

# RegexParser Guide

*This tool aim to draw the final state machine related to the regular expression given as input.*

HOW TO USE THE TOOL.....	1
HOW TO WRITE A REGULAR EXPRESSION .....	2
POSSIBLE ERRORS.....	3
USE EXAMPLE .....	5
CREDITS .....	7



# HOW TO USE THE TOOL

The tool comes in the following form:

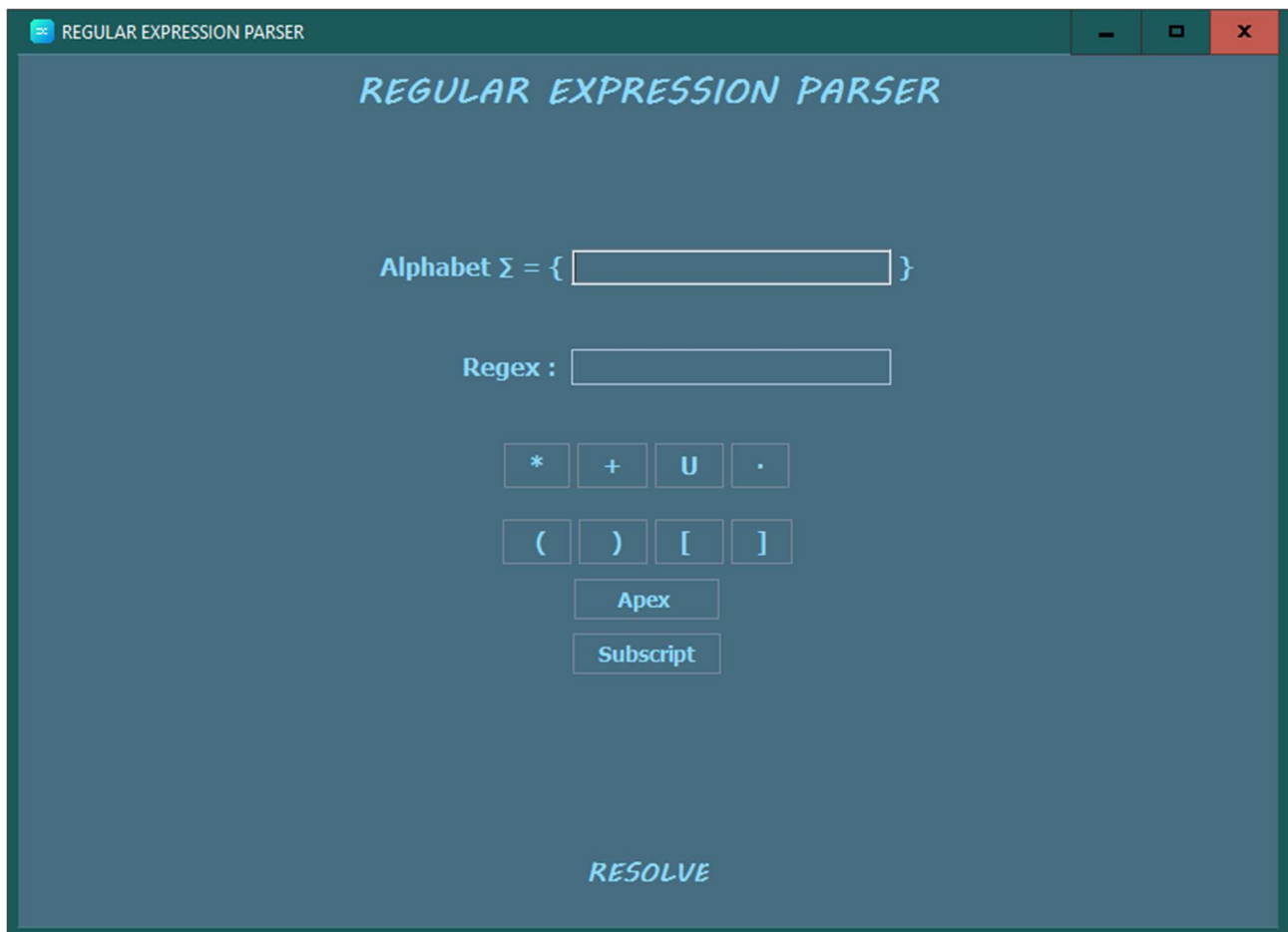
The image shows a web application window titled "REGULAR EXPRESSION PARSER". The interface has a dark teal background. At the top, there's a title bar with standard window controls. Below the title, the text "REGULAR EXPRESSION PARSER" is displayed in a light blue, stylized font. The main area contains two input fields: "Alphabet  $\Sigma$  = { " followed by a text input box, and "Regex : " followed by another text input box. Below these fields are several buttons: "\*", "+", "U", and "." in the first row; "(", ")", "[", and "]" in the second row; "Apex" in the third row; and "Subscript" in the fourth row. At the bottom center, the word "RESOLVE" is written in a light blue, stylized font.

