МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(национальный исследовательский университет)

Институт №8 «Компьютерные науки и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

Отчёт по лабораторным работам

по дисциплине «Системы программирования»

Выполнил:
Студент группы М8О-201Б-21
Голов Александр Юрьевич

Принял:
Доцент кафедры 806
Киндинова Виктория Валерьевна

Оценка:_____ **Дата:**_____

Москва, 2023

Содержание

1 Практическая работа №1 **2**
1.1 Построение автоматной грамматики 2
1.2 Построение конечного автомата 3

2 Практическая работа №2 **5**
2.1 Устранение ε-правил, бесполезных символов и цепных правил из грамматики 5
2.2 Построение МП-автомата 7
2.3 Реализация МП-автомата 7

3 Построение $LL(K)$-анализатора **8**
3.1 Реализация LL(K)-анализатора 9

1 Практическая работа №1

1.1 Построение автоматной грамматики

Формулировка задания

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

Заданный регулярный язык:

$$pattern = 192.168.1.d(1,\ 3)$$

Автоматная грамматика:

$$\begin{gathered} L(pattern) = L(\text{"}192.168.1.d(1,\ 3)\text{"}) = \\ = L(192.168.1.(0-9), \\ 192.168.1.(0-9)^2, \\ 192.168.1.(0-9)^3) \\ G(T,\ V,\ P,\ S_0) = G((192.168.1., 0, 1, 2, 3, 4, 5, 6, 7, 8, 9), \\ (S_0, A, B, C, D), \\ (p_1, p_2, p_3, p_4, p_5), S_0) \end{gathered}$$

Правила регулярной грамматики:

$$\begin{gathered} p_1:\ S_0 \to 192.168.1.\boldsymbol{A} \\ p_2:\ A \to 0\boldsymbol{B} \mid 1\boldsymbol{B} \mid \cdots \mid 9\boldsymbol{B} \\ p_3:\ B \to 0\boldsymbol{C} \mid 1\boldsymbol{C} \mid \cdots \mid 9\boldsymbol{C} \\ p_4:\ B \to 0\boldsymbol{D} \mid 1\boldsymbol{D} \mid \cdots \mid 9\boldsymbol{D} \end{gathered}$$

Пример цепочки:

$$\begin{gathered} S_0 \to^1 192.168.1.A \to^2 192.168.1.1B \to^3 192.168.1.12C \to^4 \\ \to^4 192.168.1.123D \to^5 192.168.1.123 \end{gathered}$$

1.2 Построение конечного автомата

$$
\begin{gathered}
L(KA) = L(G) \\
KA = (Q,\, \Sigma,\, \delta,\, S_0,\, F), \\
Q = (S_0, A, B, C, q_f), \\
\Sigma = (0-9, 192.168.1.), \\
S_0 = S_0, \\
F = q_f. \\
\delta = (\\
\delta_1(S_0, 192.168.1.) = (A), \\
\delta_2(A, 0) = (B), \\
\vdots \\
\delta_1 1(A, 9) = (B), \\
\delta_1 2(B, 0) = (C), \\
\vdots \\
\delta_2 1(B, 9) = (C), \\
\delta_2 2(B, \varepsilon) = (q_f), \\
\delta_2 3(C, 0) = (D), \\
\vdots \\
\delta_2 1(C, 9) = (D), \\
\delta_2 1(C, \varepsilon) = (q_f), \\
\delta_2 1(D, \varepsilon) = (q_f),)
\end{gathered}
$$

Диаграмма конечного автомата:

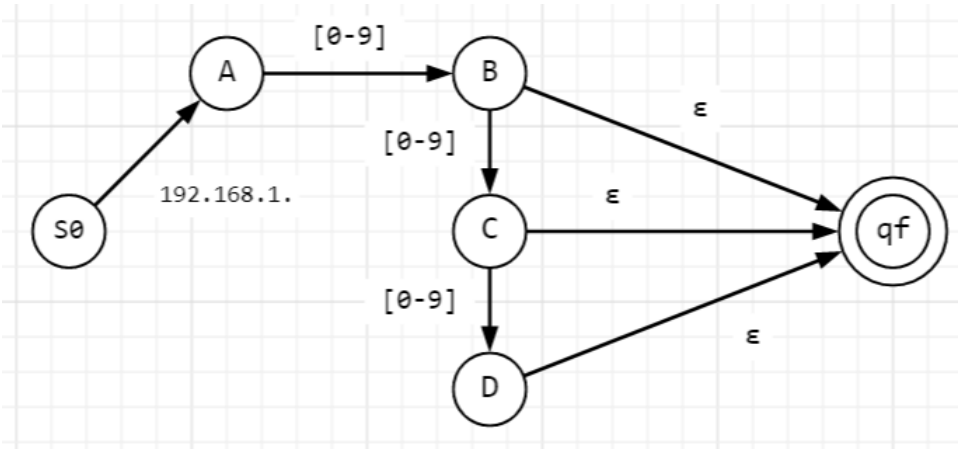

Рисунок 1 - Диаграмма конечного автомата

Листинг программы:

```
var ka1 = new FSAutomate(new List<Symbol>()
        { "S0", "A", "B", "C", "D", "qf" },
        new List<Symbol>() { "", "0", "1", "2",
                "3", "4", "5", "6", "7", "8", "9",
        "192.168.1." },
        new List<Symbol>() { "qf" },
        "S0");

ka1.AddRule("S0", "192.168.1.", "A");

for (int i = 0; i < 10; i++)
        ka1.AddRule("A", i.ToString(), "B");

for (int i = 0; i < 10; i++)
        ka1.AddRule("B", i.ToString(), "C");
        ka1.AddRule("B", "", "qf");

        for (int i = 0; i < 10; i++)
        ka1.AddRule("C", i.ToString(), "D");
        ka1.AddRule("C", "", "qf");
        ka1.AddRule("D", "", "qf");
```

2 Практическая работа №2

Заданная грамматика:

$$
\begin{gathered}
G = \{(a, b, c, d, f, \varepsilon, z, r, t, g), \\
(S, A, B, C, D, R, T, H, G, V), \\
(A \to bC, \\
B \to cA, \\
C \to dB, \\
D \to R, \\
R \to f, \\
S \to aABT, \\
T \to \varepsilon, \\
H \to z, \\
G \to r, \\
S \to gV, \\
V \to Vt, \\
V \to t \\
S\}
\end{gathered}
$$

2.1 Устранение ε-правил, бесполезных символов и цепных правил из грамматики

В данном случае очевидно, что ε-правило имеет вид $T \to \varepsilon$, удалим его. Получилось так, что нетерминал T стал непроизводящим, удалим его из правой части правила $S \to aABT$ и получим $S \to aAB$. Также зметим наличие цепного правила

$$
\begin{gathered}
D \to R, \\
R \to f
\end{gathered}
$$

Преобразуем эти правила следующим образом:

$$D \to f$$

Таким образом, получим следующие правила грамматики:

$$A \to bC,$$
$$B \to cA,$$
$$C \to dB,$$
$$D \to f,$$
$$S \to aAB,$$
$$S \to gV.$$
$$V \to t,$$
$$V \to tV',$$
$$V' \to t,$$
$$V' \to tV'$$

2.2 Построение МП-автомата

$$\text{МП} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F),$$
$$Q = \{q\},$$
$$\Sigma = T,$$
$$\Gamma = T \cup V.$$
$$q_0 = q_0,$$
$$z_0 = S_0,$$
$$F = q.$$
$$\delta = \{\delta(q, \varepsilon, A) = (q, bC),$$
$$\delta(q, \varepsilon, B) = (q, cA),$$
$$\delta(q, \varepsilon, C) = (q, dB),$$
$$\delta(q, \varepsilon, D) = (q, f),$$
$$\delta(q, \varepsilon, R) = (q, f),$$
$$\delta(q_0, \varepsilon, S) = (q, aAB),$$
$$\delta(q_0, \varepsilon, S) = (q, gv),$$
$$\delta(q, \varepsilon, V) = (q, t),$$
$$\delta(q, \varepsilon, V) = (q, tV'),$$
$$\delta(q, \varepsilon, V') = (q, t),$$
$$\delta(q, \varepsilon, V') = (q, tV;),$$
$$\delta(q, a, a) = (q, \varepsilon),\ \forall a \in \Sigma\}$$

2.3 Реализация МП-автомата

```
var regGr = new Grammar(new List<Symbol>() { "a", "b", "c", "d", "f",
new List<Symbol>() { "S", "A", "B", "C", "D", "R", "T", "V", "H", "G"
"S");
regGr.AddRule("A", new List<Symbol>() { "b", "C"});
regGr.AddRule("B", new List<Symbol>() { "c", "A" });
regGr.AddRule("C", new List<Symbol>(RST$ & $M$ \\) { "d", "D" });
regGr.AddRule("D", new List<Symbol>() { "R" });
regGr.AddRule("R", new List<Symbol>() { "f" });
regGr.AddRule("S", new List<Symbol>() { "a", "A", "B", "T" });
```

```
regGr.AddRule("T", new List<Symbol>() { "" });
regGr.AddRule("H", new List<Symbol>() { "z" });
regGr.AddRule("G", new List<Symbol>() { "r" });
regGr.AddRule("S", new List<Symbol>() { "g", "V" });
regGr.AddRule("V", new List<Symbol>() { "V", "t" });
regGr.AddRule("V", new List<Symbol>() { "t" });

Console.WriteLine("Grammar:");
regGr.Debug("T", regGr.T);
regGr.Debug("T", regGr.V);
regGr.DebugPrules();
Grammar G1 = regGr.EpsDelete();
G1.DebugPrules();
Grammar G2 = G1.unUsefulDelete();
G2.DebugPrules();
Grammar G3 = G2.ChainRuleDelete();
G3.DebugPrules();
Grammar G4 = G3.LeftRecursDelete_new6();
G4.DebugPrules();
// G4 - приведеннаяграмматика
Console.WriteLine("----------------------------------------");
Console.WriteLine("Normal Grammatic:");
G4.Debug("T", G4.T);
G4.Debug("V", G4.V);
G4.DebugPrules();
Console.Write("Start symbol: ");
Console.WriteLine(G4.S0 + "\n");
```

3 Построение $LL(K)$-анализатора

КС-грамматика $G = (T, V, P, S)$ без ε-правил называется простой $LL(1)$ грамматикой (s- грамматикой, разделенной грамматикой), если для каждого $v \in V$ 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.

3.1 Удаление правого ветвоения из грамматики

Применив алгоритм факторизации для устранения правого ветвления к исходной грамматике, получим следующий набор правил:

$$A \to bC,$$
$$B \to cA,$$
$$C \to dB,$$
$$D \to f,$$
$$S \to aAB,$$
$$S \to gV.$$
$$V \to tT,$$
$$V' \to tT',$$
$$T \to V',$$
$$T \to \varepsilon$$

Приведём аналитическое представление управляющей таблицы M:

Таблица 1 - Управляющая таблица анализатора

$A \to bC$	$FIRST(A) = b$	$M(A, b) = (bC, 1)$
$B \to cA$	$FIRST(B) = c$	$M(B, c) = (cA, 2)$
$C \to dB$	$FIRST(C) = d$	$M(C, d) = (db, 3)$
$D \to f$	$FIRST(D) = d$	$M(D, f) = (f, 4)$
$S \to aAB$	$FIRST(S) = a$	$M(S, a) = (aAb, 5)$
$S \to gV$	$FIRST(S) = g$	$M(S, g) = (V, 7)$
$V \to tT$	$FIRST(V) = g$	$M(V, t) = (tT, 8)$
$V' \to tT'$	$FIRST(V) = t$	$M(V', t) = (tT, 9)$
$T \to V'$	$FIRST(T) = t$	$M(T, t) = (V', 10)$
$T \to \varepsilon$	$FIRST(T) = \varepsilon$	$M(T, \varepsilon) = (\varepsilon, 11)$

3.2 Реализация LL(K)-анализатора

```
var LL = new Grammar(new List<Symbol>()
{ "a", "b", "c", "d", "f", "t", "g", "␣" },
new List<Symbol>()
{ "S", "A", "B", "C", "D", "R", "V", "V'", "T" },
```

```
”S”);
LL.AddRule(”A”, new List<Symbol>() { ”b”, ”C” });
LL.AddRule(”B”, new List<Symbol>() { ”c”, ”A” });
LL.AddRule(”C”, new List<Symbol>() { ”d”, ”D” });

LL.AddRule(”D”, new List<Symbol>() { ”f” });
LL.AddRule(”R”, new List<Symbol>() { ”f” });
LL.AddRule(”S”, new List<Symbol>() { ”a”, ”A”, ”B” });
LL.AddRule(”S”, new List<Symbol>() { ”g”, ”V” });
LL.AddRule(”V”, new List<Symbol>() { ”t”, ”T” });
LL.AddRule(”V’”, new List<Symbol>() { ”t”, ”T” });
LL.AddRule(”T”, new List<Symbol>() { ”V’” });
LL.AddRule(”T”, new List<Symbol>() { ”␣” });
var parser = new LLParser(LL);
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).”);
}
```