DISSERTATION

Compiler Automation Tool

Pile of sheets now just a click away

Under the guidance of

Ms. Rekha Bhatia / Mr. Munshi Yadav

Presented by :

GHAH

Gagandeep Singh (singh.gagan144@gmail.com)
Hargeet Kaur (hargeet.kaur13@gmail.com)
Amarpreet Singh (amarpreetsingh92@gmail.com)
Harpreet Kaur (harpreetmarwah19@gmail.com)

Guru Tegh Bahadur Institute of Technology

Guru Gobind Singh Indraprastha University
Dwarka, New Delhi

Year 2013

Compiler Automation Tool

DISSERTATION

Submitted in partial fulfillment of the

Requirements for the award of the degree

Of

Bachelor of Technology

In

Computer Science & Engineering

By

Amarpreet Singh (08/CSE1/2009)

Gagandeep Singh (04/CSE1/2009)

Hargeet Kaur (03/CSE1/2009)

Harpreet Kaur (15/CSE1/2009)

Under the guidance of:

Ms. Rekha Bhatia

Department of Computer Science & Engineering

Guru Tegh Bahadur Institute of Technology

Guru Gobind Singh Indraprastha University

Dwarka, New Delhi

Session 2012-2013

DECLARATION

We hereby declare that all the work presented in the dissertation entitled "**Compiler Automation Tool**" in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in **Computer Science & Engineering,** Guru Tegh Bahadur Institute of Technology, affiliated to Guru Gobind Singh Indraprastha University Delhi is an authentic record of our own work carried out under the guidance of **Ms. Rekha Bhatia**.

Date:

Amarpreet Singh(08/CSE1/2009)
Gagandeep Singh(04/CSE1/2009)
Hargeet Kaur(03/CSE1/2009)
Harpreet Kaur(15/CSE1/2009)

CERTIFICATE

This is to certify that dissertation entitled **"Compiler Automation Tool",** which is submitted by **Mr. Amarpreet Singh, Mr. Gagandeep Singh, Ms. Hargeet Kaur, Ms. Harpreet Kaur** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in **Computer Science & Engineering**, Guru Tegh Bahadur Institute of Technology, New Delhi is an authentic record of the candidate's own work carried out by them under our guidance. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.

Rekha Bhatia
(Project Guide)

P.S. Bedi
(Head of Department)
Computer Science & Engineering

Date:

ACKNOWLEDGEMENT

We would like to express our great gratitude towards our supervisor, **Ms. Rekha Bhatia** who has given us support and suggestions as well as towards **Mr. Munshi Yadav,** our compiler teacher. Without their help we could not have presented this dissertation upto the present standard. We also take this opportunity to give thanks to all others who gave us support for the project or in other aspects of our study at Guru Tegh Bahadur Institute of Technology.

Date:

Amarpreet Singh
(08/CSE1/2009)
amarpreetsingh92@gmail.com

Gagandeep Singh
(04/CSE1/2009)
singh.gagan144@gmail.com

Hargeet Kaur
(03/CSE1/2009)
hargeet.kaur13@gmail.com

Harpreet Kaur
(15/CSE1/2009)
harpreetmarwah19@gmail.com

ABSTRACT

Compiler Automation Tool is a java based application, a tool for automating the task of compilation process. With CAT, a compiler designer can design compiler for any language using user-friendly graphical interface, by specifying regular expressions, transition diagram, context free grammar and various compiler specifications to generate a compiler model. This model is then used to compile a source code displaying intermediate processing results during each phase such as stream of tokens generated by lexical phase, parsing decisions etc. Such a tool will surely provide the designer a bright light over the language being designed and help to make strategy on how the compilation process must take place. CAT is compiler field specific tool and is useful for anyone related to compiler design, development as well as learning.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

INTRODUCTION

1.1 PROJECT INTRODUCTION

1.1.1 AIM

To create a tool for automating the task of compilation process and provide a blue print for compiler construction.

1.1.2 OBJECTIVE

A compiler is a special program that processes statements written in a particular programming language and turns them into machine language or "code" that a computer processes. It consists of various phases:

- Lexical analyzer takes the source program as an input and produces a long string of tokens
- Syntax Analyzer takes an out of lexical analyzer and produces a large tree.
- Semantic analyzer takes the output of syntax analyzer and produces another tree.
- Intermediate code generator takes a tree as an input produced by semantic analyzer and produces intermediate code.

Compiler is a specialized field in computer science and is regarded as one the most complex subject of all. Only few tools exist at present date such as lex, yacc etc that are widely used for compiler construction. All the decisions regarding compiler construction are done theoretically which are then incorporated into development using these tools. However, no such commercial tool exist that may automate the task and provide an environment for decision making.

Compiler Automation Tool fills the gap by proving an environment using which one can design a blue print for compiler on fly and mould decisions to fit it in the best way as possible. With automation, the designer get a quick view of how the compiler actions

must take place and what steps can be taken to optimize and increase compiler efficiency.

1.1.3 PURPOSE, SCOPE AND APPLICABILITY

This document aims at defining the overall software requirements for the project "**Compiler Automation Tool**". Efforts have been made to define the requirements exhaustively and accurately. The final product will have features/functionalities mentioned in this document and assumptions for any additional functionality/feature should not be made by any of the parties involved in developing/testing/implementing this product. In case it is required to have some additional features, a formal change request will need to be raised and subsequently a new release of this document and/or will be produced.

1.1.3.1 PURPOSE

This specification document describes the capabilities that will be provided by the system. It also states the various required constraints by which the system will abide.

- Compiler Automation Tool is a java based application, a tool for automating the task of compilation process. With CAT, a compiler designer can design compiler for any language using user-friendly graphical interface, by specifying regular expressions, transition diagram, context free grammar and various compiler specifications to generate a compiler model. This model is then used to compile a source code displaying intermediate processing results during each phase such as stream of tokens generated by lexical phase, parsing decisions etc. Such a tool will surely provide the designer a bright light over the language being designed and help to make strategy on how the compilation process must take place. CAT Compiler is also responsible for recording compilation process and generates statistical data highlighting the efficiency of the compiler.

1.1.3.2 SCOPE

The system once functional will give users a platform to model the programming language under consideration. He can specify various regular expressions, transition diagrams, context free grammar, syntax directed translation schemes etc to build a compiler model which will be used for compiling source code .Compilation process will be recorded and statistical data will be generated highlighting the efficiency of the compiler.

The system built is user friendly and is highly interactive.

1.1.3.3 APPLICABILITY

Compiler Construction Tool uses a three phase strategy. The complete project is divided into three separate but related applications – CAT Modeler, CAT Compiler and CAT Statistic Studio. All the application forms a consumer-producer environment with each other.

- 'CAT Modeler' covers all the specification regarding a compiler construction such as lexical, syntax, semantic etc. Thus, using 'CAT Modeler' the user models the programming language under consideration. He specifics various regular expressions, transition diagrams, context free grammar, syntax directed translation schemes etc to build a compiler model. Using these specifications, CAT Modeler generate Parse tree, Non Deterministic finite automata, Deterministic finite automata, syntax tree etc and provide simulation facility to test various test cases. In the end it produces a model file of extension .mdc which in turn will be used for compiling source code.

- The 'CAT Compiler' act as a consumer which uses the model file generated by CAT Modeler as well as source code and compiles it according to the

specifications. All processing by individual phases are displayed separately giving the user an idea about hoe the compilation decisions were made. Besides, this the user can also choose to keep the track of the performance of the code being complied on basis of time consumed in compiling and other information. Once, finished with several codes, the data gathered than can be passed to CAT Statistic Studio for analysis.

- The CAT Statistic Studio obtains the performance record as gathered by CAT Compiler and processes it to generate statistic information regarding the efficiency of the model. The gathered data can used to plot graphs to represent the data graphically and judge the behavior of the model. Moreover, the data can be saved into a CAT statistic file of extension .stc which can later be uploaded and viewed again. Besides this, CAT Statistic Studio can also be used to generate a well-organized structured report of format .pdf .

1.1.4 FEATURES

- The basic aim of the application is to automate the task of compilation and provide a blue print for compiler construction. Such an application is expected to be used in 'design' phase of compiler construction development cycle.
- The application that can be used to design wider set of programming languages.
- The application provides a user-friendly graphic interface to gather specification regarding compiler construction.
- The application promotes compiler construction learning and generates strategic decisions for compiler construction.
- The application can be used as a product that can be used in compiler labs for various universities.
- The application provides graphical representation of automata.

- No coding required from user side. In fact all it need is to just think, model, generate.
- The application is cross platform. It is based on paradigm “model once compile anywhere”.
- Moreover, record compiler performance and obtain statistics.

TECHNOLOGY USED

2.1 PROGRAMMING LANGUAGE USED

Java is a small, simple, safe, object oriented, interpreted or dynamically optimized, byte coded, architectural, garbage collected, multithreaded programming language with a strongly typed exception-handling for writing distributed and dynamically extensible programs. Java is a high-level, third generation language like C, FORTRAN, Small talk, Pearl and many others.

The following features make it one of the best programming language:

- It is simple and object oriented
- It helps to create user friendly interfaces.
- It is very dynamic.
- It supports multithreading.
- It is platform independent
- It is highly secure and robust.

We have used the following features of Java in sales Process Analyzer:

- **Swings:** Swing is the primary Java GUI widget toolkit. Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform.
- **Threads:** A thread is an independent path of execution within a program. Many threads can run concurrently within a program. Every thread in Java is created and controlled by the java.lang.Thread class.
- **JDBC:** This technology is an API for the Java programming language that defines how a client may access a database. It provides methods for querying and updating data in a database. JDBC is oriented towards relational databases. A

JDBC-to-ODBC bridge enables connections to any ODBC-accessible data source in the JVM host environment.

- **File Handling:** An I/O Stream represents an input source or an output destination. A stream can represent many different kinds of sources and destinations, disk files, devices, other programs, etc. Streams support many different kinds of data simple bytes, primitive data types, localized characters, and objects. Data is transferred to devices by streams.

2.2 ENVIRONMENT USED

2.2.1 NETBEANS IDE

Net Beans refers to both a platform framework for Java desktop applications, and an integrated development environment (IDE) for developing with Java, JavaScript, PHP, Python, Groovy, C, C++, Scala, Clojure, and others.

The Net Beans IDE is written in Java and can run anywhere a compatible JVM is installed, including Windows, Mac OS, Linux, and Solaris. A JDK is required for Java development functionality, but is not required for development in other programming languages.

The Net Beans platform allows applications to be developed from a set of modular software components called *modules*. Applications based on the Net Beans platform (including the Net Beans IDE) can be extended by third party developers.

2.2.2 ADOBE PHOTOSHOP CS5

Adobe Photoshop is a graphics editing program developed and published by Adobe Systems Incorporated. Adobe Photoshop is released in two editions: **Adobe Photoshop**, and **Adobe Photoshop Extended**, with the Extended having extra 3D image creation, motion graphics editing, and advanced image analysis features. Adobe Photoshop

Extended is included in all of Adobe's Creative Suite offerings except Design Standard, which includes the Adobe Photoshop edition.

Photoshop has ties with other Adobe software for media editing, animation, and authoring.

2.2.3 COREL DRAW X5

CorelDraw is a vector graphics editor developed and marketed by Corel Corporation of Ottawa, Canada. It is also the name of Corel's Graphics Suite.

Several innovations to vector-based illustration originated with CorelDraw: a node-edit tool that operates differently on different objects, fit text-to-path, stroke-before-fill, quick fill/stroke color selection palettes, perspective projections, mesh fills and complex gradient fills.

A full range of editing tools allow the user to adjust contrast, color balance, change the format from RGB to CMYK, add special effects such as vignettes and special borders to bitmaps.

CorelDraw is capable of handling multiple pages along with multiple master layers.

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

A Software Requirements Specification (SRS) describes a complete description of the behavior of a system to be developed. It will include a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS will also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation.

Software Requirements is basically a sub-field of Software engineering that deals with the elicitation, analysis, specification, and validation of requirements for software.

The **software requirement specification** (SRS) document will enlists all necessary requirements for project development. To derive the requirements we need to have clear and thorough understanding of the products to be developed. This is prepared after detailed communications with project team and the customer.

3.1 INTRODUCTION

The aim of the project is to develop a standalone application 'Compiler Automation Tool'. The application is required to take several inputs in the form of specifying regular expressions, transition diagram, context free grammar and various compiler specifications to generate a compiler model. This model is then used to compile a source code displaying intermediate processing results during each phase. The project is also required to generates statistical data highlighting the efficiency of the compiler in each phase. The following subsection provides an overview of entire SRS.

3.1.1 PURPOSE: This SRS intended for 'Compiler Automation Tool' and its purpose is to provide the detailed description of the application specifying 'what' it will do without specifying 'how' it will do.

The SRS is intended for the developer as well for the customer who has requested the development of the application. The purpose of this SRS is achieved only if the user is able to express his views and specification confidently about the application and the developer perceive them correctly.

The user of this application will be a person having knowledge basic knowledge of tokens, regular expressions, parse trees and other intermediate stages of the compilation process.

3.1.2 SCOPE: The application is intended to cover all the specification regarding a compiler construction such as lexical, syntax, semantic etc. A modeler will be built by using the specified regular expressions, transition diagrams, context free grammar, syntax directed translation schemes etc. Thus this modeler will be used to model the programming language under consideration. Using these specifications, CAT Modeler will generate Parse tree, Non Deterministic finite automata, Deterministic finite automata, syntax tree etc. and provide simulation facility to test various test cases. In the end it will produces a model file of extension .mdc which in turn will be used for compiling source code.

3.1.3 OVERVIEW: The remainder of this document is in two chapters, the first providing a full description of the application. It lists all the functions performed by the system. The final chapter concerns details of each of the application functions and actions in full for the application developers' assistance. These two sections are cross-referenced by topic, to increase understanding by both groups involved.

3.2 OVERALL DESCRIPTION

This section of SRS describes the general factors that may affect the application and its requirements. It does not contain any specific requirement instead it is meant for providing background for those requirements which are discussed in details in later section making them easier to understand.

3.2.1 APPLICATION PERSPECTIVE: The 'Compiler AutomationTool' will be an independent and totally self-contained application that does not require development of any additional application. On the other hand, since the

application will be java based, it will require Java Runtime Environment and a java enabled browser to run the applet.

The following subsections describe how the application operates inside various constraints.

3.2.1.1 INTERFACES: The interface of the application will be a major focus of the development. Less attention will be required on algorithmic part as compared to Graphical User Interface. The application will not be intended for all kinds of users. Rather it will be confined for specific compiler-knowing users thus it must be compact, simple and easy to understand so that fresh user quickly adapt to it. Thus as a whole, the application is expected to be extremely user friendly and should not be complex, annoying users rather must create a good impression on their minds.

3.2.1.2 HARDWARE INTERFACE: No typical hardware component will be required to run this application. The stand alone application will be meant to run on any personal computer, laptops or even Notebooks (only) that have preinstalled Java Runtime Environment.

3.2.1.3 SOFTWARE INTERFACE: The following table describes the various additional software requirements:

Table 3.1 − Additional software required

Name	**Purpose/ Interface with application**	**Version Number**	**Source**
Java ™ Platform SE or Java Runtime Environment (JRE)	Java Platform, Standard Edition or Java SE is a widely used platform for programming in the Java language. It is the Java Platform used to deploy portable applications for general use. In practical	6 or higher version	Web Link : www.oracle.com/technetwork/java/javase/downloads/index.html Others: If not installed, the

	terms, Java SE consists of a virtual machine, which must be used to run Java programs, together with a set of libraries (or packages) needed to allow the use of file systems, networks, graphical interfaces, and so on, from within those programs.		browser will automatically download and install Java ™ Platform SE. This will however take some time to download and install.
Sigar	Hyperic's System Information Gatherer (SIGAR) is a cross-platform API for collecting software inventory data. SIGAR is core of HQ's auto-discovery functionality, and you can use it to extend auto-discovery behavior. SIGAR includes support for Linux, FreeBSD, Windows, Solaris, AIX, HP-UX and Mac OSX across a variety of versions and architectures. Users of the SIGAR API are given portable access to inventory and monitoring data.	1.6.4	http://www.hyperic.com/ http://code.google.com/p/magelan/downloads/detail?name=hyperic-sigar-1.6.4.zip&can=2&q=
JGraphX (JGraph 6)	The Java flavour of mxGraph is called JGraphX and starts from version 1.x, but think of it as JGraph 6 if that is easier. JGraphX enables you to produce Java Swing applications that feature interactive diagramming functionality. The core client functionality of JGraphX is a Java 5 compliable library that describes, displays and interacts with diagrams as	Version 6	http://jgraph.github.com/mxgraph/docs/manual_javavis.html

	part of your larger Java Swing application. JGraphX is primarily designed for use in a desktop environment, although Java does have web enabling features making it possible to deploy JGraphX in web environment.		

3.2.2 APPLICATION FUNCTIONS: The function of the application is simple and straight forward. The application will automate the task of compilation and provide a blue print for compiler construction. It may be used in 'design' phase of compiler construction development cycle. All processing by individual phases will be displayed separately which will give an insight view of how the compilation takes place. This way it can be used to design wider set of programming languages. CAT Compiler will also record compilation process and generate statistical data highlighting the efficiency of the compiler.

It can also be used to promote compiler construction learning and generate strategic decisions for compiler construction.

3.2.3 USER CHARACTERISTICS: The application is intended for the customer who has detailed knowledge of the various phases of the compilation process of a source code in any language. A proper specific education level, experience and technical expertise are required. However, just basic knowledge of the subject may help the user for learning the details of a compilation process.

3.3 SPECIFIC REQUIREMENTS

This section contains all the requirements at a level of detailed sufficient to enable designers to design an application to satisfy those requirements and testers to test that the application satisfies those requirements. Throughout this section, every stated

requirement should be perceived by users and operators. These requirements should include at a minimum description of every input into the application, every output from the application and all functions performed by the system in response to an input or in support of an output.

3.3.1 NON FUNCTIONAL REQUIREMENTS: This subsection contains a detailed description of all inputs and outputs for the application :

Table 3.2 – List of input data items

Name of item	Data Type	Valid Range	Source of input	Unit	Description
Programing Language	String	–	TextField	–	Name of the programming Language
Model Version	String	–	TextField	–	Model Version
Author	String	–	TextField	–	Name of the Modeler
Extensions	String	–	TextField	–	Various Extension supported by the language
Description	String	–	TextField	–	Description/ comments
Priority	Integer Array	–	Table	–	Priority for lexical analysis
Input Sets	Set	–	Table	–	Set Space
Keywords	Array of Strings	–	Table	–	Set of Keywords
Identifier Regular Expression	String	Valid Regular expression	TextField	–	Regular Expression for Identifier
Integer Regular Expression	String	Valid Regular expression	TextField	–	Regular Expression for Integer
Character Regular Expression	String	Valid Regular expression	TextField	–	Regular Expression for Character

Floating pt. Regular Expression	String	Valid Regular expression	TextField	–	Regular Expression for floating points
String Regular Expression	String	Valid Regular expression	TextField	–	Regular Expression for Strings
Punctuators	2D array of String	–	Table	–	Set of punctuators
Operators	Array of Transition Diagram	–	List and Table	–	Transition diagram data for various operators such as arithmetic, relational etc.
Other operators	Array of Strings	–	Table	–	Other operators
Comments	Transition diagram	–	List and table	–	Transition diagram for comments
Source Code	String	–	TextArea	–	Source Code

Table 3.3 – List of output data items

Name of item	Data Type	Source of output	Description
Identifier DFA	Graph	Table	Deterministic finite Automata for identifier
Integer DFA	Graph	Table	Deterministic finite Automata for integer
Character DFA	Graph	Table	Deterministic finite Automata for Character
String DFA	Graph	Table	Deterministic finite Automata for String
Floating point DFA	Graph	Table	Deterministic finite Automata for floating point.
Set of Tokens	List of Strings	TextArea	A set of tokens in form of string corresponding to source code.
Symbol Table	Table	Table	Symbol Table corresponding to source code
Error	Table	Table	Various errors in source code
Sales List	String	Label	Qualified Sales List

3.3.2 FUNCTIONAL REQUIREMENTS: This subsection defines the fundamental actions that must take place in the application in accepting and processing the inputs and in processing and generating the outputs.

3.3.2.1 VALIDITY CHECK ON THE INPUTS: The input to the application would be provided using graphical components such as TextFields. Such component takes input in form of String which later may be converted to proper data type as required. In such a case validity check is important before the inputs are processed. The table discussed above provides the necessary conversion required while fetching data from respective TexFields. Java provides an excellent way of converting data types and handling any exception occurring while conversion.

3.3.3 APPLICATION ATTRIBUTES: There are a number of quality attributes of the application that can serve as requirements for instance:

3.3.3.1 RELIABILITY: Once on live, the application must be extremely reliable. It must produce correct results independent of time. There must be no scope of user to input invalid data causing abnormal behavior. Moreover, any error occurring in any kind of input entry must immediately prompt.