Figure 1 - RegexParser application initial form

- **Alphabet  $\Sigma$ :** Textbox where write the alphabet symbols used to write the regex:
  - Only ONE-char symbols are allowed.
  - Symbols like (, ), [, ], \*,  $\epsilon$ , +, U,  $\cdot$ , \$, %, &, |, are NOT allowed in this Textbox.
  - Symbols in the Textbox are not spaced. Insert these like the following example:

This image shows a close-up of the "Alphabet  $\Sigma$  = { " input field. The text "abc" has been entered into the text input box, which is followed by a closing curly brace "}".

Figure 2 - example: insert symbol in the alphabet textbox

- **Regex:** Textbox where write the regular expression:
  - View the chapter “how to write a Regular Expression” for more details.
  - Space between symbol are NOT allowed.

- **Symbols keyboard:** keyboard that allow to write a special symbol.
- **Apex button:** button that allow to write an apex to regular expression:
  - One clicks on this button, show a textbox where insert the number you want.
  - The apex value MUST be greater than 0.
  - Only ONE-digit number are allowed.
- **Subscript button:** button that allow to write a subscript to regular expression:
  - One clicks on this button, show a textbox where insert the number you want.
  - The subscript value MUST be smaller than the apex one. It can be equal to 0.
  - Only ONE-digit number are allowed.
  - You can insert a subscript ONLY IF there is an apex in the previous position.
- **Resolve button:** button that initiate the parsing process.

## HOW TO WRITE A REGULAR EXPRESSION

In this chapter, you can see how to avoid error when writing a regular expression.

An example is useful for this aim:

$$\Sigma = \{abc\}$$

$$Regex: (ab)^*U[c]Ua^+$$

- Make sure that between ( and ) or [ and ] there is at least one alphabet symbol.
- Make sure that \*, +, ), ], :, U and the apex are not the first symbol of the regular expression.
- Make sure that (, :, [, U are not the last symbol of the regular expression.
- Make sure that the Apex value is greater than the Subscript one if this is present.
- Make sure that the number of ( or [ is equals to the number of ) or ] respectively.

In the next page, there are all errors you can see using this tool.

