Московский авиационный институт

(национальный исследовательский университет)

Факультет «Прикладная математика и физика»

**Лабораторные работы по курсу**

**«Системное программное обеспечение»**

1. Спроектировать грамматику по заданному языку L

2. Спроектировать конечный автомат, составить диаграмму переходов КА и реализовать

3. Определить свойства КА. Построить НДКА. Реализовать преобразование 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(1) анализатора построить управляющую таблицу M

10. Аналитически написать такты работы LL(1) анализатора для выведенной цепочки.

11. Реализовать управляющую таблицу M для LL(1) анализатора.

12. Построение LR(k) анализатора на основе активных префиксов и отношения OBLOW. Построить управляющую таблицу для функции перехода g(х) и действий f(u) и такты работы алгоритма.

13. Реализовать LR(k)-анализатор по управляющей таблице (g,f) для LR(k) грамматики.

14. Построение LR(k) анализатора на основе LR(0)-ситуаций и функций CLOSURE и GOTO. Построить управляющую таблицу для функции перехода g(х) и действий f(u).

15. Построить конечный автомат для переходов между ситуациями и такты работы алгоритма.

16. Реализовать LR(k)-анализатор по управляющей таблице (g,f) для LR(k) грамматики.

*Студент:* Ильиных В.М.

*Группа:*  08-201

*Руководитель:* Семёнов А. С.

*Оценка:*

*Дата:*

**Москва 2021**

**1.Спроектировать грамматику по заданному языку L**:

Вариант 9

1.1. Задан бесконечный язык L = {(01)\*-1-(01)\*+w1 | w1 ∈ {0,1}+}.

L ~ { {01-1-01+0}, {01-1-01+1}, {01-1-01+11}, {01-1-0101+1},

{0101-1-01+011}, …}

(1) Для всех регулярных языков ∀ L ⊆ Σ\*(regular(L) =>

(2) существует целое (∃ p = 9 ≥ 1 такое что

(3) для (w = 01-1-01+1 ∈ L (( |01-1-01+1| ≥ 9) =>

(4) существует (∃ x = ε, y = 01, z = -1-01+1 ∈ Σ\* такое что (w = xyz =>

1. (|01| ≥ 1, цикл y должен быть накачан хотя бы длиной 1 и

2. (|01| ≤ 9, цикл должен быть в пределах первых 9 символов и

3. для i = 1 ≥ 0, ((01)i-1-01+1 ∈ L)))))))), на x и z ограничений не накладывается

1.2. Преобразовать бесконечный язык L в язык L1 представляющий собой конечное множество цепочек, цепочки символов которого являются подмножество цепочек символов бесконечного языка. Ввести ограничения для языка, если они необходимы.

L1 = L(G)

L1 ⊂ L

Правила вывода:

1. S0 → 0A

2. A → 1B

3. B → 0A

4. B → -C

5. C → 1D

6. D → -E

7. E → 0F

8. E → +G

9. F → 1E

10. G → 0G

11. G → 1G

12. G → 0

13. G → 1

Таким образом, грамматика имеет вид G = {{0,1,+,-}, {S0,A,B,C,D,E,F,G}, P={p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12, p13}, S0}

1.3. Сгенерировать цепочки символов по языку L1. Определить свойства языка L1.

1. S0 =>1 0A =>2 01B =>4 01-C =>5 01-1D =>6 01-1-E =>7 01-1-0F =>9

01-1-01E =>8 01-1-01+G =>12 01-1-01+0

1. S0 =>1 0A =>2 01B =>4 01-C =>5 01-1D =>6 01-1-E =>7 01-1-0F =>9

01-1-01E =>8 01-1-01+G =>13 01-1-01+1G

1. S0 =>1 0A =>2 01B =>4 01-C =>5 01-1D =>6 01-1-E =>7 01-1-0F =>9

01-1-01E =>8 01-1-01+G =>11 01-1-01+1G =>13 01-1-01+11

1. S0 =>1 0A =>2 01B =>4 01-C =>5 01-1D =>6 01-1-E =>7 01-1-0F =>9

01-1-01E =>7 01-1-010F =>9 01-1-0101E =>8 01-1-0101+G =>13 01-1-0101+1

1. S0 =>1 0A =>2 01B => 3 010A => 2 0101B =>4 0101-C =>5 0101-1D =>6 0101-1-E =>7 0101-1-0F =>9 0101-1-01E =>8 0101-1-01+G =>10 0101-1-01+0G =>11

