "S2", "S1" });

pda.addDeltaRule("q0", "ε", "S", new List<Symbol>() { "q" }, new List<Symbol>() { "S3", "I" });

pda.addDeltaRule("q", "ε", "S1", new List<Symbol>() { "q" }, new List<Symbol>() { "A", "A2" });

pda.addDeltaRule("q", "ε", "S2", new List<Symbol>() { "q" }, new List<Symbol>() { "a"});

pda.addDeltaRule("q", "ε", "S3", new List<Symbol>() { "q" }, new List<Symbol>() { "z" });

pda.addDeltaRule("q", "ε", "A", new List<Symbol>() { "q" }, new List<Symbol>() { "B", "A1" });

pda.addDeltaRule("q", "ε", "A1", new List<Symbol>() { "q" }, new List<Symbol>() { "C", "D" });

pda.addDeltaRule("q", "ε", "A2", new List<Symbol>() { "q" }, new List<Symbol>() { "b" });

pda.addDeltaRule("q", "ε", "B", new List<Symbol>() { "q" }, new List<Symbol>() { "B1", "D" });

pda.addDeltaRule("q", "ε", "B1", new List<Symbol>() { "q" }, new List<Symbol>() { "c" });

pda.addDeltaRule("q", "ε", "C", new List<Symbol>() { "q" }, new List<Symbol>() { "C2", "C1" });

pda.addDeltaRule("q", "ε", "C1", new List<Symbol>() { "q" }, new List<Symbol>() { "C3", "B" });

pda.addDeltaRule("q", "ε", "C2", new List<Symbol>() { "q" }, new List<Symbol>() { "d" });

pda.addDeltaRule("q", "ε", "C3", new List<Symbol>() { "q" }, new List<Symbol>() { "f" });

pda.addDeltaRule("q", "ε", "D", new List<Symbol>() { "q" }, new List<Symbol>() { "g" });

pda.addDeltaRule("q", "ε", "I", new List<Symbol>() { "q" }, new List<Symbol>() { "I2", "I1" });

pda.addDeltaRule("q", "ε", "I1", new List<Symbol>() { "q" }, new List<Symbol>() { "I2", "I1" });

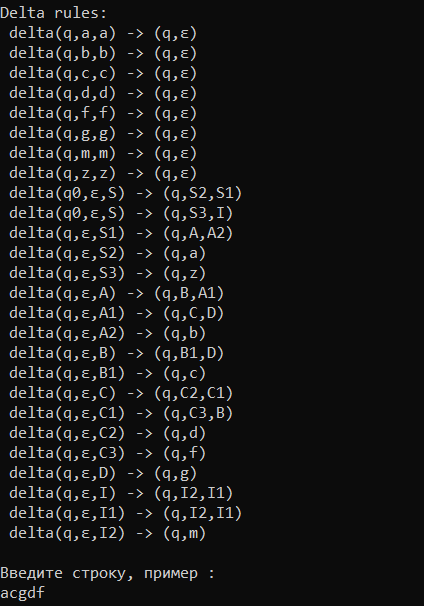
pda.addDeltaRule("q", "ε", "I2", new List<Symbol>() { "q" }, new List<Symbol>() { "m" });

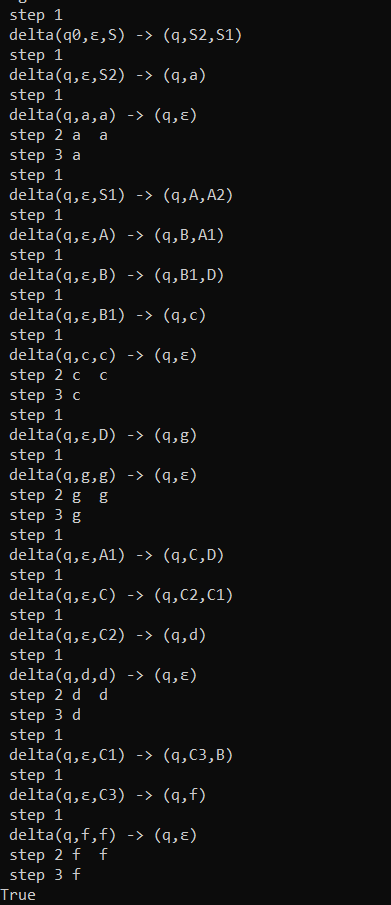
pda.Debug();

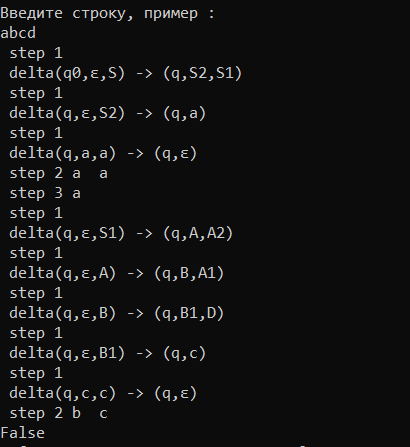
Console.WriteLine("\nВведите строку, пример :");

Console.WriteLine(pda.Execute(Console.ReadLine()).ToString());

