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 КС-грамматики цепные правила и устранить левую рекурсию
6 Определить форму КС-грамматики и сделать ее приведение
7. Спроектировать МП автомат для приведенной КС-грамматики
8. Реализовать МП автомат для приведенной КС-грамматики
9. Для LL(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. Построить замыкание множества ситуаций для пополненной LR(1) грамматики.
13. Определить функцию перехода g(x)
14. Построить каноническую форму множества ситуаций.
15. Построить управляющую таблицу для функции перехода g(x) и действий f(u).
16. Реализовать LR(1)-анализатор по управляющей таблице (g,f) для LR(1) грамматики.

Студент: Полей-Добронравова Амелия Вадимовна

Группа: М80-207Б-18

Руководитель: Семёнов А. С.

Оценка:

Дата:

Москва 2020

1.Спроектировать грамматику по заданному языку L:

1.1. Задан бесконечный регулярный язык L

(вариант 18)

$L=\{w_1 0\text{-}w_2 \mid w_1\{0,1\}^+, w_2\{0,1\}^+\}$

1.2. Преобразовать бесконечный регулярный язык(L) в подязык(L_1), цепочки символов которого являются подмножество цепочек символов бесконечного языка.

$L1=\{w_1 0\text{-}w_2 \mid w_1=\{0,1\}^+, w_2=\{0,1\}^+, |w1| <=2, |w2| = 1 \}$

1.3. Сгенерировать цепочки символов по языку L_1.

$w_1 0\text{-}w_2$ => 00-0

$w_1 0\text{-}w_2$ => 10-0

$w_1 0\text{-}w_2$ => 000-0

$w_1 0\text{-}w_2$ => 110-0

$w_1 0\text{-}w_2$ => 00-1

$w_1 0\text{-}w_2$ => 10-1

$w_1 0\text{-}w_2$ => 000-1

$w_1 0\text{-}w_2$ => 110-1

L1 = {00-0, 00-1, 10-0,

10-1,000-0,110-0,000-1,110-1,010-1,010-0,100-0,100-1}

1.4 . Спроектировать грамматику для языка L_1.

$G=(T,V,P,S_0)$

T - конечное множество терминалов T = {0, 1, -}

V - конечное множество нетерминалов $V = \{A,B,S_0\}$

P:

S_0->0A

S_0->1A

A->0C

A->1G

G->0D

C->-B

C->0D

D->-B

B->0qf

B->1qf

возможные цепочки:

S_0=>0A=>00C=>00-B=>00-0

S_0=>0A=>00C=>00-B=>00-1

S_0=>1A=>10C=>10-B=>10-0

S_0=>1A=>10C=>10-B=>10-1

S_0=>0A=>00C=>000D=>000-B=>000-0

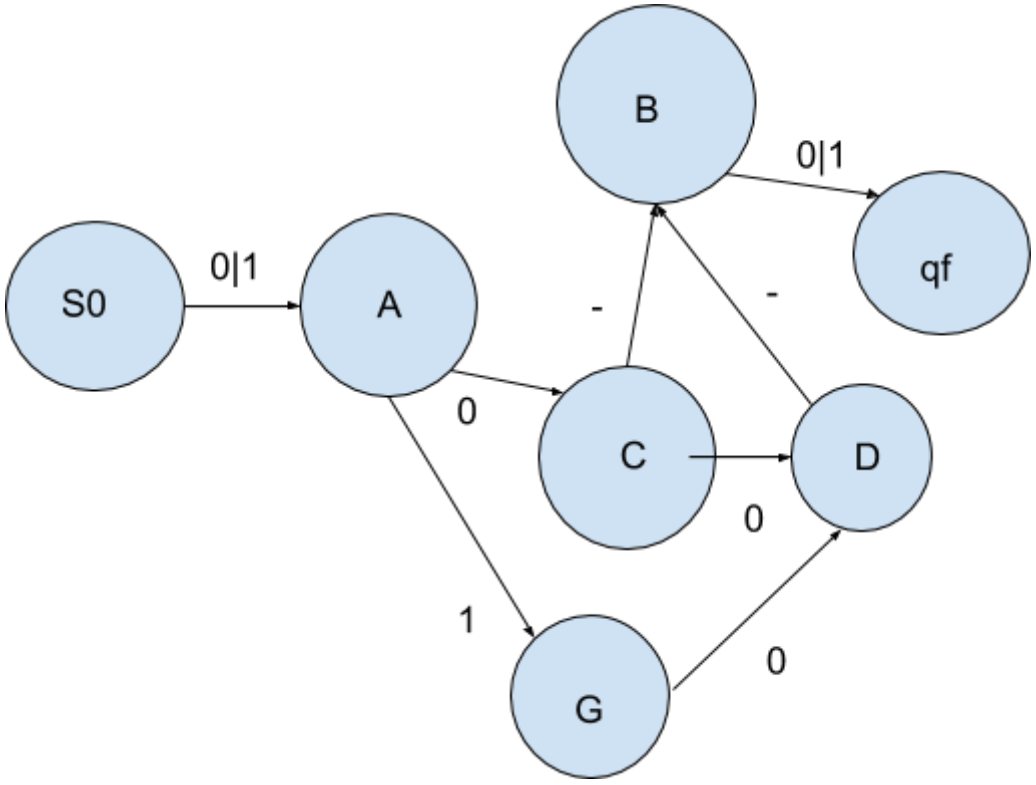
S_0=>1A=>11G=>110D=>110-B=>110-0

S_0=>0A=>00C=>000D=>000-B=>000-1

S_0=>1A=>11G=>110D=>110-B=>110-1

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

2.1. Построить диаграмму переходов и таблицу переходов по грамматике (1.3).



2.2 . Реализовать конечный автомат по диаграмме переходов.

KA={Q,Σ ,σ ,q_0,F}

Q - конечное множество состояний (= V с объединением q_F конечных состояний)

Σ - конечный алфавит входных

σ - функция переходов

q_0 - начальное состояние автомата

F - множество заключительных состояний

Σ={0,1, -}; V = {A, B, C, D, G}

σ_1 (S_0,0)={A}

σ_2 (S_0,1)={A}

σ_3 (A,0)={C}

σ_4 (B,0)={q_F}

σ_5 (B,1)={q_F}

σ_6 (C,-)={B}

σ_7 (G,0)={D}

σ_8 (C,0)={D}

σ_9 (A,1)={G}

σ_{10} (D,-)={B}

KA: (S_0,00-1) $|-^1$ (A,0-1) $|-^3$ (C,-1) $|-^6$ (B,1) $|-^5$ (q_F,ε)

L1 и L1(KA) эквивалентны

```
case "1":
                    myAutomate ka = new myAutomate(new ArrayList() { "S0",
"A", "B", "C",”D”,”G”, "qf" }, new ArrayList() { "0", "1", "-", "" },new
ArrayList() { "qf" }, "S0");
                    ka.AddRule("S0", "1", "A");
                    ka.AddRule("S0", "0", "A");
                    ka.AddRule("G", "0", "D");
                   ka.AddRule("C", "0", "D");
```

```
ka.AddRule("A", "0", "C");
ka.AddRule("A", "1", "G");
ka.AddRule("C", "-", "B");
ka.AddRule("D", "-", "B");
ka.AddRule("B", "0", "qf");
ka.AddRule("B", "1", "qf");
```

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

3.1. Сгенерировать цепочку символов по заданному языку

$L = \{w_1 0\text{-}w_2 \mid w_1=\{0,1\}^+, w_2=\{0,1\}^+|\}$

$w_1 0\text{-}w_2$ => $0w_1 0\text{-}w_2$ => $01w_1 0\text{-}w_2$ => $01w_1 0\text{-}0w_2$ =>$01w_1 0\text{-}00w_2$ =>...

Язык L нерегулярный.

Для всех регулярных языков L существует целое $p \geq 1$ такое что для всех(w ∈ L((|w|>= p) => существует (x,y, z такое что (w = xyz =>

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

2. |xy | ≤ p, цикл должен быть в пределах первых p символов и

3. для всех i ≥ 0, (xy^iz L))))))), на x и z ограничений не накладывается.