0101-1-01+01G =>13 0101-1-01+011

**2.Спроектировать конечный автомат, составить диаграмму переходов КА и реализовать**

2.1. Спроектировать грамматику для языка L1.

L(KA) = L(G)

KA = {{0,1,+,-}, {S0, A, B, C, D, E, F, G, qf}, δ, S0, qf}

1. δ(S0, 0) = {A}
2. δ(A, 1) = {B}
3. δ(B, 0) = {A}
4. δ(B, -) = {C}
5. δ(C, 1) = {D}
6. δ(D, -) = {E}
7. δ(E, 0) = {F}
8. δ(E, +) = {G}
9. δ(F, 1) = {E}
10. δ(G, 0) = {G}
11. δ(G, 1) = {G}
12. δ(G, 0) = {G, qf}
13. δ(G, 1) = {G, qf}

2.2. Построить диаграмму переходов и таблицу переходов по грамматике.













0,1



0, 1

Примеры конфигураций:

1. (S0, 01-1-01+0) ⊢1 (A, 1-1-01+0) ⊢2 (B, -1-01+0) ⊢4 (C, 1-01+0) ⊢5 (D, -01+0) ⊢6 (E, 01+0) ⊢7 (F, 1+0) ⊢9 (E, +0) ⊢8 (G, 0) ⊢12 (qf**,** ε)
2. (S0, 01-1-01+1) ⊢1 (A, 1-1-01+1) ⊢2 (B, -1-01+1) ⊢4 (C, 1-01+1) ⊢5 (D, -01+1) ⊢6 (E, 01+1) ⊢7 (F, 1+1) ⊢9 (E, +1) ⊢8 (G, 1) ⊢13 (qf**,** ε)
3. (S0, 01-1-01+11) ⊢1 (A, 1-1-01+11) ⊢2 (B, -1-01+11) ⊢4 (C, 1-01+11) ⊢5

(D, -01+11) ⊢6 (E, 01+11) ⊢7 (F, 1+11) ⊢9 (E, +11) ⊢8 (G, 11) ⊢11 (G, 1) ⊢13

(qf**,** ε)

**3.Определить свойства КА. Построить НДКА. Реализовать преобразование НДКА в ДКА.**

3.1. По заданному языку построить КА. Представить в виде диаграмм.

3.2. Реализовать преобразование НДКА в ДКА.

3.3. Реализовать конечный автомат по диаграмме переходов.

**Преобразование НДКА в ДКА**

**НДКА**

myAutomate ndka = new myAutomate(new ArrayList() { "S0", "A", "B", "C", "D", "E", "F", "G", "qf" },

new ArrayList() { "0", "1", "-", "+"}, new ArrayList() { "qf" }, "S0");

ndka.AddRule("S0", "0", "A");

ndka.AddRule("A", "1", "B");

ndka.AddRule("B", "0", "A");

ndka.AddRule("B", "-", "C");

ndka.AddRule("C", "1", "D");

ndka.AddRule("D", "-", "E");

ndka.AddRule("E", "0", "F");

ndka.AddRule("E", "+", "G");

ndka.AddRule("F", "1", "E");

ndka.AddRule("G", "0", "G");

ndka.AddRule("G", "1", "G");

ndka.AddRule("G", "0", "qf");

ndka.AddRule("G", "1", "qf");

**ДКА**

Automate config:

Q: S0 A B C D E F G Gqf

Sigma: 0 1 - +

Q0: S0

F: Gqf

DeltaList:

: (S0 , 0 ) -> A

: (A , 1 ) -> B

: (B , 0 ) -> A

: (B , - ) -> C

: (C , 1 ) -> D

: (D , - ) -> E

: (E , 0 ) -> F

: (F , 1 ) -> E

: (E , + ) -> G

: (G , 0 ) -> Gqf

: (Gqf , 0 ) -> Gqf

: (Gqf , 1 ) -> Gqf

: (G , 1 ) -> Gqf



















Примеры:

Enter line to execute :

01-1-01+1

Length: 9

i :9

curr: Gqf

chineSymbol belongs to language

Enter line to execute :

010101-1-0101+1110

Length: 18

i :18

curr: Gqf

chineSymbol belongs to language

Enter line to execute :

01-1-01+11

Length: 10

i :10

curr: Gqf

chineSymbol belongs to language

Enter line to execute :

010101-11121

Length: 12

i :8

curr: D

chineSymbol doesn't belong to language

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

4.1 Задана КС-грамматика:

P={S → b, S→ cA, A→ Ab, A → AC, A → ε, B → cB, B→ ε, C → d, C → ε}

4.2. Устранить бесполезные символы, ε-правила.

Составим множества производящих символов:

Vp1 = {S, A, B, C}

V’ = Vp

G1 = ({S, A, B, C}, {b, c, d}, P’, S),

где P’ = {S → b, S→ cA, A→ Ab, A → AC, A → ε, B → cB, B→ ε, C → d, C → ε}

Определим недостижимые символы:

VTr1 = {S}

VTr2 = {A, c, b, S}

VTr3 = {C, A, c, b, S}

VTr4 = {d, C, A, c, b, S}

VTr5 = {d, C, A, c, b, S}, символ B недостижим

V’ = V ⋂ VTr

T’ = T ⋂ VTr

G’ = {{S, A, C}, {b, c, d}, P’, S},

где P’ = {S → b, S → cA, A→ Ab, A → ε, A → AC, C → d, C → ε}

**Устранение ε-правил**

G = {{S, A, C}, {b, c, d}, P, S},

где P = {S → b, S → cA, A→ Ab, A → ε, A → AC, C → d, C → ε}

Vε0= {A, C}

Vε1= {A, C}

Составим правила без ε-правил:

P’: S → b | cA | c

A → Ab | b | A | C

C → d

Получим грамматику G’ = {{S, A, C}, {b, c, d}, P’, S}

P’: S → b |cA | c

A → Ab | b | A | C

C → d

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

**5.1. Устранение цепных правил.**

G = {{S, A, C}, {b, c, d}, P, S}

P: S → b | cA | c

A → Ab | b | A | C

C → d

Получаем VA = {A, С}

Устраняем правила вида A → B:

Для VA строим {A → Ab | b | d} и добавляем в P’

Получаем

P’: S → b | cA | c

A → Ab | b | d

В результате G’ = {{S, A}, {b, c, d}, P’, S},

P’: S → b | cA | c

A → Ab | b | d

* 1. **Устранение левой рекурсии.**

Пусть A1 = S, A2 = A. V = {A1, A2}.

Для i = 1 правила вида Ai → Aja отсутствуют

Для i = 2, j = 2 правила A → Ab | b | d α = b, β1 = b, β2 = d.

Заменим A-правила правилами

A → b | d | bA’ | dA’ и добавим в грамматику правила нового нетерминала

A’ → b | bA’

Получили правила новой грамматики G’:

P’: S → b | cA | c

A → b | d | bA’ | dA’

A’ → b | bA’

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

G = {{S, A, A’}, {b, c, d}, P, S},

P: S → b | cA | c

A → b | d | bA’ | dA’

A’ → b | bA’

КС грамматика G = (T, V, P, S) называется грамматикой в нормальной форме Грейбах, если в ней нет ε-правил, т.е. правил вида A→ ε, и каждое правило из P отличное от S → ε, имеет вид A→ aα, где a ∈ T, α ⊂ V.

