Московский авиационный институт

(национальный исследовательский институт)

Институт «Компьютерные науки и прикладная математика»

**Лабораторные работы**

**по курсу**

**«Системы программирования»**

**IV семестр**

**Практическая работа №1**

**1лаб**. Спроектировать грамматику по паттерн-модели регулярного языка.

**2лаб.** Преобразовать спроектированную грамматику в конечный автомат, составить диаграмму переходов КА и реализовать.

**3лаб.** Определить свойства КА. Изучить алгоритм преобразования НДКА в ДКА.

**Практическая работа №2**

**4лаб.** Устранить из КС-грамматики бесполезные символы и ε–правила.

**5лаб.** Устранить из KС-грамматики цепные правила и устранить левую рекурсию.

**6лаб.** Определить форму КС-грамматики и сделать ее приведение.

**7лаб.** Спроектировать МП-автомат для приведенной КС-грамматики.

**8лаб.** Реализовать МП-автомат для приведенной КС-грамматики.

**Практическая работа №3**

**9лаб.** Для LL(k) анализатора построить управляющую таблицу M.

**10лаб.** Аналитически написать правила вывода для цепочки LL(k) анализатора.

**11лаб.** Реализовать управляющую таблицу M Для LL(k) анализатора.

**Практическая работа №4**

**12лаб.** Построить множество LR(0)-таблиц не содержащих ε-правила.

**13лаб.** Для LR(k) -грамматики cпроектировать матрицу oblow.

**14лаб.** Определить функции перехода g(X).

**15лаб.** Определить функцию переноса-свертки f(u).

**16лаб.** Для функции перехода g(X) и функции переноса-свертки f(u) спроектировать управляющую таблицу.

*Студент:* Xxxx X.X.*Группа: М8О-20*

*Руководитель:* Семёнов А. С.

*Оценка:*

*Дата:*

**Москва. 2023**

**Практическая работа №1 (1-3 лаб.)**

*Лабораторные работы №1-2*

**Формулировка задания**:

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

Спроектируем грамматику для заданного языка:

**1. pattern = "\D+"**

**Автоматная грамматика:**

L(pattern) = L("\D+") = 0 | ({1, ..., 9} {0, ..., 9}\*)

G(T, V, P, S0) = G({0, 1, ..., 9}, {**S0**, **A**}, {p1, p2, p3}, **S0**)

*Правила регулярной грамматики:*

**p1:** **S0** → 1**A** | 2**A** | ... | 9**A**

**p2:** **A** → ε

**p3:** **A** → 0**A** | 1**A** | ... | 9**A**

*Пример цепочек:*

**S0** =>1 1**A** =>3 15**A** =>3 158**A** =>2 15**8**

**S0** =>1 5**A** =>3 57**A** =>3 571**A** =>3 5715**A** =>3 57153**A** =>2 57153

**S0** =>1 3**A** =>2 3

**Конечный автомат:**

L(КА) = L(G)

КА = (**Q**, **Σ**, **δ**, **S0**, **F**), где

**Q** = { **S0**, **A**, **qf** }, **Σ** = { 0, 1, ..., 9 }, **S0** = **S0**, **F** = **qf**,

**δ** = { 1. δ(**S0**, 1) = {**A**},

...

9. δ(**S0**, 9) = {**A**},

10. δ(**A**, 0) = {**A**},

...

19. δ(**A**, 9) = {**A**},

20. δ(**A**, ε) = {**qf**}

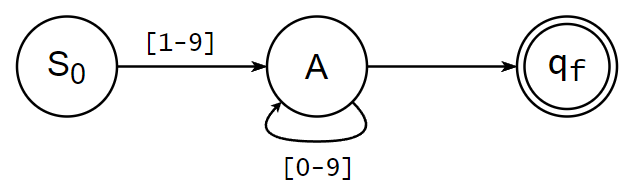
}

*Примеры конфигурации КА:*

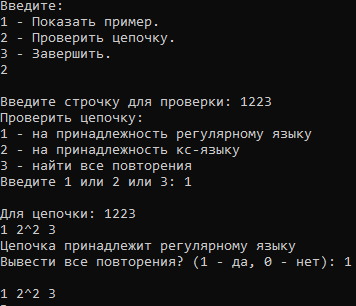
**1.** (**S0**, 158) ⸠1 (**A**, 58) ⸠15 (**A**, 8) ⸠18 (**A**, ε) ⸠20 (**qf**, ε)

**2.** (**S0**, 57153) ⸠5 (**A**, 7153) ⸠17 (**A**, 153) ⸠11 (**A**, 53) ⸠15 (**A**, 3) ⸠13 (**A**, ε) ⸠20 (**qf**, ε)

**3.** (**S0**, 3) ⸠3 (**A**, ε) ⸠20 (**qf**, ε)



*Лемма о накачке:*



**2. pattern = "one|two|three"**

**Автоматная грамматика:**

L(pattern) = L("one|two|three") = {one, two, three}

G(T, V, P, S0) = G({o, n, e, t, w, h, r}, {**S0**, **A**, ..., **G**}, {p1, ..., p10}, **S0**)

*Правила регулярной грамматики:*

**p1:** **S0** → o**A**

**p2:** **A** → n**B**

**p3:** **B** → e

**p4:** **S0** → t**C**

**p5:** **C** → w**D**

**p6:** **D** → o

**p7:** **C** → h**E**

**p8:** **E** → r**F**

**p9:** **F** → e**G**

**p10:** **G** → e

*Пример цепочек:*

**S0** =>1 o**A** =>2 on**B** =>3 one

**S0** =>4 t**C** =>5 tw**D** =>6 two

**S0** =>4 t**C** =>7 th**E** =>8 thr**F** =>9 thre**G** =>10 three

**Конечный автомат:**

L(КА) = L(G)