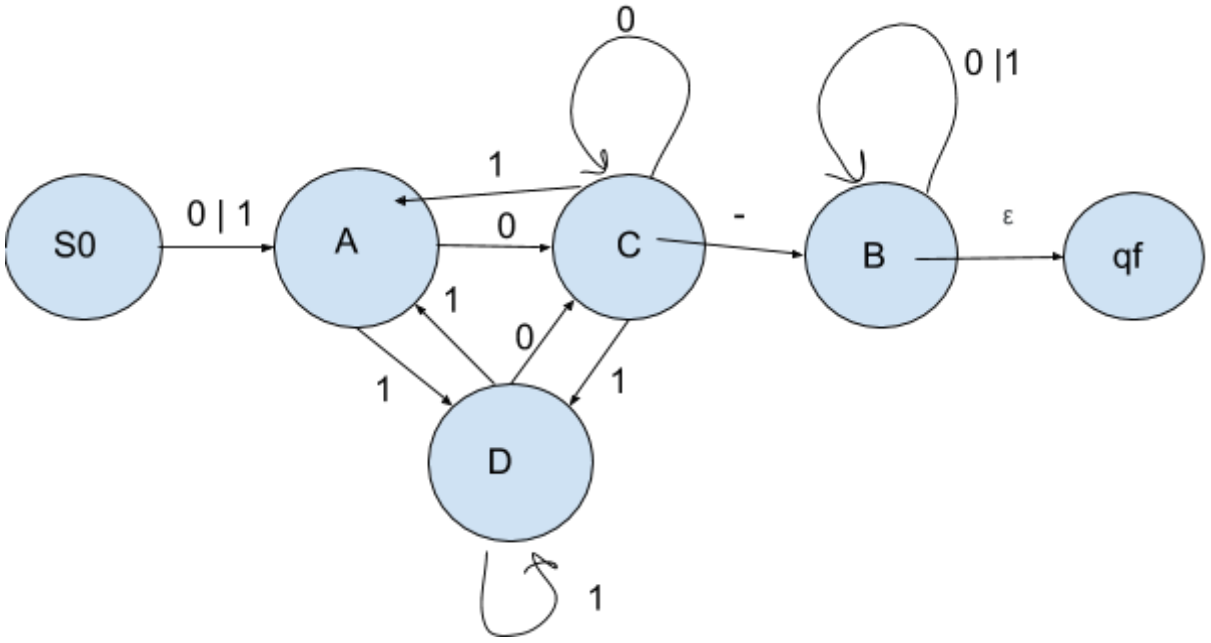
Доказательство:

существует число p >=1 такое, что для любой цепочки v из L: |v|>=p, записанной в виде v=xyz, |y|>=1, |xy|<=p, что для всех i>0 справедливо $xy^iz \in L$.

Пусть $w_3=\{0,1\}$ - один символ: либо 0, либо 1. По условию, $w_2=\{0,1\}^+$ - цепочка символов.

Пусть $x=w_3$, $y=w_3$, $z=0\text{-}w_2$, p = 4, |xy| = 2 < p, |v| >= 4 = p, тогда при любом количестве повторений y цепочка $v = xy^iz \in L$.

3.2. По заданному языку построить НДКА. Представить в виде диаграмм.



L(НДКА)=L(G)

3.3. Реализовать преобразование НДКА в ДКА.

Реализация представлена в выданном преподавателем фреймворке.

DeltaList:
: (S0 , 1) -> 1
: (1 , 1) -> 4
: (4 , 1) -> 41
: (41 , 1) -> 41
: (41 , 0) -> 3
: (3 , 1) -> 41
: (3 , 0) -> 3
: (3 , -) -> 2qf
: (2qf , 1) -> 2qf
: (2qf , 0) -> 2qf
: (4 , 0) -> 3
: (1 , 0) -> 3
: (S0 , 0) -> 1

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

4.1 Задана КС-грамматика:

P={S → cB, B → cB, B → cA, A→ C, A→ aB, C→ Ca, C→ cf, A→ ε }

4.2. Устранить бесполезные символы, ε-правила.

ε-правила: A→ ε; бесполезных символов нет.

P={S → cB, B → cB, B → cA, B → c,A→ C, A→ aB, C→ Ca, C→ cf}

4.3. Реализация алгоритма.

Реализация представлена в выданном преподавателем фреймворке.

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

5.1. Устранение цепных правил.

цепные правила: A->C; замена на {A->cf, A->Ca}

Prules:
S -> cB
B -> c
B -> cB
B -> cA
A -> aB
C -> cf
C -> Ca
A -> cf
A -> Ca

5.2. Устранение левой рекурсии.

P={S -> cB, A -> cf, A -> aB, B -> c, B -> cA, B -> cB, A -> Ca, C -> cf, C -> Ca}

Что заменяем	На что
C->Ca C->cf	C->cf | cfC’ C’->a | aC’

P = {S -> cB, A -> cf, A -> aB, B -> c, B -> cA, B -> cB, A -> Ca, C -> cf,
C->a | aC', C'->a | aC'}

5.3. Реализация алгоритма.

Реализация представлена в выданном преподавателем фреймворке.

6. Определить форму КС-грамматики и сделать ее приведение

6.1. Задана следующая грамматика:

18. T = {i, * , +,(,)}, V={S, F, L}, P = {S→F+ L, F→ (L*), F→ i, L→F}

правила G грамматики рассмотреть как правила LL(k) грамматики

6.2. Эта же грамматика в приведённом виде:

T = {i, * , +,(,)}, V={S, F, L}, P = {S -> F+L, F -> i,F -> (L*),L -> i,L -> (L*)}, т.к. F,L -бесполезные символы

Определить форму грамматики (Грейбаха и т.д.)

7. Спроектировать МП автомат для приведенной КС-грамматики

T = {i, * , +,(,)}, V={S, F, L}, P = {S -> F+L, F -> i, F -> (L*), L -> i, L -> (L*)}

как правила LR(k).

