Converting DFA to RegEx

Anescu Adelina & Cristina Haranguș

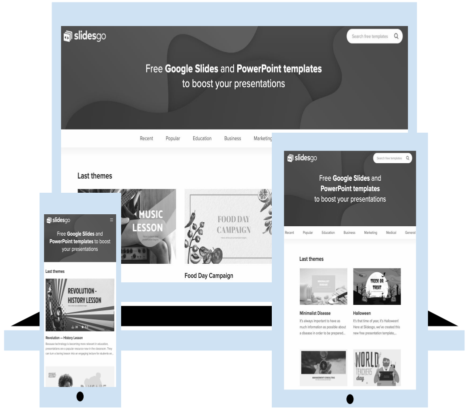


Table of contents

Motivation

We chose this project because we know that Regular Expressions are quite important and we wanted to find out more about this subject as it could be possible for us to work with them in the future (text recognition).

This project is important because we know that finite automata are very useful in the creation of compilers and interpreters techniques and regular expressions are important for pattern recognition.

*Note: to be honest, there weren’t many projects left when we wanted to pick one, so, it was more like the project chose us

DFA & RegEx

A *DFA* consists of:

- Q - a finite set state
- ∑ - a finite set of input symbols (alphabet)
- q0 - a start state (one of the elements from Q)
- F - set of accepting states
- $\delta : Q \times \Sigma \rightarrow Q$ - a transition function which takes a state and an input symbol as an argument and returns a state.

It is defined by the 5-tuple: {Q, ∑, q0,F, δ}

Regular expressions consist of constants, which denote sets of strings, and operator symbols, which denote operations over these sets.

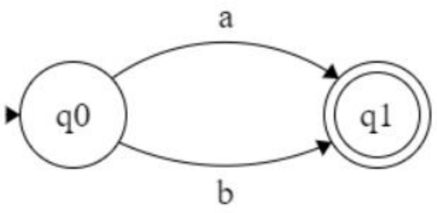
Precedence of operators (highest to lowest):

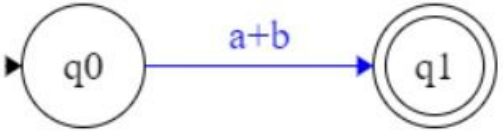
- * (star)
- .(concatenation)
- + (or)

Example: (0|1)*1 - strings of 0’s and 1’s ending with 1

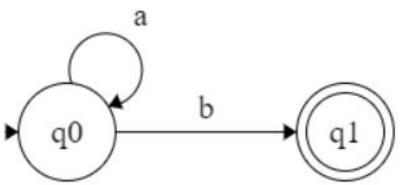
STATE ELIMINATION METHOD

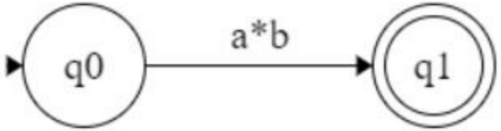
1





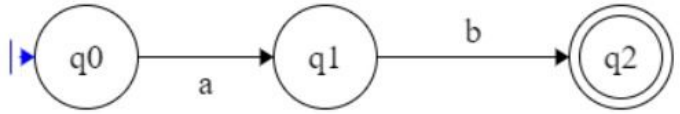
2

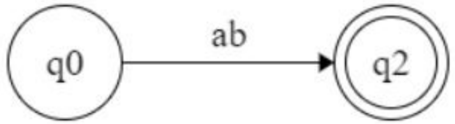




STATE ELIMINATION METHOD

3

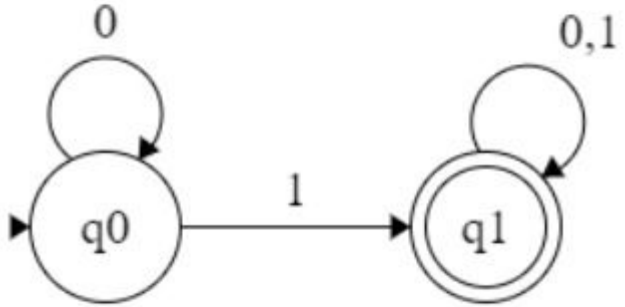


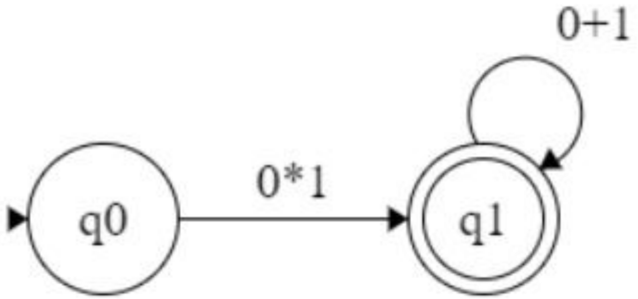


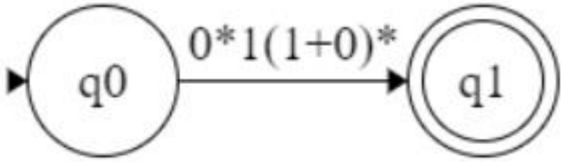
For the construction of the DFAs we used http://madebyevan.com/fsm/