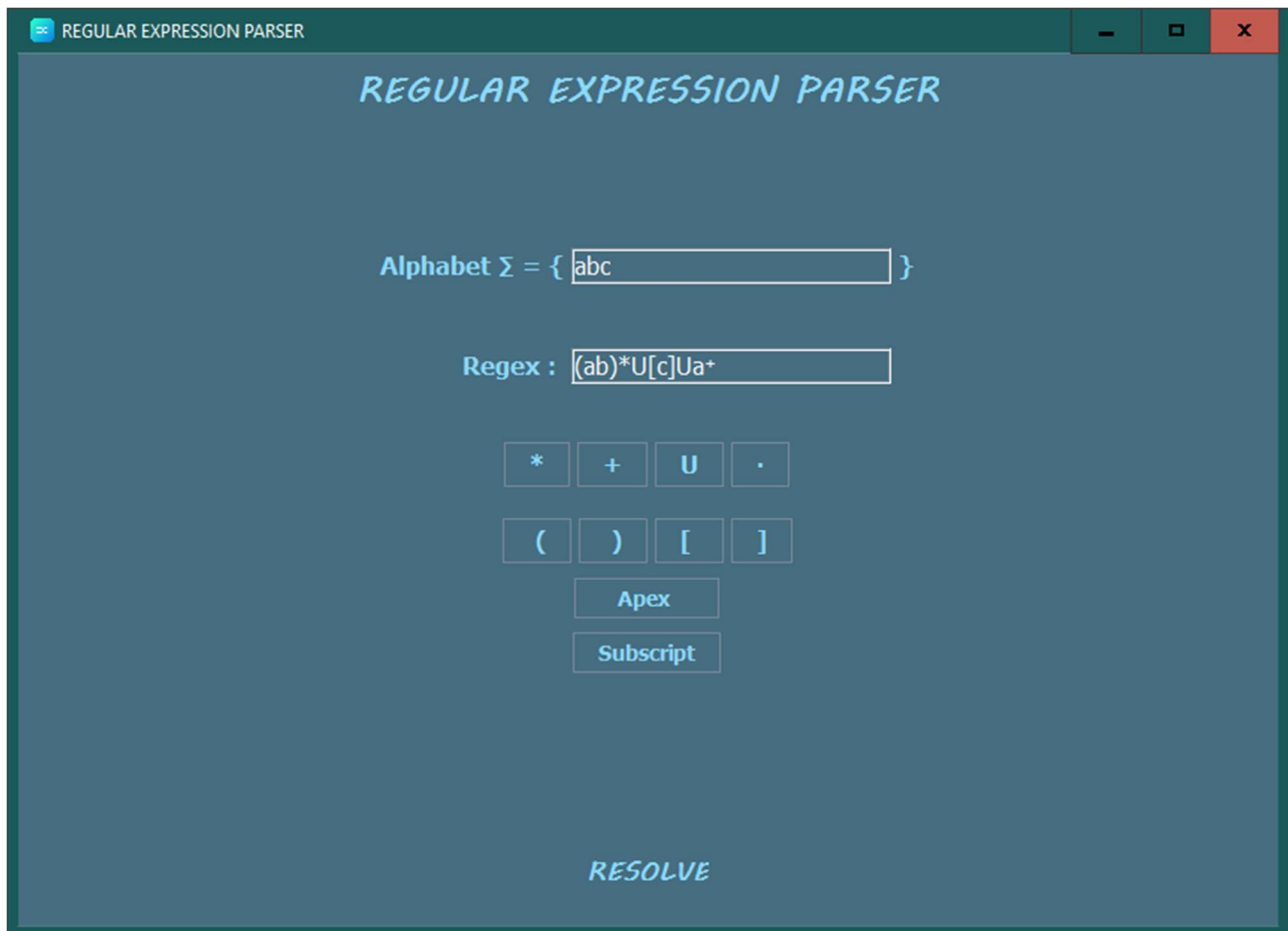
## POSSIBLE ERRORS

*The string “alphabet symbol” show the specific symbol of your use case.*

1. The Alphabet and the Regular Expression must not be empty.
2. Symbols not permitted in Regular Expression.
3. The Regular Expression must contain at least one alphabet symbol.
4. After a round bracket (, there could only be symbols: (, [, alphabet symbols.
5. The round bracket ( couldn't be the last symbol of the Regular Expression.
6. Before a round bracket ), there could only be symbols: ), ], \*, +, one apex, one subscript, alphabet symbol.
7. The round bracket ) couldn't be the first symbol of the Regular Expression.
8. After a square bracket [, there could only be symbols: (, [, alphabet symbol.
9. The square bracket [ couldn't be the last symbol of the Regular Expression.
10. Before a square bracket ], there could only be symbols: ), ], \*, +, one apex, one subscript, alphabet symbol.
11. After a square bracket ], there could only be symbols: (, ), [, ], one apex, U, ·, alphabet symbol.
12. The square bracket ] couldn't be the first symbol of the Regular Expression.
13. Before a star \*, there could only be symbols: ), alphabet symbol.
14. After a star \*, there could only be symbols: (, ), [, ], one apex, U, ·, alphabet symbol;
15. The star \* couldn't be the first symbol of the Regular Expression.
16. Before a cross +, there could only be symbols: ), alphabet symbol.
17. After a cross +, there could only be symbols: (, ), [, ], one apex, U, ·, alphabet symbol.
18. The cross + couldn't be the first symbol of the Regular Expression.
19. Before a point ·, there could only be symbols: ), ], \*, +, alphabet symbol.
20. After a point ·, there could only be symbols: (, [, alphabet symbol.
21. The point · couldn't be the first symbol of the Regular Expression.
22. The point · couldn't be the last symbol of the Regular Expression.
23. Before an Union U, there could only be symbols: ), ], \*, +, alphabet symbol.
24. After an Union U, there could only be symbols: (, [, alphabet symbol.
25. The Union U couldn't be the first symbol of the Regular Expression.
26. The Union U couldn't be the last symbol of the Regular Expression.
27. Before an Apex, there could only be symbols: ), ], alphabet symbol.
28. After an Apex, there could only be symbols: (, ), [, ], one subscript, U, ·, alphabet symbol.
29. Before a Subscript, there could only be an apex.

30. Before an Apex-Subscript pair, there must be a square bracket `]`.
31. After a Subscript, there could only be symbols: `(, ), [, ], U, ·`, alphabet symbol.
32. Apex and Subscript couldn't be the first symbols of the Regular Expression.
33. The number of `(` and `[` must be the same of `)`, `]`, respectively.
34. The Apex must be a positive one-digit number.
35. There cannot be a Subscript if there's not an Apex.
36. The Subscript must be a positive one-digit number smaller than the Apex.
37. `(, ), [, ], *, U, +, ·, ε` are not allowed like alphabet symbols.

## USE EXAMPLE



The screenshot shows a web application titled "REGULAR EXPRESSION PARSER". The interface has a dark teal background. At the top, there's a title bar with a minus, maximize, and close button. Below the title bar, the text "REGULAR EXPRESSION PARSER" is displayed in a light blue, stylized font. The main area contains two input fields: "Alphabet  $\Sigma = \{$ " followed by a text box containing "abc" and a closing brace "}", and "Regex :" followed by a text box containing "(ab)\*U[c]Ua+". Below these fields is a set of buttons for building the regex: "\*", "+", "U", and "." in the first row; "(", ")", "[", and "]" in the second row; "Apex" in the third row; and "Subscript" in the fourth row. At the bottom center, there is a large, light blue "RESOLVE" button.