КА = (**Q**, **Σ**, **δ**, **S0**, **F**), где

**Q** = { **S0**, **A**, ..., **G**, **qf** }, **Σ** = { o, n, e, t, w, h, r }, **S0** = S0, **F** = qf,

**δ** = { 1. δ(**S0**, o) = {**A**}, 6. δ(**D**, o) = {**qf**},

2. δ(**A**, n) = {**B**},

3. δ(**B**, e) = {**qf**},

4. δ(**S0**, t) = {**C**},

5. δ(**C**, w) = {**D**},

7. δ(**C**, h) = {**E**},

8. δ(**E**, r) = {**F**},

9. δ(**F**, e) = {**G**},

10. δ(**G**, e) = {**qf**}

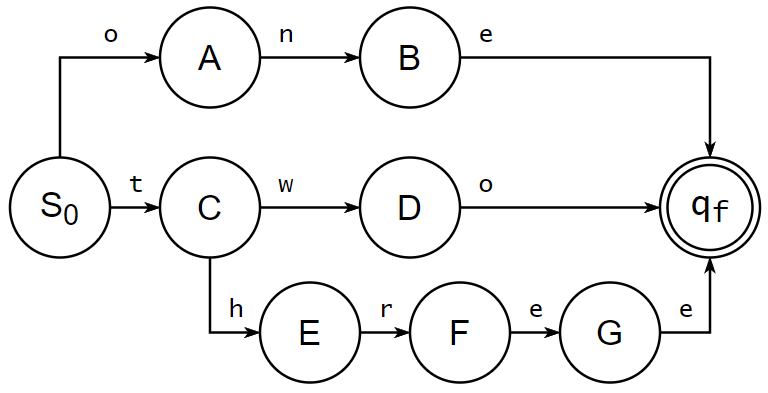
}

*Примеры конфигурации КА:*

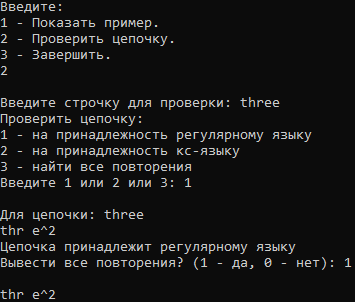
**1.** (**S0**, one) ⸠1 (**A**, ne) ⸠2 (**B**, e) ⸠3 (**qf**, ε)

**2.** (**S0**, two) ⸠4 (**C**, wo) ⸠5 (**D**, o) ⸠6 (**qf**, ε)

**3.** (**S0**, three) ⸠4 (**C**, hree) ⸠7 (**E**, ree) ⸠8 (**F**, ee) ⸠9 (**G**, e) ⸠10 (**qf**, ε)



*Лемма о накачке:*



**3. pattern = "(?i)[$s]ch[o0]{2}l"**

**Автоматная грамматика:**

L(pattern) = L("(?i)[$s]ch[o0]{2}l") = {"school", ..., "$CH00L"}

G(T, V, P, S0) = G({s, S, $, c, ..., l, L}, {**S0**, **A'**, ..., **E'**}, {p1, ..., p6}, **S0**)

*Правила регулярной грамматики:*

**p1:** **S0** → s**A'** | S**A'** | $**A'**

**p2:** **A'** → c**B'** | C**B'**

**p3:** **B'** → h**C'** | H**C'**

**p4:** **C'** → o**D'** | O**D'** | 0**D'**

**p5:** **D'** → o**E'** | O**E'** | 0**E'**

**p6:** **E'** → l | L

*Пример цепочек:*

**S0** =>1 s**A'** =>2 sc**B'** =>3 sch**C'** =>4 scho**D'** =>5 scho0**E'** =>6 scho0L

**S0** =>1 S**A'** =>2 SC**B'** =>3 SCH**C'** =>4 SCH0**D'** =>5 SCH0O**E'** =>6 SCH0Ol

**S0** =>1 $**A'** =>2 $c**B'** =>3 $ch**C'** =>4 $cho**D'** =>5 $choO**E'** =>6 $choOL

**Конечный автомат:**

L(КА) = L(G)

КА = (**Q**, **Σ**, **δ**, **S0**, **F**), где

**Q** = { **S0**, **A'**, ..., **E'**, **qf** }, **Σ** = { s, S, $, c, ..., l, L }, **S0** = S0, **F** = qf,

**δ** = { 1. δ(**S0**, s) = {**A'**},

2. δ(**S0**, S) = {**A'**},

3. δ(**S0**, $) = {**A'**},

4. δ(**A'**, c) = {**B'**},

5. δ(**A'**, C) = {**B'**},

6. δ(**B'**, h) = {**C'**},

7. δ(**B'**, H) = {**C'**},

8. δ(**C'**, o) = {**D'**},

9. δ(**C'**, O) = {**D'**},

10. δ(**C'**, 0) = {**D'**},

11. δ(**D'**, o) = {**E'**},

12. δ(**D'**, O) = {**E'**},

13. δ(**D'**, 0) = {**E'**},

14. δ(**E'**, l) = {**qf**},

15. δ(**E'**, L) = {**qf**}

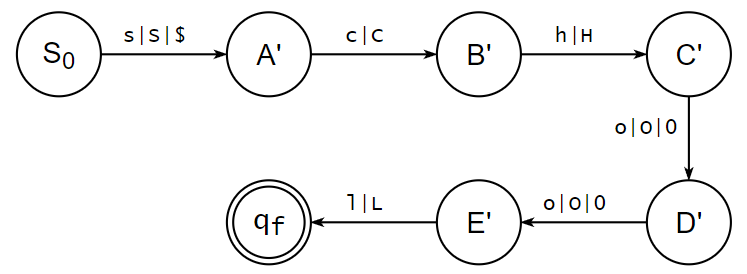
}

*Примеры конфигурации КА:*

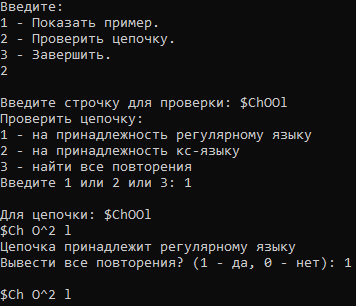
**1.** (**S0**, scho0l) ⸠1 (**A'**, cho0l) ⸠4 (**B'**, ho0l) ⸠6 (**C'**, o0l) ⸠8 (**D'**, 0l) ⸠10 (**E'**, l) ⸠14 (**qf**, ε)

**2.** (**S0**, SCH0Ol) ⸠2 (**A'**, CH0Ol) ⸠5 (**B'**, H0Ol) ⸠7 (**C'**, 0Ol) ⸠10 (**D'**, Ol) ⸠9 (**E'**, l) ⸠14 (**qf**, ε)