Форма грамматики — нормальная форма Грейбах

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

1. δ(q0, ε, S) = {(q, b), (q, cA), (q, c)}
2. δ(q, ε, A) = {(q, b), (q, bA’), (q, d), (q, dA’)}
3. δ(q, ε, A’) = {(q, b), (q, bA’)}
4. δ(q, a, a) = {(q, ε}) для всех а ∈ ∑ = {b, c, d}

Последовательность тактов для цепочки cdbb:

(q0, cdbb, S) ⊢1 (q, cdbb, cA) ⊢4 (q, dbb, A) ⊢2 (q, dbb, dA’) ⊢4 (q, bb, A’) ⊢3

(q, bb, bA’) ⊢4 (q, b, A’) ⊢3 (q, b, b) ⊢4 (q, ε, ε)

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

myMp mpA1 = new myMp(new ArrayList() { "q0", "q1", "q2", "qf" }, new ArrayList() { "b", "c", "d" },

new ArrayList() { "b", "c", "d", "S", "A", "B" },"q0", "S",

new ArrayList() { "qf" });

mpA1.addDeltaRule("q0", "", "S", new ArrayList() { "q1" }, new ArrayList() { "cA" });

mpA1.addDeltaRule("q0", "", "S", new ArrayList() { "q1" }, new ArrayList() { "b" });

mpA1.addDeltaRule("q0", "", "S", new ArrayList() { "q1" }, new ArrayList() { "c" });

mpA1.addDeltaRule("q1", "", "A", new ArrayList() { "q1" }, new ArrayList() { "dB" });

mpA1.addDeltaRule("q1", "", "A", new ArrayList() { "q1" }, new ArrayList() { "d" });

mpA1.addDeltaRule("q1", "", "B", new ArrayList() { "q1" }, new ArrayList() { "b" });

mpA1.addDeltaRule("q1", "", "B", new ArrayList() { "q1" }, new ArrayList() { "bB" });

mpA1.addDeltaRule("q1", "b", "b", new ArrayList() { "q1" }, new ArrayList() { "" });

mpA1.addDeltaRule("q1", "c", "c", new ArrayList() { "q1" }, new ArrayList() { "" });

mpA1.addDeltaRule("q1", "d", "d", new ArrayList() { "q1" }, new ArrayList() { "" });

**Пример**:

Debug Mp Deltarules :

delta(Q q0,T ,Z S) = (Q q1,Z cA)

delta(Q q0,T ,Z S) = (Q q1,Z b)

delta(Q q0,T ,Z S) = (Q q1,Z c)

delta(Q q1,T ,Z A) = (Q q1,Z dB)

delta(Q q1,T ,Z A) = (Q q1,Z d)

delta(Q q1,T ,Z B) = (Q q1,Z b)

delta(Q q1,T ,Z B) = (Q q1,Z bB)

delta(Q q1,T b,Z b) = (Q q1,Z )

delta(Q q1,T c,Z c) = (Q q1,Z )

delta(Q q1,T d,Z d) = (Q q1,Z )

Введите строку :

cdb

delta(Q q0,T ,Z S) = (Q q1,Z cA)

delta(Q q1,T c,Z c) = (Q q1,Z )

delta(Q q1,T ,Z A) = (Q q1,Z dB)

delta(Q q1,T d,Z d) = (Q q1,Z )

delta(Q q1,T ,Z B) = (Q q1,Z b)

delta(Q q1,T b,Z b) = (Q q1,Z )

True

**9. Для LL(1) анализатора построить управляющую таблицу M**

T = {i, & , ^, (, )}, V = {S, F, L}, P = {S→ F^L, S→ (S), F→ &L, F→ i, L→F}

p1: S→ F^L, FIRST(F^L) = {i, &}

p2: S→ (S), FIRST((S)) = {(}

p3: F→ &L, FIRST(&L) = {&}

p4: F→ i, FIRST(i) = {i}

p5: L→F, FIRST(F) = {&, i}

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | i | & | ^ | ( | ) | ε |
| S | F^L, 1 | F^L, 1 |  | (S), 2 |  |  |
| F | i, 4 | &L, 3 |  |  |  |  |
| L | F, 5 | F, 5 |  |  |  |  |
| i | ВЫБРОС |  |  |  |  |  |
| & |  | ВЫБРОС |  |  |  |  |
| ^ |  |  | ВЫБРОС |  |  |  |
| ( |  |  |  | ВЫБРОС |  |  |
| ) |  |  |  |  | ВЫБРОС |  |
| ⟂ |  |  |  |  |  | ДОПУСК |

**10. Аналитически написать такты работы LL(1) анализатора для выведенной цепочки.**

Рассмотрим работу алгоритма для цепочки символов &i^i, порожденной LL(1) грамматикой.

|  |  |
| --- | --- |
| Текущая конфигурация | Значение М |
| (&i^i, S⟂, ε) ⊢ | M(S, &) = F^L, 1 |
| (&i^i, F^L⟂, 1) ⊢ | M(F, &) = &L, 3 |
| (&i^i, &L^L⟂, 13) ⊢ | M(&, &) = ВЫБРОС |
| (i^i, L^L⟂, 13) ⊢ | M(L, i) = F, 5 |
| (i^i, F^L⟂, 135) ⊢ | M(F, i) = i, 4 |
| (i^i, i^L⟂, 1354) ⊢ | M(i, i) = ВЫБРОС |
| (^i, ^L⟂, 1354) ⊢ | M(^, ^) = ВЫБРОС |
| (i, L⟂, 13545) ⊢ | M(L, i) = F, 5 |
| (i, F⟂, 135454) ⊢ | M(F, i) = i, 4 |
| (i, i⟂, 135454) ⊢ | M(i, i) = ВЫБРОС |
| (ε, ⟂, 135454) | M(⟂, ε) = ДОПУСК |

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

myGrammar exemple = new myGrammar(new ArrayList() { "i", "(", ")", "&", "^", "" },

new ArrayList() { "S", "F", "L" },

"S");

exemple.AddRule("S", new ArrayList() { "F", "^", "L" });

exemple.AddRule("S", new ArrayList() { "(", "S", ")" });

exemple.AddRule("F", new ArrayList() { "&", "L" });

exemple.AddRule("F", new ArrayList() { "i" });

exemple.AddRule("L", new ArrayList() { "F" });

Пример работы алгоритма:

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

i, (, ), &, ^, ,

Также создаем строку для Эпсилон

Рассмотрим нетерминал S

Первый символ правила S -> F^L - &

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

Первый символ правила S -> F^L - i

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

Первый символ правила S -> (S) - (

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

Рассмотрим нетерминал F

Первый символ правила F -> &L - &

Это правило заносим в таблицу на пересечении строки нетерминала F и столбца терминала &

Первый символ правила F -> i - i

Это правило заносим в таблицу на пересечении строки нетерминала F и столбца терминала i

Рассмотрим нетерминал L

Первый символ правила L -> F - &

Это правило заносим в таблицу на пересечении строки нетерминала L и столбца терминала &

Первый символ правила L -> F - i

Это правило заносим в таблицу на пересечении строки нетерминала L и столбца терминала i

Введите строку:

&i^i

Успех. Строка соответствует грамматике.

135454

**12. Построить LR(k) анализатор способом кодирования магазинных символов.**

T = {i, & , ^, (, )}, V = {S, F, L}, P = {S→ F^L, S→ (S), F→ &L, F→ i, L→F}

Определим пополненную грамматику:

p0: S’ → S

p1: S→ F^L

p2: S→ (S)

p3: F → &L

p4: F → i

p5: L → F

**Определение активных префиксов**

|  |  |  |
| --- | --- | --- |
| Символ грамматики | Магазинный символ | Кодируемая цепочка |
| S | S**0**  S**2** | ⟂S  (S |
| F | F1  F5 | F  F |
| L | L1  L3 | F^  & |
| i | i4 | i |
| ( | (2 | ( |
| & | &3 | & |
| ^ | ^1 | F^ |
| ) | )2 | (S) |

**Отношение OBLOW**

OFIRST(S0) = {S0, F1, (2, &3, i4}

OFIRST(S2) = {S2, F1, (2, &3, i4}

OFIRST(F1) = {F1, &3, i4}

OFIRST(L1) = {L1, &3, i4}

OFIRST(L3) = {L3, &3, i4}

OFIRST(F5) = {F5, &3, i4}

⟂ OBLOW S0

⟂ OBLOW (2

⟂ OBLOW F1

⟂ OBLOW &3

⟂ OBLOW i4

F1 OBLOW ^1

S2 OBLOW )2

(2 OBLOW S2

(2 OBLOW F1

(2 OBLOW &3

(2 OBLOW i4

(2 OBLOW (2

^1 OBLOW L1

^1 OBLOW F5

^1 OBLOW &3

^1 OBLOW i4

&3 OBLOW L3

&3 OBLOW F5

&3 OBLOW &3

&3 OBLOW i4

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | S0 | S2 | F1 | F5 | L1 | L3 | i4 | (2 | &3 | ^1 | )2 |
| S0 |  |  |  |  |  |  |  |  |  |  |  |
| S2 |  |  |  |  |  |  |  |  |  |  | 1 |
| F1 |  |  |  |  |  |  |  |  |  | 1 |  |
| F5 |  |  |  |  |  |  |  |  |  |  |  |
| L1 |  |  |  |  |  |  |  |  |  |  |  |
| L3 |  |  |  |  |  |  |  |  |  |  |  |
| i4 |  |  |  |  |  |  |  |  |  |  |  |
| (2 |  | 1 | 1 |  |  |  | 1 | 1 | 1 |  |  |
| &3 |  |  |  | 1 |  | 1 | 1 |  | 1 |  |  |
| ^1 |  |  |  | 1 | 1 |  | 1 |  | 1 |  |  |
| )2 |  |  |  |  |  |  |  |  |  |  |  |
| ⟂ | 1 |  | 1 |  |  |  | 1 | 1 | 1 |  |  |

**13. Построить управляющую таблицу для функции перехода g(х) и действий f(u).**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Функция действий f(u) | | | | | | Функция переходов g(X) | | | | | | | |
|  | i | ( | & | ^ | ) | ⟂ | S | F | L | i | ( | & | ^ | ) |
| S0 | П | П | П | П | П | Д |  |  |  |  |  |  |  |  |
| S2 | П | П | П | П | П |  |  |  |  |  |  |  |  | )2 |
| F1 | П | П | П | П | П |  |  |  |  |  |  |  | ^1 |  |
| F5 | С(5) | С(5) | С(5) | С(5) | С(5) | С(5) |  |  |  |  |  |  |  |  |
| L1 | С(1) | С(1) | С(1) | С(1) | С(1) | С(1) |  |  |  |  |  |  |  |  |
| L3 | С(3) | С(3) | С(3) | С(3) | С(3) | С(3) |  |  |  |  |  |  |  |  |
| i4 | С(4) | С(4) | С(4) | С(4) | С(4) | С(4) |  |  |  |  |  |  |  |  |
| (2 | П | П | П | П | П |  | S2 | F1 |  | i4 | (2 | &3 |  |  |
| &3 | П | П | П | П | П |  |  | F5 | L3 | i4 |  | &3 |  |  |
| ^1 | П | П | П | П | П |  |  | F5 | L1 | i4 |  | &3 |  |  |
| )2 | С(2) | С(2) | С(2) | С(2) | С(2) | С(2) |  |  |  |  |  |  |  |  |
| ⟂ | П | П | П | П | П |  | S0 | F1 |  | i4 | (2 | &3 |  |  |

