Report

1. Team member and labor division:

黄振栋 12S151019: Analysis the regular expression and create the rules to check the text and document.

戴 蔚 12S151021: Create the rules to get the NFA, draw the NFA and document.

1. IDE：

Ubuntu 12.04, python 2.7, VIM, pygraphviz

Language: python

1. User manual

3.1 Abstract:

This program is written in Ubuntu12.04 and in python 2.7.

This program use data structure 'list' and 'dict' in principle such as

'PDA' , 'NFA','DFA' which we learned in TOC to generate Rules and NFA from expression instead of using 're' module provided by python.

This program use pygraphviz to draw the NFA and output a 'pdf'

file. Pygraphviz is a interface of 'graphviz' for python, So you should make sure that you have install pygraphviz if you want to draw a NFA.

3.2 Document:

We have 5 documents in our project directory.

The documents are 'clean.sh', 'NFA.py', 'recReg.py', 'Reg.py' and tf.

3.2.1 The file 'clean.sh' is used to clean the tmp files and other trash files, you can type './clean.sh' in Terminal in order to remove the trash files.

3.2.2 The file 'tf' is used to store the strings you want to check if it's recognized by the 'regex' you input.

3.2.3 The file 'recReg.py' is used to receive command and arguments you input and then call the other two source code files 'Reg.py' and 'NFA.py' to deal with the arguments. 'recReg.py' will create a object called 'reg' of class 'Reg' which is in 'Reg.py' and create object 'nfa' in the same way. The main function will use 'reg' to generate Rule sets to recognize strings read from 'tf' and use 'nfa' to generate a NFA according the inputed 'regex'

3.2.4 The file 'Reg.py' is source code of a class 'Reg', 'Reg' has some functions to deal with the input expression. First 'Reg' will use function 'rmbra' to remove brackets in expression, then use function 'rmBeg' to eliminate '^' in the expression and generate Rule'^', then use function 'rmTai' to eliminate '$' in the expression and generate Rule'$', then use function 'rmStar' to eliminate '\*' and and generate Rule'\*', then use function 'rmAdd' to eliminate '+' and generate Rule'+'. Then 'reg' will generate RuleSets from the input regex. To check whether the strings is recognized by the RuleSets, the function 'checkRule' will be called to prove it and 'checkRule' is in file 'recReg.py'

3.2.5 The file 'NFA.py' is source code of a class 'NFA', 'NFA' has some functions to generate RuleSets from the input regex.But you should notice that RuleSets generated by 'NFA.py' is different from those by 'Reg.py', becasue RuleSets generated by 'NFA.py' will be used to generate a NFA which will recognize the input regex. 'NFA.py' will use function 'genNFA' to generate NFA from the RuleSets. After that the function 'drawNFA' will generate a 'dot' file and then python will use pygraphviz which is a interface provided by 'Graphviz' to generate a 'pdf' file.

3.3 How-TO:

Open Terminal and enter the source code directory.

Type the command:

python recReg.py -v '(a+b)\*c' tf

If you want to generate Rules from the regex '(a+b)\*c' to recognize strings stored in tf and generate NFA for regex

Type the command:

python recReg.py '(a+b)\*c' tf

If you want to generate Rules from the regex '(a+b)\*c' to recognize strings stored in tf and don't want to generate NFA.

Example:

We have test almost all of the expression and the tested regex is listed below

(0|1)\*

a\*b

a+b

a\*

a+

(grep)^

(last)$

(a..b)

(a|b)

(a|b)\*c(m+n)\*

a.\*b

(grep)^(last)$

(010)^((a|b)+c+)(b..c)(m\*n)\*(000)$

ErrorData:

We still have bugs to be fixed.

1. Major theory
   1. Procedure flow picture



* 1. Create the rules to check the text
     1. Data structure

We used nested structure of dictionary and list to store the rules.