First example

This is a simple DFA with one final state and no transition from the last state to the first state.





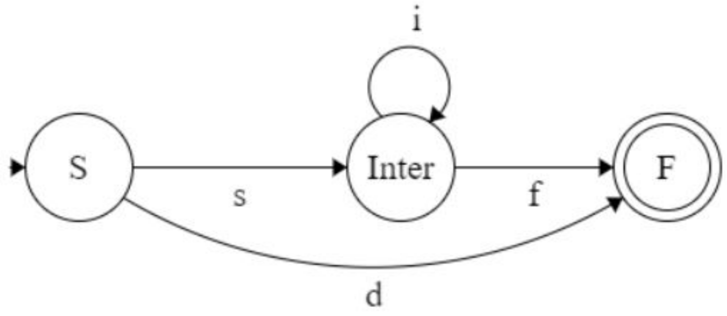


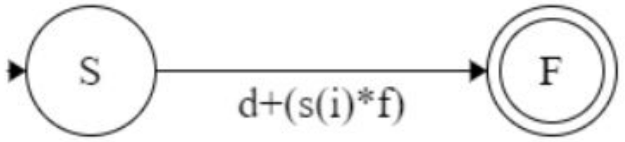
Important formulae to keep in mind!

In order to construct the REgEx from the DFA you need to use 2 formulas.

I

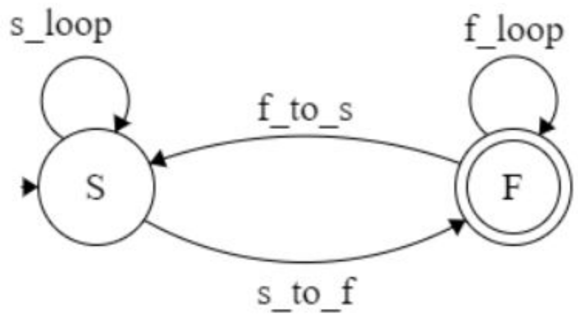
Formulae used for eliminating an intermediate state: [S]->[F]= d + (s(i)*f)

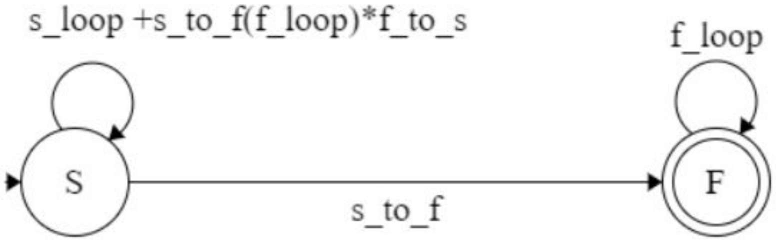


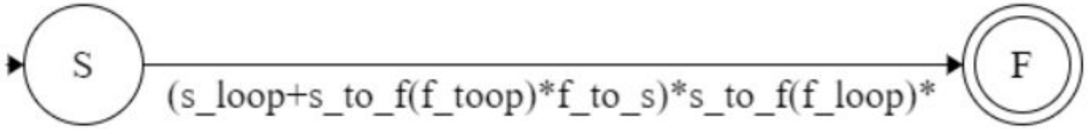


Important formulae to keep in mind!

2 Formulae used for the final RegEx when you have only the starting and final state left:







Important rules to keep in mind!

The simplification of RegularExpressions depends on two rules about how (empty set) 'ϕ' operates.

1. ϕR=Rϕ=ϕ Where ϕ is an annihilator for concatenation.

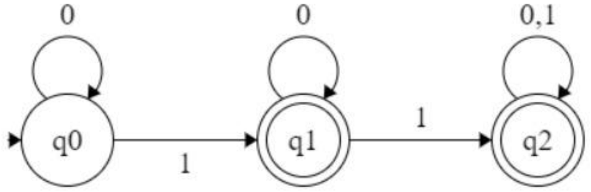
   Exemple: (0*1)+1ϕ0*   ->0*1

2. ϕ+R=R+ϕ=R Where ϕ is the identity for union.

