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:* Старцев И. Р.*Группа:* М8О-201Б-21

*Руководитель:* Киндинова В. В.

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

**Автоматная грамматика:**

L(pattern) = L("192\.168\.1\.\d{1,3}") = { 192.168.1.{0...9}, 192.168.1.({0...9})2, 192.168.1.({0...9})3 }

G(T, V, P, S0) = G({0, 1, ..., 9, .}, {**S0**, **A, B,** ..., **M**}, {p1, p2, ..., p14}, **S0**)

*Правила регулярной грамматики:*

**p1:** **S0** → 1**A**

**p2:** **A** → 9**B**

**p3:** **B** → 2**C**

**p4:** **C** → .**D**

**p5:** **D** → 1**E**

**p6:** **E** → 6**F**

**p7:** **F** → 8**G**

**p8:** **G** → .**H**

**p9:** **H** → 1**I**

**p10:** **I** → .**J**

**p11:** **J** → 0**K** | 1**K** | ... | 9**K**

**p12:** **K** → **L** | 0**L** | 1**L** | ... | 9**L**

**p13:** **L** → **M** | 0**M** | 1**M** | ... | 9**M**

**p**14: **M** → ε

*Пример цепочек:*

**S0** =>1 1**A** =>2 19**B** =>3 192**C** =>4 192.**D** =>5 192.1**E** =>6 192.16**F** =>7 192.168**G** =>8 192.168.**H** =>9 192.168.1**I** =>10 192.168.1.**J** =>11 192.168.1.1**K** =>12 192.168.1.1**L** =>13 192.168.1.1**M** =>14 192.168.1.1

**S0** =>1 1**A** =>2 19**B** =>3 192**C** =>4 192.**D** =>5 192.1**E** =>6 192.16**F** =>7 192.168**G** =>8 192.168.**H** =>9 192.168.1**I** =>10 192.168.1.**J** =>11 192.168.1.5**K** =>12 192.168.1.51**L** =>13 192.168.1.512**M** =>14 192.168.512

**S0** =>1 1**A** =>2 19**B** =>3 192**C** =>4 192.**D** =>5 192.1**E** =>6 192.16**F** =>7 192.168**G** =>8 192.168.**H** =>9 192.168.1**I** =>10 192.168.1.**J** =>11 192.168.1.0**K** =>12 192.168.1.04**L** =>13 192.168.1.04**M** =>14 192.168.04

**Конечный автомат:**

L(КА) = L(G)

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

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

**δ** = { 1. δ(**S0**, 1) = {**A**},

2. δ(**A**, 9) = {**B**},

3. δ(**B**, 2) = {**C**},

4. δ(**C**, .) = {**D**},

5. δ(**D**, 1) = {**E**},

6. δ(**E**, 6) = {**F**},

7. δ(**F**, 8) = {**G**},

8. δ(**G**, .) = {**H**},

9. δ(**H**, 1) = {**I**},

10. δ(**I**, .) = {**J**},

11. δ(**J**, 0) = {**K**},

12. δ(**J**, 1) = {**K**},

...

20. δ(**J**, 9) = {**K**},

21. δ(**K**, ε) = {**L**},

22. δ(**K**, 0) = {**L**},

23. δ(**K**, 1) = {**L**},

...

31. δ(**K**, 9) = {**L**},

32. δ(**L**, ε) = {**M**},

33. δ(**L**, 0) = {**M**},

34. δ(**L**, 1) = {**M**},

...

42. δ(**L**, 9) = {**M**},

43. δ(**M**, ε) = {**qf**},

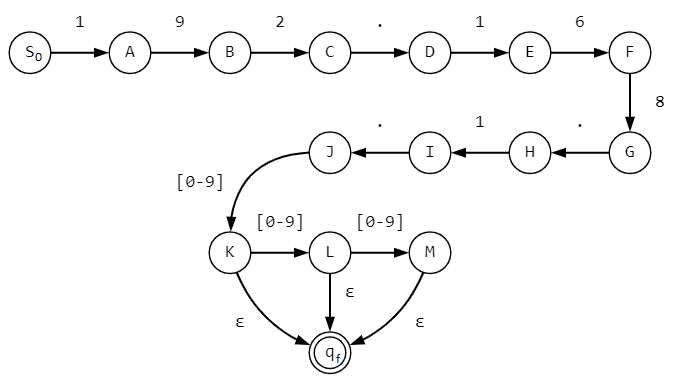
}

*Примеры конфигурации КА:*

**1.** (**S0**, 192.168.1.512) ⸠1 (**A**, 92.168.1.512) ⸠2 (**B**, 2.168.1.512) ⸠3 (**C**, .168.1.512) ⸠4 (**D**, 168.1.512) ⸠5 (**E**, 68.1.512) ⸠6 (**F**, 8.1.512) ⸠7 (**G**, .1.512) ⸠8 (**H**, 1.512) ⸠9 (**I**, .512) ⸠10 (**J**, 512) ⸠15 (**K**, 12) ⸠23 (**L**, 2) ⸠35 (**M**, ε) ⸠43 (**qf**, ε)