3.3.3.2 SECURITY: The application must be designed by taking into account the security aspects of the application. Efforts must been taken to make the application as secure as possible, eliminating any possibility of unauthorized copying and editing.

3.3.3.3 MAINTAINABILITY: No maintenance is required if the product is not disturbed by the customer i.e. if all files are intact with the application in the similar way it was delivered.

3.4 CHANGE MANAGEMENT PROCESS

The application must be designed to be flexible enough to adapt normal changes in specification such as little changes in GUI layout, changes in investment heads etc. However, it must be noted that only minor changes will be acceptable. Change in algorithm or addition of company's other product's features will cause complete change in structure of the project. Nonfunctional as well as functional part of the SRS has to be modified. This will consequently result in redesign of the algorithm as well as the layout of the application which may cause project to deviate from its intended schedule.

3.5. DOCUMENT APPROVAL

This is to declare that all the descriptions, requirements and specification stated in this SRS have been discussed and approved by both the customer and the developer. The customer agrees that the above state description is his expectation from the application and will satisfy his demand. On the other hand, the developer promises to perceive all the information stated and develop the application to its utmost accuracy and deliver it on time.

Moreover, the customer also agrees that unauthorized Copying, Reproduction, Rental, Broadcasting of this application is violation of applicable law. This application will be intended solely for customer's own organization. The customer may not decompile, reverse engineer, or disassemble the application, except as permitted by law.

SYSTEM DESIGN

4.1 CLASS DIAGRAM

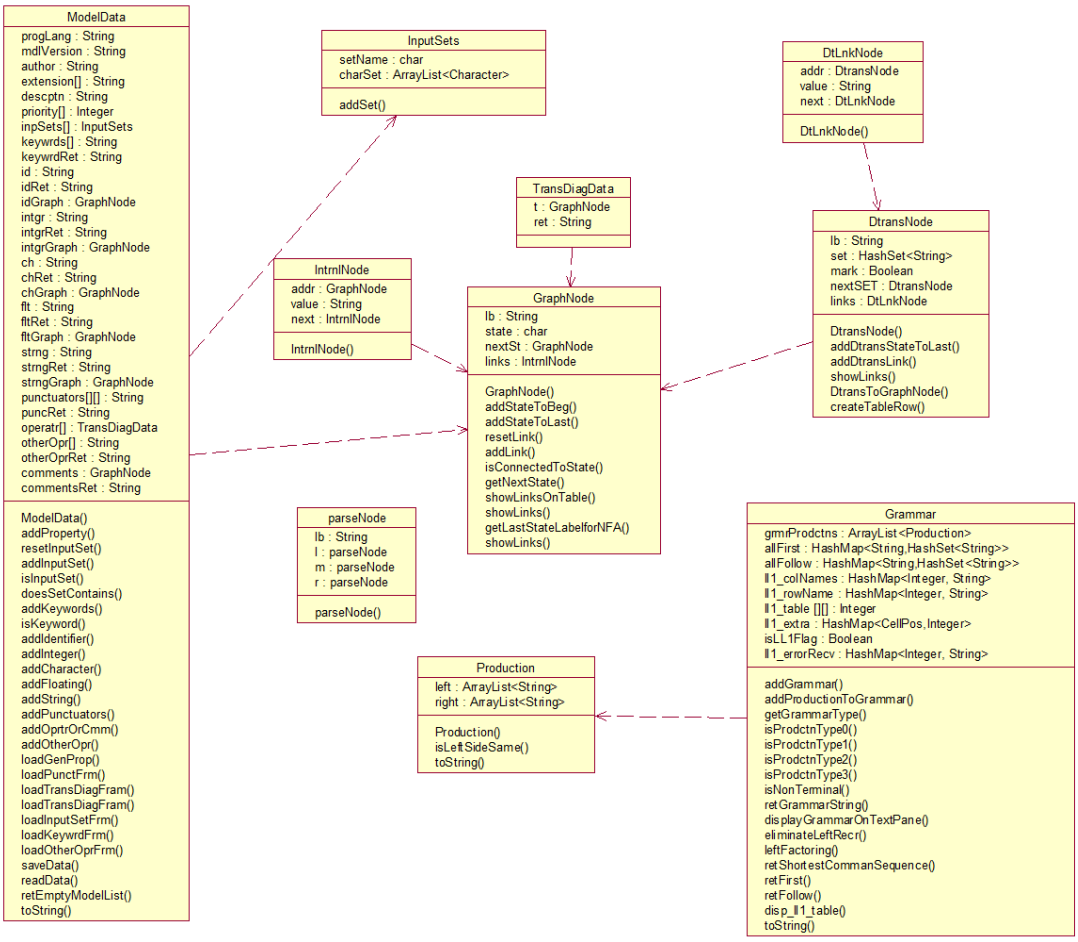


Fig 4.1 – Class Diagram

The above class diagram describes fundamentals classes used by the program to store data and methods for processing them.

- **GraphNode :** Class that describes the a single node in graph.
- **TransDiagData :** Transition diagram data that contains sets of GraphNodes and links
- **DtransNode :** Nodes corresponding to DTrans Table
- **Production :** Represents a production of the grammar containing left and right sides
- **Grammar :** Contains set of Productions and various methods to process them
- **ModelData :** Contains all model data of the user.

4.2 CAT ARCHITECTURE

Compiler Automation Tool uses a three phase strategy. The complete project is divided into three separate but related applications – CAT Modeler, CAT Compiler and CAT Statistic Studio. All the application forms a consumer-producer environment with each other. In fact, the blend of these three different application makes language modeling easier, fun and exciting.

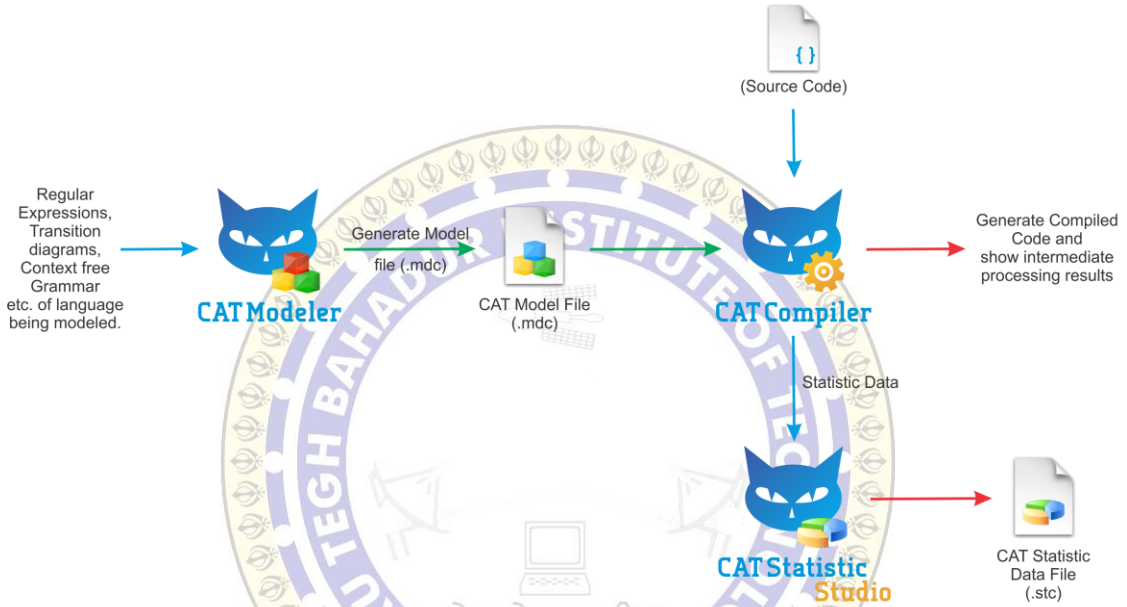


Fig 4.2 – CAT Architecture

- 'CAT Modeler' covers all the specification regarding a compiler construction such as lexical, syntax, semantic etc. Thus, using 'CAT Modeler' the user models the programming language under consideration. He specifics various regular expressions, transition diagrams, context free grammar, syntax directed translation schemes etc to build a compiler model. Using these specifications, CAT Modeler generate Parse tree, Non Deterministic finite automata, Deterministic finite automata, syntax tree etc and provide simulation facility to test various test cases. In the end it produces a model file of extension .mdc which in turn will be used for compiling source code.

- The ‘CAT Compiler’ act as a consumer which uses the model file generated by CAT Modeler as well as source code and compiles it according to the specifications. All processing by individual phases are displayed separately which gives an insight view of how the compilation took place. CAT Compiler is also responsible for recording compilation performance information to be send to CAT Statistic Studio for analysis.

- The CAT Statistic Studio is responsible of providing information regarding the efficiency of the model that was designed and used. Its receives rich set of data from CAT Compiler regarding compilation performance based on time taken to compile entire code, time taken at each phase and other such information. These information are then processed and represented graphically by plotting over a graph with selected metrics. The slop, rise or fall helps in concluding compile performance and promotes analytical learning. Moreover, the CAT Statistic Studio can also be used to summarize all detail into a pdf file in an organized way as well as allows user to gather all the files he used such as model files, statistic files, source code etc. into a single folder that is easy to be transferred from one machine to another.

4.3 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. The various Activity Diagrams for the Sales Process Analyzer are as follows:

4.3.1 SIMULATION OF DFA

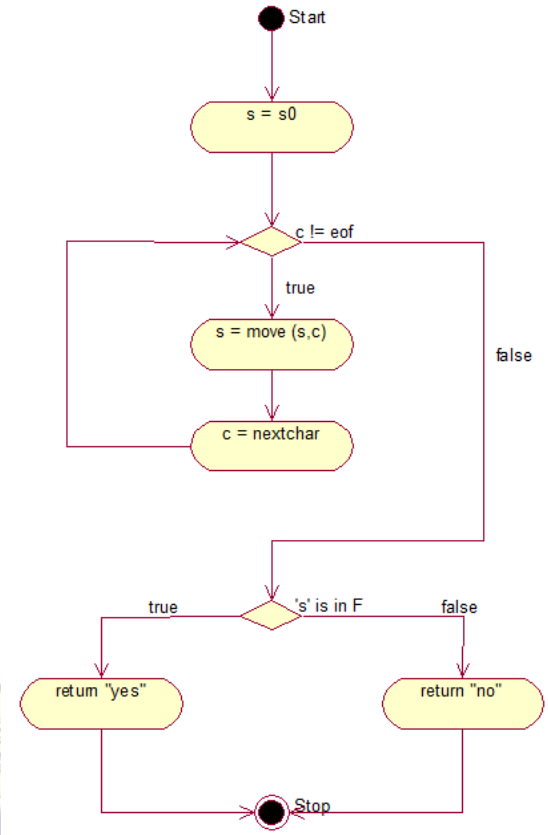


Fig 4.3 – Simulation of DFA

A deterministic finite automaton (DFA) is a special case of an NFA where:

1. There are no moves on input ε, and

2. For each state s and input symbol a, there is exactly one edge out of s labeled a.

INPUT: An input string x terminated by an end-of-file character eof. A DFA D with start state so, accepting states F, and transition function moue.

OUTPUT: Answer "yes" if D accepts x; "no" otherwise.

METHOD : Algorithm is applied to the input string x. The function move(s, c) gives the state to which there is an edge from state s on input c. The function next Char returns the next character of the input string x.

4.3.2 COMPUTATION OF ε-CLOSURE

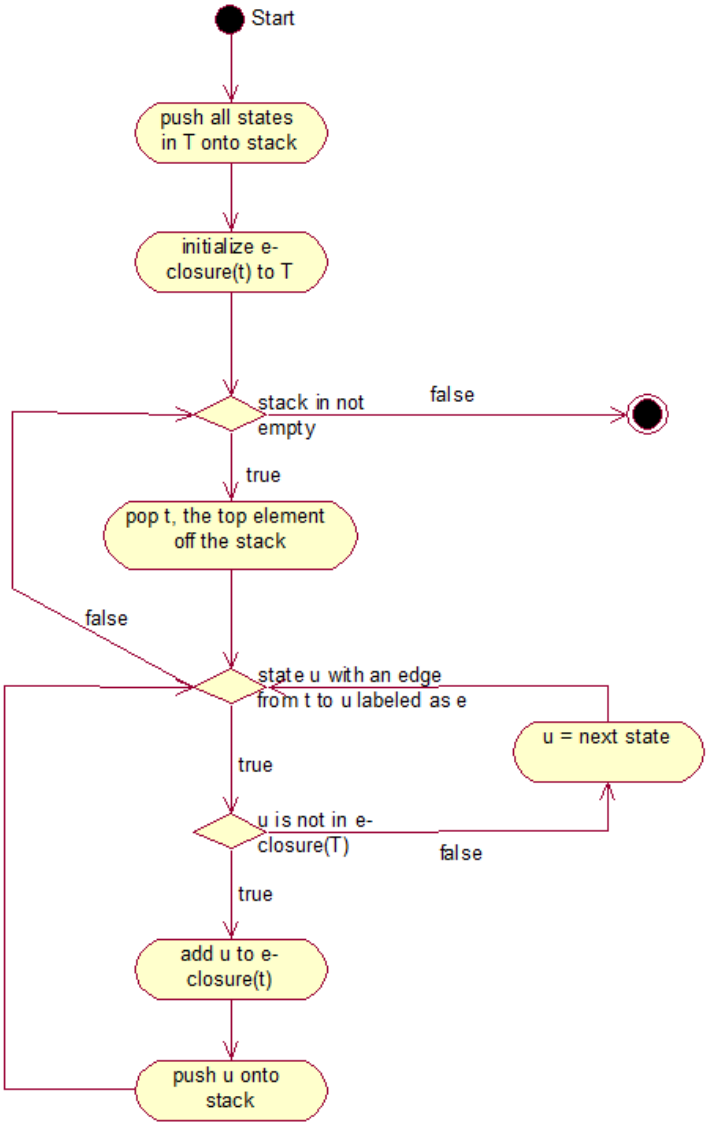


Fig 4.4 – Activity Diagram for computation of ε-closure

- ε-closure(s) is method that returns set of NFA states reachable from NFA state s on ε – transition alone.
- This method is used in conversion from NFA to DFA.
- The computation of ε-closure(T) is a typical process of searching a graph fro nodes reachable from a given set of nodes. In this case the sates of T are the given set of nodes and the graph consist of just the-labeled edges of the NFA.

- A simple algorithm to compute ε-closure(T) uses a stack to hold states whose edges have not been checked for ε-labeled transitions.

4.3.3 CONTRUCTION OF DFA FROM NFA

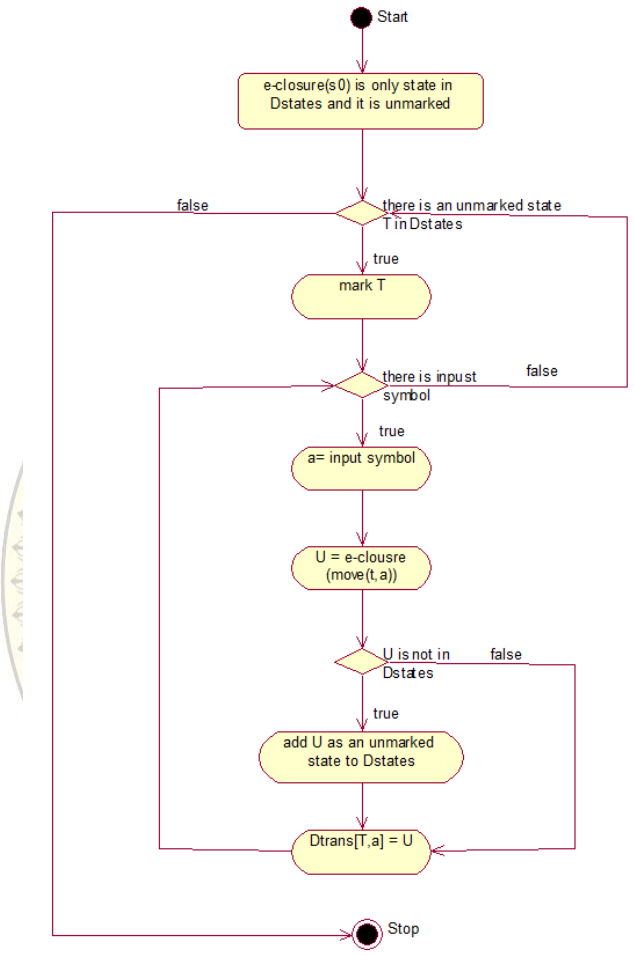


Fig 4.5 – Activity Diagram for Constructing a DFA from an NFA

The general idea behind the subset construction is that each state of the constructed DFA corresponds to a set of NFA states. After reading input ala2 - -. an, the DFA is in that state which corresponds to the set of states that the NFA can reach, from its start state, following paths labeled ala2 . . an.

INPUT: An NFA N.

OUTPUT: A DFA D accepting the same language as N.

METHOD: Algorithm constructs a transition table Dtran for D. Each state of D is a set of NFA states, and we construct Dtran so D will simulate "in parallel" all possible moves N can make on a given input string. s is a single state of N, while T is a set of states of N.

OPERATION	DESCRIPTION
ε-closure(s)	Set of NFA states reachable from NFA state s I in set T on €-transitions alone
ε-closure(T)	Set of NFA states reachable from some NFA state s in set T on €-transitions alone; = Us in T ε-closure(s).
move(T,a)	Set of NFA states to which there is a transition on input symbol a from some state s in T.

- The algorithm constructs a transition table Dtran for D. Each DFA state is a set of NFA states and we construct Dtran so that D will simulate “in parallel” all possible moves N can make on a given input string.
- The algorithm starts with initial state S0 in Dstates as unmarked and makes it way by calculating moves and ε-closure generating new states.
- Each state in DFA corresponds to set of NFA satates.

4.3.4 AN NFA FROM A REGULAR EXPRESSION (THOMPSON'S CONSTRUCTION)

The algorithm is syntax-directed, in the sense that it works recursively up the parse tree for the regular expression. For each subexpression the algorithm constructs an NFA with a single accepting state.

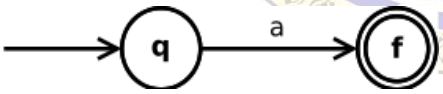
INPUT: A regular expression r over alphabet C.

OUTPUT: An NFA N accepting L(r).

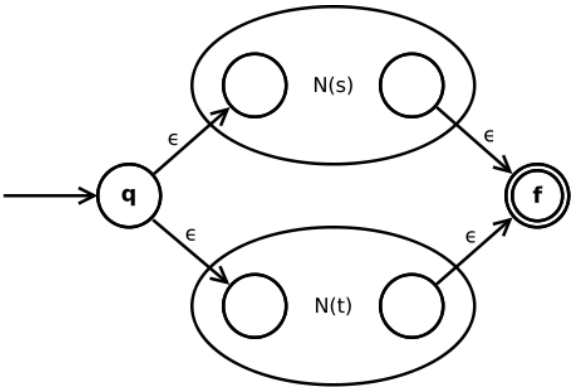
METHOD: Begin by parsing r into its constituent subexpressions. The rules for constructing an NFA consist of basis rules for handling subexpressions with no operators, and inductive rules for constructing larger NFA's from the NFA's for the immediate subexpressions of a given expression. The following rules guide the construction of NFA :

- The **expression** ϵ is converted to

- A **symbol** a**of the input alphabet** is converted to

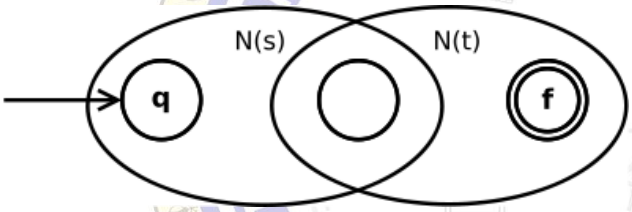


- The **union expression** $s|t$ is converted to



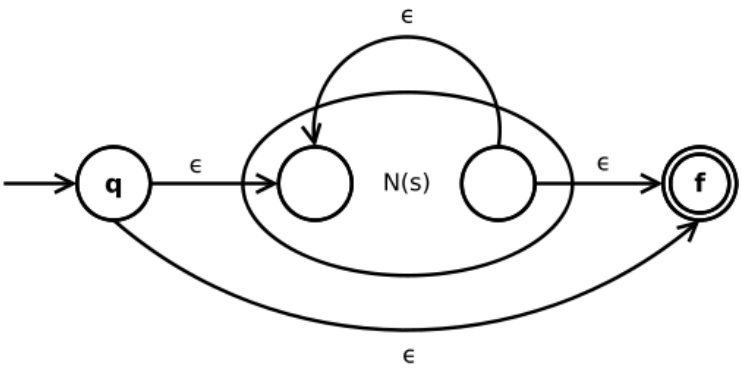
State q goes via ε either to the initial state of $N(s)$ or $N(t)$. Their final states become intermediate states of the whole NFA and merge via two ε-transitions into the final state of the NFA.