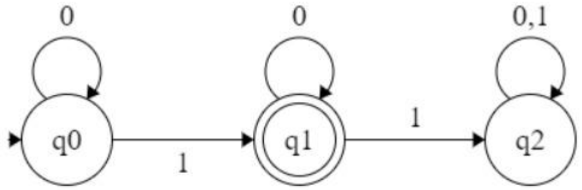
   Exemple: (0*(1+ϕ))+0 ->(0*1)+0

The second example

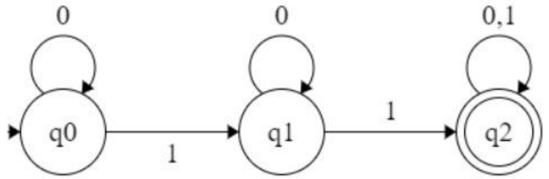
The DFA from this example has 2 final states. We're going to separate this in 2 DFA's and create the Regular expressions for each. The final result will be R.E.1+R.E.2



DFA2

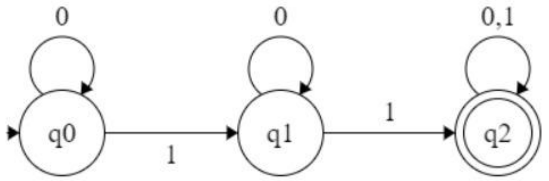


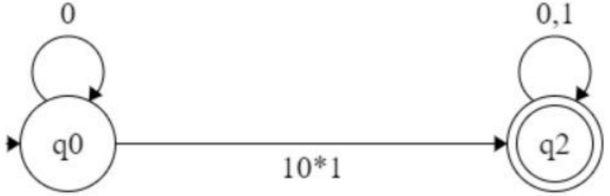
DFA1

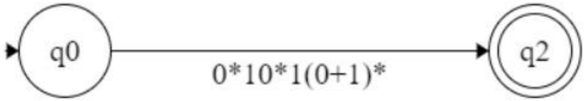


The second example - continued

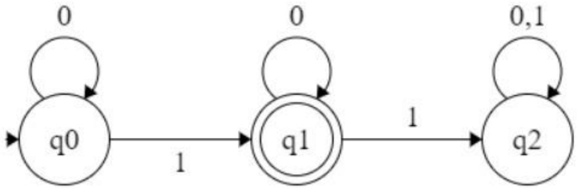
DFA2

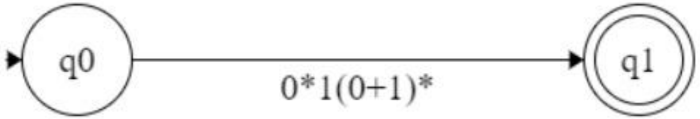






DFA1





The final result will be: DFA1 + DFA2
(0*1(0+1)*) + (0*10*1(0+1)*)

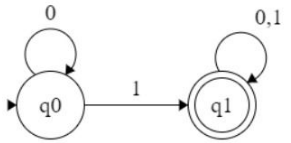
Implementation

1.We used Python

2.We created a class DFA that has states, an alphabet, starting_state, final_states, a states dictionary and also some methods/functions used for creating the RegEx.

3.We organized the table input with all the state transitions in a dictionary.

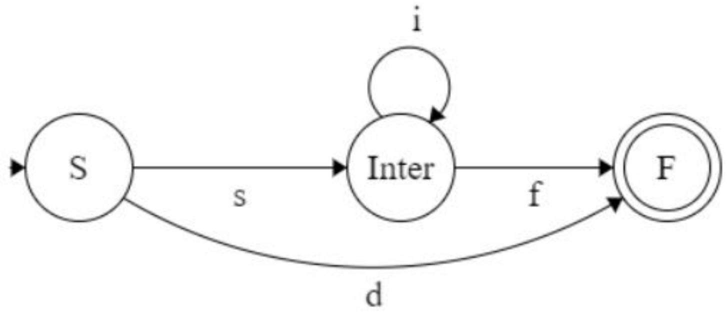
0	1
q0	q1
q1	q1



```
States Dictionary:  {'q0': {'q0': '0', 'q1': '1'},
                      'q1': {'q0': 'ϕ', 'q1': '0|1'}}
state_dict[q0][q1]=’1’
```