МП = (Q, Σ, Γ, δ, q_0, z_0, F)

Q - конечное число состояний устройства управления

Σ - конечный алфавит входных символов

Γ - конечный алфавит магазинных символов

δ - функция переходов, отображает множество Q x (Σ ∪ {ε}) x Γ в множество конечных подмножеств множества Q x Γ^*

q_0 - начальное состояние, $q_0 \in Q$

z_0 - начальный символ магазина, $z_0 \in \Gamma$

F - множество заключительных состояний, $F \subseteq Q$

Проектирование:

МП = (Q, {i, * , +,(,)}, {i, * , +,(,),S, F, L}}, δ, q_0, S, {q})

1) δ(q_0, ε, S) = {(q, F+L)}
2) δ(q, ε, F) = {(q, i), (q, (L*))}
3) δ(q, ε, L) = {(q, i), (q, (L*))}
4) δ(q, a, a) = {(q, ε)} для всех a ∈ Σ = {i, * , +,(,)}

При анализе входной цепочки i+((i*)*) МП автомат выполнит последовательность тактов:

(q_0, i+((i*)*), S) $|-^1$ (q_0, i+((i*)*), F+L) $|-^2$ (q_0, i+((i*)*), i+L) $|-^4$ (q_0, +((i*)*), +L) $|-^4$ (q_0, ((i*)*), L) $|-^3$ (q_0, ((i*)*), (L*)) $|-^4$ (q_0, (i*)*), L*)) $|-^3$ (q_0, (i*)*), (L*)*)) $|-^4$ (q_0, i*)*), L*)*)) $|-^3$ (q_0, i*)*), i*)*)) $|-^4$ (q_0, *)*), *)*)) $|-^4$ (q_0,)*),)*)) $|-^4$ (q_0, *), *)) $|-^4$ (q_0,),)) $|-^4$ (q_0, ε, ε)

Расширенный МП автомат

РМП = ({q, r}, {i, * , +,(,)}, {z_0, i, * , +,(,),S, F, L}}, δ, q_0, z_0, {r})

1) δ(q, a, ε) = {(q, a)} для всех a ∈ Σ = {i, * , +,(,)}
2) δ(q, ε, F+L) = {(q, S)}
3) δ(q, ε, (L*)) = {(q, F), (q, L)}
4) δ(q, ε, i) = {(q, F), (q, L)}
5) δ(q, ε, z_0S) = {(r, ε)}
6) δ(q, a, z_0) = {(q, a)} для всех a ∈ Σ = {i, * , +,(,)} //для начала работы

При анализе входной цепочки i+((i*)*) РМП автомат выполнит последовательность тактов:

(q, i+((i*)*), z_0) $|-^1$ (q, +((i*)*), z_0i) $|-^4$ (q, +((i*)*), z_0F) $|-^1$ (q, ((i*)*), z_0F+)

$|-^{1}$ (q, (i*)*)), z_0F+() $|-^{1}$ (q, i*)*), z_0F+(() $|-^{1}$ (q,*)*), z_0F+((i) $|-^{4}$ (q,)*), z_0F+((L)

$|-^{1}$ (q,)*), z_0F+((L*) $|-^{1}$ (q,*), z_0F+((L*)) $|-^{3}$ (q,*), z_0F+(L) $|-^{1}$ (q,), z_0F+(L*)

$|-^{1}$ (q, ε, z_0F+(L*)) $|-^{3}$ (q, ε, z_0F+L) $|-^{2}$ (q, ε, z_0S) $|-^{5}$ (q, ε, ε)

8. Реализовать МП автомат для приведенной КС-грамматики

Реализация представлена в выданном преподавателем фреймворке.

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

FIRST(A) – это множество первых терминальных символов, которыми начинаются цепочки, выводимые из нетерминала A.

FOLLOW(A) – это множество следующих терминальных символов, которые

могут встретиться непосредственно справа от нетерминала в некоторой сентенциальной форме.

Таблица M определяется на множестве

$(V \cup T \cup \{\perp\}) \times (T \cup \{\varepsilon\})$ по правилам:

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

	i	*	+	(	)	ε
S	F+L, 1			F+L, 1		
F	i, 2			(L*) , 3		
L	i, 4			(L*), 5		

i	ВЫБРОС					
*		ВЫБРОС				
+			ВЫБРОС			
(				ВЫБРОС		
)					ВЫБРОС	
⊥						ДОПУСК

Построение таблицы построчно:

p_1: S→F+ L.

FIRST(F+L) = {i, (}

M(S, i) = M(S, () = F+L, 1

p_2: F -> i.

FIRST(i) = {i}

M(F, i) = i, 2

p_3: F -> (L*).

