CS323 Compilation Principle

Assignment 1

Name | Yubin Hu
ID | 11712121
Date | 2020.09.23

Required Exercises

Exercise 1

> When a C compiler compiles the following statement, how many tokens will it generate? [5 points]
>
> ```
> int a3 = a * 3;
> ```

There are 7 tokens will be generated.

- <keyword, int>
- <id, a3>
- <id, a>
- <assign, =>
- <assign, *>
- <assign, ;>
- <number, 3>

Exercise 2

> In a string of length n (n > 0), how many of the following are there?
>
> 1. Prefixes [5 points]
> 2. Proper prefixes [5 points]
> 3. Prefixes of length m (0 < m ≤ n) [5 points]
> 4. Suffixes of length m (0 < m ≤ n) [5 points]
> 5. Proper prefixes of length m (0 < m ≤ n) [10 points]
> 6. Substrings [10 points]
> 7. Subsequences [10 points]

1. $n+1$
2. $n-1$
3. 1
4. 1

5. $res = \begin{cases} 0 & \text{m=n} \\ 1 & \text{otherwise} \end{cases}$
6. $1 + \frac{(1+n)n}{2}$
7. 2^n

Exercise 3

> Describe the languages denoted by the following regular expressions:
>
> 1. ((ε|a)*b*)* [5 points]
> 2. (a|b)*a(a|b)(a|b) [5 points]
> 3. a*ba*ba*ba* [5 points]

1. A string consisting of a and b
2. A string consisting of a and b whose third-to-last digit is a
3. A string consisting of a and b with only three

Exercise 4

> Write regular definitions or regular expressions for the following languages.
>
> 1. All strings representing valid telephone numbers in Shenzhen. A valid telephone number contains the country code (86), a hyphen, the area code 0755, another hyphen, and eight digits where the first one cannot be zero (e.g., 86-0755-88015159). [10 points]
> 2. All strings of a's and b's that start with a and end with b. [10 points]
> 3. All strings of lowercase letters that contain the five vowels in order. [10 points]

```
86-0755-[1-9][0-9]{7}
```

```
a(a|b)*b
```

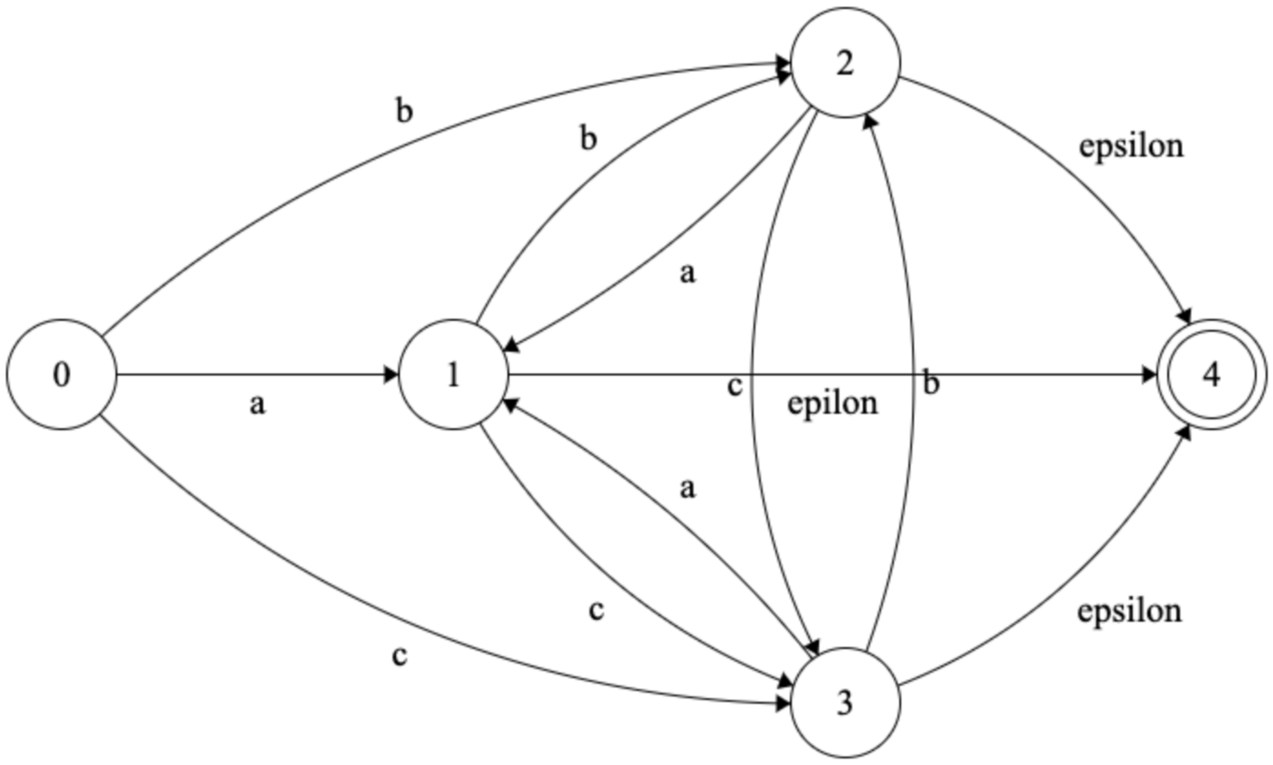
```
pattern = [b-df-hj-np-tv-z]
{pattern}*a{pattern}*e{pattern}*i{pattern}*o{pattern}*u{pattern}*
```