**3.** (**S0**, $choOL) ⸠3 (**A'**, choOL) ⸠4 (**B'**, hoOL) ⸠6 (**C'**, oOL) ⸠8 (**D'**, OL) ⸠9 (**E'**, L) ⸠15 (**qf**, ε)



*Лемма о накачке:*



*Лабораторная работа №3*

**Формулировка задания:**

Реализовать конечные автоматы, составленные в ЛР №2

**Код программы:**

FSAutomate[] automates = new FSAutomate[] {

new FSAutomate(

    new List<Symbol>() { "S01", "A1", "qf1" },

        new List<Symbol>() { "0", "1", "2", "3", "4", "5", "6", "7", "8", "9" },

        new List<Symbol>() { "qf1" },

        "S01"

    ),

    new FSAutomate(

        new List<Symbol>() { "S02", "A2", "B2", "C2", "D2", "E2", "F2", "G2", "qf2" },

      new List<Symbol>() { "o", "n", "e", "t", "w", "h", "r" },

        new List<Symbol>() { "qf2" },

        "S02"

    ),

    new FSAutomate(

        new List<Symbol>() { "S03", "A3", "B3", "C3", "D3", "E3", "qf3" },

        new List<Symbol>() { "s", "S", "$", "c", "C", "h", "H", "o", "O", "0", "l", "L" },

        new List<Symbol>() { "qf3" },

        "S03"

    ),

};

// deltas

{ // first FSA

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

foreach (string curNumber in numbers) {

automates[0].AddRule("S01", curNumber, "A1");

    automates[0].AddRule("A1", curNumber, "A1");

}

automates[0].AddRule("A1", "0", "A1");

automates[0].AddRule("A1", "", "qf1");

}

{ // second FSA

automates[1].AddRule("S02", "o", "A2");

    automates[1].AddRule("A2", "n", "B2");

    automates[1].AddRule("B2", "e", "qf2");

    automates[1].AddRule("S02", "t", "C2");

    automates[1].AddRule("C2", "w", "D2");

    automates[1].AddRule("D2", "o", "qf2");

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

    automates[1].AddRule("E2", "r", "F2");

    automates[1].AddRule("F2", "e", "G2");

    automates[1].AddRule("G2", "e", "qf2");

}

{ // third FSA

automates[2].AddRule("S03", "s", "A3");

    automates[2].AddRule("S03", "S", "A3");

    automates[2].AddRule("S03", "$", "A3");

    automates[2].AddRule("A3", "c", "B3");

    automates[2].AddRule("A3", "C", "B3");

    automates[2].AddRule("B3", "h", "C3");

    automates[2].AddRule("B3", "H", "C3");

    automates[2].AddRule("C3", "o", "D3");

    automates[2].AddRule("C3", "O", "D3");

    automates[2].AddRule("C3", "0", "D3");

    automates[2].AddRule("D3", "o", "E3");

    automates[2].AddRule("D3", "O", "E3");

    automates[2].AddRule("D3", "0", "E3");

    automates[2].AddRule("E3", "l", "qf3");

    automates[2].AddRule("E3", "L", "qf3");

}

**Пример работы программы:**

Automate 1: "\D+" (for ex.: 123)

Automate 2: "one|two|three" (for ex.: one)

Automate 3: "(?i)[$s]ch[o0]{2}l" (for ex.: school, $SCH0OL)

Enter number of automate:

1

You chose automate number 1

Enter line ("\D+", for ex.: 123) to execute

or "exit" to exit:

123

Length: 4

i: 4

curr: qf1

chineSymbol belongs to language

Enter line to execute or "exit" to exit:

423

Length: 4

i: 4

curr: qf1

chineSymbol belongs to language

Enter line to execute or "exit" to exit:

223f2342

Length: 8

i: 3

curr: A1

chineSymbol doesn't belong to language

Enter line to execute or "exit" to exit:

exit

Automate 1: "\D+" (for ex.: 123)

Automate 2: "one|two|three" (for ex.: one)

Automate 3: "(?i)[$s]ch[o0]{2}l" (for ex.: school, $SCH0OL)

Enter number of automate:

2

You chose automate number 2

Enter line ("one|two|three", for ex.: one) to execute

or "exit" to exit:

one

Length: 3

i: 3

curr: qf2

chineSymbol belongs to language

Enter line to execute or "exit" to exit:

two

Length: 3

i: 3

curr: qf2

chineSymbol belongs to language

Enter line to execute or "exit" to exit:

three

Length: 5

i: 5

curr: qf2

chineSymbol belongs to language

Enter line to execute or "exit" to exit:

exit

Automate 1: "\D+" (for ex.: 123)

Automate 2: "one|two|three" (for ex.: one)

Automate 3: "(?i)[$s]ch[o0]{2}l" (for ex.: school, $SCH0OL)

Enter number of automate:

3

You chose automate number 3

Enter line ("(?i)[$s]ch[o0]{2}l", for ex.: $SCH0OL) to execute

or "exit" to exit:

$cho0L

Length: 6

i: 6

curr: qf3

chineSymbol belongs to language

Enter line to execute or "exit" to exit:

schaol

Length: 6

i: 3

curr: C3

chineSymbol doesn't belong to language

Enter line to execute or "exit" to exit:

exit

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

*Лабораторная работа №4:*

**Формулировка задания:**

Устранить из КС-грамматики бесполезные символы и ε–правила

Исходная грамматика (a 2222, K e):

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

**T** = {ɛ, k, /, j, ), e, i, s};

**V** = {K, P, N, R, U, J, I, F, V, S, M};

**P** = {N → P/N, N → PU, U → N)U, P → kF, J → e, P → PNI, I → e, K → F, F → ɛ, R → V, V → j, F → iS, S → SeR, J → sM, M → MkR};

