МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное агенство по образованию

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики-процессов управления

Трефилов Андрей Романович

Курсовая работа

Разработка и построение конечного автомата для цепочек заданного регулярного языка

2 курс, 232 группа

Руководитель курсовой работы
А.В. Матросов
2016 г.

Санкт-Петербург, 2016 г.

1 Постановка задачи

1. Разработать распознающий КА для цепочек заданного регулярного языка и использовать его в программе поиска цепочек этого языка во входной последовательности символов.

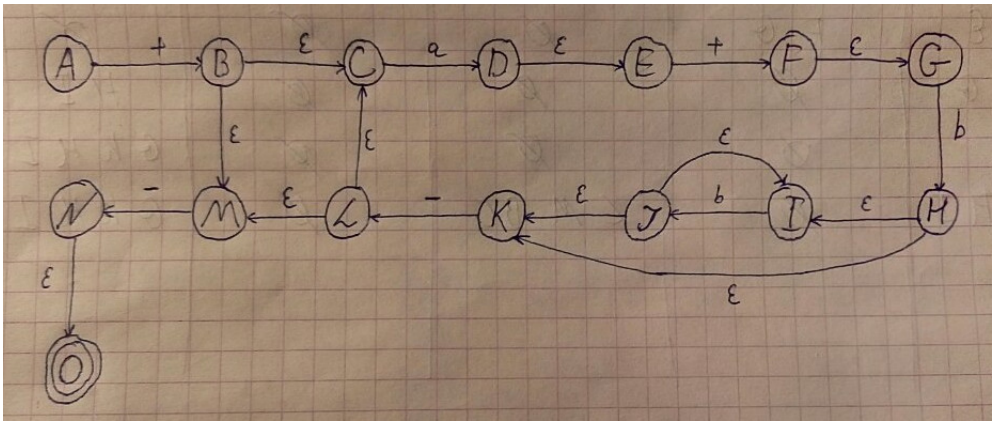
2. Построить таблицу переходов распознающего КА.

Строка символов $a, b, +, -$, начинающаяся с префикса $+$ и заканчивающаяся суффиксом $-$, между которыми располагается последовательность символов a, b, в которой после каждого символа a следует последовательность символов b, заключенная в символы $+$ и $-$.

2 Разработка автомата

По поставленной задаче составлено регулярное выражение: $+(a + b(b)^*-)^*-$.

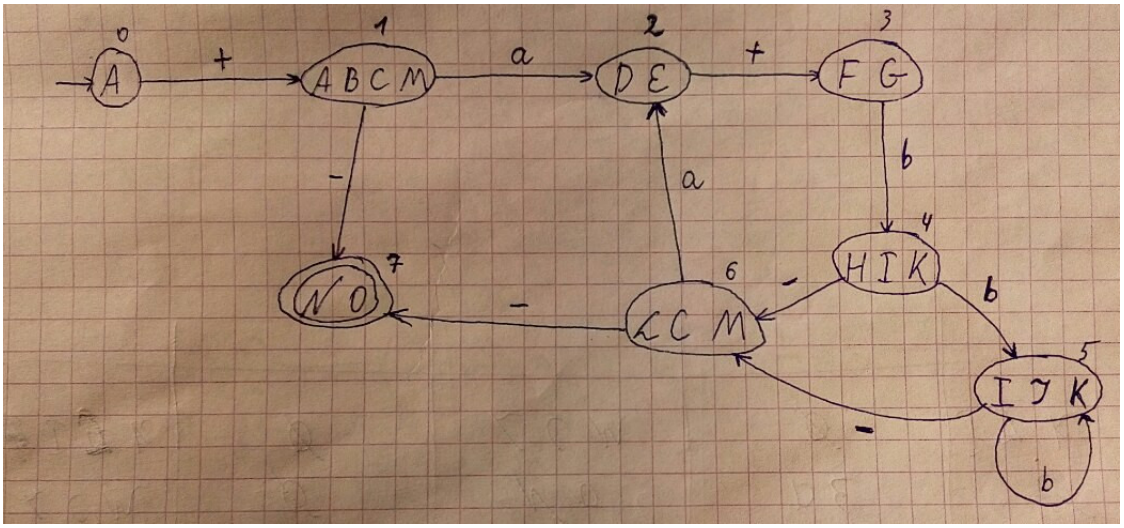
Разработан ε-НКА:

Рис. 2.1: 

Построена его таблица переходов:

	$+$	$-$	a	b
→A (0)	ABCM (1)	∅	∅	∅
ABCM (1)	∅	NO (7)	DE (2)	∅
NO* (7)	∅	∅	∅	∅
DE (2)	FG (3)	∅	∅	∅
FG (3)	∅	∅	∅	HIK (4)
HIK (4)	∅	LCM (6)	∅	IJK (5)
LCM (6)	∅	NO (7)	DE (2)	∅
IJK (5)	∅	LCM (6)	∅	IJK (5)

И получен следующий граф:

Рис. 2.2: 

3 Код программы

Язык программирования — python версии 3.5

```
step = 0 #состояния конечного автомата (см. граф выше)
line = input('Input string: ')
out = ''
for symbol in line: #проход по каждому символу из введеной строки
    if step == 0 and symbol == '+' :
        step = 1
        out += symbol
    elif step == 1 and symbol == '-':
        step = 7
        out += symbol
    elif step == 1 and symbol == 'a':
        step = 2
        out += symbol
    elif step == 2 and symbol == '+':
        step = 3
        out += symbol
    elif step == 3 and symbol == 'b':
        step = 4
        out += symbol
   elif step == 4 and symbol == 'b':
        step = 5
        out += symbol
   elif step == 4 and symbol == '-':
        step = 6
        out += symbol
   elif step == 5 and symbol == 'b':
        step = 5
        out += symbol
```

```
elif step == 5 and symbol == '-':
     step = 6
     out += symbol
elif step == 6 and symbol == 'a':
     step = 2
     out += symbol
elif step == 6 and symbol == '-':
     step = 7
     out += symbol
else:
     step = 0 #ожидаем начала новой потенциально верной цепочки
     out = '' #сбрасываем предыдущую строку, т.к. она не удовл. КА

if step == 7: #Если достигли конечного состояния
     print(out) #Вывести строку и ожидать новой
     step = 0
     out = ''
```

4 Описание работы программы

Программа представляет собой наивную реализацию переходов конечного автомата в различные состояния. Переменная *step* является хранителем состояния. При обработке каждого нового символа из введеной строки программа последовательно проверяет соответствие текущего положения и возможности перехода в другое состояние по текущему символу. Если это возможно, происходит переход, в противном случае состояние сбрасывается на начальное. При запуске программы отображается приглашение ко вводу, после которого необходимо ввести строку, в которой будет производиться поиск подстроки, заданной регулярным выражением. Если подстрока будет найдена, она будет выведена в стандартный вывод, в случае нахождения нескольких подстрок они будут выведены последовательно, каждая с новой строки. Если ни одной подстроки найдено не будет, программа завершит свою работу без вывода какой-либо информации.

5 Тестирование

Тестовая строка	Вывод программы
$+-$	$+-$
$++++---$	$+-$
$+a+b-a+bb--$	$+a+b-a+bb--$
$--a+a+b--$	$+a+b--$
$ab+a+bbbb--a+b+a+b--ab$	$+a+bbbb--$ $+a+b--$
$+$	
$+a+b+a+bb-$	
uncorrect	
$-------+$	
$+b+a+bb-+$	