**2.** (**S0**, 192.168.1.04) ⸠1 (**A**, 92.168.1.04) ⸠2 (**B**, 2.168.1.04) ⸠3 (**C**, .168.1.04) ⸠4 (**D**, 168.1.04) ⸠5 (**E**, 68.1.04) ⸠6 (**F**, 8.1.04) ⸠7 (**G**, .1.04) ⸠8 (**H**, 1.04) ⸠9 (**I**, .04) ⸠10 (**J**, 04) ⸠11 (**K**, 4) ⸠26 (**L**, ε) ⸠32 (**M**, ε) ⸠43 (**qf**, ε)

**3.** (**S0**, 192.168.1.1) ⸠1 (**A**, 92.168.1.1) ⸠2 (**B**, 2.168.1.1) ⸠3 (**C**, .168.1.1) ⸠4 (**D**, 168.1.1) ⸠5 (**E**, 68.1.1) ⸠6 (**F**, 8.1.1) ⸠7 (**G**, .1.1) ⸠8 (**H**, 1.1) ⸠9 (**I**, .1) ⸠11 (**J**, 1) ⸠11 (**K**, ε) ⸠21 (**L**, ε) ⸠32 (K, ε) ⸠43 (**qf**, ε)



*Лемма о накачке:*

Изображение выглядит как текст

Автоматически созданное описание

**2. (\W|^)stock\stips(\W|$)**

**Автоматная грамматика:**

L(pattern) = L("(\W|^)stock\s{0,3}tips(\W|$)") = { stock\stips }

G(T, V, P, S0) = G({0, 1, ..., 9, .}, {S0, A, B, ..., L},

{p1, p2, ..., p12}, S0)

*Правила регулярной грамматики:*

**p1: S0 → sA' | #Q**

**p2: Q → sA'**

**p3: A' → tB'**

**p4: B' → oC'**

**p5: C' → cD'**

**p6: D' → kG'**

**p7: G' → \sF**

**p8: F → tH'**

**p9: H' → iI'**

**p10: I' → pM'**

**p11: M' → s | sW**

**p12: W → #**

*Пример цепочек:*

**S0 → sA' → stB' → stoC' → stocD' → stockG' → stock F → stock tH' →**

**stock tiI'→ stock tipM' → stock tips**

**Конечный автомат:**

**L(KA) = L(G)**

**KA = (Q, Σ, δ, S0, F), где**

**Q = {s, t, o, c, k, i, p, \s, q, #}**

**Σ = {S0, A', B', C', D', E', F', G', H', M', Q}**

**S0 = S0**

**δ = {**

**1. δ(S0, s) = {A'},**

**2. δ(A', t) = {B'},**

**...**

**5. δ(D', k) = {E'},**

**6. δ(E', \s) = {F'},**

**7. δ(F', t) = {G'},**

**8. δ(G', i) = {H'},**

**9. δ(H', p) = {M'},**

**10. δ(M', s) = {q},**

**11. δ(Q, #) = {A'}**

**12. δ(W, #) = {q}**

**}**

*Примеры конфигурации КА:*stock\stips → A'tock\stips → ... → G'\stips → F\stips → H'ips → ... → M's → qfstock\stips → A'tock\stips → ... → G'\stips → F3\stips → F2tips → H'ips → ... → M's → qfstock\stips → A'tock\stips → ... → G'\stips → F3tips → H'ips → ... → M's → qfstocktips → A'tocktips → ...

→ G'tips → H'ips → ... → M's → qfИзображение выглядит как текст, грифельная доска

Автоматически созданное описание

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

*Лабораторная работа №4:*

**Формулировка задания:**

Устранить из КС-грамматики бесполезные символы и ε–правила

Исходная грамматика (вариант 22):

**G** = {

( a, b, c, d, f, epsilon, r, z, o ),

( S, A, B, C, D, M, W, T, J, K ),

( S → aAB, A → bC, B → cA, D → M, M → f, C → dDW, W → epsilon, T → r, J → z, B → aK, K → Ko, K → o),

S

}

p1: S → aAB

p2: A → bC

p3: B → cA

p4: D → M

p5: M → f

p6: C → dDW

p7: W → epsilon

p8: T → r

p9: J → z

p10: B → aK

p11: K → Ko

p12: K → o

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

**Устранение непроизводящих символов.**

Символ недостижимый, если он не участвует ни в одной цепочке вывода из начального символа

грамматики.

Нетерминальный символ A ∈ V называется производящим, если из него можно вывести терминальную цепочку (последовательность терминальных символов, которые могут быть выводимыми в грамматике).

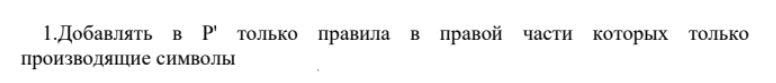
1. V1p = {M, W, T, J, K} – для этих нетерминалов нашлось хотя бы одно правило, правая часть которого не содержит нетерминалов

2. V2p = {M, W, T, J, K, D, B} – 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. V3p = {M, W, T, J, K, D, B, C}

4. V4p = {M, W, T, J, K, D, B, C, A}

5. V5p = {M, W, T, J, K, D, B, C, A, S}

6. Vp = {M, W, T, J, K, D, B, C, A, S} содержит все производящие нетерминалы грамматики, а все нетерминалы, не попавшие в него, являются непроизводящими (V - Vp = ∅ ).