Такты работы для цепочки &i^i:

(⟂, &i^i⟂, ε) ⊢ (⟂&3, i^i⟂, ε) ⊢ (⟂&3i4, ^i⟂, ε) ⊢ (⟂&3F5, ^i⟂, 4) ⊢ (⟂&3L3, ^i⟂, 45) ⊢ (⟂F1, ^i⟂, 453) ⊢ (⟂F1^1, i⟂, 453) ⊢ (⟂F1^1i4, ⟂, 453) ⊢ (⟂F1^1F5, ⟂, 4534) ⊢ (⟂F1^1L1, ⟂, 45345) ⊢ (⟂S0, ⟂, 453451) ⊢ ДОПУСК

Вывод цепочки:

S →1 F^L →5 F^F →4 F^i →3 &L^i →5 &F^i →4 &i^i

**14. Построить LR(k) анализатор способом грамматическими вхождений (замыкание множества ситуаций для пополненной LR(k) грамматики**

A0 = CLOSURE({|S’ -> •S|}) = {S’ -> •S, S -> •F^L, S -> •(S), F -> •&L, F -> •i}

|  |  |
| --- | --- |
| A0 | S’ -> •S  S -> •F^L  S -> •(S)  F -> •&L  F -> •i |
| A1 =  GOTO(A0,S) | S’ -> S• |
| A2 =  GOTO(A0,() | S -> (•S)  S -> •F^L  S -> •(S)  F -> •&L  F -> •i |
| A3 =  GOTO(A0,i) | S -> i• |
| A4 =  GOTO(A0,&) | F -> &•L  L -> •F  F -> •i  F -> •&L |
| A5 =  GOTO(A0,F) | S -> F•^L |
| A6 =  GOTO(A2,S) | S -> (S•) |
| A7 =  GOTO(A4,L) | F -> &L• |
| A8 =  GOTO(A4,F) | L -> F• |
| A9 =  GOTO(A5,^) | S -> F^•L  L -> •F  F -> •&L  F -> •i |
| A10 =  GOTO(A6,)) | S -> (S)• |
| A11 =  GOTO(A9,L) | S ->F^L• |

Построим диаграмму переходов ДКА для активных префиксов





























**15. Построить конечный автомат для переходов между ситуациями и такты работы алгоритма**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | | | | | GOTO | | |
|  | ( | ) | ^ | i | & | $ | S | F | L |
| 0 | П(2) |  |  | П(3) | П(4) |  | 1 | 5 |  |
| 1 |  |  |  |  |  | Допуск |  |  |  |
| 2 |  |  |  |  |  |  | 6 |  |  |
| 3 | С(4) | С(4) | С(4) | С(4) | С(4) | С(4) |  |  |  |
| 4 |  |  |  | П(3) |  |  |  | 8 | 7 |
| 5 |  |  | П(9) |  |  |  |  |  |  |
| 6 |  | П(10) |  |  |  |  |  |  |  |
| 7 | С(3) | С(3) | С(3) | С(3) | С(3) | С(3) |  |  |  |
| 8 | С(5) | С(5) | С(5) | С(5) | С(5) | С(5) |  |  |  |
| 9 |  |  |  |  | П(4) |  |  | 8 | 11 |
| 10 | С(2) | С(2) | С(2) | С(2) | С(2) | С(2) |  |  |  |
| 11 | С(1) | С(1) | С(1) | С(1) | С(1) | С(1) |  |  |  |

Последовательность тактов алгоритма для цепочки i^&i

(⟂0, i^&i$, ε) ⊢ (⟂03, ^&i$, ε) ⊢ (⟂05, ^&i$, 4) ⊢ (⟂059, &i$, 4) ⊢

(⟂0594, i$, 4) ⊢ (⟂05943, $, 4) ⊢ (⟂05948, $, 44) ⊢ (⟂05947, $, 445) ⊢ (⟂0598, $, 4453) ⊢ (⟂059(11), $, 44535) ⊢ (⟂01, $, 445351)