Optional Exercises

Exercise 1

> Suppose we have a alphabet Σ = {a, b, c}, write regular definitions to describe all strings over Σ without repeated letters. [Hint: You may draw an NFA for the language and convert the NFA to regular definitions.]

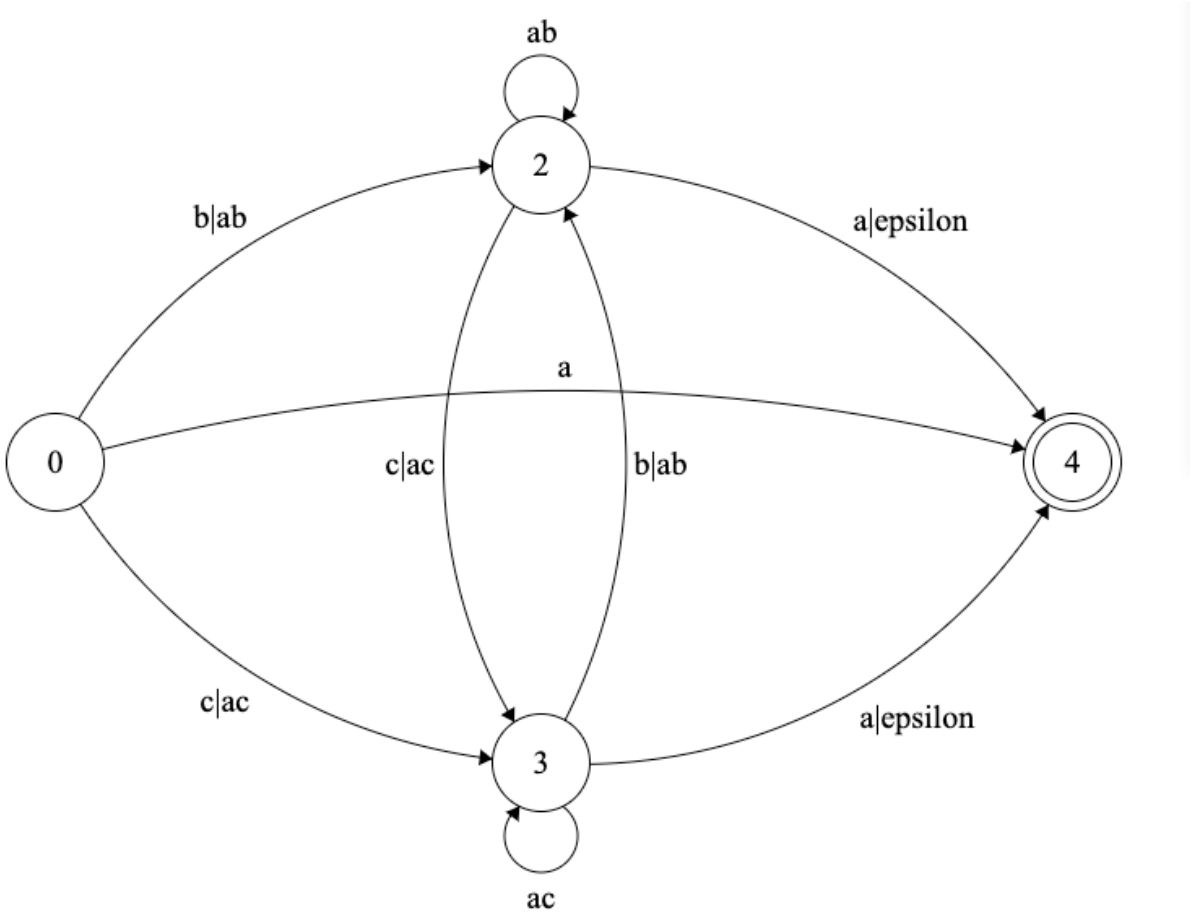
The origin NFA figure



Then, we should simplify the figure by removing the state 1, 2 , 3 to get the regular definitions.