FIRST((L*)) = {(}

M(F, () = (L*), 3

p_4: L -> i.

FIRST(i) = {i}

M(L, i) = i, 4

p_5: L -> (L*).

FIRST((L*)) = {(}

M(L, () = (L*), 5

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 M
(i+((i*)*), S, ε)	M(S, i) = F+L, 1
(i+((i*)*), F+L, 1)	M(F, i) = i, 2
(i+((i*)*), i+L, 12)	ВЫБРОС
(+((i*)*), +L, 12)	ВЫБРОС
(((i*)*), L, 12)	M(L, () = (L*), 5
(((i*)*), (L*), 125)	ВЫБРОС
((i*)*), L*), 125)	M(L, () = (L*), 5
((i*)*), (L*)*), 1255)	ВЫБРОС
(i*)*), L*)*), 1255)	M(L, i) = i, 4
(i*)*), i*)*), 12554)	ВЫБРОС
(*)*), *)*), 12554)	… Выбросы
(ε,ε , 12554)	ДОПУСК

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

Реализация представлена в выданном преподавателем фреймворке.

12. Построить замыкание множества ситуаций для пополненной LR(1)**грамматики.**

T = {i, * , +,(,)}, V={S, F, L}, P = {S -> F+L, F -> i, F -> (L*), L -> i,
L -> (L*)}

12.1. Определить пополненную LR(1) грамматику

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

Пополненная грамматика:

T = {i, * , +,(,)}, V={S, S', F, L}, P = {S' -> S, S -> F+L, F -> i, F -> (L*), L -> F}

S' - начальный символ.

Далее возможно двумя способами.

1 способ. Построение LR(k) анализатора способом использования расширенного магазинного алфавита.

Способ использования расширенного магазина состоит из трех шагов:

Шаг 1. Определение активных префиксов;

Шаг 2. Построение управляющей таблицы;

Шаг 3. Применение алгоритма «перенос-свёртка».

В приведенной выше грамматике с правилами:

p_0: S’ -> S,

p_1: S -> F+L,

p_2: F -> i,

p_3: F -> (L*),

p_4: L -> F

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

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

Символ грамматики	Магазинный символ	Кодируемая цепочка
S	S_0	⊥S
F	F_1 F_4	F F
L	L_1 L_3	F+L (L
i	i_2	i
+	$+_1$	F+
*	$*_3$	(L*
(	$(_3$	(
)	$)_3$	(L*)

Шаг 2. Построение управляющей таблицы;

Управление алгоритмом осуществляется при помощи двух функций, приведенных в таблице следующим образом:

1. Используя значение верхнего символа магазина и входного символа, алгоритм определяет значение функции действия: ПЕРЕНОС или СВЕРТКА;
2. При выполнении переноса определяется значение функции переходов, равное магазинному символу, который нужно втолкнуть в магазин;
3. Значение функции действия, равное СВЕРТКА(i), однозначно определяет этот шаг.

Работа алгоритма описывается в терминах конфигураций, представляющих собой тройки вида (αT, ax, π), где αT – цепочка магазинных символов (T- верхний символ магазина), ax – необработанная часть входной цепочки, π выход (строка из номеров правил), построенный к настоящему моменту времени.

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

Функция действий f(u) определяется на множестве $(V_p \cup \{\times(T \cup \{\varepsilon\})$ по правилам:

1. Если A → β – правило грамматики с номером i, то для конфигурации (αT, ax,π), где T кодирует цепочку β, f(u) = C(i).
2. Если A → β – правило грамматики с номером i, то для конфигурации (αT, ax,π), где T кодирует некий префикс цепочки β (но не саму основу), f(u) = П.
3. Для конфигурации (S_0, π), где S_0 кодирует цепочку S, f(u) = ДОПУСК.
4. В противном случае, f(u) = ОШИБКА.

Функция переходов g(X) определяется на множестве (Vp ∪ {×(V ∪ T ∪ {ε})

по правилам:

1. Если для конфигурации (αT, ax, π) для входного символа a ε (V ∪ T) в таблице1 существует символ ai, совместимый с цепочкой αTa, то g(X) = ai.

2. В противном случае, g(X) = ОШИБКА.

функция действий f(u)	функция переходов g(X)

	i	+	*	(	)	⊥	S	F	L	i	+	*	(	)
S_0	П	П	П	П	П	Д		F_1		i_2			$(_3$	
F_1	П	П	П	П	П						$+_1$			
F_4	С4	С4	С4	С4	С4	С4								
L_1	С1	С1	С1	С1	С1	С1								
L_3	П	П	П	П	П							$*_3$		
i_2	С2	С2	С2	С2	С2	С2								
$+_1$	П	П	П	П	П			F_4	L_1	i_2			$(_3$	
$*_3$	П	П	П	П	П									$)_3$
$(_3$	П	П	П	П	П			F_4	L_3	i_2			$(_3$	
$)_3$	С3	С3	С3	С3	С3	С3								
⊥	П	П	П	П	П	П	S_0	F_1		i_2			$(_3$	

Рассмотрим последовательность тактов алгоритма при анализе входной цепочки i+((i*)*)

(⊥, i+((i*)*)⊥, ε) |- (⊥i_2, +((i*)*)⊥, ε) |- (⊥F_1, +((i*)*)⊥, 2) |- (⊥F_1+$_1$, ((i*)*)⊥, 2) |- (⊥F_1+$_1$($_3$, (i*)*)⊥, 2) |- (⊥F_1+$_1$($_3$($_3$, i*)*)⊥, 2) |- (⊥F_1+$_1$($_3$($_3$$i_2$, *)*)⊥, 2) |- (⊥$F_1$+$_1$($_3$($_3$$F_4$, *)*)⊥, 22) |- (⊥$F_1$+$_1$($_3$($_3$$L_3$,*)*)⊥, 224) |- (⊥$F_1$+$_1$($_3$($_3$$L_3$*$_3$,)*)⊥, 224) |- (⊥$F_1$+$_1$($_3$($_3$$L_3$*$_3$)$_3$,*)⊥, 224) |- (⊥$F_1$+$_1$($_3$$F_4$,*)⊥, 2243) |- (⊥$F_1$+$_1$($_3$$L_3$,*)⊥, 22434) |- (⊥$F_1$+$_1$($_3$$L_3$*$_3$,)⊥, 22434) |- (⊥$F_1$+$_1$($_3$$L_3$*$_3$)$_3$, ⊥, 22434) |- (⊥$F_1$+$_1$$F_4$, ⊥, 224343) |- (⊥$F_1$+$_1$ L_1, ⊥, 2243434) |- (⊥S_0, ⊥, 22434341) |- ДОПУСК

2 способ. Построение LR(k) анализатора способом грамматических вхождений.

Способ использования грамматических вхождений состоит из четырёх шагов:

Шаг 1. Определение грамматических вхождений (см. алгоритм фреймворка);

Шаг 2. Построение конечного автомата из грамматических вхождений;

Шаг 3. Построение управляющей таблицы.

Шаг 4. Применение алгоритма «перенос-свёртка».

Построение:

Шаг 1. Определение грамматических вхождений

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

В приведенной выше грамматике с правилами:

p_0: S’ -> S,

p_1: S -> F+L,

p_2: F -> i,

p_3: F -> (L*),

p_4: L -> F

грамматические вхождения: $S_{0,1}$, $F_{1,1}$, $+_{1,2}$, $L_{1,3}$, $i_{2,1}$, $(_{3,1}$, $L_{3,2}$, $*_{3,3}$, $)_{3,4}$, $F_{4,1}$

Шаг 2. Построение конечного автомата из грамматических вхождений

12.2. Определить множество First для LR(1) грамматики

First = {i, (}

12.3. Определить множество LR(1) ситуаций I

LR(1)-ситуация это объект вида [A→α · β, a], где (A→αβ) ∈ P, · ∉ V ∪ T, a ∈ {$} ∪ T, $ - конец строки

LR(1)-ситуация допустима для активного префикса +, если существует вывод $S \Rightarrow_r^* \gamma A w \Rightarrow_r \gamma\alpha\beta w$, где $+ = \gamma\alpha$ и либо a - первый символ w, либо w = ε и a = $.

Ситуация *допустима*, если она допустима для какого-либо активного префикса.

Считая, что α и β ∈ {ε} ∪ T ∪ V:

I = {(S’ -> · S, $), (S’ -> S · , $), (S -> · F+L, $), (S -> F · +L, $), (S -> F+ · L, $), (S -> F+L · , $), (F -> · i, +), (F -> i · , +), (F -> · (L*), +), (L -> · F, $), (L -> F · , $), (F -> (· L*), +), (F -> (L · *), +), F -> (L* ·), +), F -> (L*) · , +), (F -> · i, +), (F -> i · , +)}

12.4. Построить замыкание clouser(I) множества ситуаций

CLOSURE(I) строится по следующим правилам:

1. Включить в CLOSURE(I) все ситуации из I.

2. Если ситуация A → α •Bβ уже включена в CLOSURE(I) и B → γ - правило грамматики, то добавить в множество CLOSURE(I) ситуацию B → • γ при условии, что там ее еще нет.

3. Повторять правило 2, до тех пор, пока в CLOSURE(I) нельзя будет включить новую ситуацию.

В данном случае CLOSURE(I) = I = {(S’ -> · S, $), (S’ -> S · , $), (S -> · F+L, $), (S -> F · +L, $), (S -> F+ · L, $), (S -> F+L · , $), (F -> · (L*), +), (L -> · F, $), (L -> F · , $), (F -> (· L*), +), (F -> (L · *), +), F -> (L* ·), +), F -> (L*) · , +), (F -> · i, +), (F -> i · , +)}

Например, CLOSURE({S’ -> · S}) = {S’ -> · S, S -> · F+L, F -> · i,F -> · (L*)}

13. Определить функцию перехода g(x)

13.1. Определить функцию перехода goto(I,x)

Функция GOTO(I,X) определяется как замыкание множества всех ситуаций [A→ αX·β], таких что [A→ α·Xβ] ∈ I

GOTO(I,S) = {S’ -> S · }

GOTO(I,i) = {F -> i · }

GOTO(I,+) = {S -> F+ · L, L -> · F, F -> · (L*),F -> · i}

GOTO(I,() = {F -> (· L*), L -> · F, F -> · (L*),F -> · i}

GOTO(I,)) = {F -> (L*) · }

GOTO(I,F) = {S -> F · +, L -> F · , F -> · (L*),F -> · i}

GOTO(I,L) = {F -> (L · *)}

Другой пример:

Пусть для грамматики G_0 некоторое множество I = {[S' → S•], [S → F•&L]}.

GOTO(I, &) = {[S → F&•L], [L→ •F], [F→ •i], [F→•*L]}.

14. Построить каноническую форму множества ситуаций.

14.1. Построить каноническую форму множества ситуаций

Процесс построения канонической системы множеств LR(0) –ситуаций можно описать с помощью следующих действий:

1. $\varphi = \varnothing$

2. Включить в φ множество A0 = CLOSURE([S'→ • S]), которое в начале «не отмечено».

3. Если множество ситуаций A, входящее в систему , «не отмечено», то:

отметить множество A;
вычислить для каждого символа X ∈ (V ∪ Σ) значение A' = GOTO(A,X);

если множество $A' \neq \varnothing$ и еще не включено в φ, то включить его в систему
множеств как «неотмеченное» множество.

4. Повторять шаг 3, пока все множества ситуаций системы φ не будут отмечены.

Построение:

A_0 = CLOSURE(|S’ -> ·S|)

A_1 = GOTO(A_0, S) = CLOSURE(|S’ -> S·|) = {S’ -> S·}

A_2 = GOTO(A_0, F) = {S -> F·+L}

A_3 = GOTO(A_0, i) = {F -> i·}

A_4 = GOTO(A_0, () = {F -> (·L*), L -> ·F, F -> ·i, F -> ·(L*)}

Для всех X ∈ (V ∪ Σ) множества GOTO(A_1, X) пусты.

