Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Кафедра информатики и прикладной математики

Формальные языки и грамматики

Домашнее задание 2

“Построение программы-распознавателя по регулярному выражению”

Вариант 19

Старался: **Шкаруба Н.Е.**

Проверил: **Лаздин А.В.**

Группа **P3218**

2016г

Требования:

По регулярному выражению: (a*|b)*|(b|c)*

- Построить недетерминированный КА
- По полученному НДА построить ДКА
- Для ДКА написать программу-распознаватель предложений языка, порождаемого регулярным выражением.

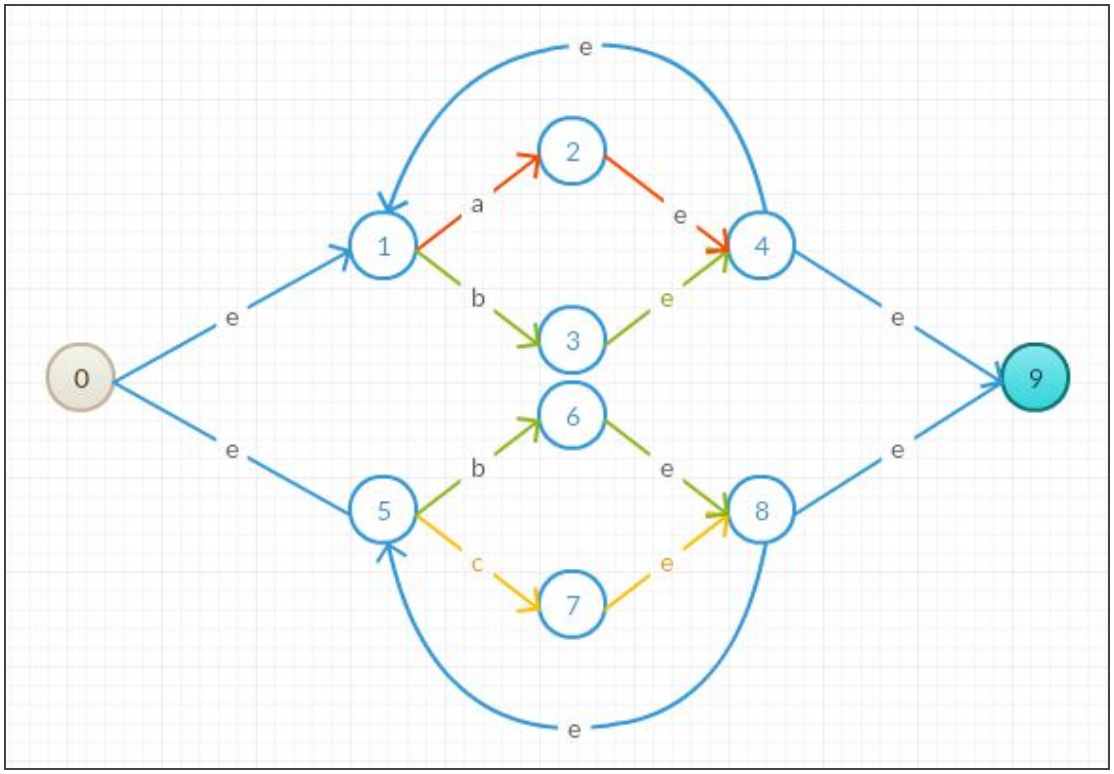
Продемонстрировать работу распознавателя на различных примерах (не менее трех правильных) предложений.

Выполнение:

Для выполнения работы необходимо совершить следующие преобразования:
РВ -> НКА -> ДКА -> Программа

Регулярное выражение:
(a*|b)*|(b|c)* = (a|b)*|(b|c)*

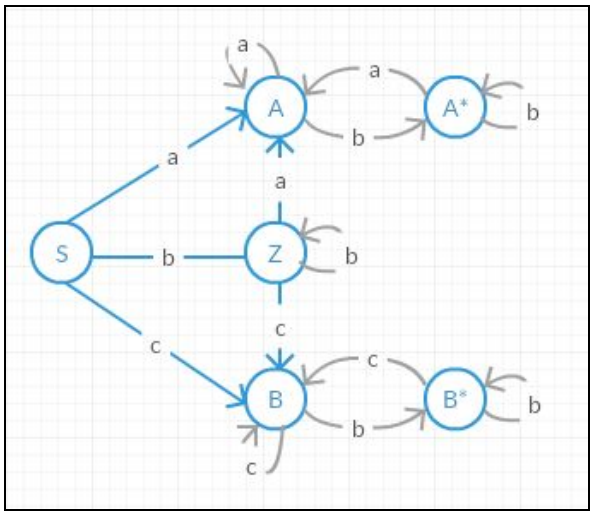
Недетерминированный конечный автомат:



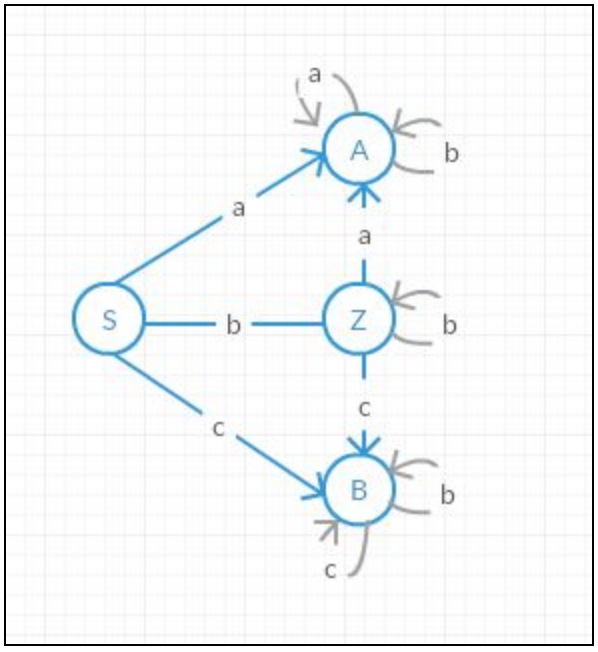
Его таблица состояний:

#	Состояние	a	b	c
S	0, 1, 5	2, 4, 1, **9**	3, 4, 1, 8, 6, 5, **9**	7, 8, 5, **9**
A	2, 4, 1, **9**	2, 4, 1, **9**	3, 4, 1, **9**	x
Z	3, 4, 1, 8, 6, 5, **9**	2, 4, 1, **9**	3, 4, 1, 8, 6, 5, **9**	7, 8, 5, **9**
B	7, 8, 5, **9**	x	6, 8, 5, **9**	7, 8, 5, **9**
A*	3, 4, 1, **9**	2, 4, 1, **9**	3, 4, 1, **9**	x
B*	6, 8, 5, **9**	x	6, 8, 5, **9**	7, 8, 5, **9**

Детерминированный конечный автомат:



Упрощённый детерминированный конечный автомат:



Код

```
class State
  def initialize(name, isFinal)
     @name = name
     @transitions = []
     @isFinal = isFinal
  end

  def addTransition(char, destination)
     @transitions.push([char, destination])
  end

  def addTransitions(transitions)
     transitions.each do |transition|
       addTransition(transition[0], transition[1])
     end
  end

  def getState(char)
    @transitions.each { |pair|
      if pair[0] == char
        return pair[1]
      end
    }
    nil
  end

  def isFinal
     @isFinal
   end
 end
```

```
class Regex
  def initialize
     @s = State.new('S', false)
     @z = State.new('Z', true)
     @a = State.new('A', true)
     @b = State.new('B', true)

     @s.addTransitions([['a', @a], ['b', @z], ['c', @b]])
     @z.addTransitions([['a', @a], ['b', @z], ['c', @b]])
     @a.addTransitions([['a', @a], ['b', @a]])
     @b.addTransitions([['b', @b], ['c', @b]])
  end

  def match(string)
     currentState = @s
```

```
    string.each_char { |nextChar|
      currentState = currentState.getState nextChar

      if currentState == nil
        return false
      end
    }

    currentState.isFinal
  end
end
```

```
#!/usr/bin/env ruby

regex = Regex.new

if ARGV.count == 0
 tests = %w(aaaa aaaab bbbb bbbba bbbbc cccc ccccb swim aaaaswim ItWillCurshYou
ComeOnProgramCrush!)
else
 tests = ARGV
end

tests.each { |test|
 puts "Is #{test} valid? : " + regex.match(test).to_s
}
```

Результат работы программы:

```
nikita@Pluto:~/Code/ITMO/Course 2/Formal_languages/Lab2(Regex_recognizer)/src(Formal_languages-develop)$ ./regex.rb
Is aaaa valid? : true
Is aaaab valid? : true
Is bbbb valid? : true
Is bbbba valid? : true
Is bbbbc valid? : true
Is cccc valid? : true
Is ccccb valid? : true
Is swim valid? : false
Is aaaaswim valid? : false
Is ItWillCurshYou valid? : false
Is ComeOnProgramCrush! valid? : false
```

```
nikita@Pluto:~/Code/ITMO/Course 2/Formal_languages/Lab2(Regex_recognizer)/src(Formal_languages-develop)$ ./regex.rb hello aababab bcbcb
cbcccb works perfectly !
Is hello valid? : false
Is aababab valid? : true
Is bcbcbcbcccb valid? : true
Is works valid? : false
Is perfectly valid? : false
Is ! valid? : false
```