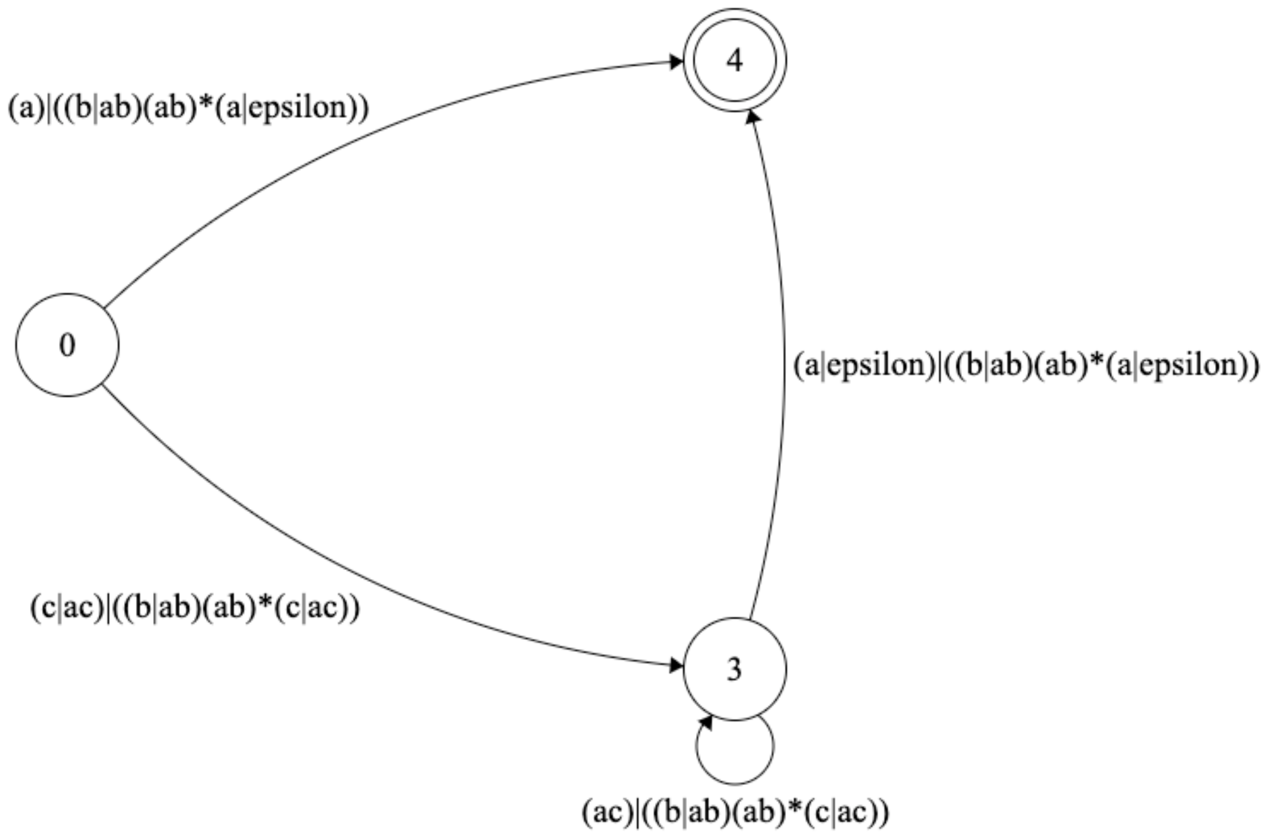
Remove state 1

edge	**Regular expression**
0 -> 2	$b\|ab$
0 -> 3	$c\|ac$
0 -> 4	a
2 -> 2	ab
2 -> 3	$c\|ac$
2 -> 4	$a\|\epsilon$
3 -> 2	$b\|ab$
3 -> 3	ac
3 -> 4	$a\|\epsilon$



Remove state 2

edge	Regular expression
0 -> 3	$(c\|ac)\|((b\|ab)(ab)^*(c\|ac))$
0 -> 4	$(a)\|((b\|ab)(ab)^*(a\|\epsilon))$
3 -> 3	$(ac)\|((b\|ab)(ab)^*(c\|ac))$
3 -> 4	$(a\|\epsilon)\|((b\|ab)(ab)^*(a\|\epsilon))$


4
(a)|((b|ab)(ab)*(a|epsilon))
0
(a|epsilon)|((b|ab)(ab)*(a|epsilon))
(c|ac)|((b|ab)(ab)*(c|ac))
3
(ac)|((b|ab)(ab)*(c|ac))

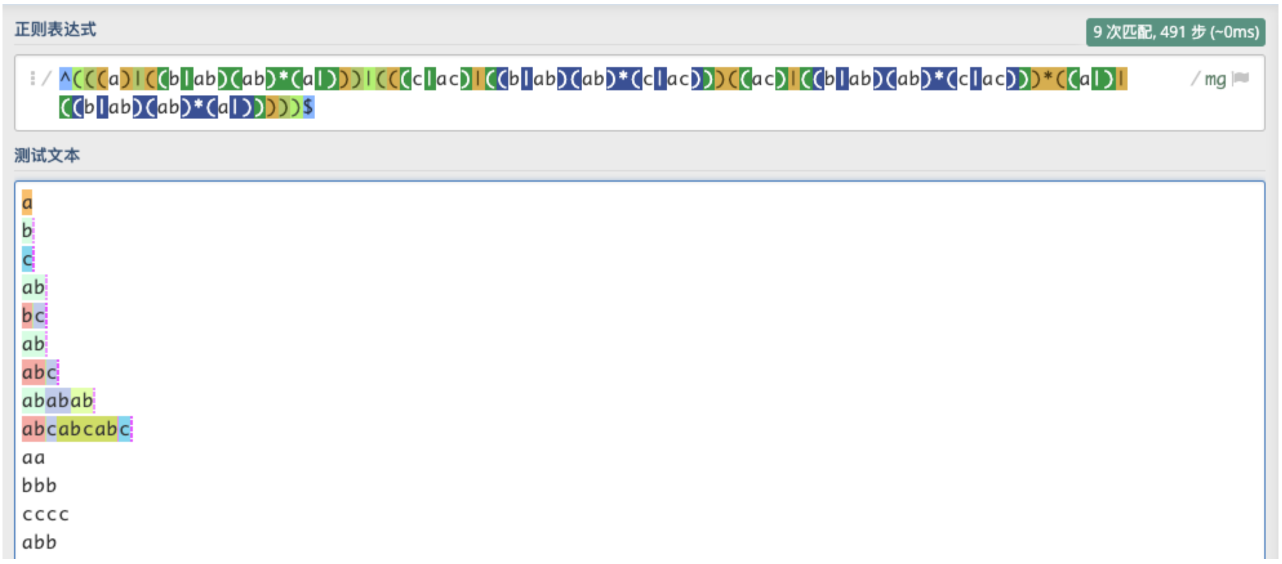
Remove state 3

edge	Regular expression
0 -> 4	$(((a)\|((b\|ab)(ab)^*(a\|\epsilon)))\|(((c\|ac)\|((b\|ab)(ab)^*(c\|ac)))((ac)\|((b\|ab)(ab)^*(c\|ac))) \ast ((a\|\epsilon)\|((b\|ab)(ab)^*(a\|\epsilon)))))$

So, from NFA to regular expression, the result is
$(((a)|((b|ab)(ab)^*(a|\epsilon)))|(((c|ac)|((b|ab)(ab)^*(c|ac)))((ac)|((b|ab)(ab)^*(c|ac))) \ast ((a|\epsilon)|((b|ab)(ab)^*(a|\epsilon)))))$
.

regex101

https://regex101.com/r/5C0PEp/1


正则表达式

9 次匹配, 491 步 (~0ms)

```
/^(((a)|((b|ab)(ab)*(a|)))|(((c|ac)|((b|ab)(ab)*(c|ac)))((ac)|((b|ab)(ab)*(c|ac)))*((a|)|((b|ab)(ab)*(a|)))))$/mg
```

测试文本

```
a
b
c
ab
bc
ab
abc
ababab
abcabcabc
aa
bbb
cccc
abb
```