- The **concatenation expression** st is converted to



The initial state of $N(s)$ is the initial state of the whole NFA. The final state of $N(s)$ becomes the initial state of $N(t)$. The final state of $N(t)$ is the final state of the whole NFA.

- The **Kleene star expression** s^* is converted to



An ε-transition connects initial and final state of the NFA with the sub-NFA $N(s)$ in between. Another ε-transition from the inner final to the inner initial state of $N(s)$ allows for repetition of expression s according to the star operator.

- The **parenthesized expression** (s) is converted to $N(s)$ itself.

4.3.5 ELIMINATION OF LEFT RECURSION

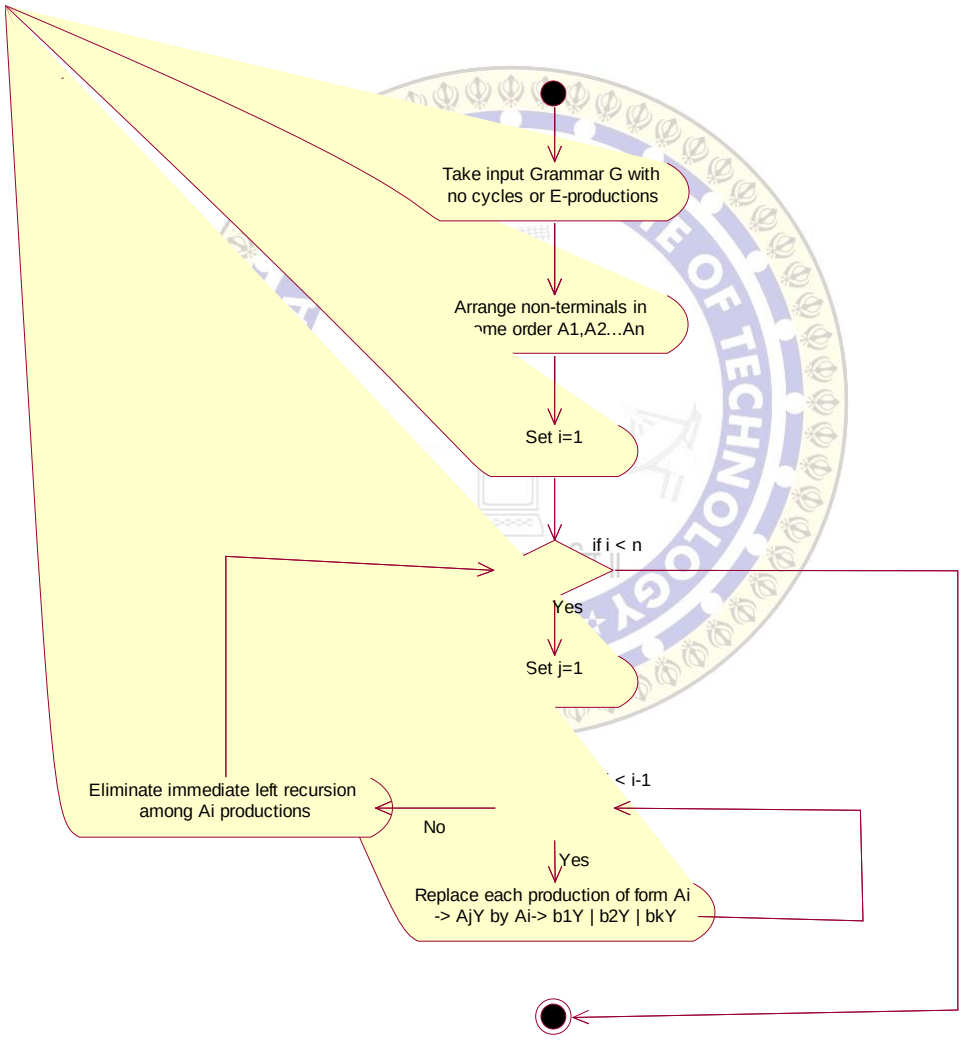


Fig 4.6 – Activity Diagram for Elimination of Left Recursion

A grammar is left recursive if it has a nonterminal A such that there is a + derivation A * Aα for some string a. Top-down parsing methods cannot handle left-recursive grammars, so a transformation is needed to eliminate left recursion.

INPUT: Grammar G with no cycles or e-productions.

OUTPUT: An equivalent grammar with no left recursion.

METHOD: The resulting non-left-recursive grammar may have E-productions.

4.3.6 LEFT FACTORING

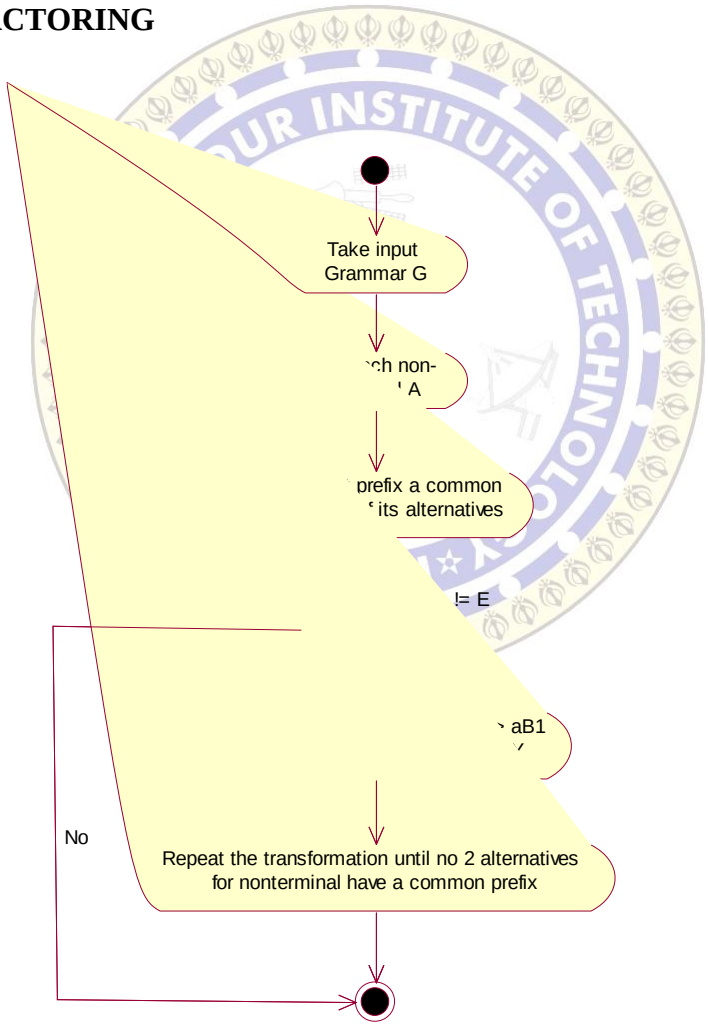


Fig 4.7 – Activity Diagram for Left Factoring

4.3.7 NONRECURSIVE PREDICTIVE PARSING

A nonrecursive predictive parser can be built by maintaining a stack explicitly, rather than implicitly via recursive calls. The parser mimics a leftmost derivation. If w is the input that has been matched so far, then the stack holds a sequence of grammar symbols α.

INPUT: A string w and a parsing table M for grammar G.

OUTPUT: If w is in L(G), a leftmost derivation of w; otherwise, an error indication.

METHOD: Initially, the parser is in a configuration with w$ in the input buffer and the start symbol S of G on top of the stack, above $.

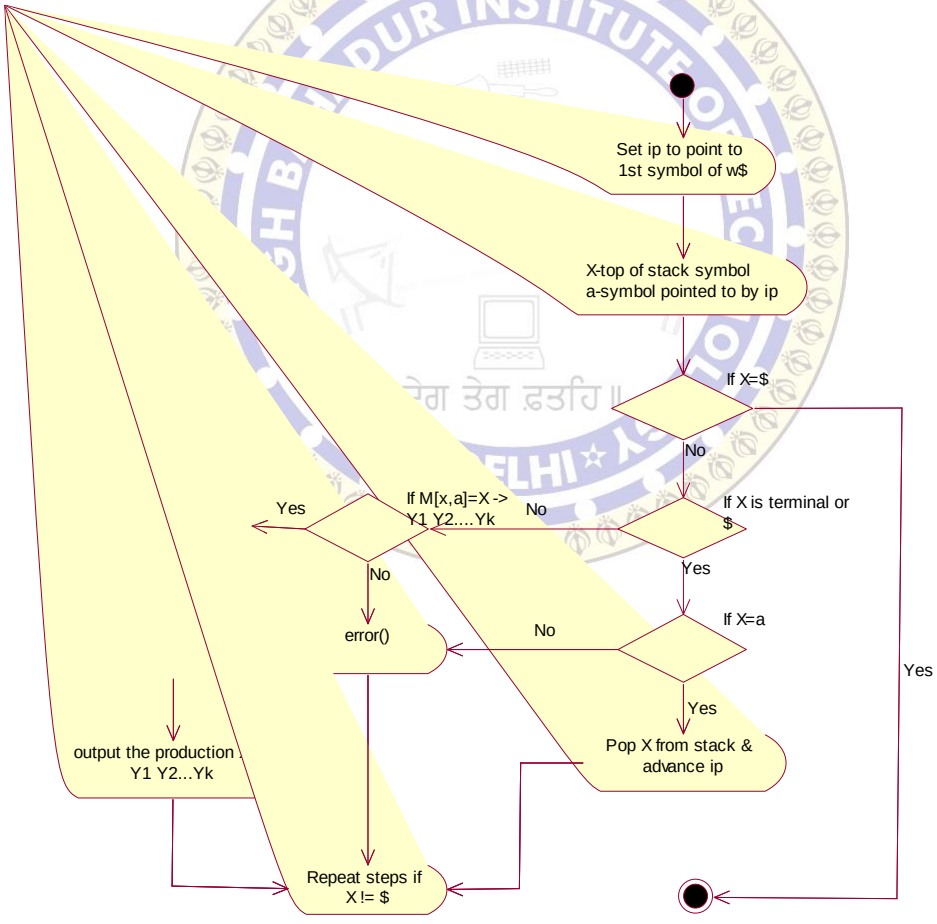


Fig 4.8 – Activity Diagram for Non Recursive Predictive Parsing

4.3.8 FIRST AND FOLLOW

To compute FIRS(X) for all grammar symbols X, apply the following rules until no more terminals or ε can be added to any FIRST set.

1. If X is terminal, the FIRST(X) is {X}
2. If X → ε is a production, then add to ε FIRST(X)
3. If X is a nonterminal and X → Y_1 Y_2 …. Y_k is a production, then place a in FIRST(X) if for some i, a is in FIRST(Y_i) and ε is in all of FIRST(Y_1)…...FIRST(Y_{i-1})

To compute FOLLOW(A) for all terminals A, apply the following rules until nothing can be added to any FOLLOW set.

1. Place $ in FOLLOW(S), where S is the start symbol and $ is the input right endmarker.
2. If there is a production A → αBβ, then everything in FIRST(β) except for ε is placed in FOLLOW(B).
3. If there is a production A → αB, or a production A → αB, or a production A→αBβ ehre FIRST(β) contains ε,then everything in FOLLOW(A) is in FOLLOW(B).

4.3.9 CONSTRUCTION OF LL(1) TABLE

Activities for constructing LL(1) table are as follows :

- The algorithm considers each production of the grammar of the form A → α and performs various computation.
- It consider each a in FIRST(α) and adds is to the table at A,a
- If ε is in FIRST(α) , A → α is added to the table at A ,b where b is terminal in FOLLOW(A).
- However, if ε is in FIRST(α) and $ in FOLLOW(A), A → α is added at A,$

- All the remaining empty entries are marked as error.

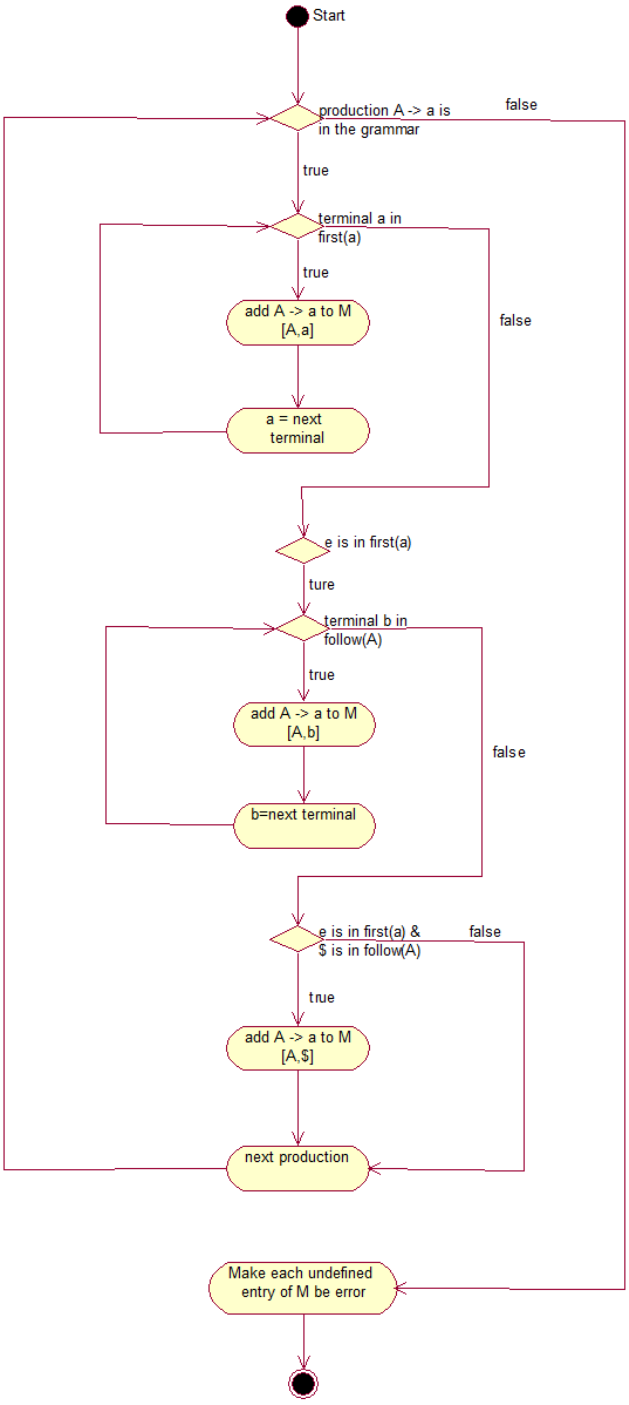


Fig 4.9 – Construction of LL(1) Table

4.3.10 CONSTRUCTION OF CLOSURE (SLR)

If I is a set of items for a grammar G, then CLOSURE(I) is the set of items constructed from I by the two rules:

1. Initially, add every item in I to CLOSURE(I).

2. If A -> α.Bβ is in CLOSURE(I) and B-> γ is a production, then add the item B-> .γ to CLOSURE(I) if, it is not already there. Apply this rule until no more new items can be added to CLOSURE(I).

A convenient way to implement the function closure is to keep a boolean array added, indexed by the nonterminals of G, such that added[B] is set to true if and when we add the item B-> .γ for each B-production B-> γ.

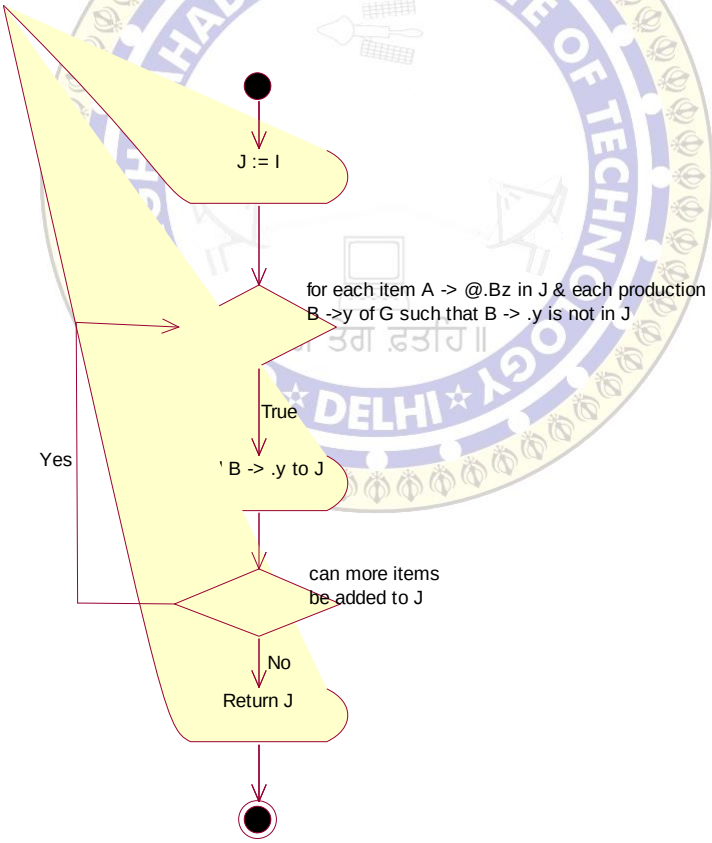


Fig 4.10 – Activity Diagram for computation of Closure (SLR)

4.3.11 CONSTRUCTION OF SLR SETS-OF-ITEMS

This algorithm is used to construct C, the canonical collection of sets of LR(0) items for an augmented grammar G'. Algorithm uses Goto Operation . GOTO(I,X) is defined to be the closure of the set of all items [A -> αX.β] such that [A -> α .Xβ] is in I, where I is a set of items and X is a grammar symbol.

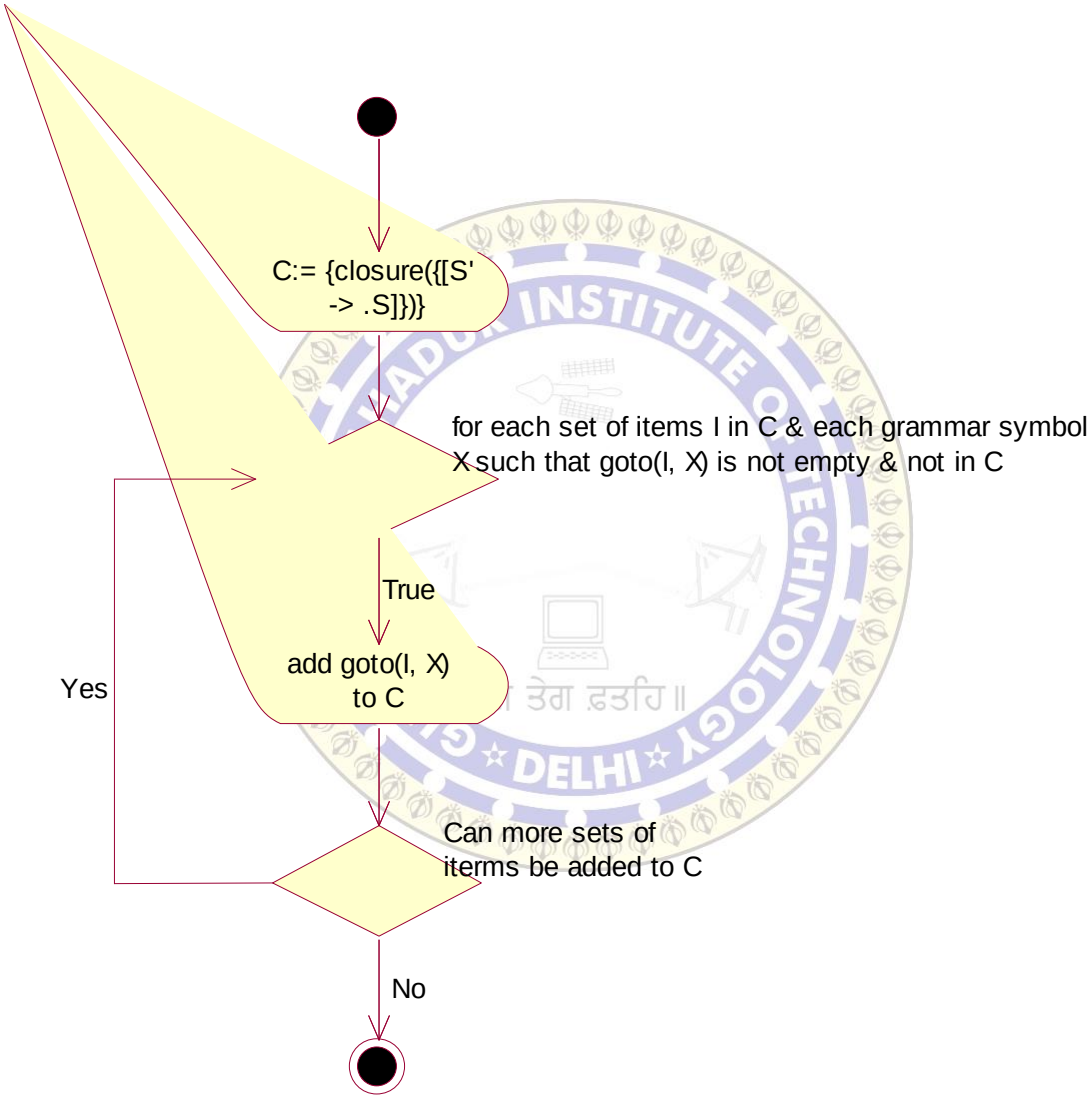


Fig 4.11 – Activity Diagram for constructing set-of-items of SLR

4.3.12 CONSTRUCTION OF SLR PARSING TABLE

The SLR method begins with LR(0) items and LR(0) automata,. That is, given a grammar, G, we augment G to produce G', with a new start symbol S'. From G', we construct C, the canonical collection of sets of items for G' together with the GOTO function.

INPUT: An augmented grammar G'.

OUTPUT: The SLR-parsing table functions ACTION and GOTO for G'.

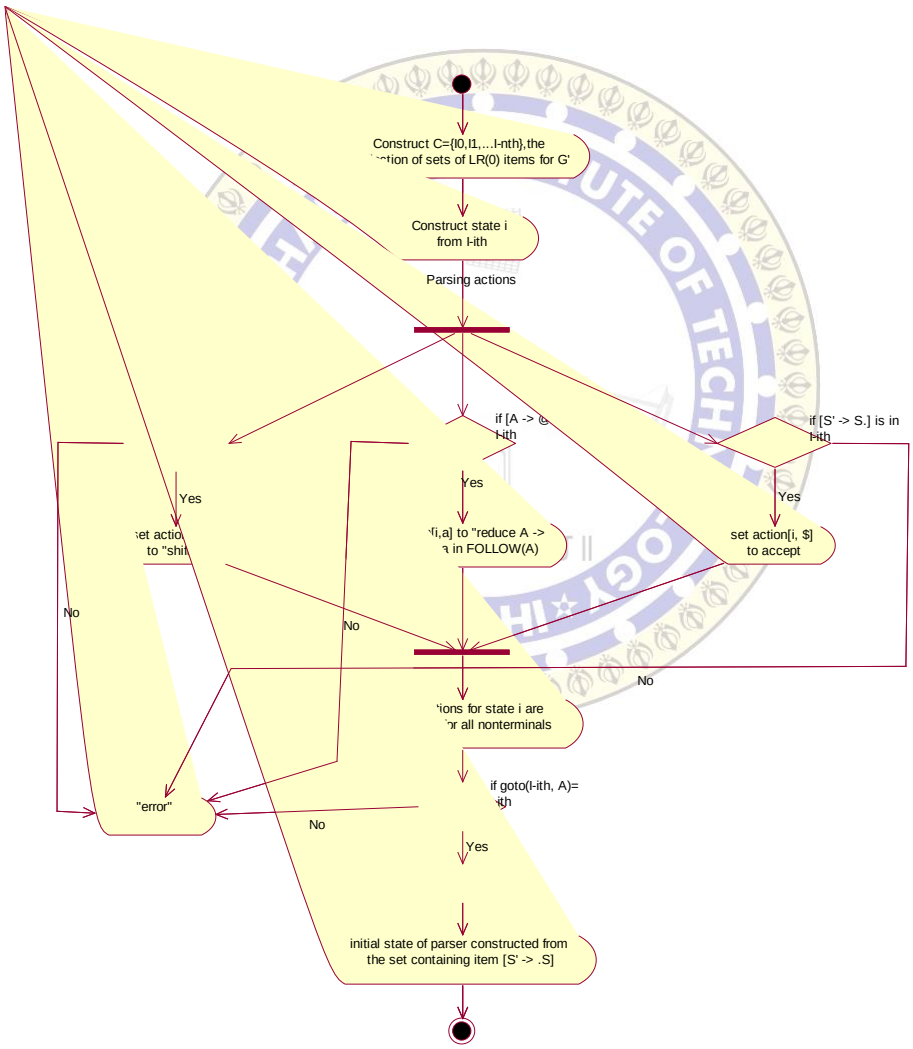


Fig 4.12 – Activity Diagram for constructing SLR pasrsing Table

4.3.13 CONSTRUCTION OF THE CANONICAL LR PARSING TABLE

The "canonical-LR" or just "LR" method, which makes full use of the lookahead symbol(s). This method uses a large set of items, called the LR(1) items**.** An LR parser using canonical LR(1) parsing table is called a canonical-LR(1) parser.

INPUT: An augmented grammar GI.

OUTPUT: The canonical-LR parsing table functions ACTION and GOT0 for G'.

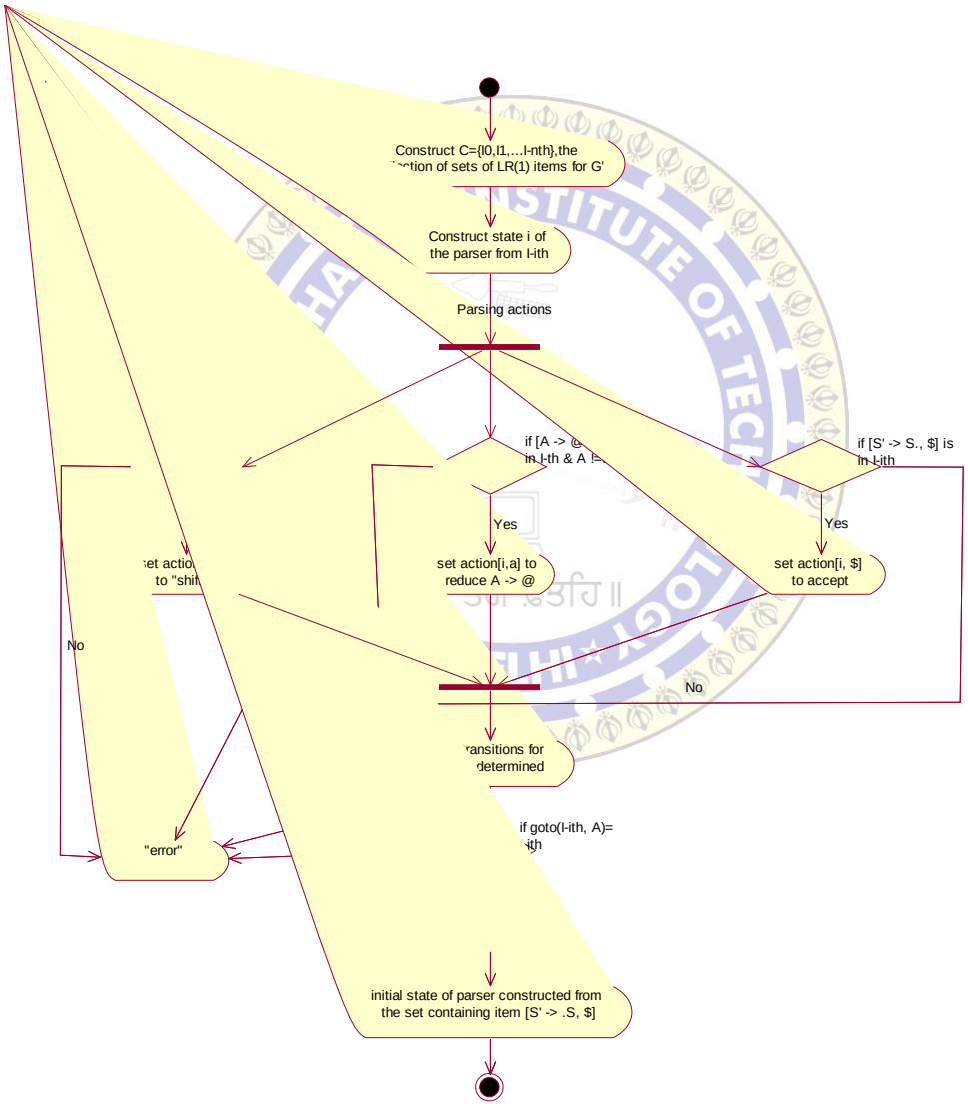


Fig 4.13 – Activity Diagram for constructing CLR parsing Table

4.3.14 LALR TABLE CONSTRUCTION

LALR table construction is easy but space consuming. This method is often used in practice, because the tables obtained by it are considerably smaller than the canonical LR tables, yet most common syntactic constructs of programming languages can be expressed conveniently by an LALR grammar.

INPUT: An augmented grammar G'.

OUTPUT: The LALR parsing-table functions ACTION and GOT0 for G'.

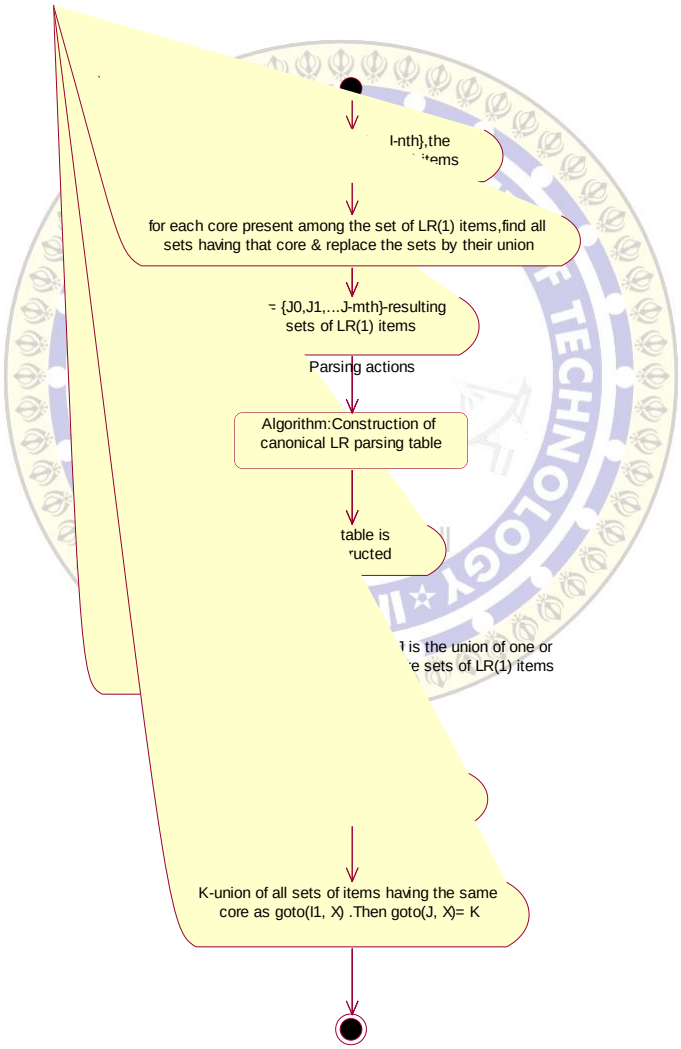


Fig 4.14 – Activity Diagram for constructing LALR parsing table

TESTING

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs (errors or other defects).

Software testing can be stated as the process of validating and verifying that a software program/application/product:

- meets the requirements that guided its design and development;
- works as expected
- Can be implemented with the same characteristics.

Different software testing techniques have been discussed in this section to investigate and verify that "Compiler Automation Tool" meets the various requirements and works efficiently as expected.

5.1 Functional Testing (Black Box Testing)

A functional testing is testing technique based on the functionality of the program and involves only observation of the output for certain input values. There is no attempt to analyze the code, which produces the output. We ignore the internal structure of the code and therefore, functional testing is also referred to as **black box testing** in which contents of the black box are not known. Functionality of the black box is understood completely in terms of input and output. Thus, in functional or black box testing, we are interested in functionality rather than internal structure.

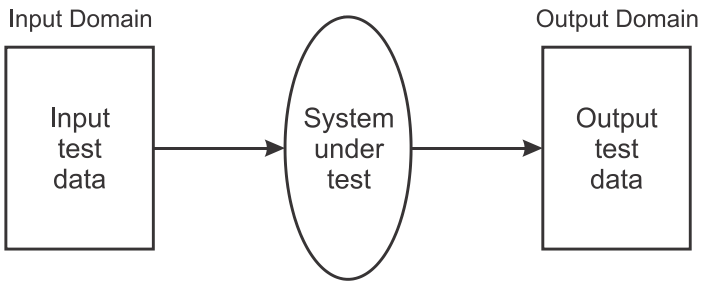


Fig 5.1 − Black box Testing

The following table describes the functional testing of various modules of Compiler Automation Tool:

Table 5.1 – Functional Testing

Test ID	**Test Case Scenario**	**Test Steps**	**Expected Results**	**Actual Result**	**Pass/ Fail**
Module 1 : Calculation of First and Follow					
01	To verify that the first of a grammar is calculated accurately.	(a) Enter the following grammar : S → U | V U → T a U | T a T V → T b V | T b T T→ a T b T | b T a T | ε	First(S) => {a, b} First(U) => {a, b} First(V) => {a, b} First(T) => {a, b, ε}	First(S) => {a, b} First(U) => {a, b} First(V) => {a, b} First(T) => {a, b, ε}	Pass
02		(b) Enter the	First(A) =>	First(A) =>	Pass

		following grammar : A → A c | B B → ε	{c, ε} First(B) => {ε}	{c, ε} First(B) => {ε}	
03	To verify that the follow of a grammar is calculated accurately.	(c) Enter the following grammar : A → A B C | a | ε B → b | ε C → c	Follow(A) => {b, c, $} Follow(B) => {c} Follow(C) => {b, c, $}	Follow(A) => {b, c, $} Follow(B) => {c} Follow(C) => {}	Fail(earlier when follow(A) was not being included in follow(C))
04				Follow(A) => {b, c, $} Follow(B) => {c} Follow(C) => {b, c, $}	Pass(now)
Module 2 : Eliminating Left Recursion in a Grammar					
05	To examine whether left recursion is removed from a grammar correctly.	(a) Enter a left recursive grammar. E → E + T | T T → T * F | F F → (E) | id	Displays the grammar after removal of left recursion. E → T E'	The application displays the correct grammar as output. E → T E' E' → + T E' | ε	Pass

			E' → + T E' | ε T → F T' T' → * F T' | ε F → (E) | id | ε	T → F T' T' → * F T' | ε F → (E) | id | ε	
06		(b) Enter a grammar without any left recursion. S → i E t S Z | a Z → e S | ε E → b	Displays the original grammar after elimination. S → i E t S Z | a Z → e S | ε E → b	Response as expected. S → i E t S Z | a Z → e S | ε E → b	Pass
07		(c) To verify that a right recursive grammar also produces the correct result. A → A A | a A | a	The application removes left recursion correctly. A → a A A' | a A' A' → A A' |	Result as desired. A → a A A' | a A' A' → A A' | ε	Pass

			ε		
08		(d) To verify that indirect left recursion is also eliminated. S → A a | b A → A c | S d | ε	The application must remove indirect left recursion. S → A a | b A → b d A' | A' A' → c A' | a d A' | ε	Result as expected. S → A a | b A → b d A' | A' A' → c A' | a d A' | ε	Pass
Module 3 : Left Factoring a Grammar					
09	To test the left factoring of a grammar.	(a) Enter a non deterministic grammar. S → a b B c | a b D c B → b | ε D → d | ε	Returns a left factored grammar. S → a b S' S' → B c | D c B → b | ε D → d | ε	Result as expected. S → a b S' S' → B c | D c B → b | ε D → d | ε	Pass
10		(b) Enter a non deterministic grammar.	Returns a left factored grammar.	Result obtained is wrong.	Fail (earlier when left factoring was done

		S → b S S a a S | b S S a S b | b S b | a	S → b S S" | a S" → S a S' | b S' → a S | S b	S → b S S a S' | b S b | a S' → a S | S b	in one step only and further the grammar obtained was not checked.)
11				S → b S S" | a S" → S a S' | b S' → a S | S b	Pass
Module 4 : LL(1) Table Generation					
12	To test that the LL(1) table being constructed for the given grammar is correct and	(a) Enter the following grammar : S → (A) | a A | ε B → b A | ε	The module must generate the LL(1) table as expected.	All entries in the table were accurate and complete.	Pass
13		(b) Enter the	The grammar	Though the	Fail

	shows duplicate entries in a cell if any).	following grammar : S → i E t S Z | a Z → e S | ε E → b	must generate table as expected with duplicate entries as the grammar is ambiguous.	grammar is not LL(1), the table did not show duplicate entries in a cell.	(duplicate entries must be handled efficiently)
14				All entries were accurate and duplicate entries were also shown.	Pass
Module 5 : SLR(1) Table Generation					
15	To test that the SLR(1) item sets, the DFA and the table being constructed for the given grammar is correct and shows duplicate entries in a	(a) Enter the following grammar : E → E + T | T T → T * F | F F → (E) | id	Correct item sets, DFA and SLR(1) table must be created.	Output as expected.	Pass
16		(b) Enter the following grammar : X → Y Z | a Y → b Z | ε	Correct item sets, DFA and SLR(1) table must be created.	Output as expected.	Pass

	cell if any).	$Z \rightarrow \varepsilon$			
Module 6 : CLR(1) Table Generation					
17	To test that the CLR(1) item sets, the DFA and the table being constructed for the given grammar is correct and shows duplicate entries in a cell if any).	(a) Enter the following grammar : $S \rightarrow x$ $S \rightarrow y$ $S \rightarrow z$ $S \rightarrow S + S$ $S \rightarrow S - S$ $S \rightarrow S * S$ $S \rightarrow S / S$ $S \rightarrow (S)$	30 LL(1) item states with accurate links in a DFA and a complete CLR(1) table for the corresponding grammar.	Wrong item sets and an incorrect CLR(1) table constructed.	Fail (left recursive grammars have to be handled separately while creating LR(1) item sets)
18				Output as expected.	Pass
19		(b) Enter the following grammar : $S \rightarrow C\ C$ $C \rightarrow e\ C \mid d$	Correct item sets, DFA for the states and CLR(1) table constructed.	Output as expected.	Pass
Module 7 : LALR(1) Table Generation					

20	To test that the LALR(1) item sets, the DFA and the table being constructed for the given grammar is correct and shows duplicate entries in a cell if any).	(a) Enter the following grammar : S → A a | b A c | B c | b B a A → d B → d	Correct LALR item sets and LALR table with reduce reduce conflict.	Output as expected.	Pass
21		(b) Enter the following grammar : S → A a | b A c | d c | b d a A → c	Correct LALR item sets and LALR table without any conflict.	Output as expected.	Pass

5.2 Non-functional Testing

Non-functional testing is the testing of a software application for its non-functional requirements. The names of many non-functional tests are sometimes often used interchangeably because of the overlap in scope between various non-functional requirements. Some of the non-functional testing of 'Compiler Automation Tool' are as follows:

5.2.1 Compatibility Testing: is testing conducted on the application to evaluate the application's compatibility with the computing environment. It determines how well software performs in a particular hardware / software / operating system / network environment and different combinations of above.

(a) Operating System and Processor: With respect to operating system and processor the compatibility of 'Compiler Automation Tool' can be studied on the bases of 'average boot-up time of the application'. Once successfully loaded the response time of the application is nearly negligible.

Table 5.2 – OS and Processor compatibility

Operating System	Processor	Rating (out of 5)
Windows 7 Ultimate	Intel® Core™ i3 CPU 540 @ 3.07GHz	5
Windows Xp SP2	Intel® Core™ i3 CPU 540 @ 3.07GHz	4
Windows Xp SP2	Intel® Pentium 4 @ 2.0 Ghz	3

Conclusion: The application is compatible with almost every operating system and processor. However, it is recommended to work with atleast Windows Xp as OS and Dual core processors.

5.3 Heuristic Testing

A heuristic is a commonsense guideline used to increase the probability of solving a problem by directing one's attention to things that are likely to matter. Software testers don't know exactly where the bugs are going to hide. However software testers know where bugs have been found before. We can use that knowledge of past discoveries and the nature of what we seek to create heuristics that help us narrow in on areas most likely to contain the treasure.

Heuristic-based testing may not give us concrete answers, but it can guide us to the important things to test. Heuristics can also be used in automation to provide information to guide human testers.

We have used the above explained heuristic approach in testing our application ‘Compiler Automation Tool’.

To use heuristics in testing, we have created a list of open-ended questions and guidelines. This is not just a pass/fail list of test criteria. It is not a list of specific test steps. Instead it can be used to guide our test scripting and exploration for bugs. We have developed general heuristics that we can apply to all our testing and specific heuristics that apply to specific applications.

LIMITATION AND FUTURE SCOPE

6.1 LIMITATIONS

Following are the limitations of the project:

- Although CAT is designed to model a large set of programming languages, yet there may be some languages that seem difficult to be described and study using CAT.
- CAT captures only general features of the language under consideration. There are certain specific features related to the languages that are difficult to model or perhaps CAT does not provide any way of modeling them.
- There exist certain limitations with respect to CAT Modeler. These are as follows:
  - Regular expressions does not take into account '?' (zero or one instance). Expressions using '?' have to be expanded using '|'. This increases the length and complexity of the expression.
  - Limited set space is available. User can use maximum of 25 (a-y) set.
  - Wastage of set space when specifying single character. For instance, when specifying quote in regular expression, user must first define a set containing only quote character. This causes wastage of limited set space.
  - Difficult to model operator whose nature depends upon the operands. For example, '>>' operator is regarded as shift operator as well as extraction operator in C++.
  - Does not allow multiple line strings. These are considered as errors later by CAT Compiler.
  - There is nearly no way for CAT to identify Ambiguous grammar entered by the user. Thus, the user must be very careful while entering grammar.
- CAT Compiler may take time to load long codes.

- No other language other than English is supported.

6.2 FUTURE SCOPE

This project has been designed to work on a large scale and in real time environments. According to our survey, there rarely exist any such software in the market that automates the compilation process and describes the internal working the process. In fact, this fact was the inspiration to design such software and fill the needy gap.

Once fully operational, CAT is can to be used as designing platform for new developing programming languages. This tool is expected to be widely used in design phase for any complier development. In addition, CAT is expected to gain popularity in educational institutes for providing learning grounds for compiler students and faculty. In fact, it is the aim of the team to deliver CAT as compiler lab software. Thus, CAT is a tool that will surely be useful for anyone who is related to compiler field.

So according to our analysis the future scope of this project is bright and is expected to be appreciated by whosoever uses this software.

In contrast, various features can be included in the project to increases its capabilities and accommodate to wider range of languages:

- The specific nature of languages can be consider in more intelligence way.
- The models designed by the user can be converted to specific language source code which can be directly used in compiler construction thereby saving users affords for separate implementation after he is satisfied with his model.

REFERENCES

- Herbert Schildt, "The Complete Reference Java – Seventh Edition", TataMcGraw-Hill , 2011.
- Katy Sierra and Bert Bates, "Head first Java" , 2nd Edition.
- Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman , "Compilers Principle, Techniques and Tools, Low Price Edition, Person Education Asia, 2002
- "Compiler Construction/Dealing with errors", *http://en.wikibooks.org/wiki/Compiler_Construction/Dealing_with_errors*
- *"*Lexical Analysis", *http://www.eecg.toronto.edu/~csc467/project/phase1.html*
- "Error Handlings", *http://www.cs.vassar.edu/~cs331/lexical-analyzer/error.html*
- *"*Compiler Errors*", http://bcook.cs.georgiasouthern.edu/windowsce/cpp3.htm*
- "ASCII Codes for the Greek Alphabet", *http://ancienthistory.about.com/od/greeklanguage/a/ASCIIGreek.htm*
- "ASCII Code List for numpad Characters", *http://chexed.com/ComputerTips/asciicodes.php*
- *"*HTML Symbols Entities Reference*", http://www.ascii-code.com/html-symbol.php*
- *"*Formal Languages and Grammer*", http://www.cse.ohio-state.edu/~gurari/theory-bk/theory-bk-onese2.html*
- *"*Context-free grammer", *http://en.wikipedia.org/wiki/Context-free_grammar*
- "Left Recursion and Left Factoring", *http://cs.stackexchange.com/questions/2696/left-recursion-and-left-factoring-which-one-goes-first*
- *"*How to left factor context-free grammer*", http://stackoverflow.com/questions/9540117/how-to-left-factor-a-context-free-grammar*
- *"*Left Factoring*", http://code.google.com/p/blacc/wiki/DesignCommonLeft*
- *"*Program to compute first and follow of a grammer*", wiki.answers.com/Q/Program_to_compute_first_and_follow_of_a_grammar*

- *"*C code to implement first and follow*",* *http://wiki.answers.com/Q/C_code_to_implement_first_and_follow*
- *"*Type 3 grammer-Regular expression", *http://stackoverflow.com/questions/8826802/type-3-grammar-regular-expression*
- *"*Operator-precedence Parsing*",* *http://zeus.cs.pacificu.edu/ryand/cs480/2005/ch4g.html*
- "Syntax and Semantics", *http://cseweb.ucsd.edu/classes/sp05/cse130/lecture_notes/syntax_semantics_student.txt*
- "JDesktopPane-how to get active frame", *http://stackoverflow.com/questions/8542609/jdesktoppane-how-to-get-active-frame*
- "JInternalFrames, JDesktopPane, and DesktopManager 2*",* *http://www.java2s.com/Code/Java/Swing-JFC/InterestingthingsusingJInternalFramesJDesktopPaneandDesktopManager2.htm*
- "How to remove row being edited from JTable", *http://stackoverflow.com/questions/6349386/remove-row-being-edited-from-jtable*
- "How to add line numbers to JTextArea", *http://www.javaprogrammingforums.com/java-swing-tutorials/915-how-add-line-numbers-your-jtextarea.html*
- *"*Text Component Line number", *http://tips4java.wordpress.com/2009/05/23/text-component-line-number/*
- "How to add text different color on JTextPane", *http://stackoverflow.com/questions/6068398/how-to-add-text-different-color-on-jtextpane*
- "How to Use Formatted Text Fields", *http://docs.oracle.com/javase/tutorial/uiswing/components/formattedtextfield.html*
- "How to create Adobe like TabbedPane in Swing", *http://www.jroller.com/santhosh/date/20050617#adobe_like_tabbedpane_in_swing*

- "How to import a Text file content to a JTextArea in a Java application", *http://stackoverflow.com/questions/7033589/how-to-import-a-text-file-content-to-a-jtextarea-in-a-java-application*
- "JTextPane appending a new string", *http://stackoverflow.com/questions/4059198/jtextpane-appending-a-new-string*
- "How to get multiple file extensions in the drop-down box", *http://stackoverflow.com/questions/2450223/jfilechooser-multiple-file-filters*
- *"*Java Access Modifiers*",* *http://www.tutorialspoint.com/java/java_access_modifiers.htm*
- "How to copy an object in Java", *http://stackoverflow.com/questions/869033/how-do-i-copy-an-object-in-java*
- "How to save a text from a jTextArea (ie Save As) into a new .txt file", *http://stackoverflow.com/questions/9690686/save-a-the-text-from-a-jtextarea-ie-save-as-into-a-new-txt-file*
- "How to hide JTable Column Headers", *http://tech.chitgoks.com/2010/03/01/hide-jtable-column-headers/*
- *"*Time in 12-hr format*", http://www.java2s.com/Tutorial/Java/0040__Data-Type/Timein12hourformat.htm*
- "How to set own date patterns", *http://www.java2s.com/Tutorial/Java/0040__Data-Type/SimpleDateFormat.htm*
- *"*JTextPane*",* *http://www.java2s.com/Tutorial/Java/0240__Swing/0680__JTextPane.htm*
- "How to get column number and line number of cursor's current position", *http://stackoverflow.com/questions/5139995/java-column-number-and-line-number-of-cursors-current-position*
- *"*How to make Split Pane using Swing*", http://www.java-tips.org/java-se-tips/javax.swing/how-to-make-split-pane-using-swing.html*
- *"*Tokens in Java Programs*",* *http://www.cs.cmu.edu/~pattis/15-1XX/15-200/lectures/tokens/lecture.html*
- "How to change name of node in JTree", *http://stackoverflow.com/questions/5121122/change-name-of-node-in-jtree*

- "How to expand all Nodes in JTree", *http://www.javalobby.org/java/forums/t61157.html*
- "JOptionPane-Dialog", *http://www.java2s.com/Tutorial/Java/0240__Swing/0960__JOptionPane-Dialog.htm*
- "How to bring JInternalFrame to the front", *http://stackoverflow.com/questions/5081552/bring-jinternalframe-to-the-front*
- "How to write Windows Registry using Java code/program", *http://www.javaquery.com/2011/07/how-to-write-windows-registry-using.html*
- "How to set icon for JFrame", *http://www.youtube.com/watch?v=40ikcEonWng*
- "How to check CPU and Memory Usage in Java", *http://stackoverflow.com/questions/74674/how-to-do-i-check-cpu-and-memory-usage-in-java*
- "Rendering an image in a JTable column", *http://www.java2s.com/Code/Java/Swing-JFC/RenderinganimageinaJTablecolumn.htm*
- "How to set up the JTree to have different icons for each node.", *http://www.daniweb.com/software-development/java/threads/117886/jtree-with-different-icons-for-each-node*
- "How to Use Tables(combobox)", *http://docs.oracle.com/javase/tutorial/uiswing/components/table.html#combobox*
- "How to convert Milliseconds to "X mins, x seconds" in Java", *http://stackoverflow.com/questions/625433/how-to-convert-milliseconds-to-x-mins-x-seconds-in-java*
- "How many hardware details can a Java Applet Discover", *http://stackoverflow.com/questions/1011063/how-many-hardware-details-can-a-java-applet-discover*
- "How to compile a java program into an exe." , *http://stackoverflow.com/questions/2011664/compiling-a-java-program-into-an-exe*

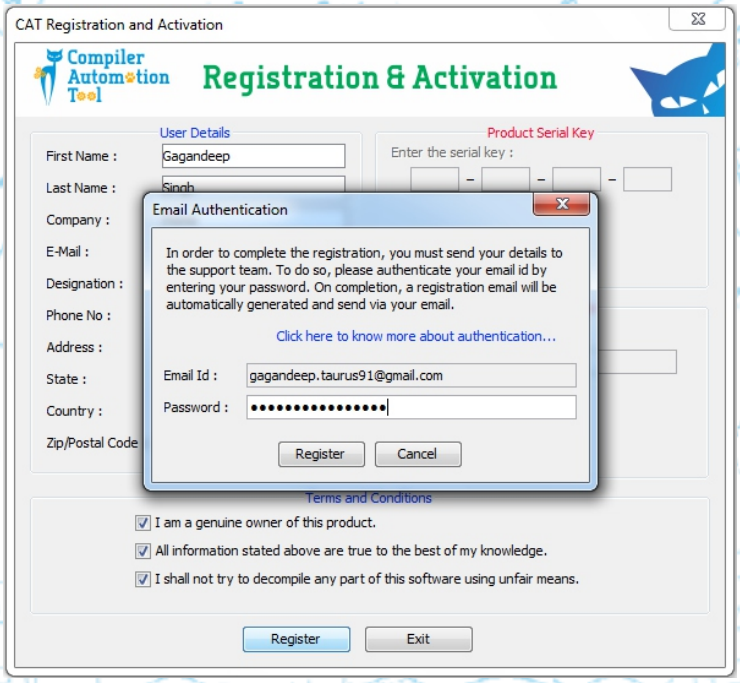
- "SIGAR - System Information Gatherer And Reporter API", *http://support.hyperic.com/display/SIGAR/Home*
- "Using Java to get OS-level system information", *http://stackoverflow.com/questions/25552/using-java-to-get-os-level-system-information*
- "How to iterate through HashMap in Java", *http://stackoverflow.com/questions/1066589/java-iterate-through-hashmap*
- "How to remove gradient from bars?", *http://www.jfree.org/phpBB2/viewtopic.php?f=3&t=27932*
- "How to display customized values on a bar in a bar chart using jfree chart?", *http://stackoverflow.com/questions/13625740/how-to-display-customized-values-on-a-bar-in-bar-chart-using-jfree-chart*
- "How to use CipherOutputStream ?", *http://www.java2s.com/Tutorial/Java/0490__Security/UsingCipherOutputStream.htm*
- "How to maximize a JFrame through code ?", *http://stackoverflow.com/questions/5258207/how-to-maximize-a-jframe-through-code"*
- "Sending Email through Gmail Server with SSL", *http://www.javatpoint.com/example-of-sending-email-using-java-mail-api-through-gmail-server*
- "Sending Email using JavaMail API", *http://www.javatpoint.com/example-of-sending-email-using-java-mail-api*
- "How to get the separate digits of an int number ?", *http://stackoverflow.com/questions/3389264/how-to-get-the-separate-digits-of-an-int-number*
- "Is it acceptable to leave the javamail session Transport open?", *http://stackoverflow.com/questions/4334027/is-it-acceptable-to-leave-the-javamail-session-transport-open*
- "iText : Organizing content in tables", *http://itextpdf.com/examples/iia.php?id=76*

- "How to set column width of a table in iText?", *http://www.kodejava.org/examples/833.html*
- "How to set the table cell width in iText java pdf", *http://stackoverflow.com/questions/14778165/set-the-table-cell-width-in-itext-java-pdf*
- "How to add an iText image to a pdf document", *http://tutorials.jenkov.com/java-itext/image.html*
- "How to get the directory that the currently executing jar file is in?", *http://stackoverflow.com/questions/2837263/how-do-i-get-the-directory-that-the-currently-executing-jar-file-is-in*
- "How to set the look and feel of application's GUI ?", *http://docs.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html#available*
- "How to shuffle characters in a string ?", *http://stackoverflow.com/questions/3316674/how-to-shuffle-characters-in-a-string*
- "How to remove whitespaces from strings in Java", *http://stackoverflow.com/questions/5455794/removing-whitespace-from-strings-in-java*
- "Java Swing font chooser" , *http://stackoverflow.com/questions/2120228/java-swing-font-chooser*
- "How to hide file from Java Program", *http://javarevisited.blogspot.in/2012/01/how-to-hide-file-java-program-example.html*
- "System Properties", *http://docs.oracle.com/javase/tutorial/essential/environment/sysprop.html*
- "How to make the borders of table invisible in iText", *http://stackoverflow.com/questions/4900514/an-invisible-border-of-pdfptable*
- "Adding PNG to table cell" , *http://www.java2s.com/Code/Java/PDF-RTF/AddingPNGtotablecell.htm*
- "Adding image to a table cell", *http://www.java2s.com/Code/Java/PDF-RTF/AddingImagetoaTableCell.htm*

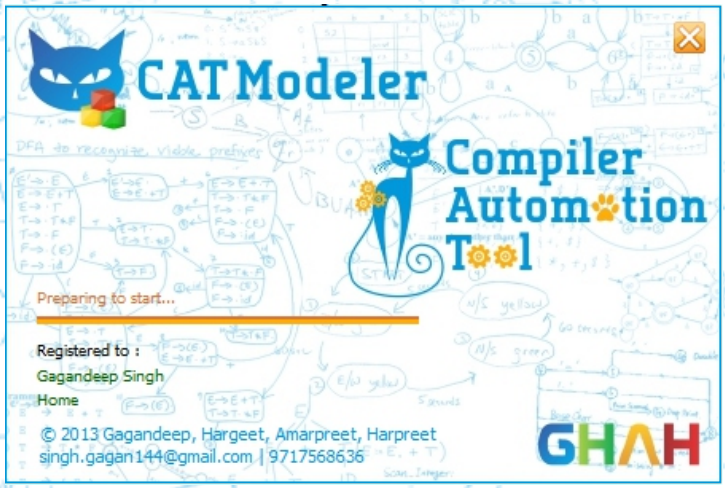
- "Java image scaling", *http://stackoverflow.com/questions/2905323/java-image-scaling*
- "How to add jfreecharts to a pdf file ?", *http://www.jfree.org/phpBB2/viewtopic.php?t=13673*
- "How to convert JAR to EXE", *http://www.youtube.com/watch?v=mARUFRknTYQ*
- "launch4j: cross platform java executable wrapper", *http://launch4j.sourceforge.net/*
- "ResourcesExtract", *http://www.nirsoft.net/utils/resources_extract.html*
- "Convert JAR file to EXE executable", *http://viralpatel.net/blogs/convert-jar-to-exe-executable-jar-file-to-exe-converting/*
- "Jar2Exe", *http://www.jar2exe.com/*
- "Catch Events after JSplitPane Divider Location is moved", *http://tech.chitgoks.com/2009/11/20/catch-events-after-jsplitpane-divider-location-is-moved/*
- "Detecting when JSplitPane divider is being dragged,not component being resized", *http://stackoverflow.com/questions/14426472/detecting-when-jsplitpane-divider-is-being-dragged-not-component-being-resized*
- "How to get the graph in center", *http://forum.jgraph.com/questions/4884/center-graph-grapheditor-example*
- "How to extract all classes loaded in the JVM into a single JAR", *http://www.nakov.com/blog/2008/08/27/extract-all-classes-loaded-in-the-jvm-into-a-single-jar/*
- "JGraphX User Manual", *http://jgraph.github.io/mxgraph/docs/manual_javavis.html*
- "JGraph-Java Graph drawing component", *www.jgraph.com/jgraph.html*

- "JGraphX- auto organize of cells and bidirectional edges", *http://stackoverflow.com/questions/5603306/jgraphx-auto-organise-of-cells-and-bidirectional-edges*
- "How to get number of digits in an int?" , *http://stackoverflow.com/questions/1306727/way-to-get-number-of-digits-in-an-int*
- "Parsing Algorithm", *http://cs.nyu.edu/courses/spring01/V22.0480-002/parse.html*
- "Lexical and Syntax Analysis of Programming Languages", *http://www.pling.org.uk/cs/lsa.html*
- "SLR(1), LR(1) and LALR(1) Grammars", *http://lambda.uta.edu/cse5317/notes/node20.html*
- "SLR Parse table in JFlap", *http://www.jflap.org/tutorial/grammar/slr/index.html*
- "LR parser", *http://en.wikipedia.org/wiki/LR_parser*
- "How is the following grammar LR(1) but not SLR(1)", *http://stackoverflow.com/questions/10505717/how-is-the-following-grammar-lr1-but-not-slr1*
- "How to run a .Jar Java File", *http://www.wikihow.com/Run-a-.Jar-Java-File*
- "How to change the OK Cancel Button to custom string in java ?", *http://stackoverflow.com/questions/8763075/change-the-ok-cancel-string-in-joptionpane*
- "How to get value after decimal point from a double value in C# ?", *http://stackoverflow.com/questions/4604980/how-to-get-value-after-decimal-point-from-a-double-value-in-c*
- "Selecting the multiple rows randomly in JTable", *http://stackoverflow.com/questions/13526711/selecting-the-multiple-rows-randomly-in-jtable*
- "How to create multi-line column header", *http://www.javarichclient.com/multiline-column-header/*

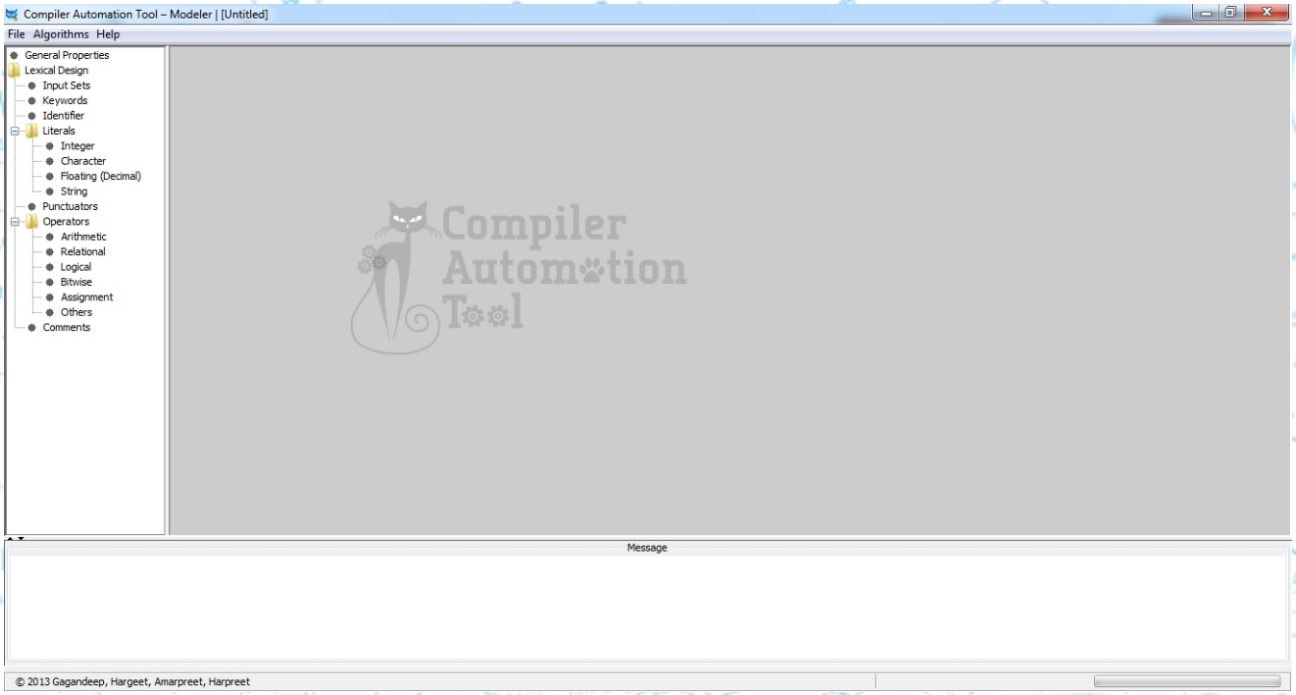
- "How to change the node image of a JTree dynamically", *http://stackoverflow.com/questions/2199987/changing-the-node-image-of-a-jtree-dynamically*
- "How to create a popup menu in java", *http://www.roseindia.net/java/example/java/swing/PopUpMenu.shtml*
- "How to add a new variable in the Windows-Registry", *http://stackoverflow.com/questions/9305470/how-to-add-a-new-variable-in-the-windows-registry*
- "JFreeChart: Bar Chart demo", *http://www.java2s.com/Code/Java/Chart/JFreeChartBarChartDemo.htm*
- "Disabling user edits in a JTable", *http://www.java2s.com/Tutorial/Java/0240__Swing/DisablingUserEditsinaJTable.htm*
- "How to change line colour in XY line chart", *http://www.jfree.org/phpBB2/viewtopic.php?f=3&t=27967*
- "How to customize subtitle position in JFreeChart", *http://stackoverflow.com/questions/6933187/customize-subtitle-position-jfreechart*



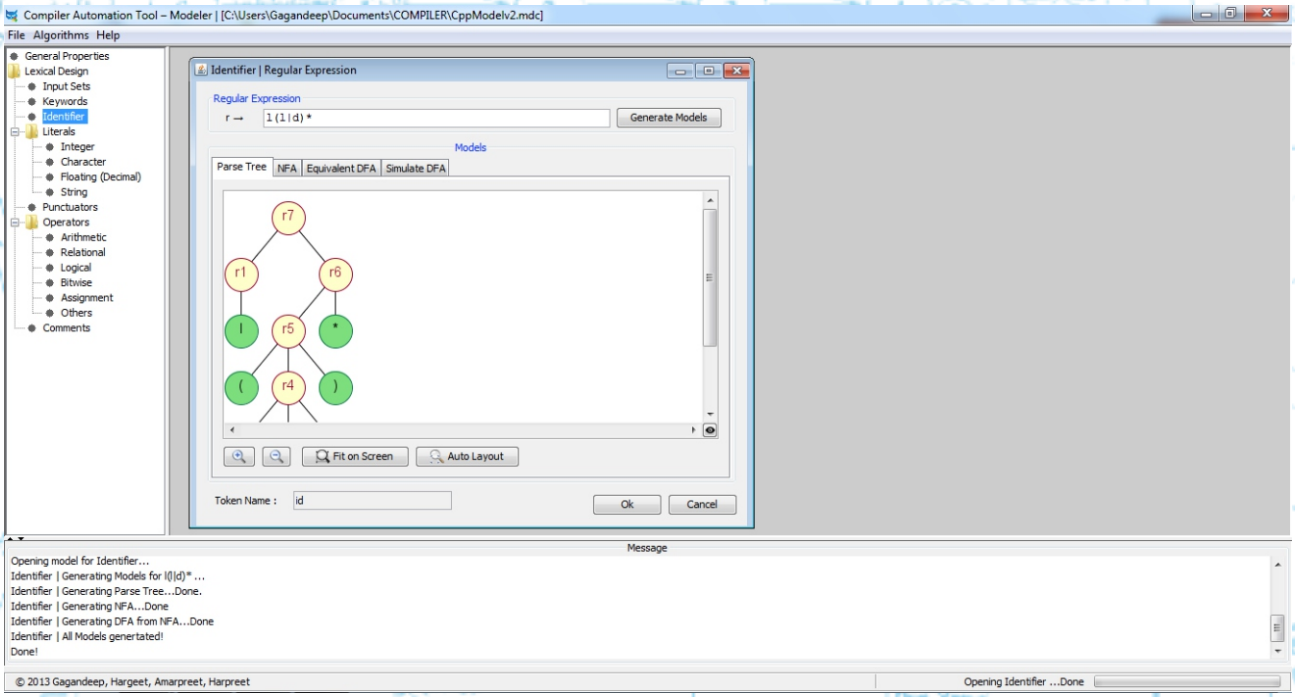
ScreenShot 1 - The user registers for the compiler automation tool by submitting its name, email id, phone number, address etc and thus agreeing to the terms and conditions specified.



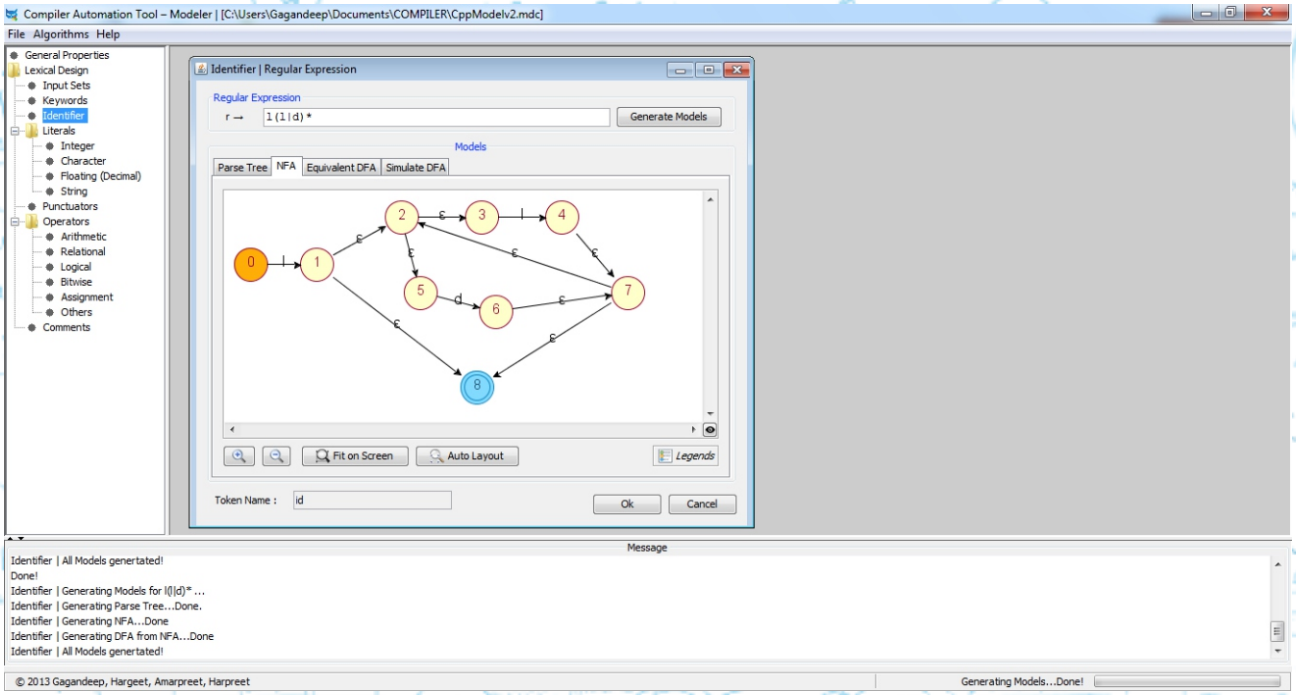
ScreenShot 2 - BootUp splash screen for CAT Modeler showing the progress made to start the application. Besides, CAT Compiler and CAT Statistic Studio stats similarly.



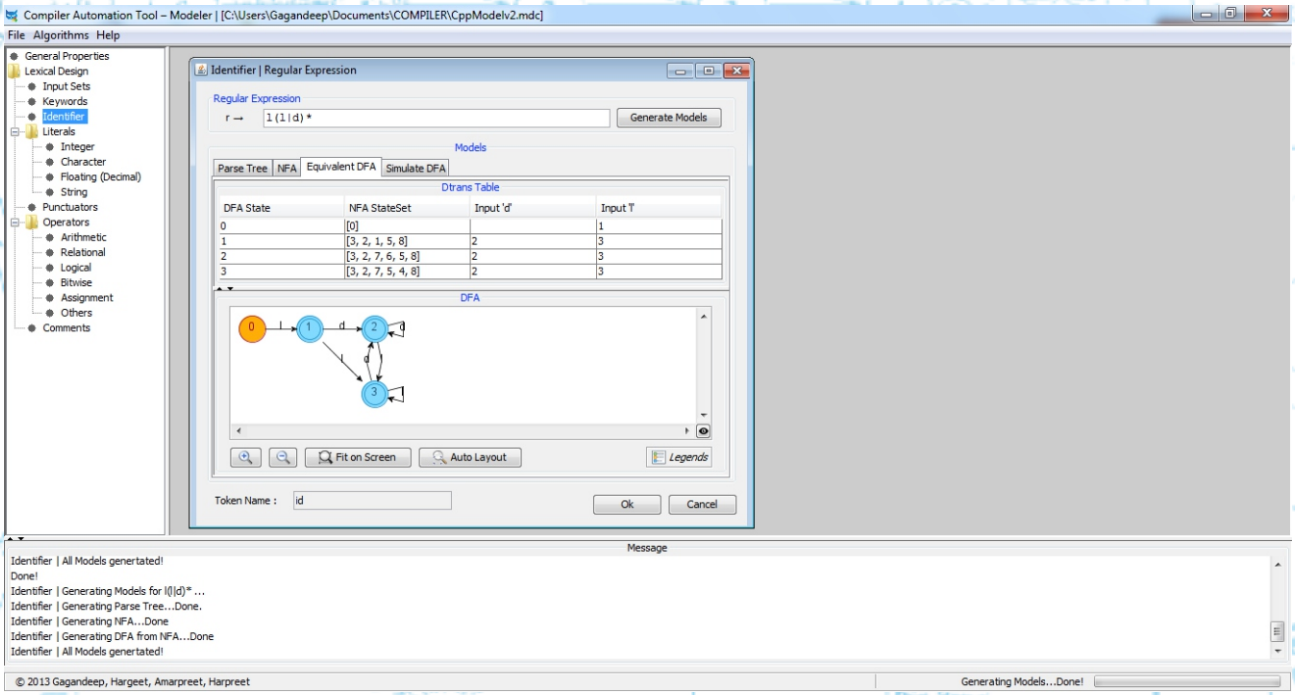
ScreenShot 3 - GUI for CAT Modeler showing various option to specifying modeling details.



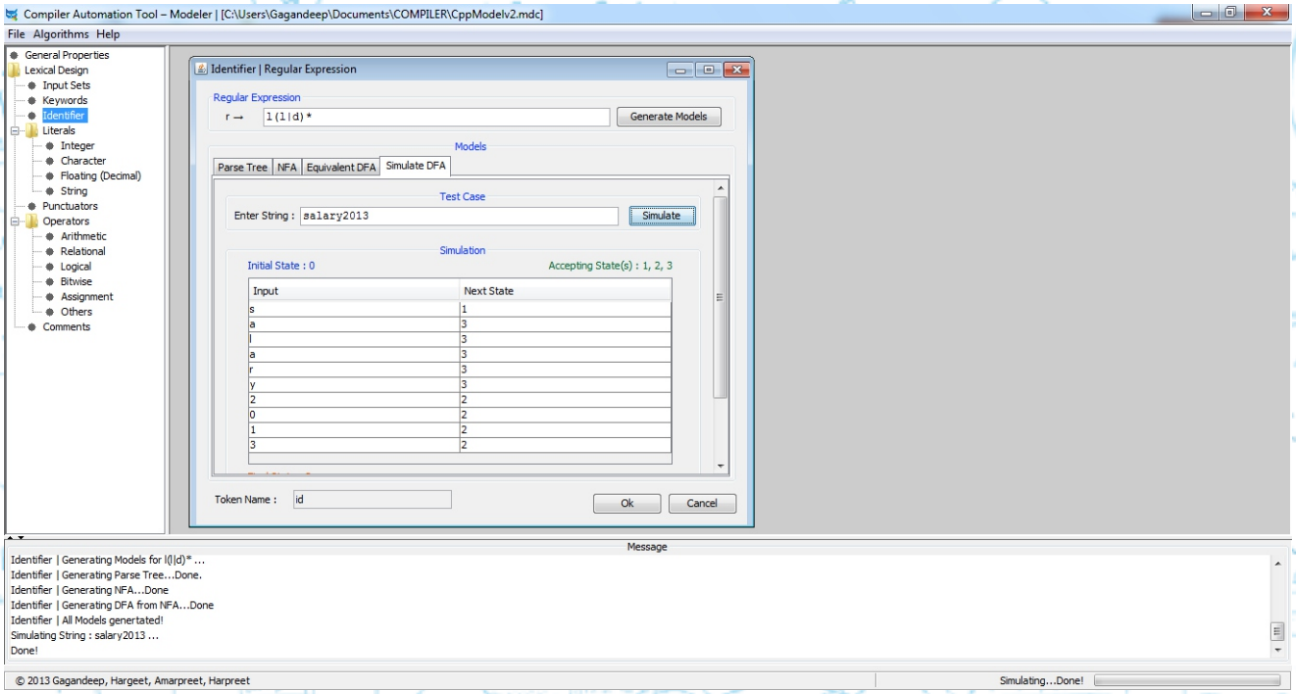
ScreenShot 4 - Parse tree generated according to user's regular expression.



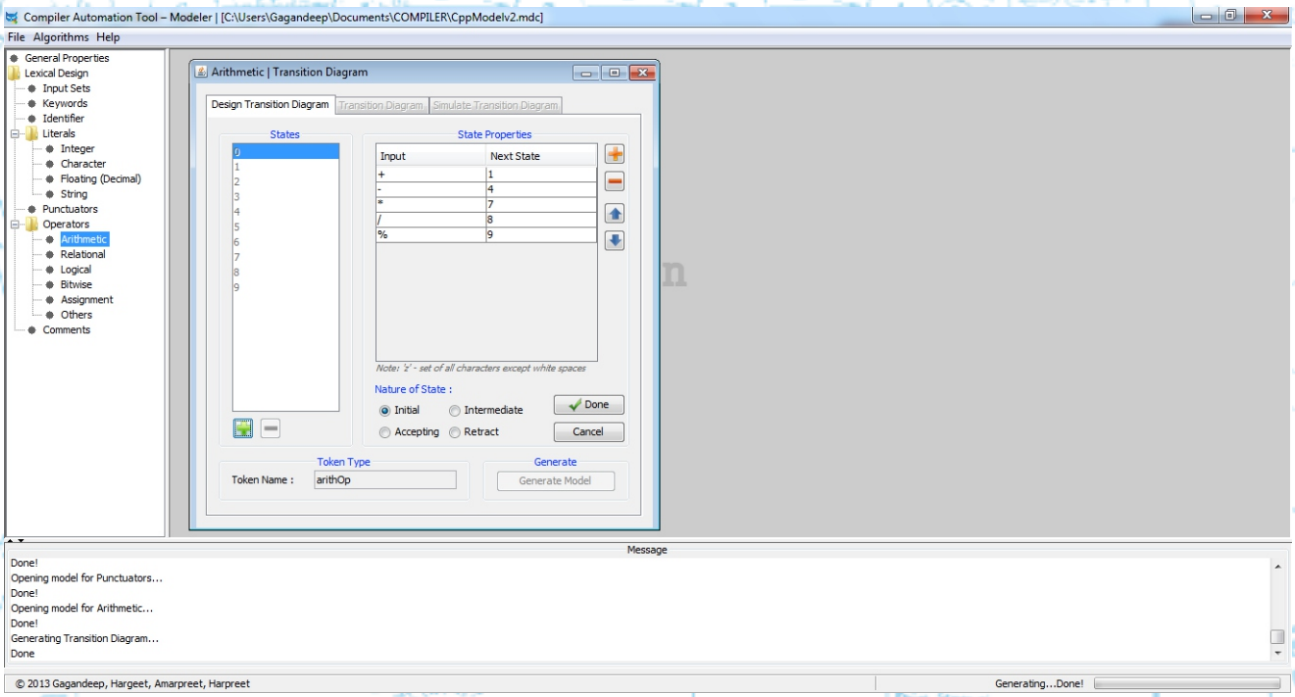
ScreenShot 5 - NFA i.e. non-deterministic finite automata is generated for the regular expressions given by the user.



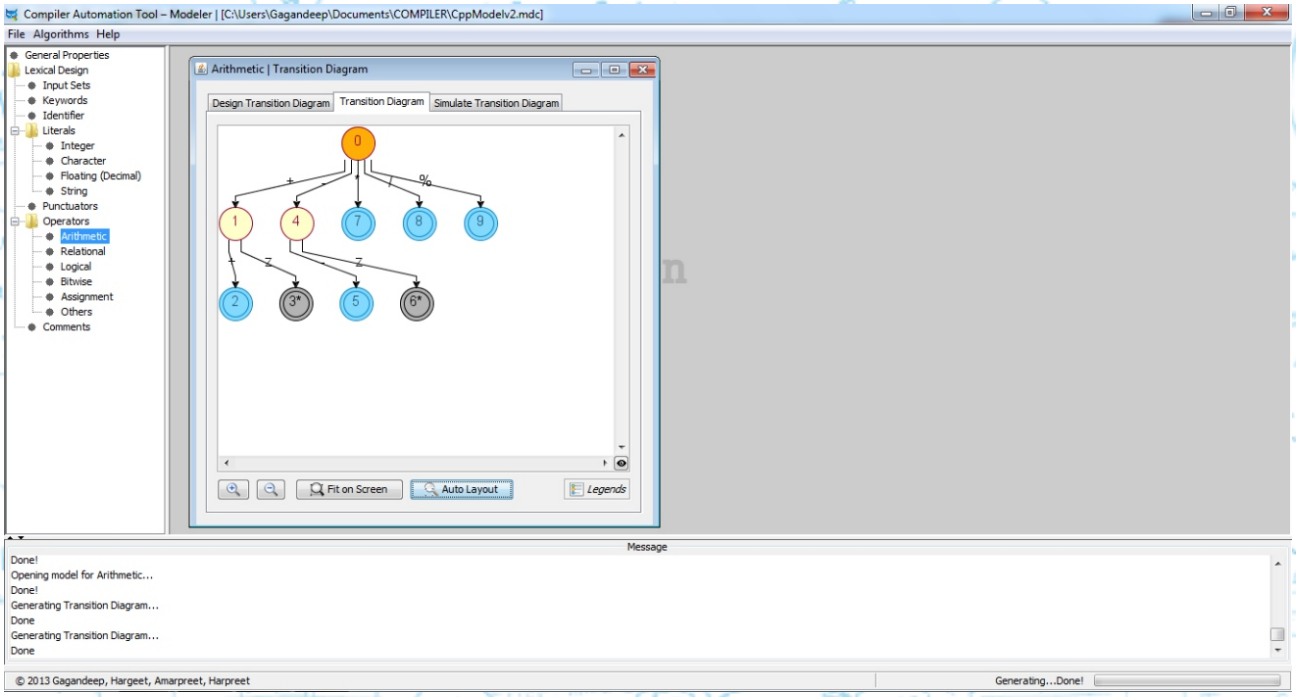
ScreenShot 6 - Equivalent DFA of the NFA. It also shows the Dtrans table with DFA state, NFA state set and inputs as its attributes.



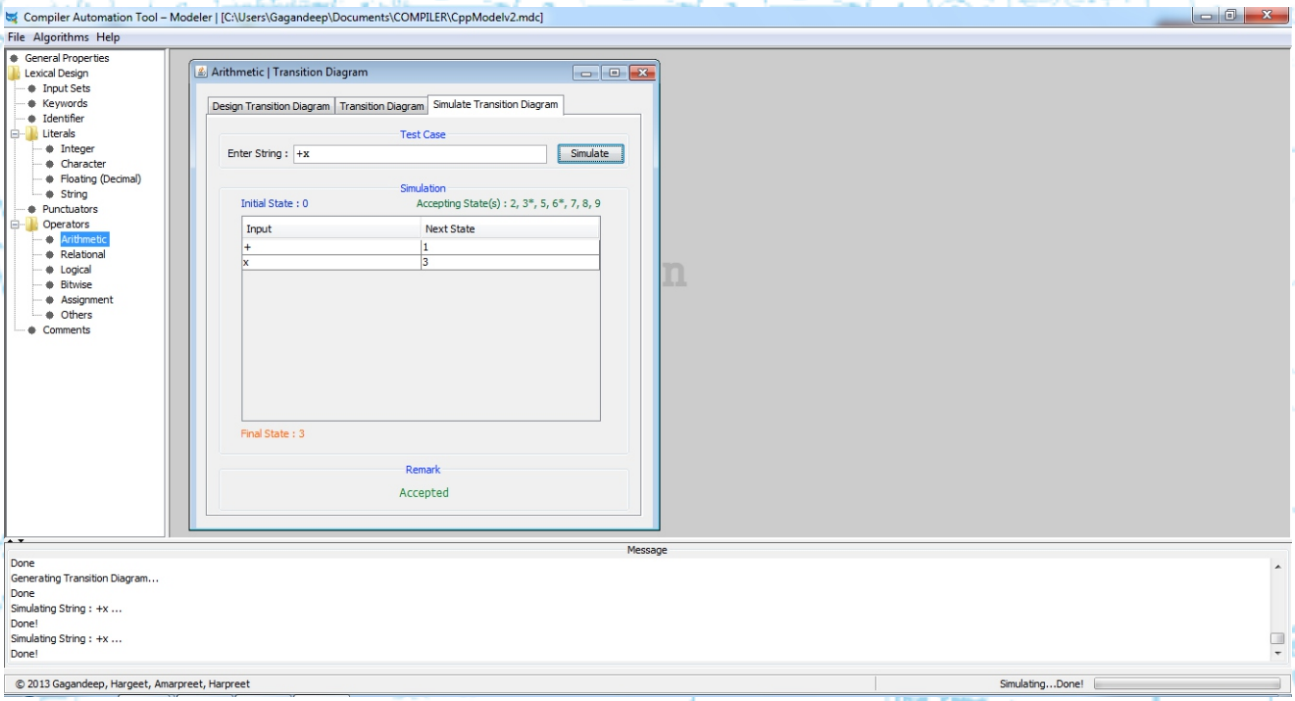
ScreenShot 7 - DFA is simulated where user enters a test string. On clicking the simulate button, a table is generated depicting input and next state. It also shows the accepting state(s).



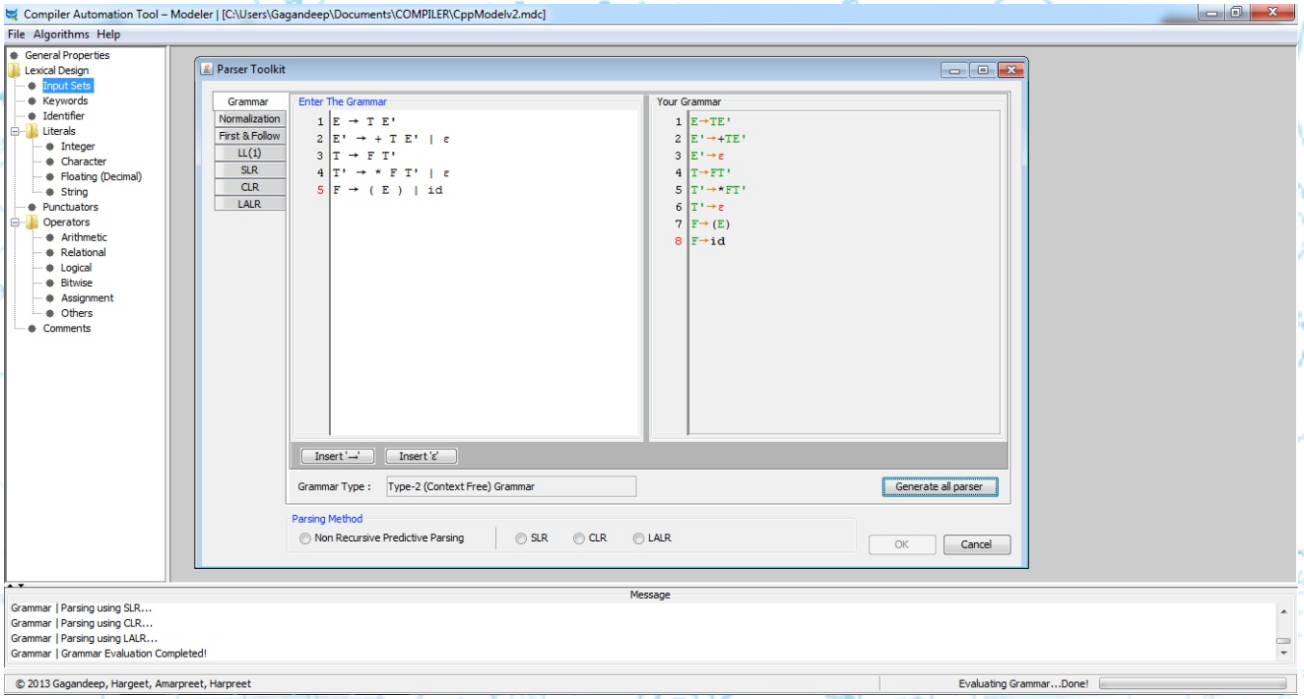
ScreenShot 8 - User designing transition diagram for arithmetic expression.



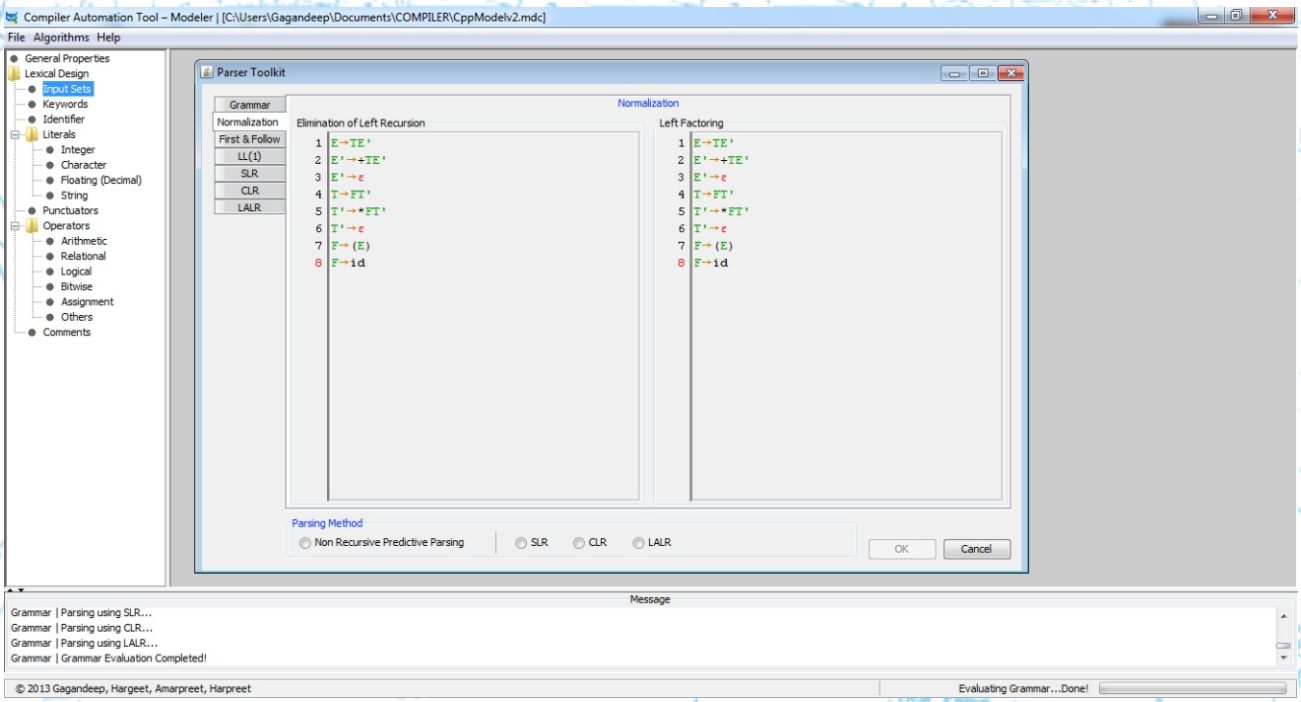
ScreenShot 9 - Graphical representation of the transition diagram as designed by the user.



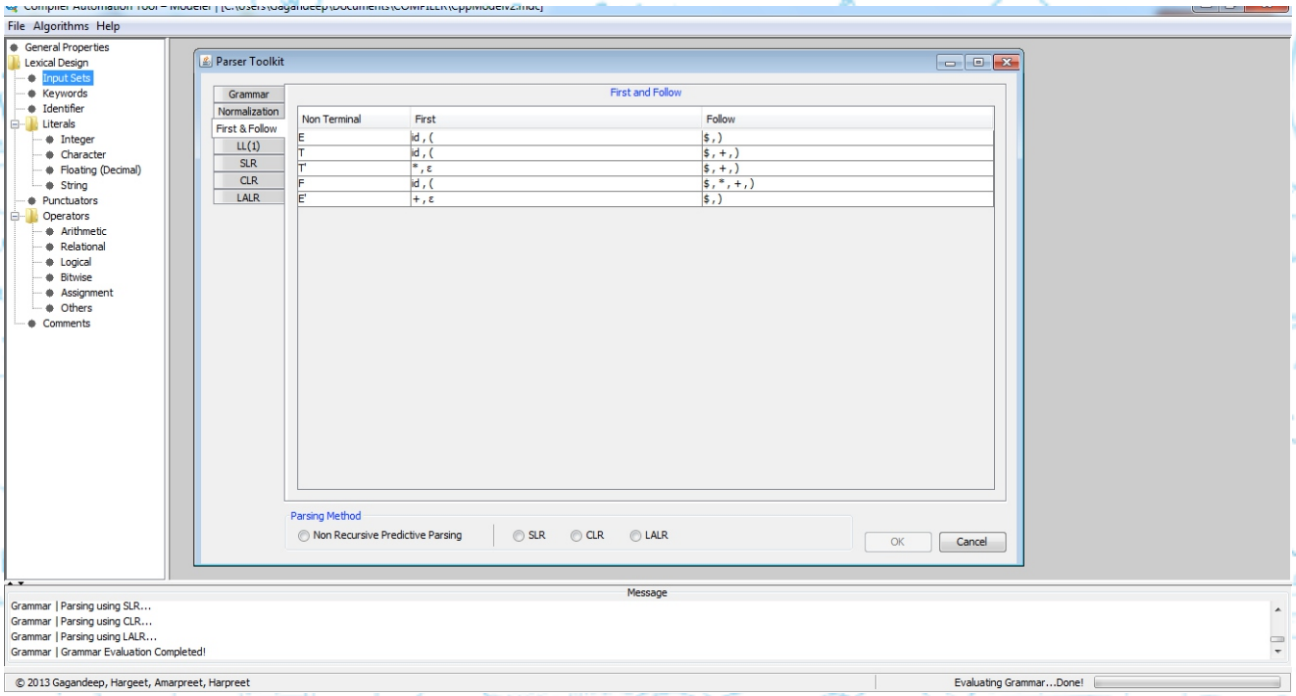
ScreenShot 10 - Simulation of the transition diagram against a test string. The depicts the various jumps from one state to another as the input is read.



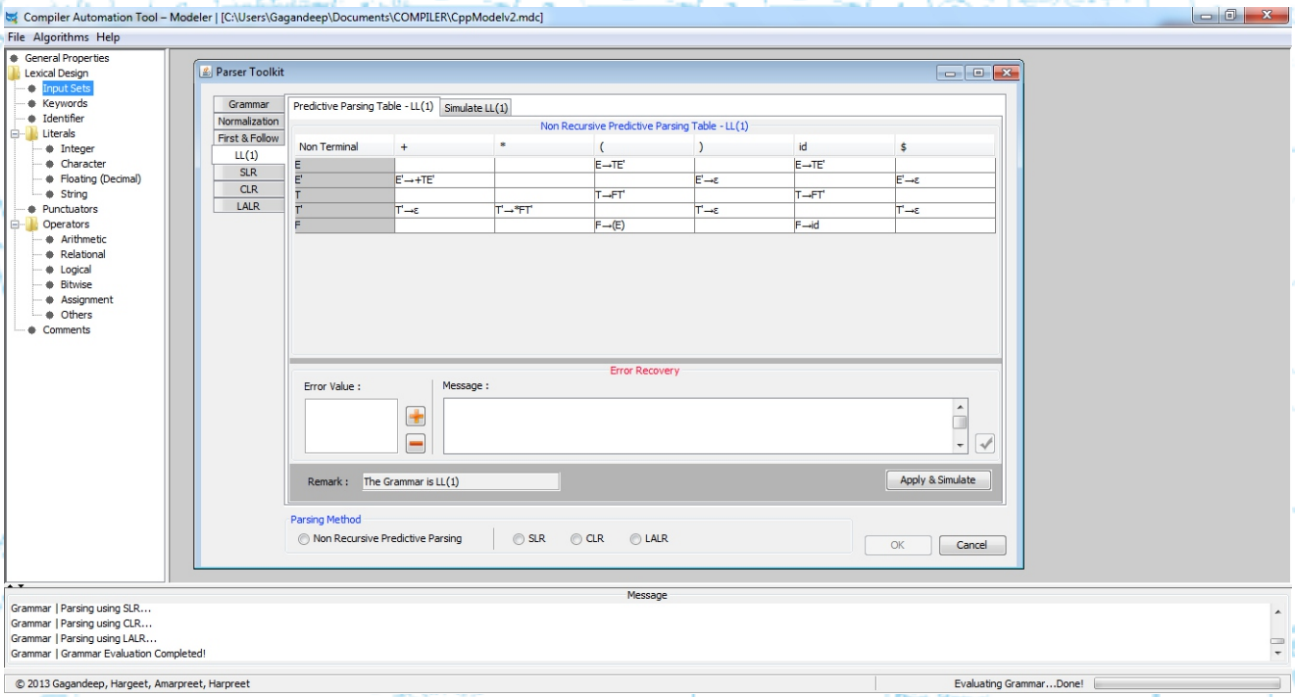
ScreenShot 11 - Parser toolkit where the user enters the grammar and generates normalized grammar and all parser.



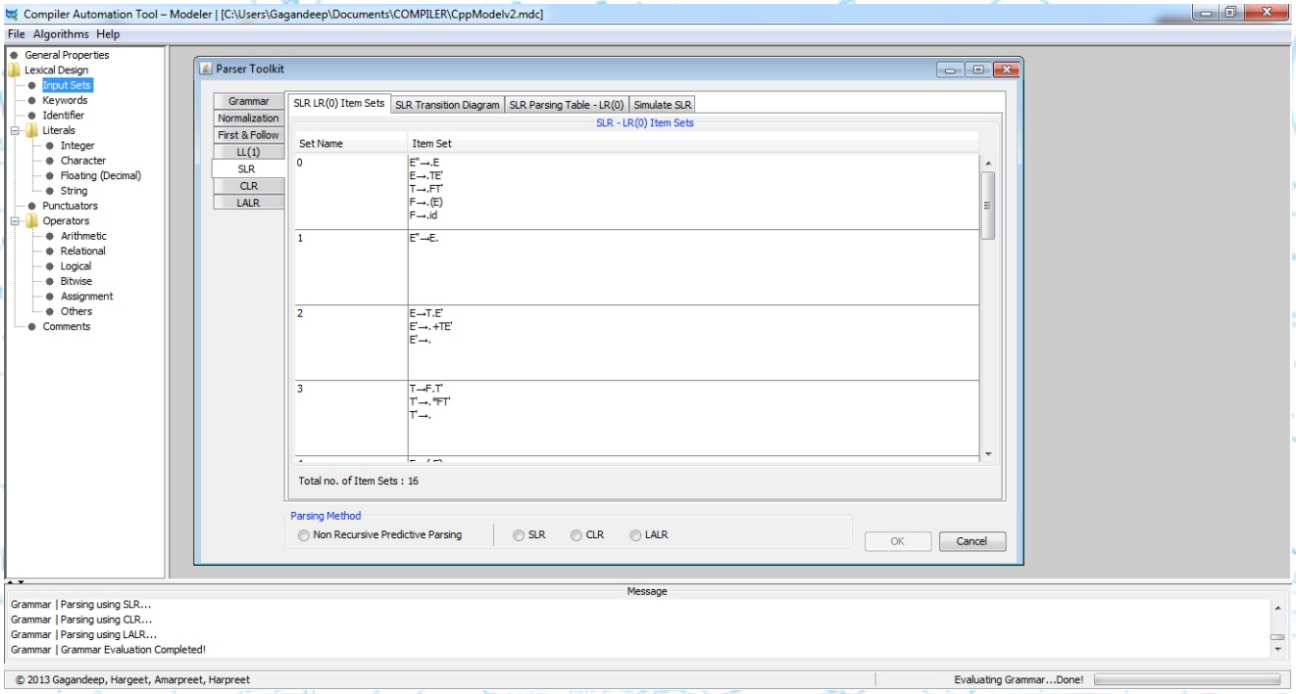
ScreenShot 12 - Equivalent left recursion elimination and left factoring of the grammar entered.



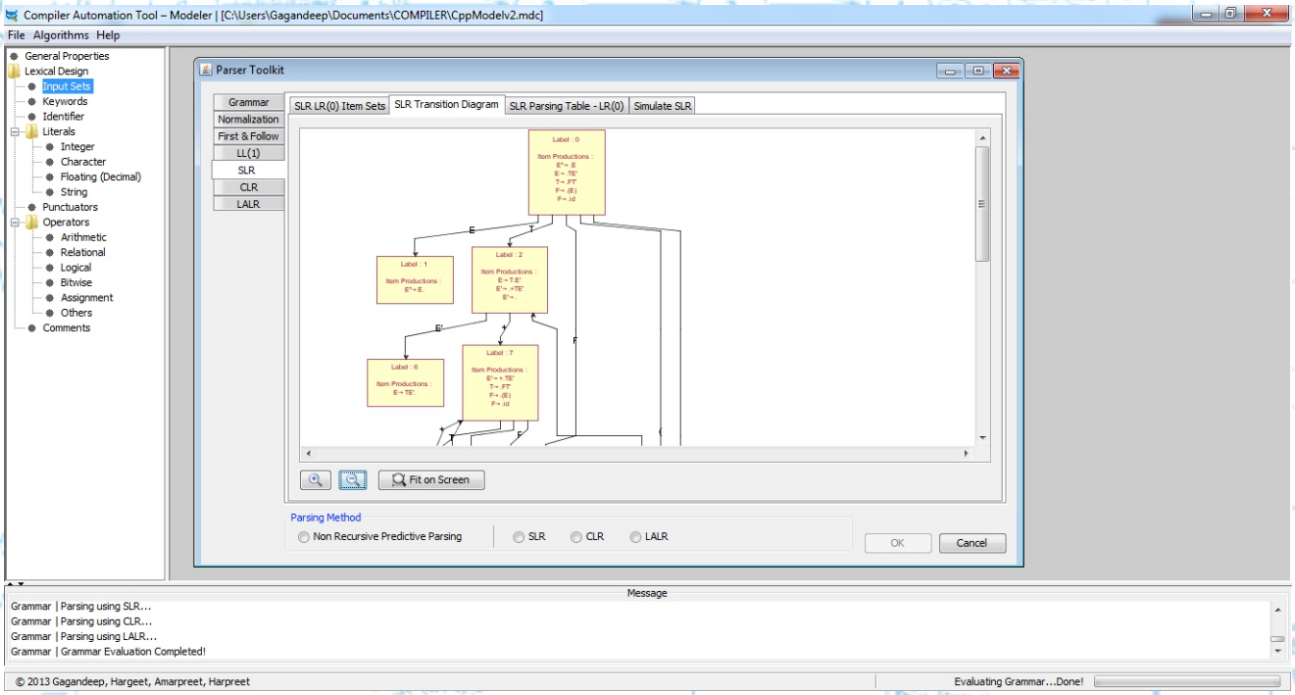
ScreenShot 13 - Calculation of first and follow of the grammar.



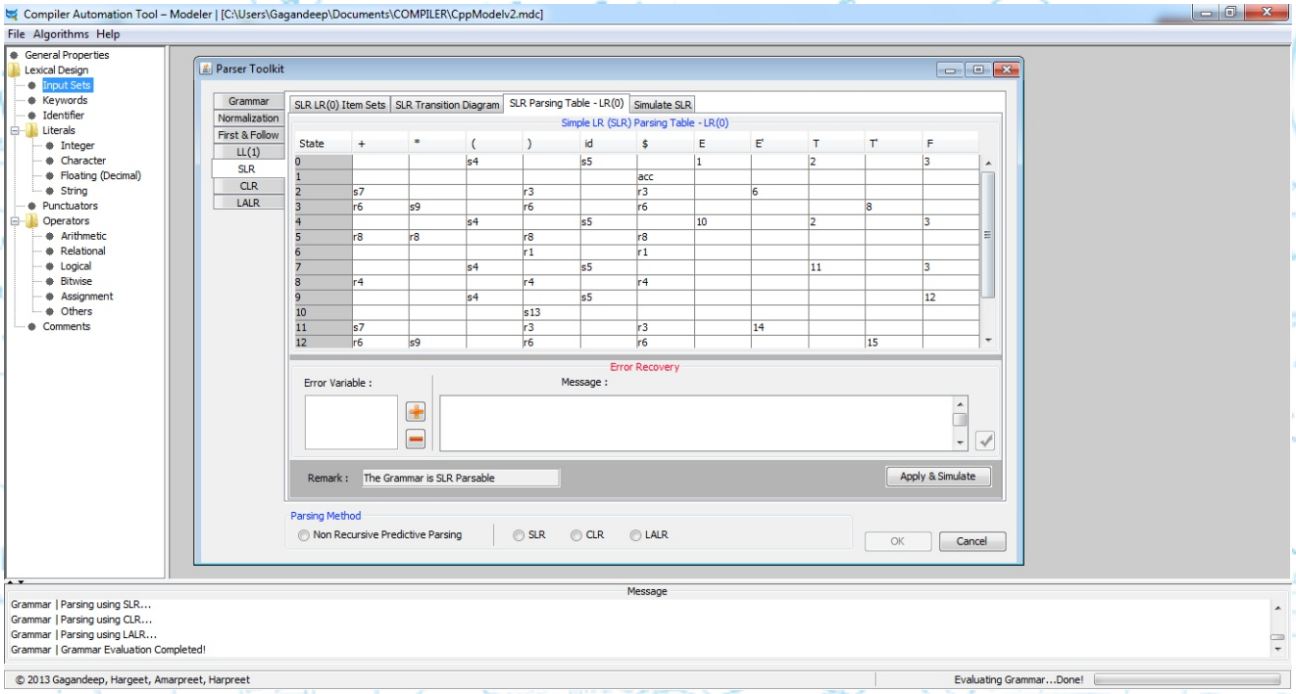
ScreenShot 14 - Predictive parsing table (LL1) of the grammar.



