Regular expression to NFA table

Version 1.0.0

Language: Python2.7

A regular expression to non-finite automata converter.

> I did not include the math portion in this version of the write up document. Did not have the time.

Usage:

Standard Usage:

```
$ python REtoNFA.py
```

Stdin Usage:

```
Linux/Windows/IOS:
    $ python REtoNFA.py < file.txt

Powershell:
    >  Get-Content file.txt | python REtoNFA.py
```

Example

```
 $ python REtoNFA.py
Enter Regular Expression: abba

        D |  a  |  b  |  e
        --------------------
        >0|  1  |  /  |  /
        1 |  /  |  2  |  /
        2 |  /  |  3  |  /
        3 |  4  |  /  |  /
        *4|  /  |  /  |  /

$ python REtoNFA.py
Enter Regular Expression: aab+bbb

        D |  a  |  b  |  e
        --------------------
        >0|  1  |  4  |  /
        1 |  2  |  /  |  /
        2 |  /  |  3  |  /
        *3|  /  |  /  |  /
        4 |  /  |  5  |  /
        5 |  /  |  3  |  /

$ python REtoNFA.py
Enter Regular Expression: a*+b*

        D |  a  |  b  |  e
        --------------------
        >0|  0  |  0  |  1
        *1|  /  |  /  |  /
```

Regular Expression Format:

`Format: Any combination of accepted characters.`

String	Represents
αβ	α then β
α+β	α or β
α^*	0 or more α's
α (ρ) β	ρ has priority over β

> **Note:** The output dictionary will have all transition values, even it they are not represented in the default print table.

Accepted Characters:

> Note: This program will only display the transitions for a, b, and ε in its default printout table.

This program will accept almost all characters *(See non-accepted characters for exceptions)*.

Char	Represents
a-z, A-Z, 0-9, ε	Literal Character
+	Or operation
*	Kleene Star
( )	Grouping

Non-Accepted Characters:

Some characters may break the program. Those characters include but are not limited to:

Char	Represents	Will Cause
\$	Final Node in return dict	Unwanted Final Node Declaration
&	Epsilon transition in return dict	Unwanted Epsilon Transition
\t	End of RE	Premature Termination of program
α+	any string ending with a plus	Breaking of Program

How it works:

At its core, this program is a simple case of parsing a string, identifying cases, and performing the proper function for each case.

After the program recieves a string in a proper format from the user, The string gets passed to the `parse()` function, where an end-tag(`\t`) is appended to the end of the string and then is passed to the `parseRecur()` function.

Once in `parseRecur()` , the string is first checked if it only contains the terminal character, if yes then terminate. Otherwise the string is broken down into 3 pieces: * alpha = First character of string * beta = Rest of string * nu = Next character following alpha

Now, alpha is checked to see if it is in `[+,(,),*]` , if its one of these characters, perform the neccessary operation ***(See math section for details of each operation)***. Otherwise, append `{curpose: {alpha: nextpos}` to the final dictionary. return `parseRecur()` until the terminating character is found.

Then return the final dictionary of key-value pairs.

Time Spent

- 108 Hours Total
  - Began working on April 27

- Figured out math on May 7
- Began rudament programming on May 8
- Completed program on May 13

License & copyright