For example:

The CFG is: A -> B | a, B -> ! (! stand for ε, below as the same)

So the left of the expression is the key of the dictionary, the right is the value of the dictionary. { ‘A’ : [ ‘B’ , ‘a’ ] , ‘B’: ‘!’}

* + 1. Create the rules from regular expression

First, we put the brackets off.

We check the order is left to right. If there have nested brackets,

we first handling the inside brackets, then handling the outside brackets. Next, we choose a capital which didn’t used to replace the brackets. For, example:

exp = (010)^((a | b)+c+)\*(b..c)\*(m\*n)(000)$

A -> 010, B -> a | b, C -> B + c+, D -> b..c, E -> m\*n, F -> 000

So, we can replace the exp without the brackets:

exp = A^C\*D\*EF$

Next, we put the “^” , “$”, “\*”, “+”, “|”, “.”,off.

We can choose a capital which didn’t used to replace the “C\*”.

For example:

H -> C\*, we choose “H” replace the C\*.

H -> CH | !

For the other special symbol, we also used this method to put them off.

Final, we can get the rule. For example:

S -> AHDIF

The fully process of create the rules is that:

exp = (010)^((a | b)+c+)\*(b..c)\*(m\*n)(000)$

rmBra(put the brackets off)

{'A': '010', 'C': 'B+c+', 'B': 'a|b', 'E': 'm\*n', 'D': 'b..c', 'F': '000', 'S': 'A^C\*DE\*F$'}

rmBeg(put the “^” off)

{'A': '010', 'C': 'B+c+', 'B': 'a|b', 'E': 'm\*n', 'D': 'b..c', 'F': '000', 'S': 'AC\*DE\*F$'}

rmTai(put the “$” off)

{'A': '010', 'C': 'B+c+', 'B': 'a|b', 'E': 'm\*n', 'D': 'b..c', 'F': '000', 'S': 'AC\*DE\*F'}

rmStar(put the “\*” off)

{'A': '010', 'C': 'B+c+', 'B': 'a|b', 'E': 'Gn', 'D': 'b..c', 'G': 'mG|!', 'F': '000', 'I': 'EI|!', 'H': 'CH|!', 'S': 'AHDIF'}

rmAdd(put the “+” off)

{'A': '010', 'C': 'JK', 'B': 'a|b', 'E': 'Gn', 'D': 'b..c', 'G': 'mG|!', 'F': '000', 'I': 'EI|!', 'H': 'CH|!', 'K': 'c|cK', 'J': 'B|BJ', 'S': 'AHDIF'}

rmDot(put the “.” off)

{'A': '010', 'C': 'JK', 'B': 'a|b', 'E': 'Gn', 'D': 'bLMc', 'G': 'mG|!', 'F': '000', 'I': 'EI|!', 'H': 'CH|!', 'K': 'c|cK', 'J': 'B|BJ', 'M': ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '1', '2', '3', '4', '5', '6', '7', '8', '9', '0'], 'L': ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '1', '2', '3', '4', '5', '6', '7', '8', '9', '0'], 'S': 'AHDIF'}

rmVer(put the “|” off)

{'A': '010', 'C': 'JK', 'B': ['a', 'b'], 'E': 'Gn', 'D': 'bLMc', 'G': ['mG', '!'], 'F': '000', 'I': ['EI', '!'], 'H': ['CH', '!'], 'K': ['c', 'cK'], 'J': ['B', 'BJ'], 'M': ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '1', '2', '3', '4', '5', '6', '7', '8', '9', '0'], 'L': ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '1', '2', '3', '4', '5', '6', '7', '8', '9', '0'], 'S': 'AHDIF'}

the Regular for regex (010)^((a|b)+c+)\*(b..c)(m\*n)\*(000)$ is:

{'A': '010', 'C': 'JK', 'B': ['a', 'b'], 'E': 'Gn', 'D': 'bLMc', 'G': ['mG', '!'], 'F': '000', 'I': ['EI', '!'], 'H': ['CH', '!'], 'K': ['c', 'cK'], 'J': ['B', 'BJ'], 'M': ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '1', '2', '3', '4', '5', '6', '7', '8', '9', '0'], 'L': ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '1', '2', '3', '4', '5', '6', '7', '8', '9', '0'], 'S': 'AHDIF'}

* + 1. Check the text

The check process is simulate the process of the PDA distinguish.