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

p1: S → aAB

p2: A → bC

p3: B → cA

p4: D → M

p5: M → f

p6: C → dDW

p7: W → epsilon

p8: T → r

p9: J → z

p10: B → aK

p11: K → Ko

p12: K → o

G1 = ({a, b, c, d, f, epsilon, r, z, o}, {S, A, B, C, D, M, W, T, J, K}, P', S)

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

****

VT1r = {S}

VT2r = {S, a, A, B}

VT3r = {S, a, A, B, b, C, c, K}

VT4r = {S, a, A, B, b, C, c, K, o, d, D, W}

VT5r = {S, a, A, B, b, C, c, K, o, d, D, W, M, epsilon }

VT6r = {S, a, A, B, b, C, c, K, o, d, D, W, M, epsilon, f}

Так, не входящие в VTr символы {T, J, r, z} недостижимы

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

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

p1: S → aAB

p2: A → bC

p3: B → cA

p4: D → M

p5: M → f

p6: C → dDW

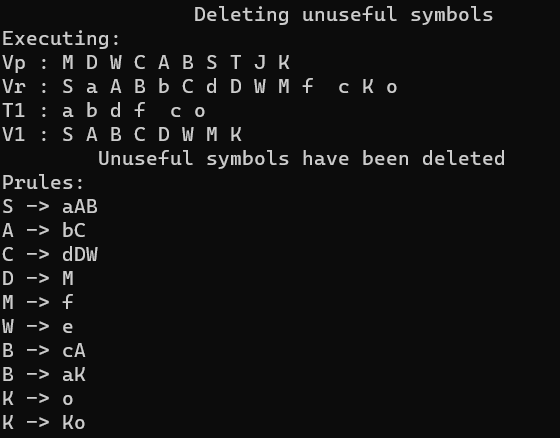
p7: W → epsilon

p8: B → aK

p9: K → Ko

p10: K → o

Получаем G' = ({a, b, c, d, f, epsilon, o}, {S, A, B, C, D, M, W, K}, P'', S)



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

Составим G" на основе алгоритма удаления эпсилон правил

G' = ({a, b, c, d, f, epsilon, o}, {S, A, B, C, D, M, W, K}, P, S), где P = {S → aAB, A → bC, B → cA, D → M, M → f, C → dDW, W → epsilon, B → aK, K → Ko, K → o}

G" = ({a, b, c, d, f, epsilon, o}, {S, A, B, C, D, M, W, K}, P, S), где P' = {S → aAB, A → bC, B → cA, D → M, M → f, C → dDW, C → dD, B → aK, K → Ko, K → o}

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

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

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

Цепные правила:

D → M, M → f

V1x = {M}

Правила D → M, M → f поменяем на D → f

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

**Устранить из KС-грамматики левую рекурсию.**

Левая рекурсия: K → Ko

Заменим правило K → Ko, на правила: K → o | oK’, K’ → o | oK’

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

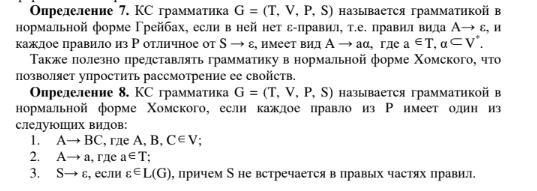
**Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание Приведенная грамматика:**

*Лабораторная работа №6*

**Формулировка задания:**

Определить форму КС-грамматики и сделать ее приведение



Приведенная грамматика не является грамматикой в нормальной форме Хомского, так как присутствует правило: **S** → a**AB.**

Приведенная грамматика является грамматикой в нормальной форме Грейбах, так как в ней нет – epsilon-правил, и каждое правило имеет вид **A** → aα, где aT, α .

*Лабораторная работа №7*

**Формулировка задания:** спроектировать МП-автомат для приведённой КС-грамматики.

Так, МП = (Q, Σ, Г, δ, q0, z0, F) – это семерка объектов, где

Изображение выглядит как текст, Шрифт, снимок экрана, алгебра

Автоматически созданное описание

Приведённая грамматика:

G = (**T**, **V**, **P**, **S**), где

{

**T** = { a, b, c, d, f, o },

**V** = { S, A, B, C, D, K, K`},

**S0** = S

**P**:

p1: S → aAB

p2: A → bC

p3: C → dD

p4: D → f

p5: B → cA

p6: B → aK

p7: K → o

p8: K → oK`

p9: K` → o

p10: K` → oK`

}

Вывод цепочки:

S0 =>1 aAB =>2 abCB =>3 abdDB =>4 abdfB =>5 abdfcA =>2 abdfcbC =>3 abdfcbdD =>4 abdfcbdf

S0 =>1 aAB =>2 abCB =>3 abdDB =>4 abdfB =>6 abdfaK =>8 abdfaoK` =>10 abdfaooK` =>9 abdfaooo

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, o },

{ a, b, c, d, f, o, S, A, B, C, D, K, K` },

**δ**,

q0,

S,

{q}

)

Строим функции переходов:

p1: S → aAB 1: δ(q0, ε, S) = (q, aAB)

p2: A → bC 2: δ(q, ε, A) = (q, bC)

p3: C → dD 3: δ(q, ε, C) = (q, dD)

p4: D → f 4: δ(q, ε, D) = (q, f)

p5: B → cA 5: δ(q, ε, B) = (q, cA)

p6: B → aK 6: δ(q, ε, B) = (q, aK)

p7: K → o 7: δ(q, ε, K) = (q, o)

p8: K → oK` 8: δ(q, ε, K) = (q, oK`)

p9: K` → o 9: δ(q, ε, K`) = (q, o)

p10: K` → oK` 10: δ(q, ε, K`) = (q, oK`)

11: δ(q, a, a) = (q, ε), a ∈ Σ

Последовательность тактов МП-автомата abdfcbdf:

(q0, abdfcbdf, S) ⸠1 (q, abdfcbdf, aAB) ⸠11 (q, bdfcbdf, AB) ⸠2 (q, bdfcbdf, bCB) ⸠11 (q, dfcbdf, CB) ⸠3 (q, dfcbdf, dDB) ⸠11 (q, fcbdf, DB) ⸠4 (q, fcbdf, fB) ⸠11 (q, cbdf, B) ⸠5 (q, cbdf, cA) ⸠11 (q, bdf, A) ⸠2 (q, bdf, bC) ⸠11 (q, df, C) ⸠3 (q, df, dD) ⸠11 (q, f, D) ⸠4 (q, f, f) ⸠11 (q, ε, ε)

*Лабораторная работа №8*

**Формулировка задания:**

Реализовать спроектированный МП-автомат для приведённой КС-грамматики.

**Код программы:**

case "7": {

var CFGrammar = new Grammar(new List<Symbol>() { "a", "b", "c", "d", "f", "o" },

new List<Symbol>() { "S","A","B","C", "D", "K", "K`" },

"S");

CFGrammar.AddRule("S",new List<Symbol>() { "a", "A", "B" });

CFGrammar.AddRule("A",new List<Symbol>() { "b","C" });

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

CFGrammar.AddRule("D",new List<Symbol>() { "f" });

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

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

CFGrammar.AddRule("K",new List<Symbol>() { "o" });

CFGrammar.AddRule("K",new List<Symbol>() { "o", "K`" });

CFGrammar.AddRule("K`",new List<Symbol>() { "o" });

CFGrammar.AddRule("K`",new List<Symbol>() { "o", "K`" });

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;

}

**Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание**

Изображение выглядит как текст, снимок экрана, меню, Шрифт

Автоматически созданное описание

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

*Лабораторная работа №9:*

**Формулировка задания:**

Для LL(k) анализатора построить управляющую таблицу M.

LL-грамматика (или грамматика с левой рекурсией) - это тип формальной грамматики, используемый в теории формальных языков и синтаксического анализа. Она названа так, потому что в процессе синтаксического анализа символы обрабатываются слева направо, и она обладает свойством левой рекурсии.

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.

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Исходная грамматика:

G = (**T**, **V**, **P**, **S**),

{

**T** = { a, b, c, d, f, o },

**V** = { S, A, B, C, D, K, K`},

**S0** = S

**P** {

p1: S → aAB

p2: A → bC

p3: C → dD

p4: D → f

p5: B → cA

p6: B → aK

p7: K → o

p8: K → oK`

p9: K` → o

p10: K` → oK`

}

}

Можно привести грамматику к форме LL(1) путем факторизации. Левая факторизация предполагает переписывание правил производства, когда необходимо выбрать из двух альтернативных продукций для нетерминала A. Вместо того чтобы сделать выбор на основе уже прочитанных символов входного потока, мы можем отложить принятие решения до тех пор, пока не будет прочитано достаточно символов для правильного выбора.

Gф = (**T**, **V**, **P’**, **S**),

{

**T** = { a, b, c, d, f, o },

**V** = { S, A, B, C, D, K, K`, T, T` },

**S0** = S

**P’**{

p1: S → aAB

p2: A → bC

p3: C → dD

p4: D → f

p5: B → cA

p6: B → aK

p7: K → oT

p8: T → K`

p9: T → epsilon

p10: K` → oT`

p11: T` → K`

p12: T` → epsilon

}

}

**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** | **o** | **epsilon** |
| **S** | (aAB, 1) |  |  |  |  |  |  |
| **A** |  | (bC, 2) |  |  |  |  |  |
| **B** | (aK,6) |  | (cA, 5) |  |  |  |  |
| **C** |  |  |  | (dD, 3) |  |  |  |
| **D** |  |  |  |  | (f,4) |  |  |
| **K** |  |  |  |  |  | (oT, 7) |  |
| **K’** |  |  |  |  |  | (oT`, 10) |  |
| **T** |  |  |  |  |  | (oT`, 10) | (epsilon, 9) |
| **T`** |  |  |  |  |  | (oT`, 10) | (epsilon, 11) |
| **a** | ВЫБРОС |  |  |  |  |  |  |
| **b** |  | ВЫБРОС |  |  |  |  |  |
| **c** |  |  | ВЫБРОС |  |  |  |  |
| **d** |  |  |  | ВЫБРОС |  |  |  |
| **f** |  |  |  |  | ВЫБРОС |  |  |
| **o** |  |  |  |  |  | ВЫБРОС |  |
| **Ʇ** |  |  |  |  |  |  | ДОПУСК |

Пустые клетки в таблице означают ОШИБКУ.

Аналитичекое представление для таблицы М:

|  |  |  |
| --- | --- | --- |
| Правило грамматики | Множество | Значение М |
| p1: S → aAB | FIRST(S) = {a} | M(S, a) = aAB, 1 |
| p2: A → bC | FIRST(A) = {b} | M(A, b) = bC, 2 |
| p3: C → dD | FIRST(C) = {d} | M(C, d) = dD, 3 |
| p4: D → f | FIRST(D) = {f} | M(D, f) = f, 4 |
| p5: B → cA | FIRST(B) = {c} | M(B, c) = cA, 5 |
| p6: B → aK | FIRST(B) = {a} | M(B, a) = aK, 6 |
| p7: K → oT | FIRST(K) = {o} | M(K, o) = oT, 7 |
| p8: T → K` | FIRST(T) = {o} | M(T, o) = K`, 8 |
| p9: T → epsilon | FOLLOW(T) = {epsilon} | M(T, epsilon) = epsilon, 9 |
| p10: K` → oT` | FIRST(K`) = {o} | M(K`, o) = oT`, 10 |
| p11: T` → K` | FIRST(T`) = {o} | M(T`, o) = K`, 11 |
| p12: T` → epsilon | FOLLOW(T`) = {epsilon} | M(T`, epsilon) = epsilon, 12 |

*Лабораторная работа №10:*

**Формулировка задания:**

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

Шаг 1. Алгоритм находится в начальной конфигурации ((abdfaoo), S­0Ʇ, epsilon), где S0 = S

Значение управляющей таблицы M(S, a) = (aAB, 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 |
| (abdfaoo, S⊥, epsilon) ⸠ | M(S, a) = aAB, 1 |
| (abdfaoo, aAB⊥, 1) ⸠ | M(a, a) = ВЫБРОС |
| (bdfaoo, AB⊥, 1) ⸠ | M(A, b) = bC, 2 |
| (bdfaoo, bCB⊥, 12) ⸠ | M(b, b) = ВЫБРОС |
| (dfaoo, CB⊥, 12) ⸠ | M(C, d) = dD, 3 |
| (dfaoo, dDB⊥, 123) ⸠ | M(d, d) = ВЫБРОС |
| (faoo, DB⊥, 123) ⸠ | M(D, f) = f, 4 |
| (faoo, fB⊥, 1234) ⸠ | M(f, f) = ВЫБРОС |
| (aoo, B⊥, 1234) ⸠ | M(B, a) = aK, 6 |
| (aoo, aK⊥, 12346) | M(a, a) = ВЫБРОС |
| (oo, K⊥, 12346) | M(K, o) = oT, 7 |
| (oo, oT⊥, 123467) | M(o, o) = ВЫБРОС |
| (o, T⊥, 123467) | M(T, o) = K`, 8 |
| (o, K`⊥, 1234678) | M(K`, o) = oT`, 10 |
| (o, oT`⊥, 123467810) | M(o, o) = ВЫБРОС |
| (epsilon, T`⊥, 123467810) | M(T`, epsilon) = epsilon, 12 |
| (epsilon, ⊥, 12346781012) | M(epsilon, ⊥) = ДОПУСК |

*Лабораторная работа №11:*

**Формулировка задания:**

Реализовать управляющую таблицу M Для LL(k) анализатора.

**Код программы:**

case "9.1": {

var LL = new Grammar(new List<Symbol>() { "a","b","c","d","f", "o", " " },

new List<Symbol>() { "S", "A", "B", "C", "D", "K", "K`", "T", "T`" },

"S");

LL.AddRule("S",new List<Symbol>() { "a","A","B" });

LL.AddRule("A", new List<Symbol>() { "b", "C" });

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

LL.AddRule("D", new List<Symbol>() { "f" });

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

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

LL.AddRule("K", new List<Symbol>() { "o", "T" });

LL.AddRule("T", new List<Symbol>() { "K`" });

LL.AddRule("T", new List<Symbol>() { " " });

LL.AddRule("K`", new List<Symbol>() { "o", "T`" });

LL.AddRule("T`", new List<Symbol>() { "K`" });

LL.AddRule("T`", new List<Symbol>() { " " });

var parser = new LLParser(LL);

Console.WriteLine("Пример вводимых строк: (i+i), (i+\*i)");

Console.WriteLine("Введите 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);

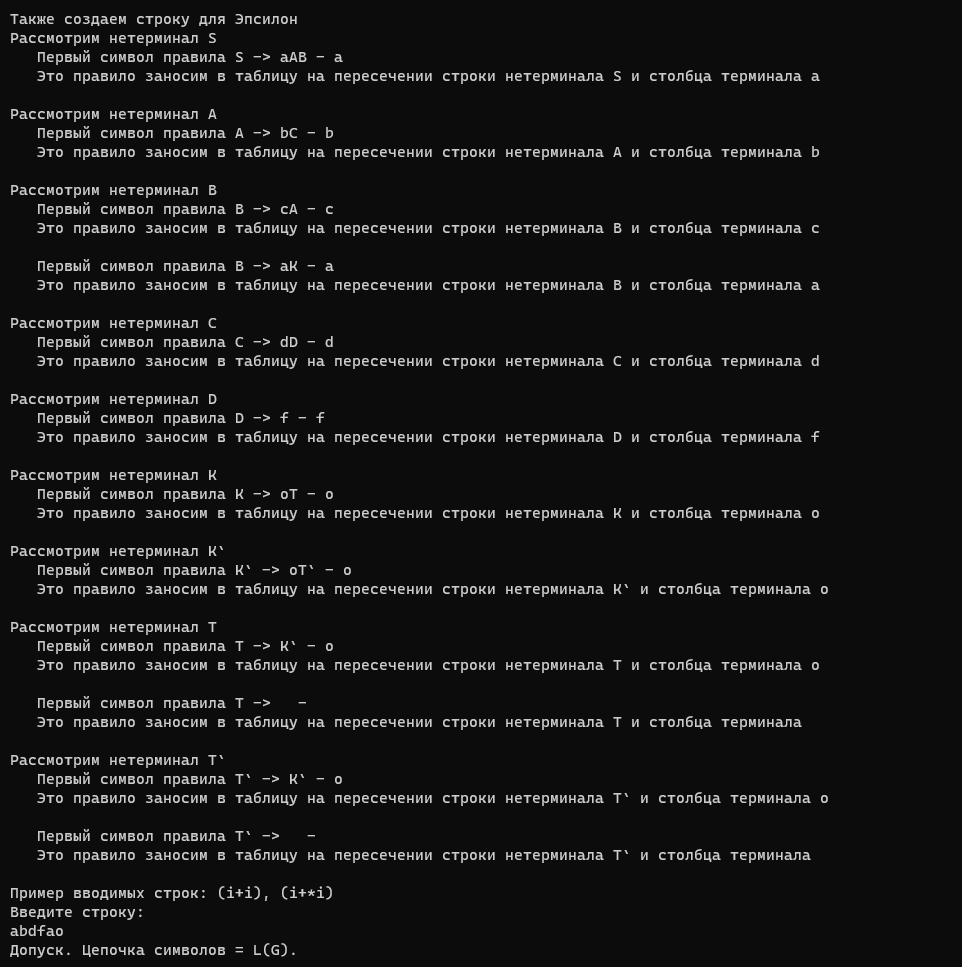
} else {

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

}

break;

}

**Результат работы программы:**

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

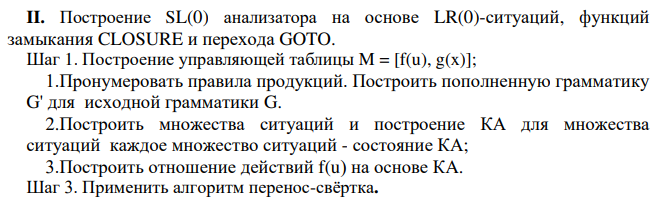
**Формулировка задания:**

Построить множество LR(0)-таблиц не содержащих ε-правила. Определить функции перехода g(X). Определить функцию переноса-свертки f(u). Для функции перехода g(X) и функции переноса-свертки f(u) спроектировать управляющую таблицу.

Существует **два способа построения LR(k) анализаторов**:

1. На основе активных префиксов (построения расширенного магазинного алфавита) и отношения OBLOW;
2. Построение SL(0) анализатора на основе LR(0)-ситуаций, функций замыкания CLOSURE и перехода GOTO;

Построим вторым способом LR(k) анализатор для заданной грамматики:



Изображение выглядит как текст, снимок экрана, Шрифт, алгебра

Автоматически созданное описание

G = (**T**, **V**, **P**, **S**), где

**T** = { a, b, c, d, f, o }, **V** = { S, A, B, C, D, K, K`}, **S0** = S

**P** {

p1: S → aAB

p2: A → bC

p3: C → dD

p4: D → f

p5: B → cA

p6: B → aK

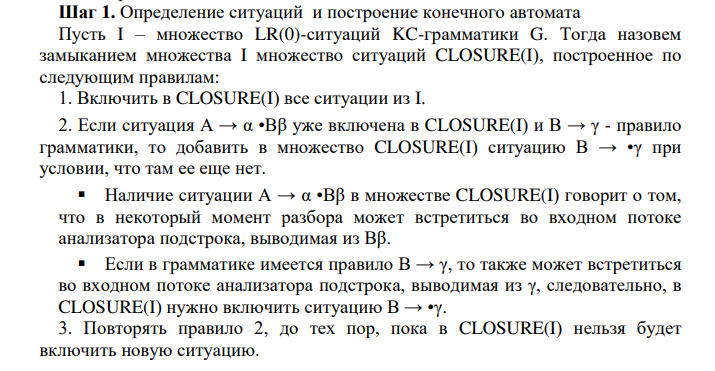
p7: K → o

p8: K → oK`

p9: K` → o

p10: K` → oK`

}



Пополненная грамматика G содержит еще одно правило: S' → S

G = (**T**, **V**, **P**, **S**), где

**T** = { a, b, c, d, f, o }, **V** = { S, A, B, C, D, K, K`}, **S0** = S

**P** {

p0: S’ → S

p1: S → aAB

p2: A → bC

p3: C → dD

p4: D → f

p5: B → cA

p6: B → aK

p7: K → o

p8: K → oK`

p9: K` → o

p10: K` → oK`

}

CLOSURE(S'→ •S) ={S'→ •S, S → •aAB}

I0= {S'→ •S,S → •aAB}

Изображение выглядит как текст, Шрифт, снимок экрана, алгебра

Автоматически созданное описание

**Функция переходов GOTO**

I0 = CLOSURE(S'→ •S) ={S'→ •S, S → •aAB}

GOTO(I0, a) = {S → a•AB, A → •bC} = I1

GOTO(I0, S) = {S'→ S•} = I2

GOTO(I1, b) = {A → b•C, C → •dD} = I3

GOTO(I1, A) = {S → aA•B, B → •cA, B → •aK} = I4

GOTO(I3, d) = {C → d•D, D → •f} = I5

GOTO(I3, C) = {A → bC•} = I6

GOTO(I4, a) = {B → a•K, K → •o, K → •oK`} = I7

GOTO(I4, c) = {B → c•A, A → •bC} = I8

GOTO(I4, B) = {S → aAB•} = I9

GOTO(I5, f) = {D → f•} = I10

GOTO(I5, D) = {C → dD•} = I11

GOTO(I7, o) = {K → o•, K → o•K`, K` → •o, K` → •oK`} = I12

GOTO(I7, K) = {B → aK•} = I13

GOTO(I8, A) = {B → cA•} = I14

GOTO(I12, o) = {K` → o•, K` → o•K`, K` → •o, K` → •oK`} = I15

GOTO(I12, K`) = {K → oK`•} = I16

GOTO(I15, K`) = {K` → oK`•} = I17

Изображение выглядит как текст, снимок экрана, Шрифт, алгебра

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

𝜑 = {

I0 = {S'→ •S, S → •aAB},

I1 = {S → a•AB, A → •bC},

I2 = {S'→ S•},

I3 = {A → b•C, C → •dD},

I4 = {S → aA•B, B → •cA, B → •aK}

I5 = {C → d•D, D → •f},

I6 = {A → bC•},

I7 = {B → a•K, K → •o, K → •oK`},

I8 = {B → c•A, A → •bC},

I9 = {S → aAB•},

I10 = {D → f•},

I11 = {C → dD•},

I12 = {K → o•, K → o•K`, K` → •o, K` → •oK`},

I13 = {B → aK•},

I14 = {B → cA•},

I15 = {K` → o•, K` → o•K`, K` → •o, K` → •oK`},

I16 = {K → oK`•},

I17 = {K` → oK`•}

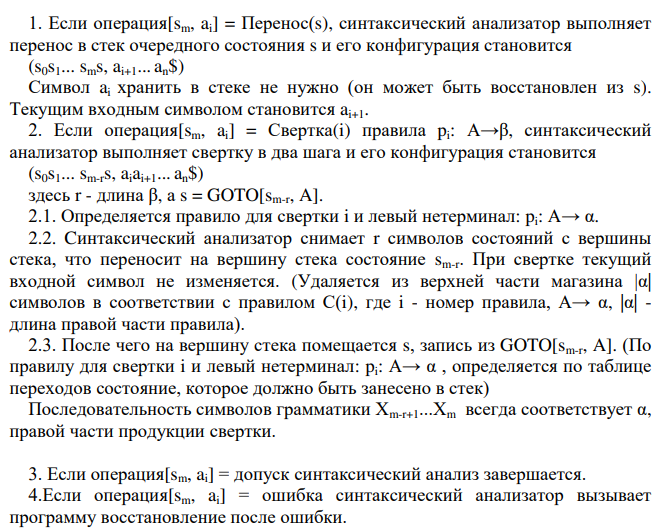
}

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание



**Управляющая таблица**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **I** | **f(u)** | | | | | | | **g(x)** | | | | | | |
| **a** | **b** | **с** | **d** | **f** | **o** | ┴ | **S** | **A** | **B** | **C** | **D** | **K** | **K’** |
| **0** | **П,1** |  |  |  |  |  |  | **2** |  |  |  |  |  |  |
| **1** |  | **П,3** |  |  |  |  |  |  | **4** |  |  |  |  |  |
| **2** |  |  |  |  |  |  | **Д** |  |  |  |  |  |  |  |
| **3** |  |  |  | **П,5** |  |  |  |  |  |  | **6** |  |  |  |
| **4** | **П,7** |  | **П,8** |  |  |  |  |  |  | **9** |  |  |  |  |
| **5** |  |  |  |  | **П,10** |  |  |  |  |  |  | **11** |  |  |
| **6** | **С,2** | **С,2** | **С,2** | **С,2** | **С,2** | **С,2** | **С,2** |  |  |  |  |  |  |  |