**S0** = K.

p1: N → P/N

p2: N → PU

p3: U → N)U

p4: P → kF

p5: J → e

p6: P → PNI

p7: I → e

p8: K → P

p9: F → ɛ

p10: R → V

p11: V → j

p12: F → iS

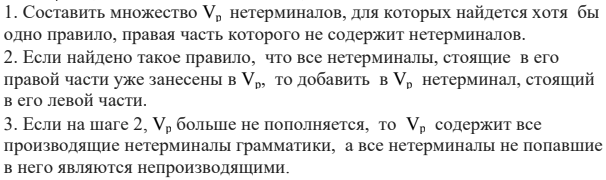
p13: S → SeR

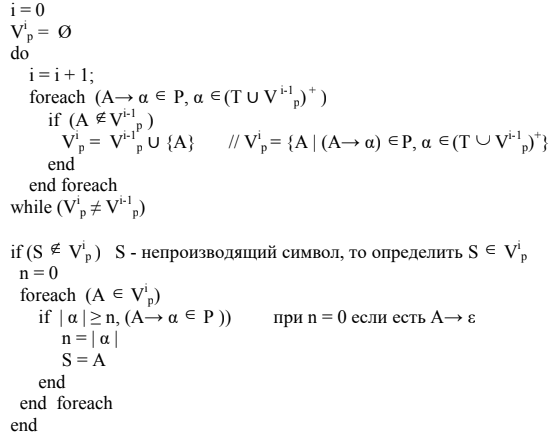
p14: J → sM

p15: M → MkR

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

**Устранение непроизводящих символов.**

****

****

1. V1p = {J, I, F, V} – для этих нетерминалов нашлось хотя бы одно правило, правая часть которого не содержит нетерминалов

2. V2p = {J, I, F, V, P, K, R} – 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. Vp = {J**,** I, F, V, P, K, R} содержит все производящие нетерминалы грамматики, а все нетерминалы, не попавшие в него, являются непроизводящими (V - Vp = {N, U, S, M}).



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

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

p1: P → kF

p2: J → e

p3: I → e

p4: K → P

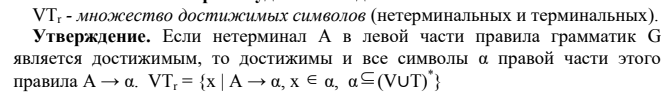
p5: F → ɛ

p6: R → V

p7: V → j

G1 = ({ɛ, e, j, k}, {J, I, F, V, P, K, R}, P', K)

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

****

****

VT1r = {K}

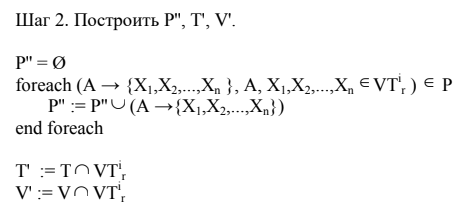
VT2r = {K, P}

VT2r = {K, P, k, F}

VT2r = {K, P, k, F, ɛ}

Так, не входящие в VTr символы {N, R, U, J, I, V, S, M, /, j, ), e, i, s} недостижимы

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

****

P'' = {p1, p2, p3}:

p1: P → kF

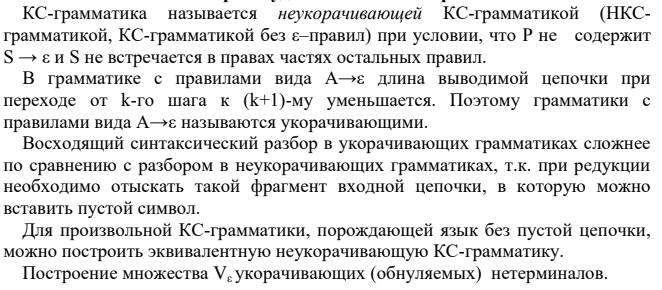
p2: K → P

p3: F → ɛ

Получаем G' = ({k, ɛ}, {K, P, F}, P, K), где P = {P → kF, K → P, F → ɛ}

L(G') = {k}

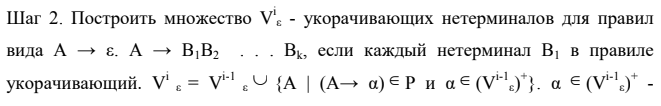
**Устранить из КС-грамматики ε–правила.**

****

*Построим множество укорачивающих нетерминалов согласно алгоритму:*

**

V0ɛ = {F} (множество укорачивающих нетерминалов для правил вида A → ε)

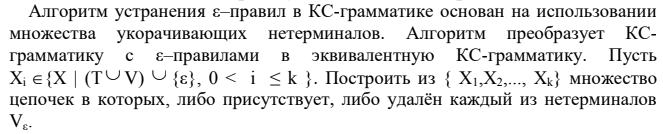




Необходимо построить множество V1ɛ укорачивающих нетерминалов для правилвида A → B1B2...Bk, если каждый нетерминал B1 в правилеукорачивающий. Однако в данном примере таких правил нет.

Итак, множество укорачивающих терминалов: Vɛ = {F}.

*Составим G" на основе алгоритма удаления эпсилон правил:*



G' = ({k, ɛ}, {K, P, F}, P, K), где P = {P → kF, K → P, F → ɛ}

G" = ({k}, {K, P}, P', K), где P' = {P → k, K → P}

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

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

****

Цепные правила:

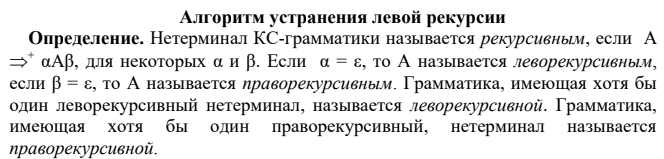
K → P, P → k

V1x = {P | K → P}



Правила K → P, P → k “заменим” на K → k

В сгенерированном варианте КС-грамматике отсуствует левая рекурсия

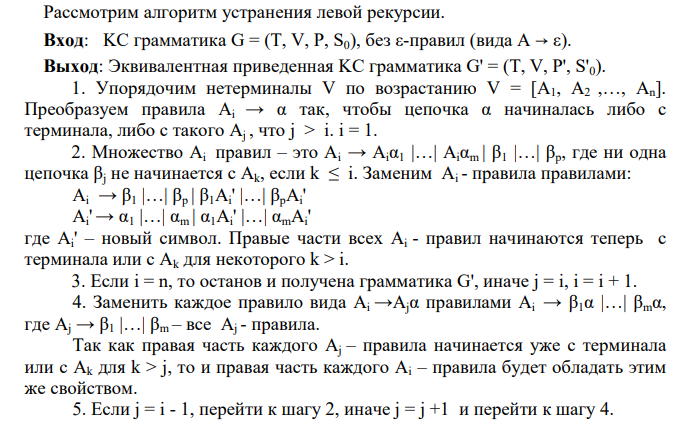


**Устранение из KС-грамматики левой рекурсии.**

Нетерминал КС-грамматики называется рекурсивным, если A =>+ αAβ, для некоторых α и β. Если α = ε, то A называется леворекурсивным.