Figure 3- example: insert alphabet and regular expression

After one click on the RESOLVE button, you can see the result like in the next page.

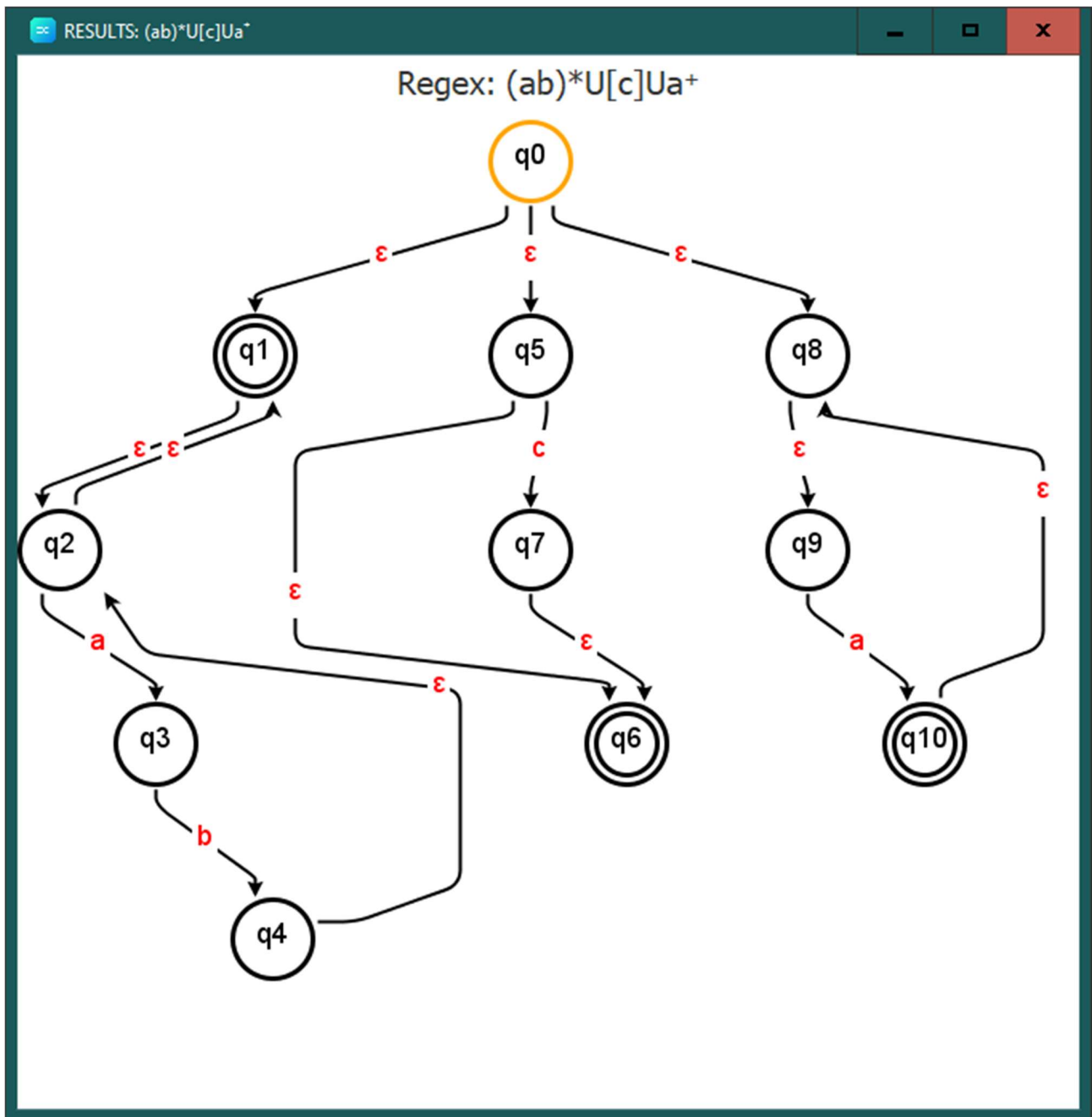


Figure 4 - example: result



## CREDITS

Tool designed and developed by

Dott. Davide Cesani @ Università degli Studi di Bergamo

*d.cesani@studenti.unibg.it*

Dott. Federico Nespoli @ Università degli Studi di Bergamo

*f.nespoli1@studenti.unibg.it*

as end-of-class project.

Class:

Formal languages and Compilers

Prof. Giuseppe Psaila