**Пример работы программы:**







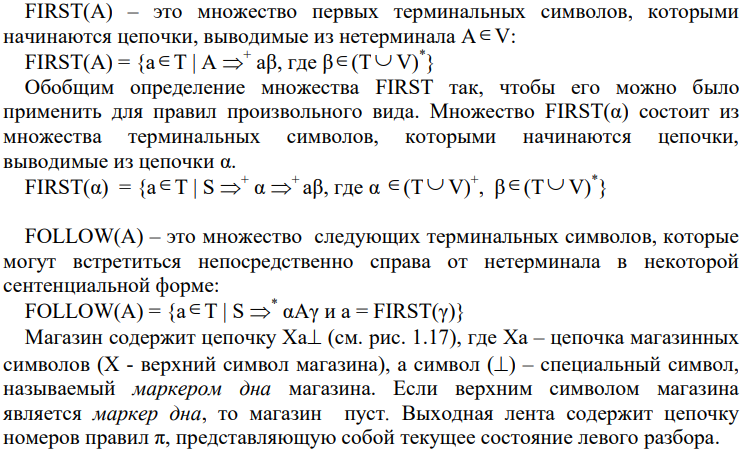
**Практическая работа №3 (лабораторные 9-11)**

*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 ∈ 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.

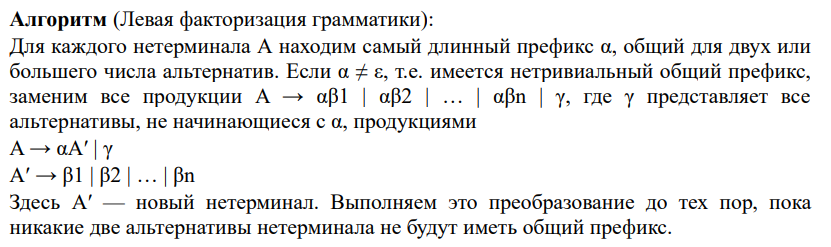
****

Исходная грамматика:

G = (**T**, **V**, **P**, **S**), где

**T** = { a,b,c,d,f,g,m,z }, **V** = { S,S1,A,A1,B,C,C1,D,I,I1}, **S0** = S

**P=**{S → aS1, S → zI, S1 → Ab, A → BA1, A1 → CD, B → cD, C → dC1, C1 → fB, D → g, I → mI1, I1 → mI1 | m}



1. Для I1 → mI1 | m введём новый нетерминал I2 и добавим правила I1 → mI2, I2 → ε | I1

Таким образом, получили LL(1)-грамматику:

G = ({ a,b,c,d,f,g,m,z }, **{**S,S1,A,A1,B,C,C1,D,I,I1,I2**}**, **P**, **S**), где

**P = {**

**P**1: S → aS1

**P**2: S → zI

**P**3: S1 → Ab

**P**4: A → BA1

**P**5: A1 → CD

**P**6: B → cD

**P**7: C → dC1

**P**8: C1 → fB

**P**9: D → g

**P**10: I → mI1

**P**11: I1 → mI2

**P**12: I2 → ε

**P**13: I2 → I1

**}**

acgdfcggb

**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** U **T** U {Ʇ}) × (**T** U {ε}) по правилам:

1. Если A → β – правило вывода грамматики с номером i, то M(А, a) = (β, i) для всех a ≠ ε, принадлежащих множеству FIRST(β). Если ε ∈FIRST(β), то M(А, b) = (β, i) для всех b ∈ FOLLOW(A).
2. M(a, a) = ВЫБРОС для всех a ∈ **T**.
3. M(Ʇ, ε) = ДОПУСК.
4. В остальных случаях M(X, a) = ОШИБКА для X(**V** U **T** U {Ʇ}) и a ∈ **T** U {ε}

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **a** | **b** | **c** | **d** | **f** | **g** | **m** | **z** | **ε** |
| **S** | aS1, 1 |  |  |  |  |  |  | zI, 2 |  |
| **S1** |  |  | Ab, 3 |  |  |  |  |  |  |
| **A** |  |  | BA1, 4 |  |  |  |  |  |  |
| **A1** |  |  |  | CD, 5 |  |  |  |  |  |
| **B** |  |  | cD, 6 |  |  |  |  |  |  |
| **C** |  |  |  | dC1, 7 |  |  |  |  |  |
| **C1** |  |  |  |  | fB, 8 |  |  |  |  |
| **D** |  |  |  |  |  | g, 9 |  |  |  |
| **I** |  |  |  |  |  |  | mI1, 10 |  |  |
| **I1** |  |  |  |  |  |  | mI2, 11 |  |  |
| **I2** |  |  |  |  |  |  | I1, 13 |  | ε, 12 |
| **a** | ВЫБРОС |  |  |  |  |  |  |  |  |
| **b** |  | ВЫБРОС |  |  |  |  |  |  |  |
| **c** |  |  | ВЫБРОС |  |  |  |  |  |  |
| **d** |  |  |  | ВЫБРОС |  |  |  |  |  |
| **f** |  |  |  |  | ВЫБРОС |  |  |  |  |
| **g** |  |  |  |  |  | ВЫБРОС |  |  |  |
| **m** |  |  |  |  |  |  | ВЫБРОС |  |  |
| **z** |  |  |  |  |  |  |  | ВЫБРОС |  |
| **Ʇ** |  |  |  |  |  |  |  |  | ДОПУСК |

Пустые клетки в таблице означают ОШИБКУ.