Грамматика рассматриваемого языка не содержит левой рекурсии.

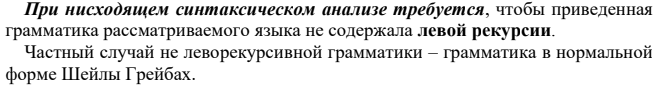
Если бы левая рекурсия содержалась – её необходимо устранить согласно алгоритму:



*Лабораторная работа №6*

**Формулировка задания:**

Определить форму КС-грамматики и сделать ее приведение



КС-грамматика в нормальной форме Шейлы Грейбах.

Приведённый вид грамматики:

Так, Gпр = ({k}, {K}, {K → k}, K)

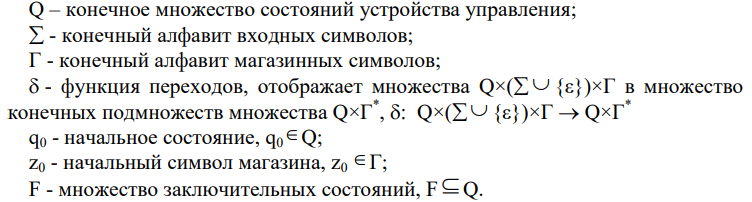
*Лабораторная работа №7*

**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, 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

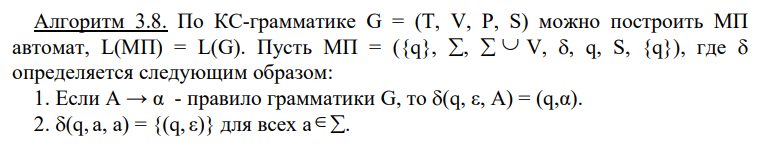
p5: D → bE

p6: E → hF

p7: F → gG

p8: G → cH

p9: H → i



L(МП) = L(G)

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

**Q** = {q}, **Σ** = **T**, **Г** = **T** U **V**, **δ** = **δ**, **q0** = **q0**, **z0** = **S0**, **F** = {q}

МП = ({q}, {a, b, c, d, f, g, h, i}, {a, b, c, d, f, g, h, i, K, A, B, C, D, E, F, G, H}, **δ**, q0, K, {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: 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: δ(q0, ε, K) = (q, fA)  2: δ(q, ε, A) = (q, cB)  3: δ(q, ε, B) = (q, aC)  4: δ(q, ε, C) = (q, dD)  5: δ(q, ε, D) = (q, bE)  6: δ(q, ε, E) = (q, hF)  7: δ(q, ε, F) = (q, gG)  8: δ(q, ε, G) = (q, cH)  9: δ(q, ε, H) = (q, i)  10: δ(q, a, a) = (q, ε) для всех a ∈ Σ |

Последовательность тактов МП-автомата fcadbhgci:

(q0, fcadbhgci, K) ⸠1 (q, cadbhgci, fA) ⸠2 (q, adbhgci, cB) ⸠3 ((q, dbhgci, aC) ⸠4 (q, bhgci, dD) ⸠5 (q, hgci, bE) ⸠6 (q, gci, hF) ⸠7 (q, ci, gG) ⸠8 (q, i, cH) ⸠9 (q, i, i) ⸠10 (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", "h", "i" },

    new List<Symbol>() { "K", "A", "B", "C", "D", "E", "F", "G", "H" },

    "K"

);

cFGr.AddRule("K", new List<Symbol>() { "f", "A" });

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

cFGr.AddRule("B", new List<Symbol>() { "a", "C" });

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

cFGr.AddRule("D", new List<Symbol>() { "b", "E" });

cFGr.AddRule("E", new List<Symbol>() { "h", "F" });

cFGr.AddRule("F", new List<Symbol>() { "g", "G" });

cFGr.AddRule("G", new List<Symbol>() { "b", "H" });

cFGr.AddRule("H", new List<Symbol>() { "i" });

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

cFGr.DebugPrules();

var pDA = new PDA(

    new List<Symbol>() { "q0", "q1", "qf" },

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

    new List<Symbol>() { "a", "b", "c", "d", "f", "g", "h", "i", "K", "A", "B", "C", "D", "E", "F", "G", "H" },

    "q0",

    "K",

    new List<Symbol>() { "qf" }

);

pDA.addDeltaRule("q0", "f", "K", new List<Symbol>() { "q" }, new List<Symbol>() { "f", "A" });

pDA.addDeltaRule("q", "c", "f", new List<Symbol>() { "q" }, new List<Symbol>() { "c", "B" });

pDA.addDeltaRule("q", "a", "c", new List<Symbol>() { "q" }, new List<Symbol>() { "a", "C" });

pDA.addDeltaRule("q", "d", "a", new List<Symbol>() { "q" }, new List<Symbol>() { "d", "D" });

pDA.addDeltaRule("q", "b", "d", new List<Symbol>() { "q" }, new List<Symbol>() { "b", "E" });

pDA.addDeltaRule("q", "h", "b", new List<Symbol>() { "q" }, new List<Symbol>() { "h", "F" });

pDA.addDeltaRule("q", "g", "h", new List<Symbol>() { "q" }, new List<Symbol>() { "g", "G" });

pDA.addDeltaRule("q", "c", "g", new List<Symbol>() { "q" }, new List<Symbol>() { "c", "H" });

pDA.addDeltaRule("q", "i", "c", new List<Symbol>() { "q" }, new List<Symbol>() { "f", "f" });

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

pDA.Debug();

Console.WriteLine("\nВведите строку (например fcadbhgci):\n");

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

Console.ReadLine();