Проверка:

S →1 F^L →5 F^F →3 &L^F →5 &F^F →4 &i^F →4 &i^i

**16. Реализовать LR(k)-анализатор по управляющей таблице (g,f) для LR(k) грамматики.**

Terminals = "^&i()";

NonTerminals = "SFL";

Grammar.Add("S F^L");

Grammar.Add("S (S)");

Grammar.Add("F &L");

Grammar.Add("F i");

Grammar.Add("L F");

Execute();

Cоздана последовательность С:

I0 { П .S,$; S .F^L,$; S .(S),$; F .&L,^; F .i,^ }

I1 { F &.L,^; L .F,^; F .&L,^; F .i,^ }

I2 { F i.,^ }

I3 { S (.S),$; S .F^L,); S .(S),); F .&L,^; F .i,^ }

I4 { П S.,$ }

I5 { S F.^L,$ }

I6 { F &L.,^ }

I7 { S F^.L,$; L .F,$; F .&L,$; F .i,$ }

I8 { F &.L,$; L .F,$; F .&L,$; F .i,$ }

I9 { F i.,$ }

I10 { S (.S),); S .F^L,); S .(S),); F .&L,^; F .i,^ }

I11 { S (S.),$ }

I12 { L F.,^ }

I13 { S F^L.,$ }

I14 { S (S).,$ }

I15 { S (S.),) }

I16 { S F.^L,) }

I17 { F &L.,$ }

I18 { S F^.L,); L .F,); F .&L,); F .i,) }

I19 { F &.L,); L .F,); F .&L,); F .i,) }

I20 { F i.,) }

I21 { S (S).,) }

I22 { L F.,$ }

I23 { S F^L.,) }

I24 { L F.,) }

I25 { F &L.,) }

Создана ACTION таблица

ACTION[18, i] = s 20

ACTION[7, &] = s 8

ACTION[3, i] = s 2

ACTION[2, ^] = r F i

ACTION[1, i] = s 2

ACTION[0, &] = s 1

ACTION[7, i] = s 9

ACTION[6, ^] = r F &L

ACTION[5, ^] = s 7

ACTION[4, $] = a

ACTION[11, )] = s 14

ACTION[10, &] = s 1

ACTION[9, $] = r F i

ACTION[8, &] = s 8

ACTION[15, )] = s 21

ACTION[14, $] = r S (S)

ACTION[13, $] = r S F^L

ACTION[12, ^] = r L F

ACTION[19, &] = s 19

ACTION[1, &] = s 1

ACTION[17, $] = r F &L

ACTION[16, ^] = s 18

ACTION[23, )] = r S F^L

ACTION[22, $] = r L F

ACTION[21, )] = r S (S)

ACTION[20, )] = r F i

ACTION[25, )] = r F &L

ACTION[10, i] = s 2

ACTION[0, i] = s 2

ACTION[19, i] = s 20

ACTION[10, (] = s 10

ACTION[18, &] = s 19

ACTION[8, i] = s 9

ACTION[0, (] = s 3

ACTION[24, )] = r L F

ACTION[3, &] = s 1

ACTION[3, (] = s 10

Создана GOTO таблица

GOTO[10, S] = 15

GOTO[3, F] = 16

GOTO[1, F] = 12

GOTO[0, F] = 5

GOTO[7, F] = 22

GOTO[1, L] = 6

GOTO[10, F] = 16

GOTO[8, L] = 17

GOTO[8, F] = 22

GOTO[19, F] = 24

GOTO[3, S] = 11

GOTO[19, L] = 25

GOTO[18, L] = 23

GOTO[18, F] = 24

GOTO[7, L] = 13

GOTO[0, S] = 4

Введите строку:

i^&i

Введена строка: i^&i$

Процесс вывода:

F->i

F->i

L->F

F->&L

L->F

S->F^L

Строка допущена