Аналитичекое представление для таблицы М:

|  |  |  |
| --- | --- | --- |
| Правило грамматики | Множество | Значение М |
| **p1**: S → aS1 | FIRST(S) = {a} | M(S, a) = aS1, 1 |
| **p2**: S → zI | FIRST(S) = {z} | M(S, z) = zI, 2 |
| **p3**: S1 → Ab | FIRST(S1) = {c} | M(S1, c) = Ab, 3 |
| **p4**: A → BA1 | FIRST(A) = {c} | M(A, c) = BA1, 4 |
| **p5**: A1 → CD | FIRST(A1) = {d} | M(A1, d) = CD, 5 |
| **p6**: B → cD | FIRST(B) = {c} | M(B, c) = cD, 6 |
| **p7**: C → dC1 | FIRST(C) = {d} | M(C, d) = dC1, 7 |
| **p8**: C1 → fB | FIRST(C1) = {f} | M(C1, f) = fB, 8 |
| **p9**: D → g | FIRST(D) = {g} | M(D, g) = g, 9 |
| **p10**: I → mI1 | FIRST(I) = {m} | M(I, m) = mI1, 10 |
| **p11**: I1 → mI2 | FIRST(I1) = {m} | M(I1, m) = mI2, 11 |
| **p12**: I2 → ε | FIRST(I2) = {ε} | M(I2, ε) = ε, 12 |
| **p13**: I2 → I1 | FIRST(I2) = {m} | M(I1, m) = I1, 13 |

*Лабораторная работа №10:*

**Формулировка задания:**

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

Рассмотрим работу алгоритма для цепочки (fcadbhgci)

Шаг 1. Алгоритм находится в начальной конфигурации ((acgdfcggb), S­Ʇ, ε), где S0 = K

Значение управляющей таблицы M(K, f) = (A, 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 М |
| (acgdfcggb, S⊥, ε) ⸠ | M(S, a) = aS1, 1 |
| (acgdfcggb, aS1⊥, 1) ⸠ | M(a, a) = ВЫБРОС |
| (cgdfcggb, S1⊥, 1) ⸠ | M(S1, c) = Ab, 3 |
| (cgdfcggb, Ab⊥, 13) ⸠ | M(A, c) = BA1, 4 |
| (cgdfcggb, BA1b⊥, 134) ⸠ | M(B, c) = cD, 6 |
| (cgdfcggb, cDA1b⊥, 1346) ⸠ | M(c, c) = ВЫБРОС |
| (gdfcggb, DA1b⊥, 1346) ⸠ | M(D, g) = g, 9 |
| (gdfcggb, gA1b⊥, 13469) ⸠ | M(g, g) = ВЫБРОС |
| (dfcggb, A1b⊥, 13469) ⸠ | M(A1, d) = CD, 5 |
| (dfcggb, CDb⊥, 134695) ⸠ | M(C, d) = dC1, 7 |
| (dfcggb, dC1Db⊥, 1346957) ⸠ | M(d, d) = ВЫБРОС |
| (fcggb, C1Db⊥, 1346957) ⸠ | M(C1, f) = fB, 8 |
| (fcggb, fBDb⊥, 13469578) ⸠ | M(f, f) = ВЫБРОС |
| (cggb, BDb⊥, 13469578) ⸠ | M(B, c) = cD, 6 |
| (cggb, cDDb⊥, 134695786) ⸠ | M(c, c) = ВЫБРОС |
| (ggb, DDb⊥, 134695786) ⸠ | M(D, g) = g, 9 |
| (ggb, gDb⊥, 1346957869) ⸠ | M(g, g) = ВЫБРОС |
| (gb, Db⊥, 1346957869) ⸠ | M(D, g) = g, 9 |
| (gb, gb⊥, 13469578699) ⸠ | M(g, g) = ВЫБРОС |
| (b, b⊥, 13469578699) ⸠ | M(b, b) = ВЫБРОС |
| (ε, ⊥, 13469578699) ⸠ | M(ε, ⊥) = ДОПУСК |

Так как M(ε, ⊥) = ДОПУСК, то цепочка acgdfcggb принадлежит языку и последовательность номеров правил 13469578699 на выходной ленте – это её разбор.

*Лабораторная работа №11:*

**Формулировка задания:**

Реализовать управляющую таблицу M Для LL(k) анализатора.