* 1. Create rules to construct the NFA
     1. Data structure

The rules structure is the same as rules structure in the check

text. What’s more, we used multiple nested structure of dictionary and list to store the state and condition. We choose the state which the arrows get out to the key of dictionary, then the list of a dictionary to the value of dictionary. The dictionary nested in the list, the key is the state which the arrows get in and the value is the condition that the get out state directs to the get in state. For example:



{ 1 : [ { 2 : ‘A’ } ] , 2: [ { 3 : ‘B’ } ] }

* + 1. Create the rules

This create rules is mostly the same as the create rules to check

the text, like the main idea and handling some special symbols. However, there are some little different exist. For example:

exp = (010)^((a | b)+c+)\*(b..c)\*(m\*n)(000)$

The final rule is as following:

{'A': '010', 'C': 'LM', 'B': 'a|b', 'E': 'In', 'D': 'b..c', 'G': 'GH|!', 'F': '000', 'I': 'm\*', 'H': '.', 'K': 'E\*', 'J': 'C\*', 'M': 'c+', 'L': 'B+', 'S': 'GAJDKF'}

* + 1. Construct the NFA

First, we construct two state, the start state and the end state.

Then link the start state to end state by the condition S.



Next, we ergodic the S, and expand the state and conditions for

each non terminal , until there didn’t have non terminal.

The final state and condition dictionary is as following:

{1: [{3: '!'}], 3: [{10: '!'}, {12: '!'}], 4: [{14: '!'}], 5: [{6: '!'}, {18: '!'}], 6: [{20: '!'}], 7: [{8: '!'}, {25: '!'}], 8: [{27: '!'}], 9: [{2: '!'}], 10: [{31: '!'}], 11: [{4: '!'}], 12: [{13: '!'}], 13: [{4: '!'}], 14: [{15: '0'}], 15: [{16: '1'}], 16: [{17: '0'}], 17: [{5: '!'}], 18: [{33: '!'}], 19: [{6: '!'}, {18: '!'}], 20: [{21: 'b'}], 21: [{22: '.'}], 22: [{23: '.'}], 23: [{24: 'c'}], 24: [{7: '!'}], 25: [{36: '!'}], 26: [{8: '!'}, {25: '!'}], 27: [{28: '0'}], 28: [{29: '0'}], 29: [{30: '0'}], 30: [{9: '!'}], 31: [{32: '.'}], 32: [{11: '!'}], 33: [{39: '!'}], 34: [{41: '!'}], 35: [{19: '!'}], 36: [{37: '!'}, {43: '!'}], 37: [{38: 'n'}], 38: [{26: '!'}], 39: [{45: '!'}, {47: '!'}], 40: [{34: '!'}, {39: '!'}], 41: [{42: 'c'}], 42: [{35: '!'}, {41: '!'}], 43: [{44: 'm'}], 44: [{37: '!'}, {43: '!'}], 45: [{46: 'a'}], 46: [{40: '!'}], 47: [{48: 'b'}], 48: [{40: '!'}]}