**Пример работы программы:**

Debug KC-Grammar Prules:

K -> fA

A -> cB

B -> aC

C -> dD

D -> bE

E -> hF

F -> gG

G -> bH

H -> i

delta(q0,f,K) -> (q,f,A)

delta(q,c,f) -> (q,c,B)

delta(q,a,c) -> (q,a,C)

delta(q,d,a) -> (q,d,D)

delta(q,b,d) -> (q,b,E)

delta(q,h,b) -> (q,h,F)

delta(q,g,h) -> (q,g,G)

delta(q,c,g) -> (q,c,H)

delta(q,i,c) -> (q,f,f)

delta(q,ε,f) -> (qf,ε)

Введите строку (например fcadbhgci):

**> fcadbhgci**

step 1

delta(q0,f,K) -> (q,f,A)

step 2 f f

step 3 f

step 1

delta(q,c,f) -> (q,c,B)

step 2 c c

step 3 c

step 1

delta(q,a,c) -> (q,a,C)

step 2 a a

step 3 a

step 1

delta(q,d,a) -> (q,d,D)

step 2 d d

step 3 d

step 1

delta(q,b,d) -> (q,b,E)

step 2 b b

step 3 b

step 1

delta(q,h,b) -> (q,h,F)

step 2 h h

step 3 h

step 1

delta(q,g,h) -> (q,g,G)

step 2 g g

step 3 g

step 1

delta(q,c,g) -> (q,c,H)

step 2 c c

step 3 c

step 1

delta(q,a,c) -> (q,a,C)

step 2 i a

step 4

delta(q,i,c) -> (q,f,f)

step 1

delta(q,c,f) -> (q,c,B)

step 2 ε c

step 4

delta(q,ε,f) -> (qf,ε)

step 5 f

**True**

Введите строку (например fcadbhgci):

**> fsdjnfj**

step 1

delta(q0,f,K) -> (q,f,A)

step 2 f f

step 3 f

step 1

delta(q,c,f) -> (q,c,B)

step 2 s c

**False**

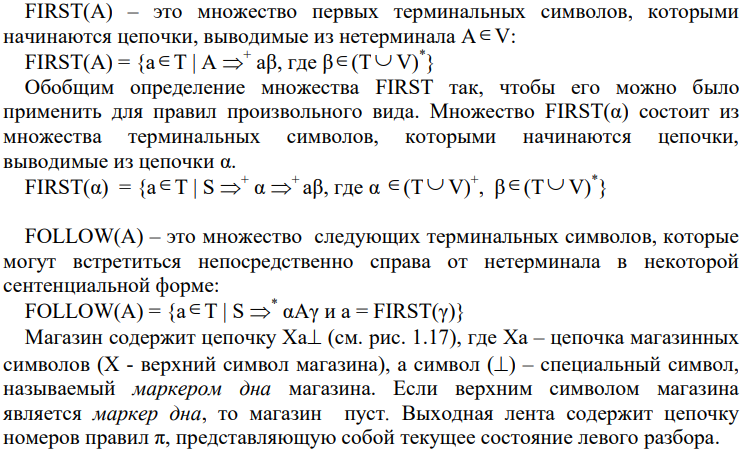
**Практическая работа №3 (лабораторные 9-11)**

*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, 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

**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** | **h** | **i** | **ε** |
| **K** |  |  |  |  | A, 1 |  |  |  |  |
| **A** |  |  | B, 2 |  |  |  |  |  |  |
| **B** | C, 3 |  |  |  |  |  |  |  |  |
| **C** |  |  |  | B, 4 |  |  |  |  |  |
| **D** |  | E, 5 |  |  |  |  |  |  |  |
| **E** |  |  |  |  |  |  | F, 6 |  |  |
| **F** |  |  |  |  |  | G, 7 |  |  |  |
| **G** |  |  | H, 8 |  |  |  |  |  |  |
| **H** |  |  |  |  |  |  |  | ε, 9 |  |
| **a** | ВЫБРОС |  |  |  |  |  |  |  |  |
| **b** |  | ВЫБРОС |  |  |  |  |  |  |  |
| **c** |  |  | ВЫБРОС |  |  |  |  |  |  |
| **d** |  |  |  | ВЫБРОС |  |  |  |  |  |
| **f** |  |  |  |  | ВЫБРОС |  |  |  |  |
| **g** |  |  |  |  |  | ВЫБРОС |  |  |  |
| **h** |  |  |  |  |  |  | ВЫБРОС |  |  |
| **i** |  |  |  |  |  |  |  | ВЫБРОС |  |
| **Ʇ** |  |  |  |  |  |  |  |  | ДОПУСК |
|  |  |  |  |  |  |  |  |  |  |

Пустые клетки в таблице означают ОШИБКУ.

Аналитичекое представление для таблицы М:

|  |  |  |
| --- | --- | --- |
| Правило грамматики | Множество | Значение М |
| p1: K → fA | FIRST(K) = {f} | M(K, f) = A, 1 |
| p2: A → cB | FIRST(A) = {c} | M(A, c) = B, 2 |
| p3: B → aC | FIRST(B) = {a} | M(B, a) = C, 3 |
| p4: C → dD | FIRST(C) = {d} | M(C, d) = D, 4 |
| p5: D → bE | FIRST(D) = {b} | M(D, b) = E, 5 |
| p6: E → hF | FIRST(E) = {h} | M(E, h) = F, 6 |
| p7: F → gG | FIRST(F) = {g} | M(F, g) = G, 7 |
| p8: G → cH | FIRST(G) = {c} | M(G, c) = H, 8 |
| p9: H → i | FIRST(H) = {i} | M(H, i) = ε, 9 |

*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 ((fcadbhgci), S­0Ʇ, 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 |
| (fcadbhgci, A⊥, 1) ⸠ | M(K, f) = A, 1 |
| (cadbhgci, B⊥, 12) ⸠ | M(A, c) = B, 2 |
| (adbhgci, C⊥, 123) ⸠ | M(B, a) = C, 3 |
| (dbhgci, D⊥, 1234) ⸠ | M(C, d) = D, 4 |
| (bhgci, E⊥, 12345) ⸠ | M(D, b) = E, 5 |
| (hgci, F⊥, 123456) ⸠ | M(E, h) = F, 6 |
| (gci, G⊥, 1234567) ⸠ | M(F, g) = G, 7 |
| (ci, H⊥, 12345678) ⸠ | M(G, c) = H, 8 |
| (i, ⊥, 123456789) ⸠ | M(H, i) = ε, 9 |
| (ε, ⊥, 12345678910) | M(ε, ⊥) = ДОПУСК |

*Лабораторная работа №11:*

**Формулировка задания:**

Реализовать управляющую таблицу M Для LL(k) анализатора.