**Код программы:**



**Результат работы программы:**

9.2

Создадим таблицу. Сначала создадим по столбцу для каждого из этих терминалов:

a, b, c, d, f, g, m, z, ε,

Также создаем строку для Эпсилон

Рассмотрим нетерминал S

Первый символ правила S -> aS1 - a

Это правило заносим в таблицу на пересечении строки нетерминала S и столбца терминала a

Первый символ правила S -> zI - z

Это правило заносим в таблицу на пересечении строки нетерминала S и столбца терминала z

Рассмотрим нетерминал S1

Первый символ правила S1 -> Ab - c

Это правило заносим в таблицу на пересечении строки нетерминала S1 и столбца терминала c

Рассмотрим нетерминал A

Первый символ правила A -> BA1 - c

Это правило заносим в таблицу на пересечении строки нетерминала A и столбца терминала c

Рассмотрим нетерминал A1

Первый символ правила A1 -> CD - d

Это правило заносим в таблицу на пересечении строки нетерминала A1 и столбца терминала d

Рассмотрим нетерминал B

Первый символ правила B -> cD - c

Это правило заносим в таблицу на пересечении строки нетерминала B и столбца терминала c

Рассмотрим нетерминал C

Первый символ правила C -> dC1 - d

Это правило заносим в таблицу на пересечении строки нетерминала C и столбца терминала d

Рассмотрим нетерминал C1

Первый символ правила C1 -> fB - f

Это правило заносим в таблицу на пересечении строки нетерминала C1 и столбца терминала f

Рассмотрим нетерминал D

Первый символ правила D -> g - g

Это правило заносим в таблицу на пересечении строки нетерминала D и столбца терминала g

Рассмотрим нетерминал I

Первый символ правила I -> mI1 - m

Это правило заносим в таблицу на пересечении строки нетерминала I и столбца терминала m

Рассмотрим нетерминал I1

Первый символ правила I1 -> mI2 - m

Это правило заносим в таблицу на пересечении строки нетерминала I1 и столбца терминала m

Рассмотрим нетерминал I2

Первый символ правила I2 -> ε - ε

Это правило заносим в таблицу на пересечении строки нетерминала I2 и столбца терминала ε

Первый символ правила I2 -> I1 - m

Это правило заносим в таблицу на пересечении строки нетерминала I2 и столбца терминала m

Введите строку:

acgdfcggb

Приступаю к чтению цепочки символов...

(System.Collections.Generic.List`1[RaGlib.Core.Symbol], S, )

В вершине стека нетерминал S

В таблице разбора, в клетке [S,a] существует правило...

Извлекаю из стека элемент и заношу все терминалы и нетерминалы

найденного в таблице правила в стек в порядке обратном порядку их следования в правиле.

Вот так: S1a

Использовано правило под номером 1

(acgdfcggb, aS1EoS, 1)

В вершине стека находится терминал a

И данный терминал равен вершине стека...

Извлекаю из стека верхний элемент, распознаю символ входной последовательности...

ВЫБРОС!

(cgdfcggb, S1EoS, 1)

В вершине стека нетерминал S1

В таблице разбора, в клетке [S1,c] существует правило...

Извлекаю из стека элемент и заношу все терминалы и нетерминалы

найденного в таблице правила в стек в порядке обратном порядку их следования в правиле.

Вот так: bA

Использовано правило под номером 3

(cgdfcggb, AbEoS, 13)

В вершине стека нетерминал A

В таблице разбора, в клетке [A,c] существует правило...

Извлекаю из стека элемент и заношу все терминалы и нетерминалы

найденного в таблице правила в стек в порядке обратном порядку их следования в правиле.

Вот так: A1B

Использовано правило под номером 4

(cgdfcggb, BA1bEoS, 134)

В вершине стека нетерминал B

В таблице разбора, в клетке [B,c] существует правило...

Извлекаю из стека элемент и заношу все терминалы и нетерминалы

найденного в таблице правила в стек в порядке обратном порядку их следования в правиле.

Вот так: Dc

Использовано правило под номером 6

(cgdfcggb, cDA1bEoS, 1346)

В вершине стека находится терминал c

И данный терминал равен вершине стека...

Извлекаю из стека верхний элемент, распознаю символ входной последовательности...

ВЫБРОС!

(gdfcggb, DA1bEoS, 1346)

В вершине стека нетерминал D

В таблице разбора, в клетке [D,g] существует правило...

Извлекаю из стека элемент и заношу все терминалы и нетерминалы

найденного в таблице правила в стек в порядке обратном порядку их следования в правиле.

Вот так: g

Использовано правило под номером 9

(gdfcggb, gA1bEoS, 13469)

В вершине стека находится терминал g

И данный терминал равен вершине стека...

Извлекаю из стека верхний элемент, распознаю символ входной последовательности...

ВЫБРОС!

(dfcggb, A1bEoS, 13469)

В вершине стека нетерминал A1

В таблице разбора, в клетке [A1,d] существует правило...

Извлекаю из стека элемент и заношу все терминалы и нетерминалы

найденного в таблице правила в стек в порядке обратном порядку их следования в правиле.

Вот так: DC

Использовано правило под номером 5

(dfcggb, CDbEoS, 134695)

В вершине стека нетерминал C

В таблице разбора, в клетке [C,d] существует правило...

Извлекаю из стека элемент и заношу все терминалы и нетерминалы

найденного в таблице правила в стек в порядке обратном порядку их следования в правиле.

Вот так: C1d

Использовано правило под номером 7

(dfcggb, dC1DbEoS, 1346957)

В вершине стека находится терминал d

И данный терминал равен вершине стека...

Извлекаю из стека верхний элемент, распознаю символ входной последовательности...

ВЫБРОС!

(fcggb, C1DbEoS, 1346957)

В вершине стека нетерминал C1

В таблице разбора, в клетке [C1,f] существует правило...

Извлекаю из стека элемент и заношу все терминалы и нетерминалы

найденного в таблице правила в стек в порядке обратном порядку их следования в правиле.

Вот так: Bf

Использовано правило под номером 8

(fcggb, fBDbEoS, 13469578)

В вершине стека находится терминал f

И данный терминал равен вершине стека...

Извлекаю из стека верхний элемент, распознаю символ входной последовательности...

ВЫБРОС!

(cggb, BDbEoS, 13469578)

В вершине стека нетерминал B

В таблице разбора, в клетке [B,c] существует правило...

Извлекаю из стека элемент и заношу все терминалы и нетерминалы

найденного в таблице правила в стек в порядке обратном порядку их следования в правиле.

Вот так: Dc

Использовано правило под номером 6

(cggb, cDDbEoS, 134695786)

В вершине стека находится терминал c

И данный терминал равен вершине стека...

Извлекаю из стека верхний элемент, распознаю символ входной последовательности...

ВЫБРОС!

(ggb, DDbEoS, 134695786)

В вершине стека нетерминал D

В таблице разбора, в клетке [D,g] существует правило...

Извлекаю из стека элемент и заношу все терминалы и нетерминалы

найденного в таблице правила в стек в порядке обратном порядку их следования в правиле.

Вот так: g

Использовано правило под номером 9

(ggb, gDbEoS, 1346957869)

В вершине стека находится терминал g

И данный терминал равен вершине стека...

Извлекаю из стека верхний элемент, распознаю символ входной последовательности...

ВЫБРОС!

(gb, DbEoS, 1346957869)

В вершине стека нетерминал D

В таблице разбора, в клетке [D,g] существует правило...

Извлекаю из стека элемент и заношу все терминалы и нетерминалы

найденного в таблице правила в стек в порядке обратном порядку их следования в правиле.

Вот так: g

Использовано правило под номером 9

(gb, gbEoS, 13469578699)

В вершине стека находится терминал g

И данный терминал равен вершине стека...

Извлекаю из стека верхний элемент, распознаю символ входной последовательности...

ВЫБРОС!

(b, bEoS, 13469578699)

В вершине стека находится терминал b

И данный терминал равен вершине стека...

Извлекаю из стека верхний элемент, распознаю символ входной последовательности...

ВЫБРОС!

(, EoS, 13469578699)

Допуск. Цепочка символов = L(G).

13469578699

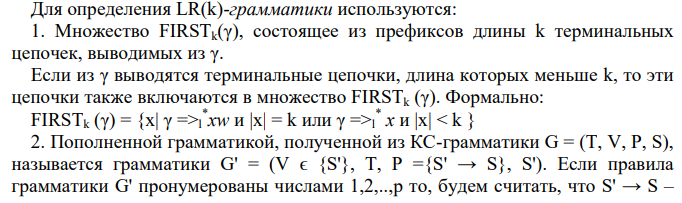
**Практическая работа №4 (лабораторные 12-16)**

**Формулировка задания:**

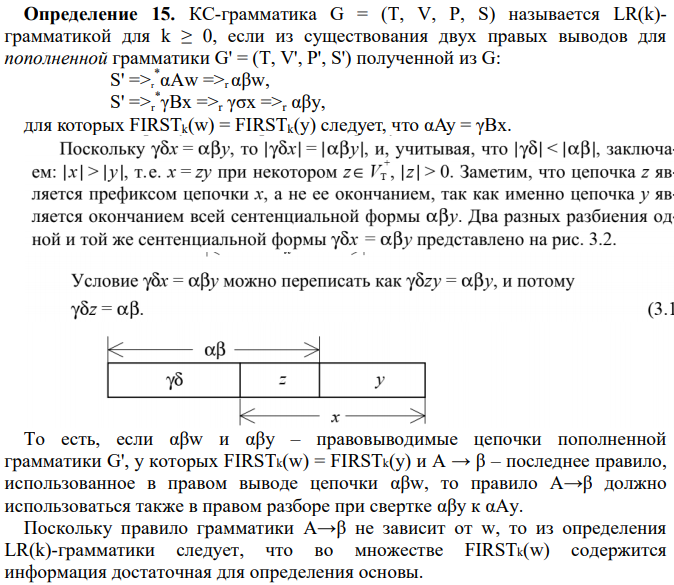
Построить управляющую таблицу M для LR(k)-грамматики, написать правило вывода выделенной строки. Описать работу алгоритма LR(k) анализатора. Построить LR(k) анализатор на основе грамматического вхождения.

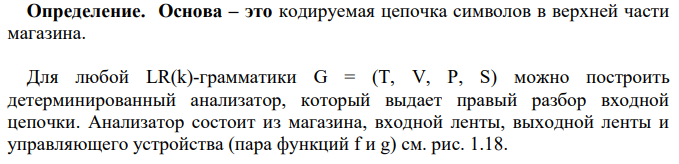
**Синтаксический LR-анализатор** анализирует входную цепочку слева направо (L), и строит правый (R) вывод грамматики.