Deleting the intermediary states

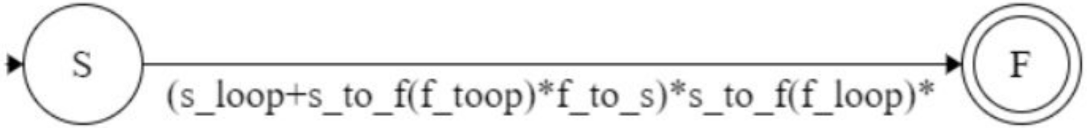
```
dict_states[i][j] = '(' + dict_states[i][j] + ')'+ '|'+
             '(''(' + dict_states[i][inter] + ')'+ '(' +
         dict_states[inter][inter] + ')'+'*',
              '(' + dict_states[inter][j] + ')'+')'
```



[S]->[F]= d + (s(i)*f)

Creating the Regular Expression

```
RegEx='(' + starting_loop  + '|'  + '(' + starting_to_final + ')' +
'(' + final_loop + ')' +
        '(' + final_to_starting + ')' + ')' + '' + '(' +
    starting_to_final + ')'+ '(' + final_loop + ')'+ ''
```



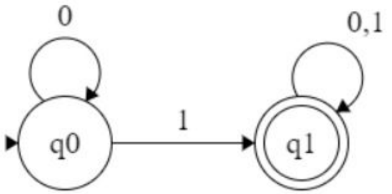
For a DFA with more final states

For a DFA that has more final states, we will use the formula:

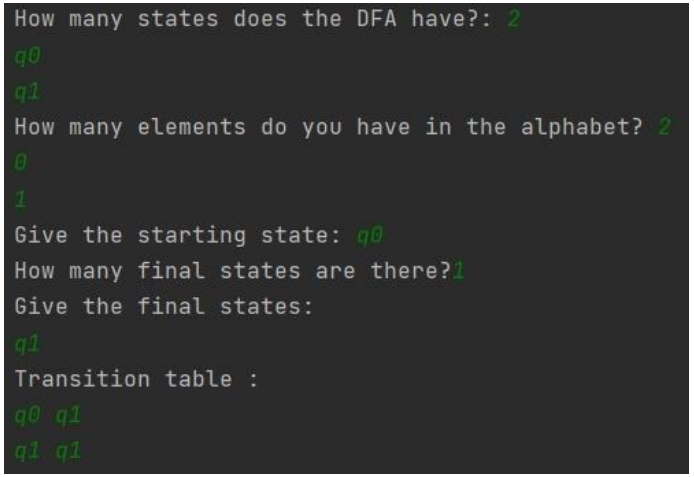
```
for f in final_states:
    dfa = DFA(states, alphabet, starting_state, [f],
transition_funct)
    r += '|' + dfa.toRegEx()
```

If dfa has more final_states then:

```
  {   1. create objects dfa=class DFA for each final state
      2. concatenate “ | dfa” to the RegEx
}
```



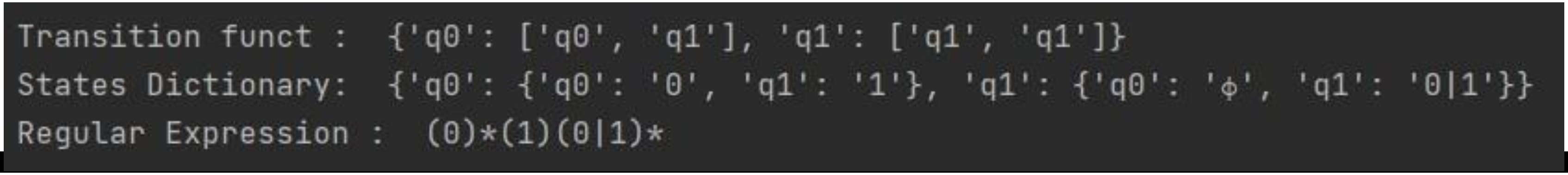
This is our DFA



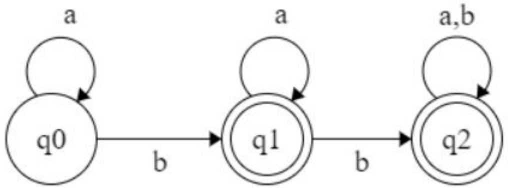
```
How many states does the DFA have?: 2
q0
q1
How many elements do you have in the alphabet? 2
0
1
Give the starting state: q0
How many final states are there?1
Give the final states:
q1
Transition table :
q0 q1
q1 q1
```