**Код программы:**

var lLGr = new Grammar(

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

    new List<Symbol>() { "K", "A", "B", "C", "D", "E", "F", "G", "H" },

    "K"

);

lLGr.AddRule("K", new List<Symbol>() { "f", "A" });

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

lLGr.AddRule("B", new List<Symbol>() { "a", "C" });

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

lLGr.AddRule("D", new List<Symbol>() { "b", "E" });

lLGr.AddRule("E", new List<Symbol>() { "h", "F" });

lLGr.AddRule("F", new List<Symbol>() { "g", "G" });

lLGr.AddRule("G", new List<Symbol>() { "c", "H" });

lLGr.AddRule("H", new List<Symbol>() { "i" });

var parser = new LLParser(lLGr);

Console.WriteLine("Пример вводимой строки: fcadbhgci\nВведите строку: ");

string stringChain = Console.ReadLine();

var chain = new List<Symbol> {};

foreach (var x in stringChain) {

chain.Add(new Symbol(x.ToString()));

}

if (parser.Parse(chain)) {

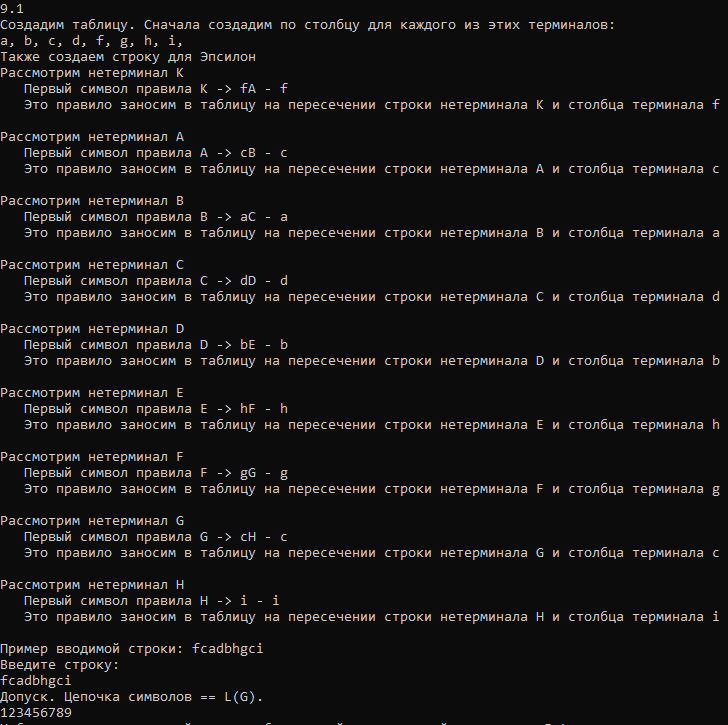
Console.WriteLine("Допуск. Цепочка символов == L(G).");

    Console.WriteLine(parser.OutputConfigure);

} else {

    Console.WriteLine("Не допуск. Цепочка символов != L(G).");

}

**Результат работы программы:**

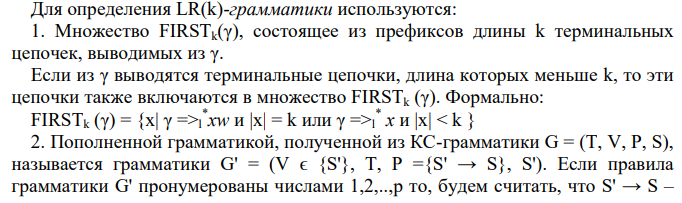
**Практическая работа №4 (лабораторные 12-16)**

**Формулировка задания:**

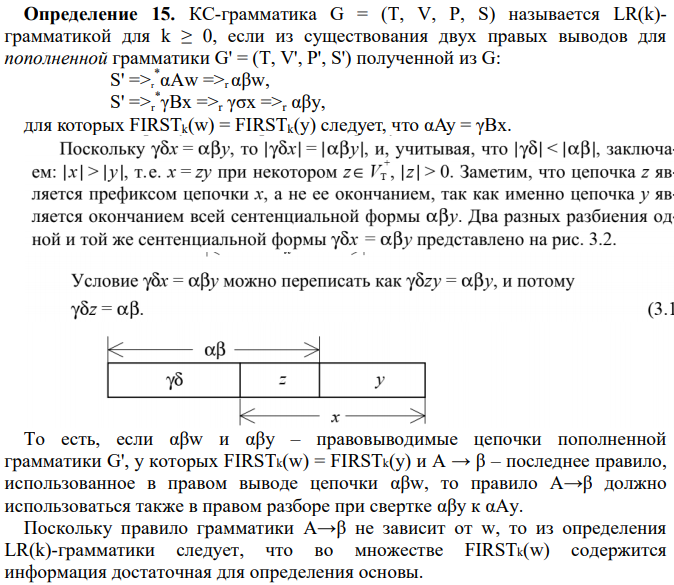
Построить управляющую таблицу M для LR(k)-грамматики, написать правило вывода выделенной строки. Описать работу алгоритма LR(k) анализатора. Построить LR(k) анализатор на основе грамматического вхождения.

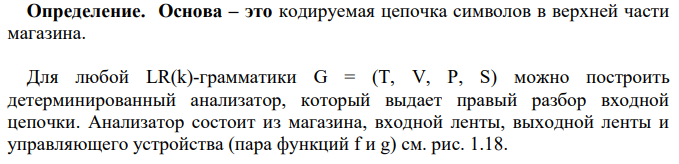
**Синтаксический LR-анализатор** анализирует входную цепочку слева направо (L), и строит правый (R) вывод грамматики.