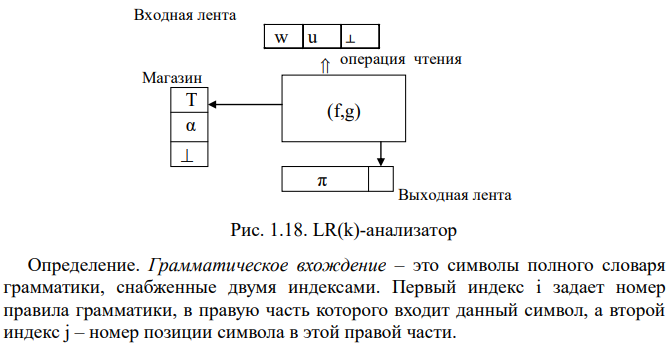
Грамматики, для которых можно построить детерминированный восходящий анализатор, называются LR(k)-*грамматиками* (входная цепочка читается слева (Left) направо, выходом анализатора является правый (Right) разбор, k – число символов входной цепочки, на которое можно “заглянуть” вперёд для выделения основы).











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**), где

**T** = {a, b, c, d, f, g, h, i}, **V** = {K, A, B, C, D, E, F, G, H}, **S0** = K

**P**:

p1: K → fA

p2: A → cB

p3: B → aC

p4: C → dD

p5: D → bE

p6: E → hF

p7: F → gG

p8: G → cH

p9: H → i

1. **Построение LR(k) анализатора на основе активных префиксов и отношения OBLOW:**

Непосредственно из правил вывода грамматики получим:

A1 OBLOW f1, B2 OBLOW c2, C3 OBLOW a3, D4 OBLOW d4, E5 OBLOW b5, F6 OBLOW h6, G7 OBLOW g7, H8 OBLOW c8

Из определения отношения OBLOW следует, что ⊥ OBLOW Yj ⬄ Yj ∈ OFIRST(S0 = K). Из K можно вывести цепочку K => f1A1. Следовательно, OFIRST(S0 = K) = {a1, S0}, и

⊥ OBLOW f1, ⊥ OBLOW K

Рассмотрим правило грамматики с номером (1). Из определения отношения OBLOW следует, что:

f1 OBLOW Yj для всех Yj ∈ OFIRST(A1).

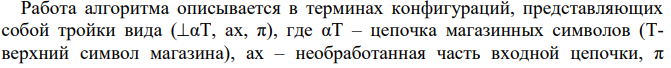
Из A можно вывести цепочки:

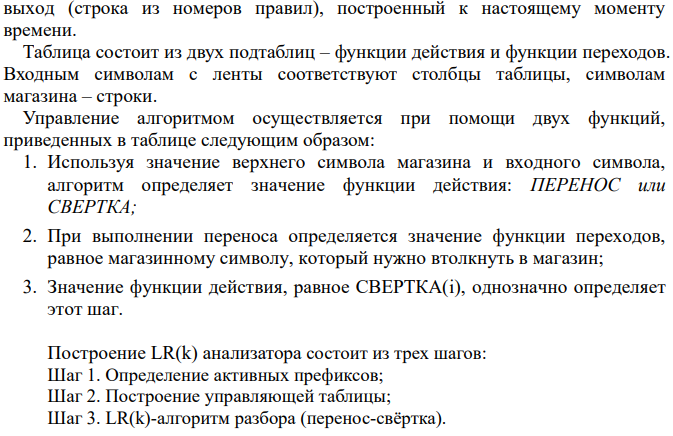
A =>2 c2B2 =>3 c2a3C3 =>4 c2a3d4D4 =>5 c2a3d4b5E5 =>6 c2a3d4b5h6F6 =>7 c2a3d4b5h6g7G7 =>8 c2a3d4b5h6g7c8H8 =>9 c2a3d4b5h6g7c8i9

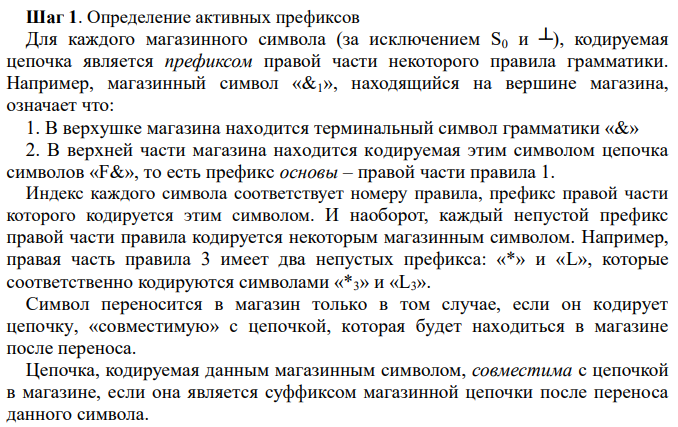
Следовательно, OFIRST(A­1) = {c2} и f1 OBLOW c2

Поступая подобным образом для остальных правил, получим матрицу отношения OBLOW:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **OBLOW** | S0 = K | a3 | b5 | c2 | c8 | d4 | f1 | g7 | h6 | i9 | A1 | B2 | C3 | D4 | E5 | F6 | G7 | H8 |
| S0 = K |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| a3 |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| b5 |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| c2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| c8 |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| d4 |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| f1 |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| g7 |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| h6 |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| i9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| A1 |  |  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |
| B2 |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| C3 |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| D4 |  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| E5 |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| F6 |  |  |  |  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |
| G7 |  |  |  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |
| H8 |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ⊥ | 1 |  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |







p1: K → fA

p2: A → cB

p3: B → aC

p4: C → dD

p5: D → bE

p6: E → hF

p7: F → gG

p8: G → cH

p9: H → i

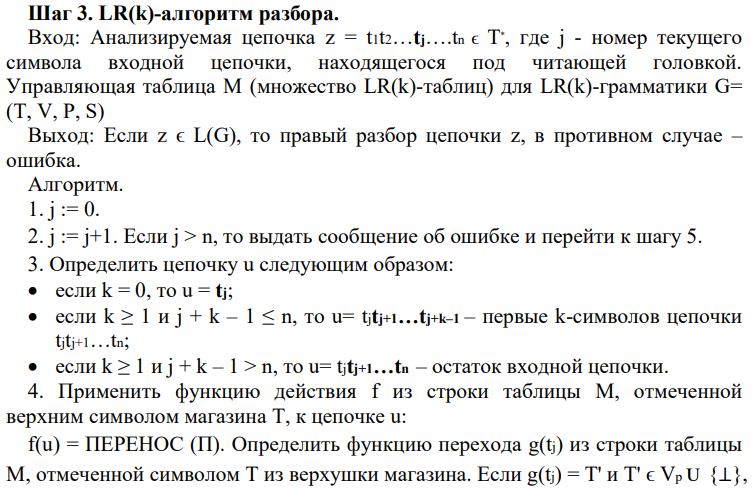
Таблица закодированных символов

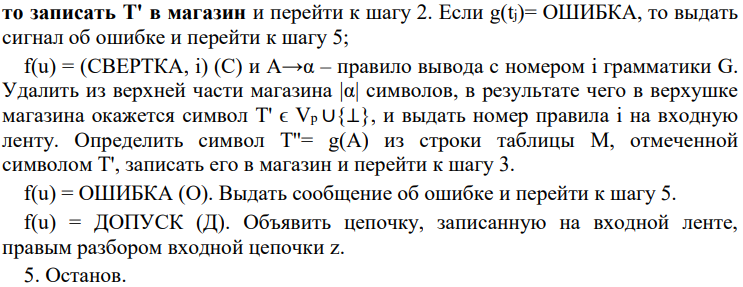
|  |  |  |  |
| --- | --- | --- | --- |
| Символ грамматики | Магазинный символ | Кодируемая цепочка | Операции |
| S0 = K | ­K0  A1 | ⊥K  fA | Д  П |
| A | A­1  B­2 | A  cB | Д  П |
| B | B2  C3 | B  aC | Д  П |
| C | C3  D4 | C  dD | Д  П |
| D | D4  E5 | D  bE | Д  П |
| E | E5  F6 | E  hF | Д  П |
| F | F6  G7 | F  gG | Д  П |
| G | G7  H8 | G  cH | Д  П |
| a | a3 | fca | С3 |
| b | b5 | fcadb | С5 |
| c | c2  c8 | fc  fcadbhgc | С2  С8 |
| d | d4 | fcad | С4 |
| f | f1 | f | С1 |
| g | g7 | fcadbhg | С7 |
| h | h6 | fcadbh | С6 |
| i | i9 | fcadbhgci | С9 |

**Основа** – кодируемая цепочка символов в верхней части магазина.

Магазинный алфавит построен таким образом, что для каждого магазинного символа (за исключением S0 и ⊥), кодируемая им цепочка является *префиксом* правой части некоторого правила грамматики.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | функция действий f(u) | | | | | | | | | функция переходов g(X) | | | | | | | | | | | | | | | |
| a | b | c | d | f | g | h | i | ⊥ | K | A | B | C | D | E | F | G | a | b | c | d | f | g | h | i |
| K0 | П | П | П | П | С(1) | П | П | П | Д |  | A1 |  |  |  |  |  |  |  |  |  |  | f1 |  |  |  |
| A1 | П | П | C(2) | П | П | П | П | П |  |  |  | B2 |  |  |  |  |  |  |  | c2 |  |  |  |  |  |
| B2 | C(3) | П | П | П | П | П | П | П |  |  |  |  | C3 |  |  |  |  | a3 |  |  |  |  |  |  |  |
| C3 | П | П | П | C(4) | П | П | П | П |  |  |  |  |  | D4 |  |  |  |  |  |  | d4 |  |  |  |  |
| D4 | П | C(5) | П | П | П | П | П | П |  |  |  |  |  |  | E5 |  |  |  | b5 |  |  |  |  |  |  |
| E5 | П | П | П | П | П | П | C(6) | П |  |  |  |  |  |  |  | F6 |  |  |  |  |  |  |  | h6 |  |
| F6 | П | П | П | П | П | C(7) | П | П |  |  |  |  |  |  |  |  | G7 |  |  |  |  |  | g7 |  |  |
| G7 | П | П | C(8) | П | П | П | П | П |  |  |  |  |  |  |  |  |  |  |  | c8 |  |  |  |  |  |
| a3 | П | П | П | П | П | П | П | П |  |  |  | B2 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| b5 | П | П | П | П | П | П | П | П |  |  |  |  |  | D4 |  |  |  |  |  |  |  |  |  |  |  |
| c2 | П | П | П | П | П | П | П | П |  |  | A1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| c8 | П | П | П | П | П | П | П | П |  |  |  |  |  |  |  |  | G7 |  |  |  |  |  |  |  |  |
| d4 | П | П | П | П | П | П | П | П |  |  |  |  | C3 |  |  |  |  |  |  |  |  |  |  |  |  |
| f1 | П | П | П | П | П | П | П | П |  | K0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| g7 | П | П | П | П | П | П | П | П |  |  |  |  |  |  |  | F6 |  |  |  |  |  |  |  |  |  |
| h6 | П | П | П | П | П | П | П | П |  |  |  |  |  |  | E5 |  |  |  |  |  |  |  |  |  |  |
| i9 | П | П | П | П | П | П | П | П |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ⊥ | П | П | П | П | П | П | П | П |  | K0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |





Рассмотрим работу алгоритма на примере цепочки (fcadbhgci):

(⊥(fcadbhgci)⊥,ɛ) ⸠ (⊥(­1,fcadbhgci)⊥,ɛ) ⸠ (⊥(­1f1,cadbhgci)⊥,ɛ) ⸠  
(⊥(­1A1,cadbhgci)⊥,1) ⸠ (⊥(­1c2,adbhgci)⊥,1) ⸠ (⊥(­1B2,adbhgci)⊥,12) ⸠  
(⊥(­1a3,dbhgci)⊥,12) ⸠ (⊥(­1C3,dbhgci)⊥,123) ⸠ (⊥(­1d4,bhgci)⊥,123) ⸠  
(⊥(­1D4,bhgci)⊥,1234) ⸠ (⊥(­1b5,hgci)⊥,1234) ⸠ (⊥(­1E5,hgci)⊥,12345) ⸠  
(⊥(­1h6,gci)⊥,12345) ⸠ (⊥(­1F6,gci)⊥,123456) ⸠ (⊥(­1g7,ci)⊥,123456) ⸠  
(⊥(­1G7,ci)⊥,1234567) ⸠ (⊥(­1c8,i)⊥,1234567) ⸠ (⊥(­1H8,i)⊥,12345678) ⸠  
(⊥(­1i9),⊥,12345678) ⸠ (⊥K,⊥,123456789) ⸠ ДОПУСК.

Управляющая таблица:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Операции | | | | | | | | | GOTO | | | | | | | | | |
|  | a | b | c | d | f | g | h | i | $ | K | A | B | C | D | E | F | G | H | S’ |
| 0 |  |  |  |  | С(1) |  |  |  |  | 2 |  |  |  |  |  |  |  |  |  |
| 1 |  |  | С(3) |  |  |  |  |  |  |  | 4 |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  | допуск |  |  |  |  |  |  |  |  |  |  |
| 3 | С(5) |  |  |  |  |  |  |  |  |  |  | 6 |  |  |  |  |  |  |  |
| 4 | П(1) | П(1) | П(1) | П(1) | П(1) | П(1) | П(1) | П(1) | П(1) |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  | С(7) |  |  |  |  |  |  |  |  | 8 |  |  |  |  |  |  |
| 6 | П(2) | П(2) | П(2) | П(2) | П(2) | П(2) | П(2) | П(2) | П(2) |  |  |  |  |  |  |  |  |  |  |
| 7 |  | С(9) |  |  |  |  |  |  |  |  |  |  |  | 10 |  |  |  |  |  |
| 8 | П(3) | П(3) | П(3) | П(3) | П(3) | П(3) | П(3) | П(3) | П(3) |  |  |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  | С(11) |  |  |  |  |  |  |  | 12 |  |  |  |  |
| 10 | П(4) | П(4) | П(4) | П(4) | П(4) | П(4) | П(4) | П(4) | П(4) |  |  |  |  |  |  |  |  |  |  |
| 11 |  |  |  |  |  | С(13) |  |  |  |  |  |  |  |  |  | 14 |  |  |  |
| 12 | П(5) | П(5) | П(5) | П(5) | П(5) | П(5) | П(5) | П(5) | П(5) |  |  |  |  |  |  |  |  |  |  |
| 13 |  | С(15) |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 16 |  |  |
| 14 | П(6) | П(6) | П(6) | П(6) | П(6) | П(6) | П(6) | П(6) | П(6) |  |  |  |  |  |  |  |  |  |  |
| 15 |  |  |  |  |  |  |  | С(17) |  |  |  |  |  |  |  |  |  | 18 |  |
| 16 | П(7) | П(7) | П(7) | П(7) | П(7) | П(7) | П(7) | П(7) | П(7) |  |  |  |  |  |  |  |  |  |  |
| 17 | П(9) | П(9) | П(9) | П(9) | П(9) | П(9) | П(9) | П(9) | П(9) |  |  |  |  |  |  |  |  |  |  |
| 18 | П(8) | П(8) | П(8) | П(8) | П(8) | П(8) | П(8) | П(8) | П(8) |  |  |  |  |  |  |  |  |  |  |