This is how the program gets the input we want from the DFA

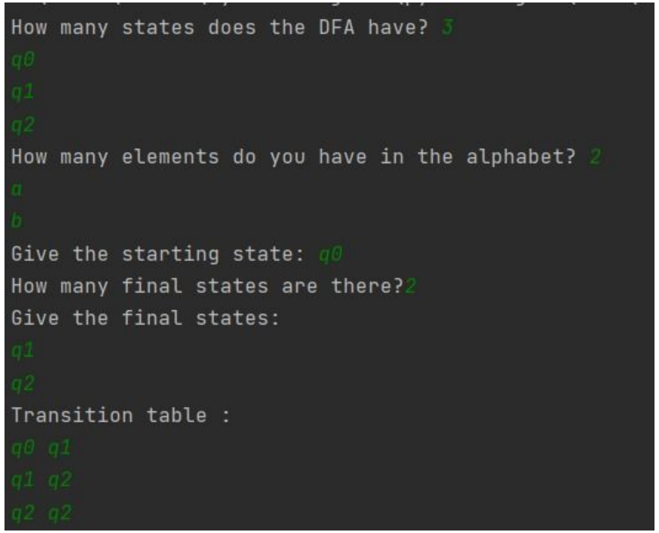
This is the output that we get. Besides the Regular expression we also added the dictionary representation of the transition table.



```
Transition funct :  {'q0': ['q0', 'q1'], 'q1': ['q1', 'q1']}
States Dictionary:  {'q0': {'q0': '0', 'q1': '1'}, 'q1': {'q0': 'ϕ', 'q1': '0|1'}}
Regular Expression :  (0)*(1)(0|1)*
```



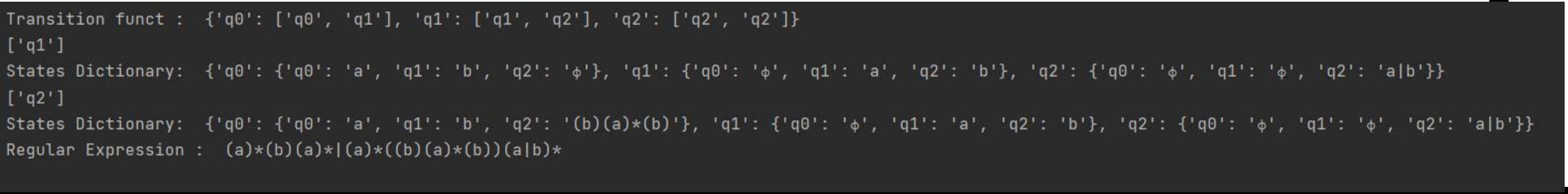
This is our second DFA



```
How many states does the DFA have? 3
q0
q1
q2
How many elements do you have in the alphabet? 2
a
b
Give the starting state: q0
How many final states are there?2
Give the final states:
q1
q2
Transition table :
q0 q1
q1 q2
q2 q2
```

This is how the program gets the input we want from this DFA

Since this is a bit more complicated than the other one, the regular expression is longer. This is the output of the DFA from the second example.



```
Transition funct :  {'q0': ['q0', 'q1'], 'q1': ['q1', 'q2'], 'q2': ['q2', 'q2']}
['q1']
States Dictionary:  {'q0': {'q0': 'a', 'q1': 'b', 'q2': 'ϕ'}, 'q1': {'q0': 'ϕ', 'q1': 'a', 'q2': 'b'}, 'q2': {'q0': 'ϕ', 'q1': 'ϕ', 'q2': 'a|b'}}
['q2']
States Dictionary:  {'q0': {'q0': 'a', 'q1': 'b', 'q2': '(b)(a)*(b)'}, 'q1': {'q0': 'ϕ', 'q1': 'a', 'q2': 'b'}, 'q2': {'q0': 'ϕ', 'q1': 'ϕ', 'q2': 'a|b'}}
Regular Expression :  (a)*(b)(a)*|(a)*((b)(a)*(b))(a|b)*
```

Bibliography

-Introduction to Automata Theory, Languages, and Computation - John E. Hopcroft (Author)

- Slides from the courses

-https://www.youtube.com/watch?v=fyJumUEITGY

- https://www.youtube.com/watch?v=57MwzSmBZpw&feature=youtu.be

- https://www.youtube.com/watch?v=4jZprdSQuWU

- https://www.gatevidyalay.com/dfa-to-regular-expression-examples-automata/

- https://github.com/vinyasns/dfa-to-regex

- http://madebyevan.com/fsm/