Car's movement using Compiling design

Jenish Patel • 04.13.2021•CS453

Overview

Lexical Analyzer(scanner)

- **Lax**
- **flex**

Parser(syntax analyzer)

- Context free grammar
- Parser tree

Semantics Analyzer

Car Modul - Topic area 1

Car fundamentals

- Window: Open|Close
- Door: Open|Close
- Brake: On|Off
- Throttle: Pump|Release
- Transmission: 0|1|2|3|4

Explanation

- The main goal is we need to design the compiler for the car's fundamentals operation.
- Compiler has a two main different step, every step has his unique process and every process has his output, also the process needs error less previous output for began the next step.

Introduction - Topic area 2

Two step of compiler

- Lexical Analyzer(scanner)
- Parser tree(syntax analyzer)
- Semantic Analyzer
- Intermediate Code generator
- Code optimizer
- Target code generation

Explain

- Scanner,syntax and semantic analyzer are belongs from front end
- Intermediate code generator, code optimizer, target code generator are belongs from back end
- Front end are scan the program in different way and prompt a error
- Back end are generate a code for machine

Introduction

Further info of front end

- Lexical analyzer(scanner) create a table that contains a Tokens value and non-token value
- Example of: print(a)

  Lexical analyser create a table were function : print

  Variable: a

  Open bracket: (

  Close bracket :)

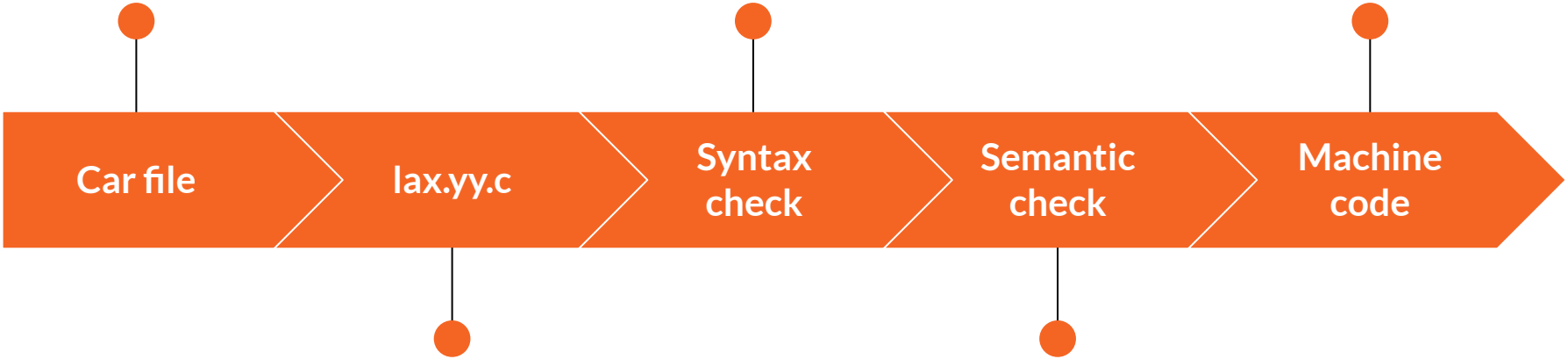
- Sytex analyzer(parser) get a table from lax file
- After that according the Context free grammar and production rule the parser crate a tree and check that the code is follow the correct production rules
- The context free grammar and the production rules was created by the user itself.

Design of Compiler

Lexical Analyzer scan the car table,get Token value and create a table

Parser get a lex tokens, he create a parser tree based on production rules

Optimize the code and generate the machine code



User needs to crate the production rules for the parser

Semantic analyzer check the grammar of the code

Car's Table

Drive

- Drive is the start symbol

Door

- Open
- Close

Window

- Up
- Down

Brake

- On
- Off

Throttle

- Pump
- Release

Transmission

- 0 (stop the car)
- 1
- 2
- 3
- 4

First step: Lexical Analyzer

- We can simplify our program to better understand
- https://github.com/Jenishbh/CS453/blob/master/Week5/CS453_Week5_hw1_Jenish_19549.pdf
- In this program we have our library where we can have out tokens value and the main program simply compare the value with our tokens and prompt the user understandable output
- As the same process we create our lexical file using flex code
- On the lexical file we need to assign our condition in the square bracket like on our car's table we have Door and the door can be [Open|Close], so the condition(as our library in uppers code) we define the language in the lex file and the lex file return the specific value
- https://github.com/Jenishbh/CS453/blob/master/Week5/CS453_week5_HW2_Jenish_19549/CS453_week5_hw2_screenshot_jenish_19549.pdf
- For understand the rules of the flex file follow this link : https://www.geeksforgeeks.org/flex-fast-lexical-analyzer-generator/

Second step : Parser(Syntax)

- After generating the lexical file from flex the process goes further to check the syntax analyzing
- At the point the user needs to create the production rules based on his needs, let's take a look of one example
- https://github.com/Jenishbh/CS453/blob/master/Week11/CS453_week11_Quiz_Jenish_19549.pdf
- Here the user crate the S->S+S that what production rules called, that rules are continually check by the parser to maintain the syntax
- Also the production rules must be has a starting node
- The starting node is the final part of the result
- Here is the car's production rules
- https://github.com/Jenishbh/CS453/blob/master/Week11/Production_rule_CS453%20(1).pdf

Step 2 : Parser(Syntax)

- After the Production rule created the parser create a parser tree for checking that the starting nodes meet the production rules requirement
- The parser tree has two way of creation, bottom up and top-down
- https://github.com/Jenishbh/CS453/blob/master/Week11/CS453_Week11_Hw2_parser_tree_Jenish_19549.pdf
- Here I create a production rule of car and check were my starting node is follow the production rules or not and then I create my parser top-down tree

Goals for next step

1. Create Yaac file