| **7** |  |  |  |  |  | **П,12** |  |  |  |  |  |  | **13** |  |
| **8** |  | **П,3** |  |  |  |  |  |  | **14** |  |  |  |  |  |
| **9** | **С,1** | **С,1** | **С,1** | **С,1** | **С,1** | **С,1** | **С,1** |  |  |  |  |  |  |  |
| **10** | **С,4** | **С,4** | **С,4** | **С,4** | **С,4** | **С,4** | **С,4** |  |  |  |  |  |  |  |
| **11** | **С,3** | **С,3** | **С,3** | **С,3** | **С,3** | **С,3** | **С,3** |  |  |  |  |  |  |  |
| **12** | **С,7** | **С,7** | **С,7** | **С,7** | **С,7** | **П,15** | **С,7** |  |  |  |  |  |  | **16** |
| **13** | **С,6** | **С,6** | **С,6** | **С,6** | **С,6** | **С,6** | **С,6** |  |  |  |  |  |  |  |
| **14** | **С,5** | **С,5** | **С,5** | **С,5** | **С,5** | **С,5** | **С,5** |  |  |  |  |  |  |  |
| **15** | **С,9** | **С,9** | **С,9** | **С,9** | **С,9** | **П,15** | **С,9** |  |  |  |  |  |  | **17** |
| **16** | **С,8** | **С,8** | **С,8** | **С,8** | **С,8** | **С,8** | **С,8** |  |  |  |  |  |  |  |
| **17** | **С,10** | **С,10** | **С,10** | **С,10** | **С,10** | **С,10** | **С,10** |  |  |  |  |  |  |  |

**Шаг 3**. Применение алгоритма «перенос-свёртка» для разбора цепочки символов на ленте.

Изображение выглядит как текст, снимок экрана, Шрифт, информация

Автоматически созданное описание

Распознавание цепочки abdfaoo:

(0, abdfaoo┴, ε) ⊢П,1(0 1, bdfaoo┴, ε) ⊢П,3 (0 1 3, dfaoo┴, ε) ⊢П,5 (0 1 3 5, faoo┴, ε) ⊢П,10 (0 1 3 5 10, aoo┴, ε) ⊢С,4 (0 1 3 5 11, aoo┴, 4) ⊢С,3 (0 1 3 6, aoo┴, 4 3) ⊢С,2 (0 1 4, aoo┴, **4 3 2**) ⊢П,7 (0 1 4 7, oo┴, 4 3 2) ⊢П,12 (0 1 4 7 12, o┴, 4 3 2) ⊢П,15 (0 1 4 7 12 15, ┴, 4 3 2) ⊢C,9 (0 1 4 7 12 16, ┴, 4 3 2 9) ⊢C,8

(0 1 4 7 13, ┴, 4 3 2 9 8) ⊢C,6 (0 1 4 9, ┴, 4 3 2 9 8 6) ⊢C,1

(0 2, ┴, 4 3 2 9 8 6 1) ⊢Д

Получили правый разбор входной цепочки: 1 6 8 9 2 3 4

Для проверки построим цепочку по заданному правому выводу: S =>1 aAB =>6 aAaK =>8 aAaoK` =>9 aAaoo =>2 abCaoo =>3 abdDaoo =>4 abdfaoo

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, меню, черно-белый

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Изображение выглядит как снимок экрана, текст, черный

Автоматически созданное описание

Код программы:

case "14":

var LR0Grammar = new SLRGrammar(new List<Symbol>() { "a", "b", "c", "d", "f", "o" },

new List<Symbol>() { "S", "A", "B", "C", "D", "K", "K`" },

new List<Production>(),

"S");

LR0Grammar.AddRule("S",new List<Symbol>() { "a", "A", "B" });

LR0Grammar.AddRule("A",new List<Symbol>() { "b", "C" });

LR0Grammar.AddRule("C",new List<Symbol>() { "d", "D" });

LR0Grammar.AddRule("D",new List<Symbol>() { "f" });

LR0Grammar.AddRule("B",new List<Symbol>() { "c", "A" });

LR0Grammar.AddRule("B", new List<Symbol>() { "a", "K" });

LR0Grammar.AddRule("K", new List<Symbol>() { "o" });

LR0Grammar.AddRule("K", new List<Symbol>() { "o", "K`" });

LR0Grammar.AddRule("K`", new List<Symbol>() { "o" });

LR0Grammar.AddRule("K`", new List<Symbol>() { "o", "K`" });

LR0Grammar.Construct();

LR0Grammar.Inference();

break;