A_5 = GOTO(A_2, +) = {S -> F+·L, L -> ·F, F -> ·i, F -> ·(L*)}

GOTO(A_3, X) = ∅

GOTO(A_4, i) = A_3

A_6 = GOTO(A_4, L) = {F -> (L·*)}

A_7 = GOTO(A_4, F) = {L -> F·}

GOTO(A_4, () = A_4

A_8 = GOTO(A_5, L) = {S -> F+L·}

A_9 = GOTO(A_5, F) = {S -> F·}

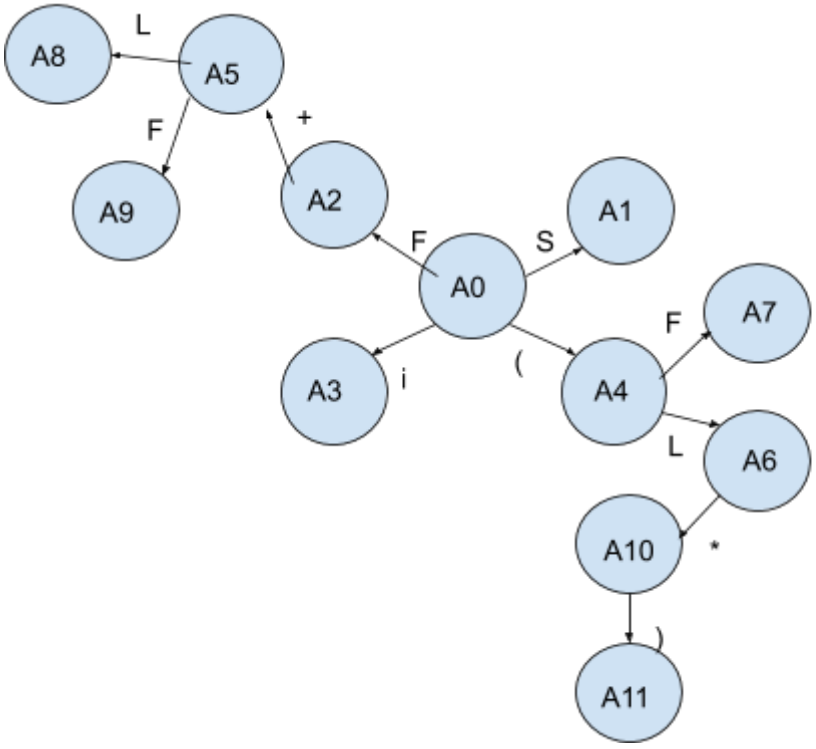
A_{10} = GOTO(A_6, *) = {F -> (L*·)}

A_{11} = GOTO(A_{10},)) = {F -> (L*)·}

A_0	S’ -> ·S S -> ·F+L F -> ·i F -> ·(L*)
A_1	S’ -> S·

A_2	S -> F · +L
A_3	F -> i ·
A_4	F -> (· L*) L -> · F F -> · i F -> · (L*)
A_5	S -> F+ · L L -> · F F -> · i F -> · (L*)
A_6	F -> (L · *)
A_7	L -> F ·
A_8	S -> F+L ·
A_9	S -> F ·
A_{10}	F -> (L* ·)
A_{11}	F -> (L*) ·

14.2. Построить диаграмму переходов автомата



Шаг 3. Построение управляющей таблицы.

15. Построить управляющую таблицу для функции перехода g(x) и действий f(u).

1. Если $f(\alpha, aw) = ПЕРЕНОС$, то входной символ переносится в верхушку магазина и читающая головка сдвигается на один символ вправо. В терминах конфигураций этот процесс описывается так:
   $(\alpha, aw, \pi) \vdash^s (\alpha a, w, \pi)$ для $\alpha \in (N \cup \Sigma \cup \{\perp\})^*$, $w \in (\Sigma \cup \{\varepsilon\})^*$ и $\pi \in \{1, \ldots, p\}^*$.
2. Если $f(\alpha\beta, w) = СВЕРТКА$, $g(\alpha\beta, w) = i$ и $A \rightarrow \beta$ — правило грамматики с номером i, то цепочка β заменяется правой частью правила с номером i, а его номер помещается на выходную ленту, т. е. $(\alpha\beta, w, \pi) \vdash^r (\alpha A, w, \pi i)$.
3. Если $f(\alpha, w) = ДОПУСК$, то $(\alpha, w, \pi) \vdash^s ДОПУСК$.

	i	+	*	(	)	S	F	L	$
0	П(3)			П(4)		1	2		
1									допуск
2		П(5)							

3	C(2)	C(2)	C(2)	C(2)	C(2)				C(2)
4							7	6	
5							9	8	
6			П(10)						
7	C(4)	C(4)	C(4)	C(4)	C(4)				
8	C(1)	C(1)	C(1)	C(1)	C(1)				C(1)
9	C(4)	C(4)	C(4)	C(4)	C(4)				C(4)
10					П(11)				
11	C(3)	C(3)	C(3)	C(3)	C(3)				C(3)

16. Реализовать LR(1)-анализатор по управляющей таблице (g,f) для LR(1) грамматики.

Реализация представлена в выданном преподавателем фреймворке.

Шаг 4. Применение алгоритма «перенос-свёртка».

Работа алгоритма описывается в терминах конфигураций, представляющих собой тройки вида (αT, ax, π), где αT – цепочка магазинных символов (T- верхний символ магазина), ax – необработанная часть входной цепочки, π – выход, построенный к настоящему моменту времени.

Рассмотрим последовательность тактов алгоритма при анализе входной цепочки i+((i*)*)

(⊥0, i+((i*)*), ε) |- (⊥0i3, +((i*)*), ε) |- (⊥0F2, +((i*)*), 2) |- (⊥0F2+5, ((i*)*), 2) |- (⊥0F2+5(4, (i*)*), 2) |- (⊥0F2+5(4(4, i*)*), 2) |- (⊥0F2+5(4(4i3, *)*), 2) |- (⊥0F2+5(4(4F7, *)*), 22) |- (⊥0F2+5(4(4L6, *)*), 224) |- (⊥0F2+5(4(4L6*10,)*), 224) |- (⊥0F2+5(4(4L6*10)11, *), 224) |- (⊥0F2+5(4F7, *), 2243) |- (⊥0F2+5(4L6, *), 22434) |- (⊥0F2+5(4L6*10,), 22434) |- (⊥0F2+5(4L6*10)11, $, 22434) |- (⊥0F2+5F7, $, 224343) |- (⊥0F2+5L8, $, 2243434) |- (⊥0S1, $, 22434341) |- ДОПУСК