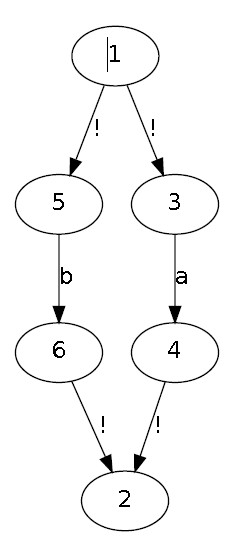
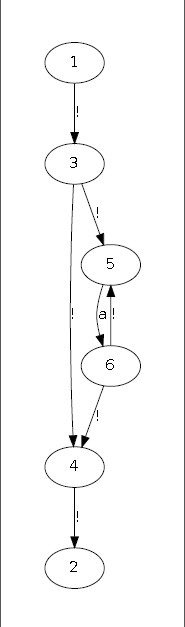
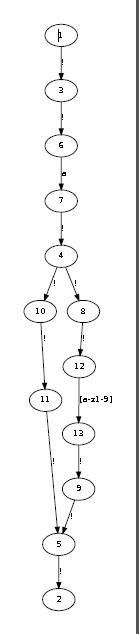
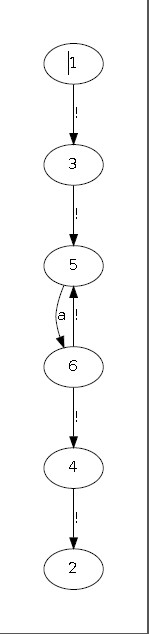
* 1. Draw the NFA

We use the pygraphviz to draw the NFA by the state and condition

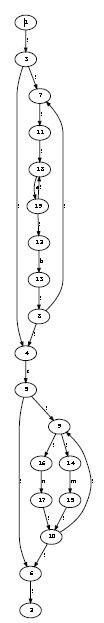
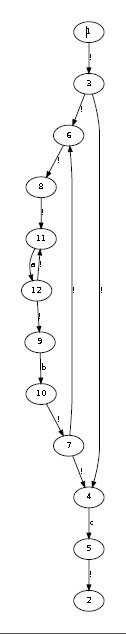
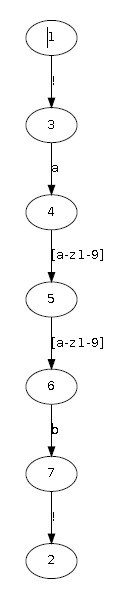
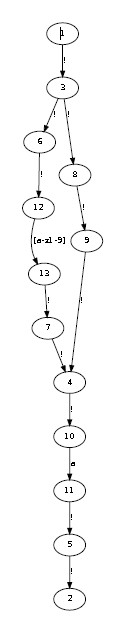
dictionary.

1. Testing result picture

5.1 Some NFA of simple regular expression:



Exp=a+ Exp=a^ Exp=a\* Exp=a|b



Exp=a$ Exp=a..b Exp=(a+b)\*c Exp=(a+b)\*c(m^|n)\*

5.2 Standard testing result

The standard testing result is in the “toc\_test” file. In this file, we

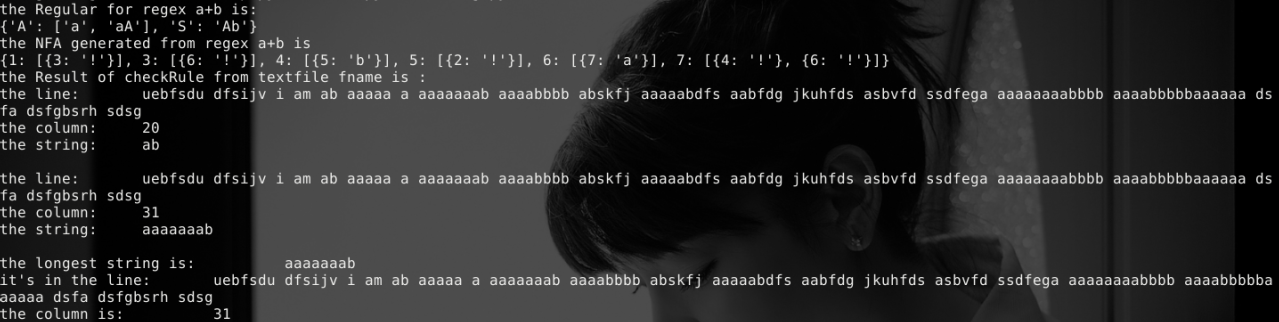
gived some testing result and the PDA.

Example:

Input:

C:\Users\Administrator\AppData\Roaming\Tencent\Users\409257922\QQ\WinTemp\RichOle\UTVL9@9CP1XOE4@9$2$[X2Y.jpg

Output:



PDA:

