VC Lexer Report

20021327 Nguyen Tien Dat[1] and 20020277 Nguyen Ha An[2]

[1]VNU UET
[2]VNU UET

April 17, 2024

1 Introduction

This report show the design and result of a **table-driven lexer**. The approach is the usage of transition table to simulate the mechanism of a finite state machines.

2 Design

2.1 Pipeline

The main component of the lexer are the lexer code written in python3 and the transition table. The output will be a list of token written in .vctok files. Below is the pipeline for the lexer.

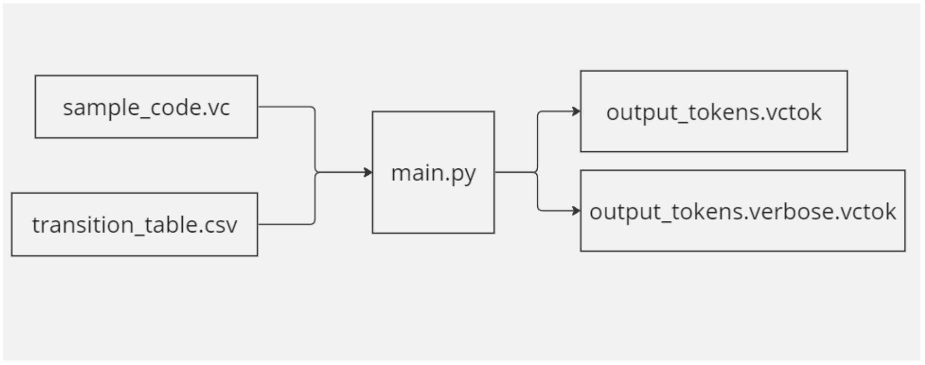


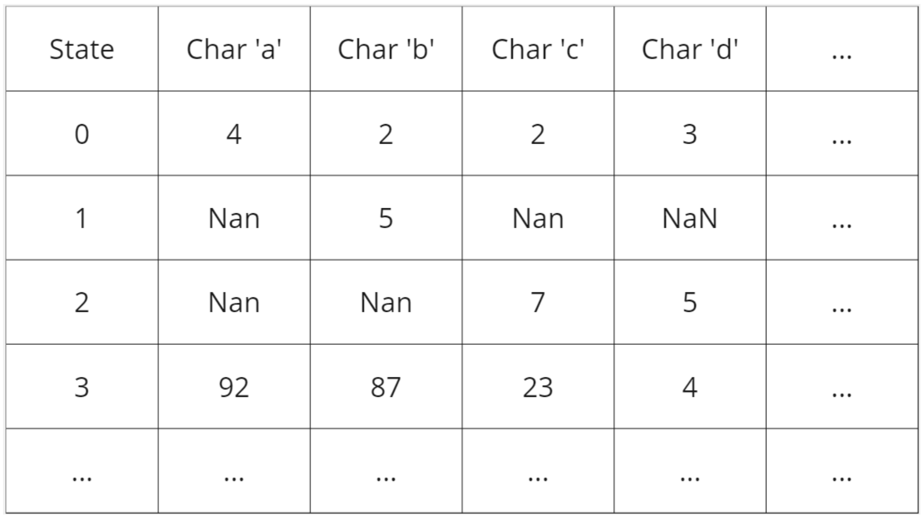
Figure 1: Lexer Pipeline

2.2 Transition Graph

In order to create the Transition Table, we first create the Transition Graph. The graph is generated by using the Graphviz tool which uses DOT language to draw graph at ease.

2.3 Transition Table

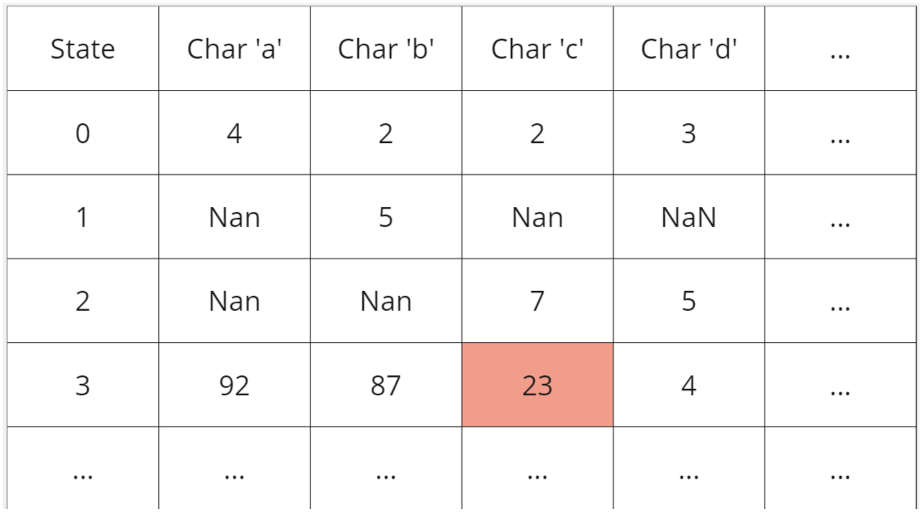
This table simulates DFA by looking information of the next stage with the input character. The Transition Table is derived from the Transition Graph before. The first row are the single character. The first column are states. Below is the format of the table.



State	Char 'a'	Char 'b'	Char 'c'	Char 'd'	...
0	4	2	2	3	...
1	Nan	5	Nan	NaN	...
2	Nan	Nan	7	5	...
3	92	87	23	4	...
...	...	...	...	...	...

Figure 2: Table Format

NaN(Not a Number) values mean there is no next state. This is due to the reading of blank cells in .csv when read in Pandas DataFrame will create NaN values.



State	Char 'a'	Char 'b'	Char 'c'	Char 'd'	...
0	4	2	2	3	...
1	Nan	5	Nan	NaN	...
2	Nan	Nan	7	5	...
3	92	87	23	4	...
...	...	...	...	...	...

Figure 3: For example, at state 3 given the character 'c' will result in state 23 (highlighted red).

3 Implementation

3.1 Overview

This is the graph represent the implementation.

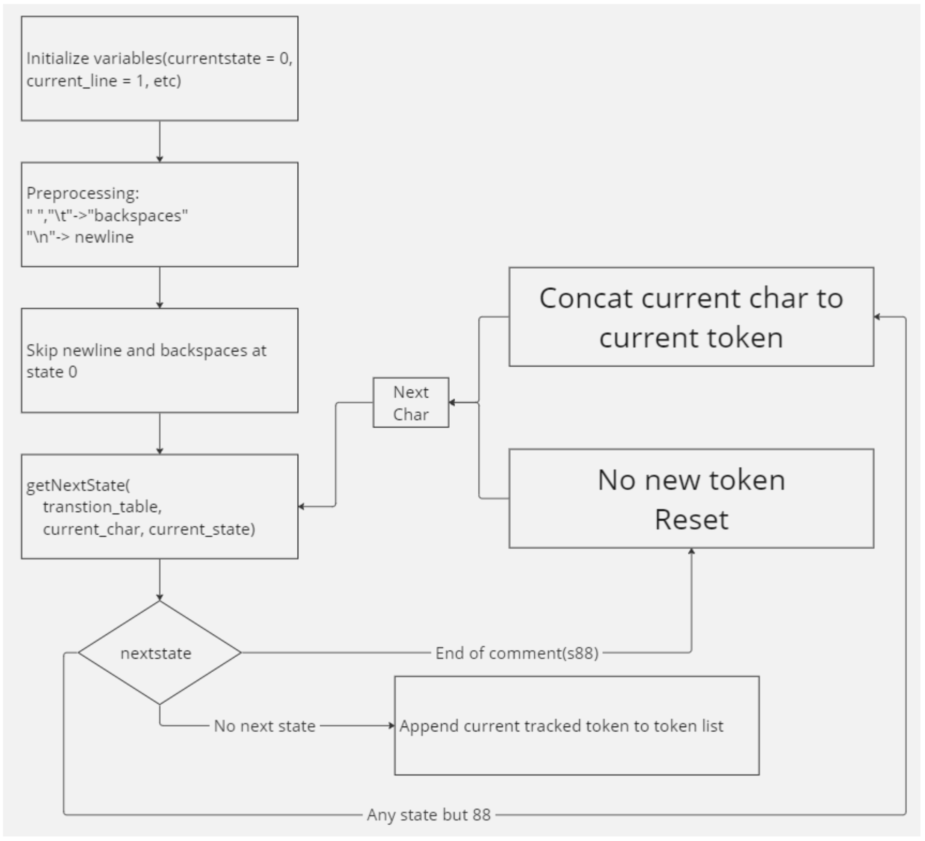

Initialize variables(currentstate = 0, current_line = 1, etc)
Preprocessing: " ","\t"->"backspaces" "\n"-> newline
Skip newline and backspaces at state 0
getNextState(transtion_table, current_char, current_state)
nextstate
Next Char
Concat current char to current token
No new token Reset
End of comment(s88)
No next state
Append current tracked token to token list
Any state but 88

Figure 4: Implementation

3.2 Details

1. First, the VC source code and transition table is read. Initialization of some variable.

```
    char_iterator = 0
    current_char = source_code[char_iterator]
    current_state = 0
    current_token = ''
    current_state = 0
    current_token = ''
```

2. Next, some preprocessing task is done so we can look up at the next state.

```
if current_char == ' ' or current_char == '\t':
    current_char = 'whitespaces'
if current_char == '\n':
    current_char = 'newline'
```

3. If we at state 0 and the current character is backspace, tab or newline, we will skip to next character without doing anything

4. If the current state is clossing of comment, we will do nothing and reset the token.

5. If there is no next state, the token will be added to the list

6. If there is next state, the token will be updated.

```
if current_char == 'whitespaces':
    current_char = ' '
current_token += current_char
current_state = next_state
```

4 Result

The lexer works well as expected. The token output is the same when tested with given VC code sample. Some minor problem still exist include incorrect character tracking at some places.