   





Accept input string from user.

Construct and display the parsing table

Generate; display augmented grammar

Generate LR(1) set of items for the grammar.

Compute GOTO and action functions.

User chooses a grammar from predefined set

The LR parser is a non-recursive, shift-reduce, bottom-up parser. It does a rightmost derivation in reverse. It uses a wide class of context-free grammar which makes it the most efficient syntax analysis technique.

LR parsers are also known as LR(k) parsers, where L stands for left-to- right scanning of the input stream; R stands for the construction of right-most derivation in reverse, and k denotes the number of look ahead symbols to make decisions.

**Front end:**

1. HTML5 +CSS3: General functionality, rendering and navigation of pages.
2. JavaScript: Front end validations of input grammar.
3. Bootstrap: To achieve graphic utilities.
4. jQuery library: User interface interactons, effects and widgets.

**Back end:**

(a)Implementation in C#

(b)Server side scripting in PHP

**DATA-FLOW DIAGRAMS**

**CONCLUSION**

The CLR parser has been successfully implemented.

The final application interface represents a visual figure of each step of the process of CLR parsing. The grammar for the parser is assumed i.e. fixed in the initial phase and the user is given the freedom of entering any input string.

The parser would generate LR(1) set of items and display the same. It would then generate a parsing table when the generated item set is fed to it.

**REFERENCES**

[1] Compilers: Principles, techniques and tools by Alfred Rao, Ravi Sethi, Jeffrey D Ullman

[2] LR(1) Parsing. Handout written by Maggie Johnson and Revised by Julie Zelenski.Stanford University

[3] Clemson University: [www.cs.clemson.edu/parsing.pdf](http://www.cs.clemson.edu/parsing.pdf)

[4] Bottom Up Parsers – Tutorials Point

[5] Compiler Design Lectures by Ravi Chandra Babu

**INTRODUCTION**

**IMPLEMENTATION DISCUSSION**

**ABSTRACT**

The objective of this project is to implement an efficient and robust CLR parser using a high level programming language, that is capable of correctly parsing any input fed to it.

The project work includes a set of predefined grammar and an interface which would convert each phase of the parsing process into a visual representation and would display onto webpage. The implementation is pretty straight forward and simple. Then it would take any input string belonging to the grammar language and show the acceptance or rejection of that input string.

SLR and the more-general methods LALR parser and CLR parser have identical methods and similar tables at parse time; they differ only in the mathematical grammar analysis algorithms used by the parser generator tool. SLR and LALR generators create tables of identical size and identical parser states.

LALR generators calculate lookahead sets by a more precise method based on exploring the graph of parser states and their transitions. This method considers the particular context of the current parser state. It customizes the handling of each grammar occurrence of some nonterminal S.

**RESULTS**

**METHODS**

**ABSTRACT**

**M.S.Ramaiah Institute of Technology**

**(Autonomous Institute, Affiliated to VTU)**

**Department of Computer Science and Engineering**

**EMULATOR FOR CLR PARSER**

( 1MS13CS036-Anvitha G K Bhat, 1MS13CS038- Deepak Jayaprakash, 1MS13CS042-Gayathri.R )