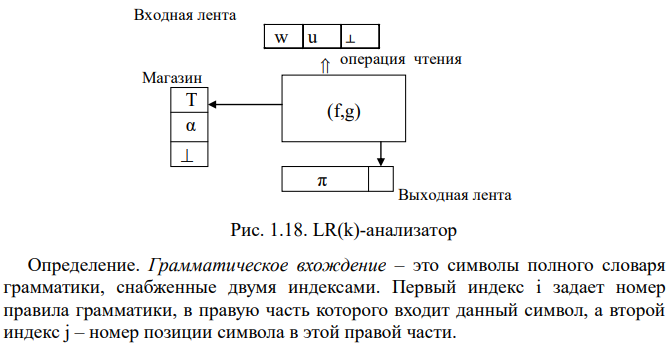
Грамматики, для которых можно построить детерминированный восходящий анализатор, называются LR(k)-*грамматиками* (входная цепочка читается слева (Left) направо, выходом анализатора является правый (Right) разбор, k – число символов входной цепочки, на которое можно “заглянуть” вперёд для выделения основы).











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).

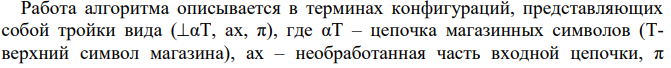
Из A можно вывести цепочки:

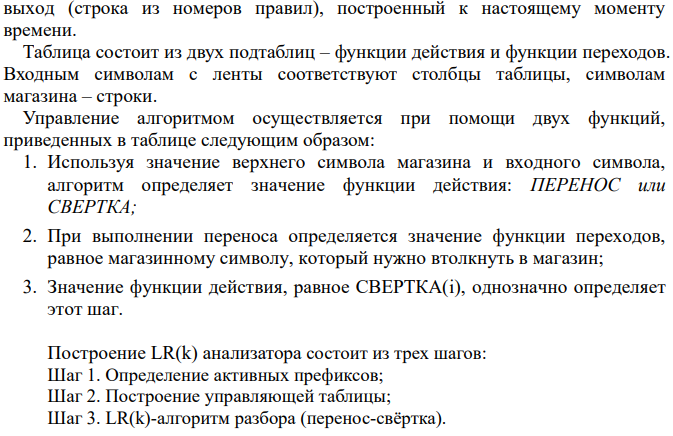
A =>2 c2B2 =>3 c2a3C3 =>4 c2a3d4D4 =>5 c2a3d4b5E5 =>6 c2a3d4b5h6F6 =>7 c2a3d4b5h6g7G7 =>8 c2a3d4b5h6g7c8H8 =>9 c2a3d4b5h6g7c8i9

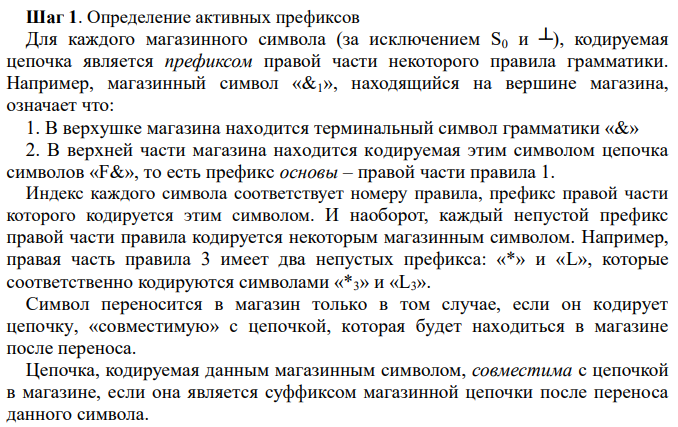
Следовательно, OFIRST(A­1) = {c2} и f1 OBLOW c2

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

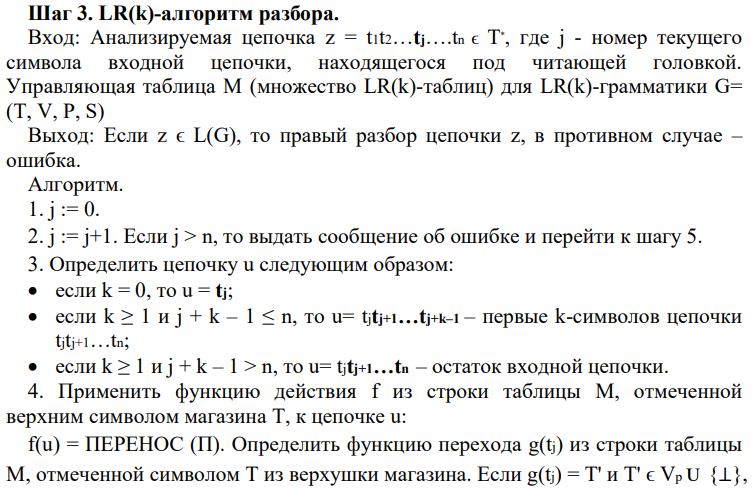
Таблица закодированных символов

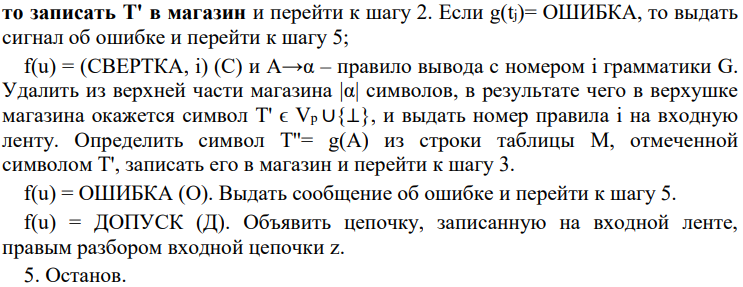
|  |  |  |  |
| --- | --- | --- | --- |
| Символ грамматики | Магазинный символ | Кодируемая цепочка | Операции |
| S0 = K | ­K0  A1 | ⊥K  fA | Д  П |
| A | A­1  B­2 | A  cB | Д  П |
| B | B2  C3 | B  aC | Д  П |
| C | C3  D4 | C  dD | Д  П |
| D | D4  E5 | D  bE | Д  П |
| E | E5  F6 | E  hF | Д  П |
| 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) |  |  |  |  |  |  |  |  |  |  |