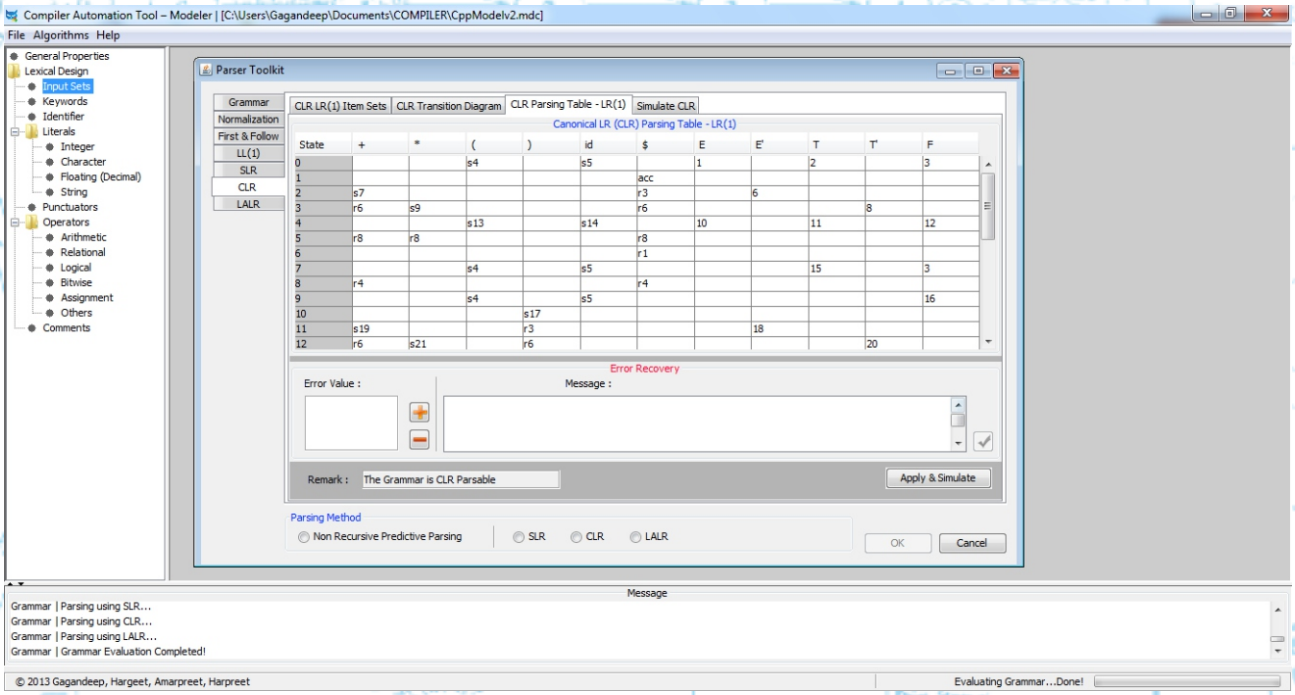
ScreenShot 15 - SLR item states of the grammar as the first step to generate SLR parsing table.



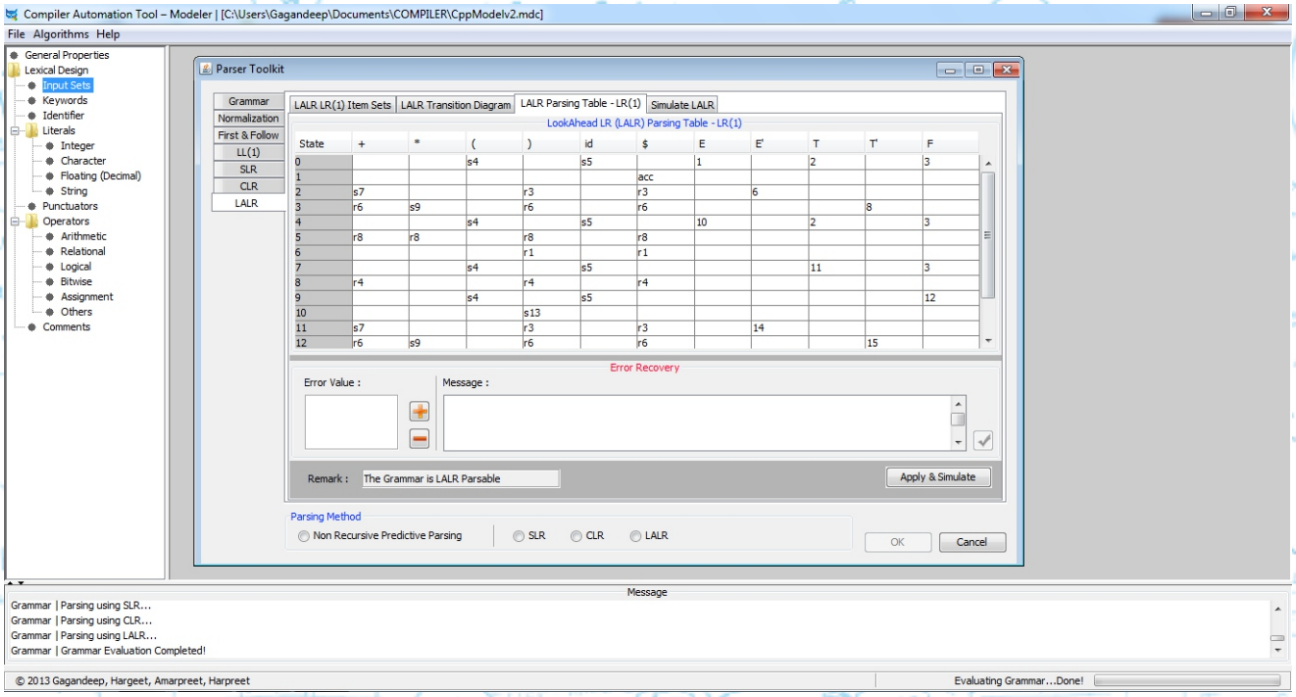
ScreenShot 16 - Graphical representation of SLR transition diagram obtained from item states.



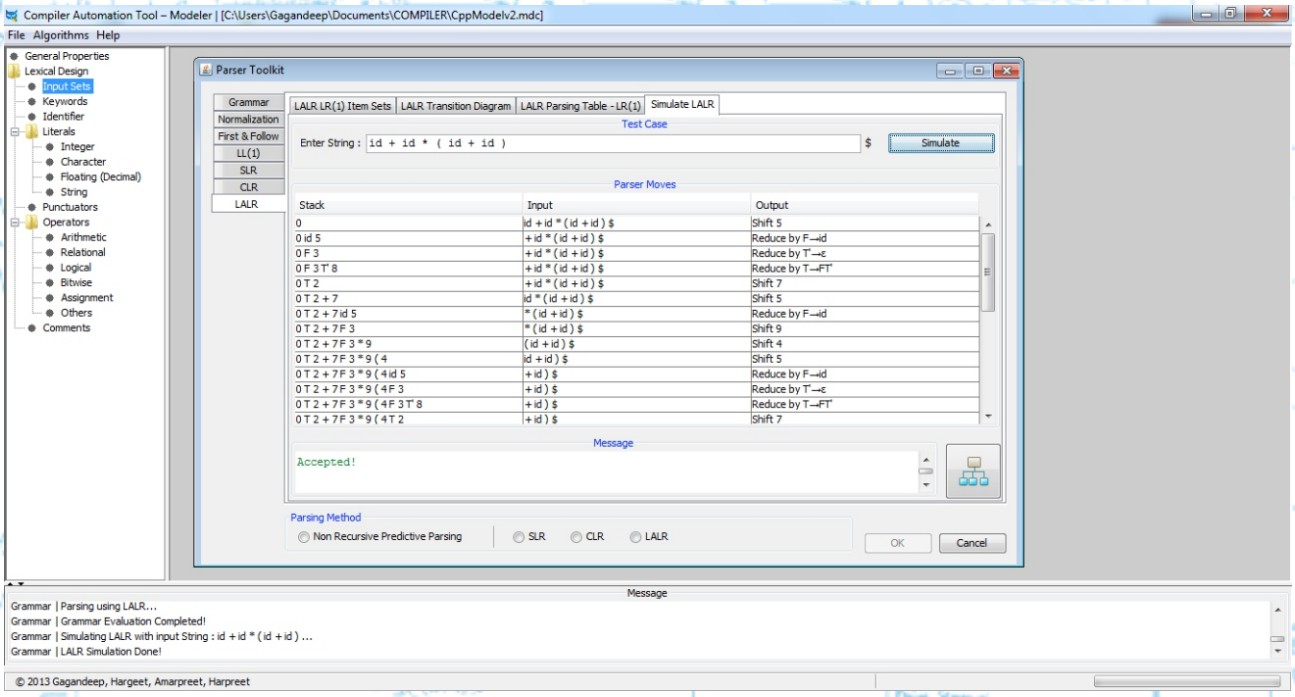
ScreenShot 17 - SLR Parsing table that can be used to parse a string of token to check whether it is accepted by the grammar or not.



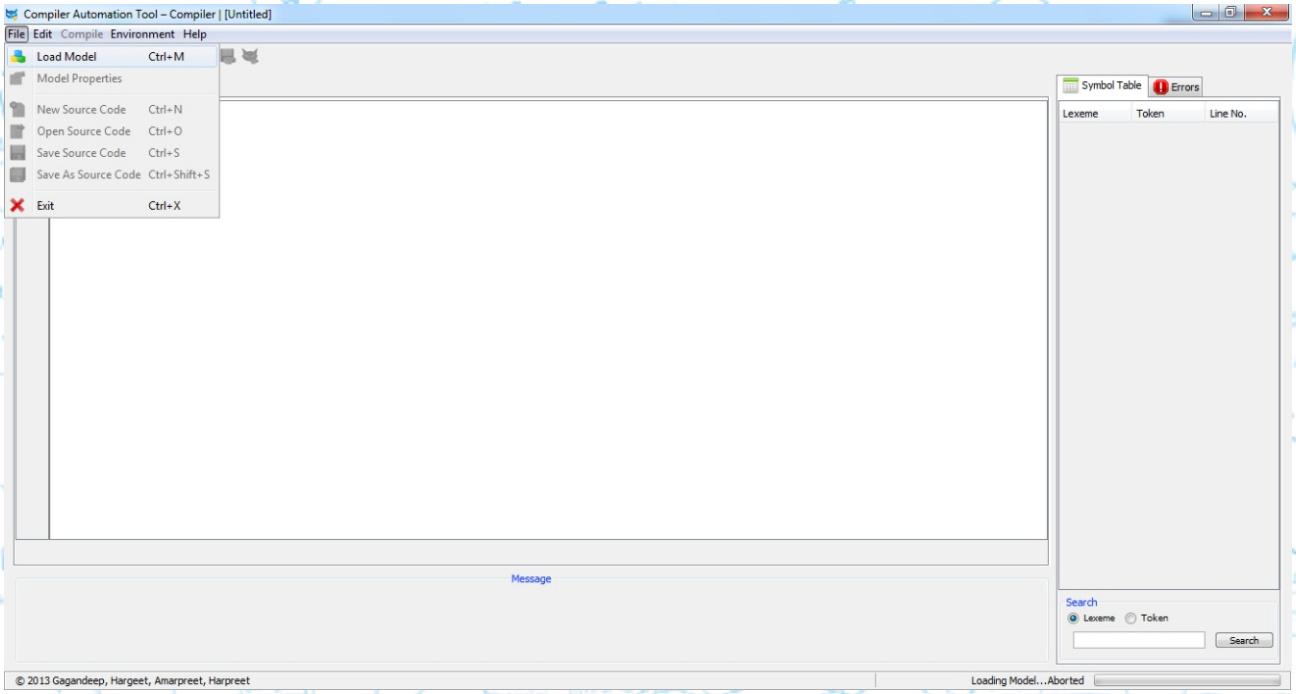
ScreenShot 18 - CLR Parsing table.



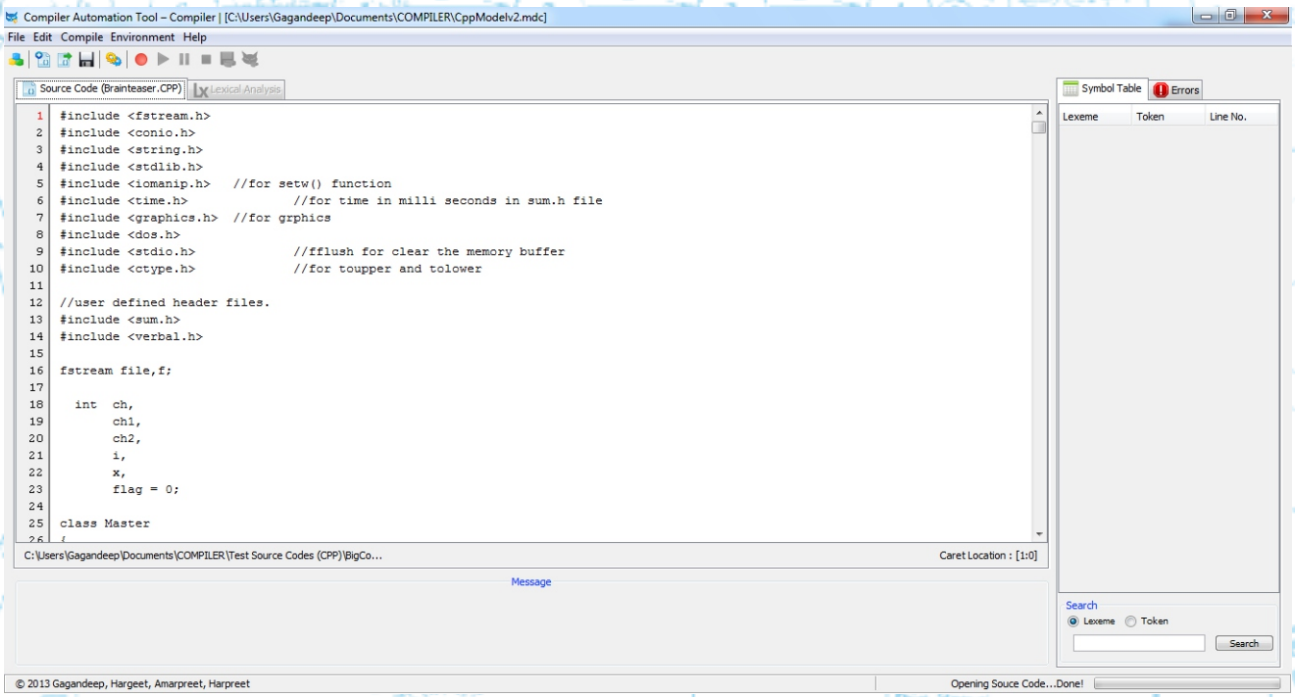
ScreenShot 19 - LALR Parsing table.



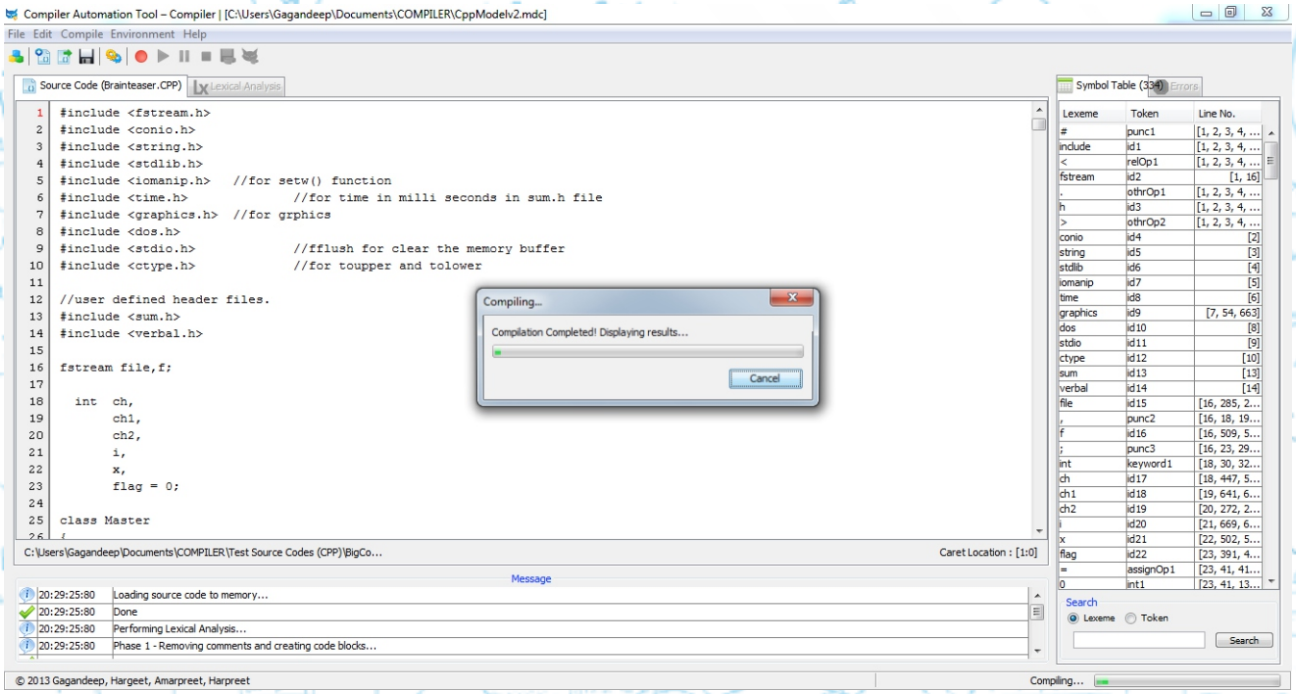
ScreenShot 20 - Simulation of LALR. The user enter a string of token to be tested against the parsing table generated for the grammar. On accepting, an accepted message is shown.



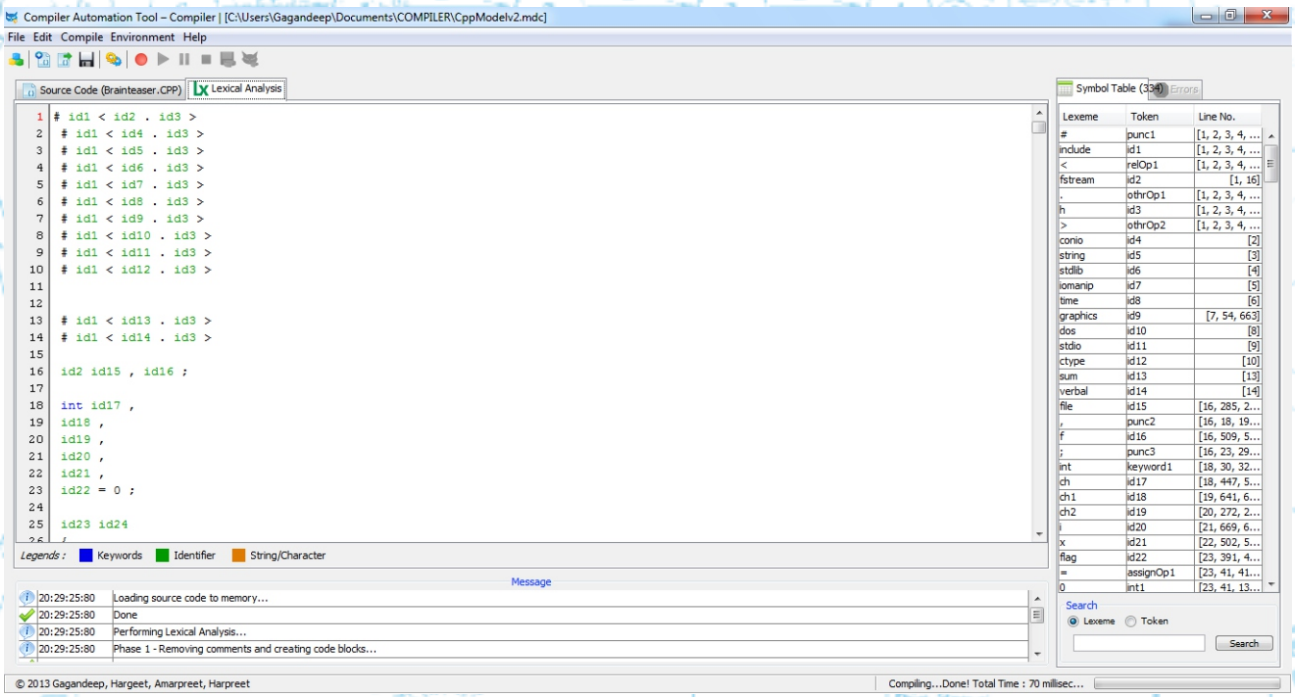
ScreenShot 21 - Graphical User Interface for CAT Compiler. The user loads the model designed in CAT Modeler and apply various source code test cases to check the working of compiler.



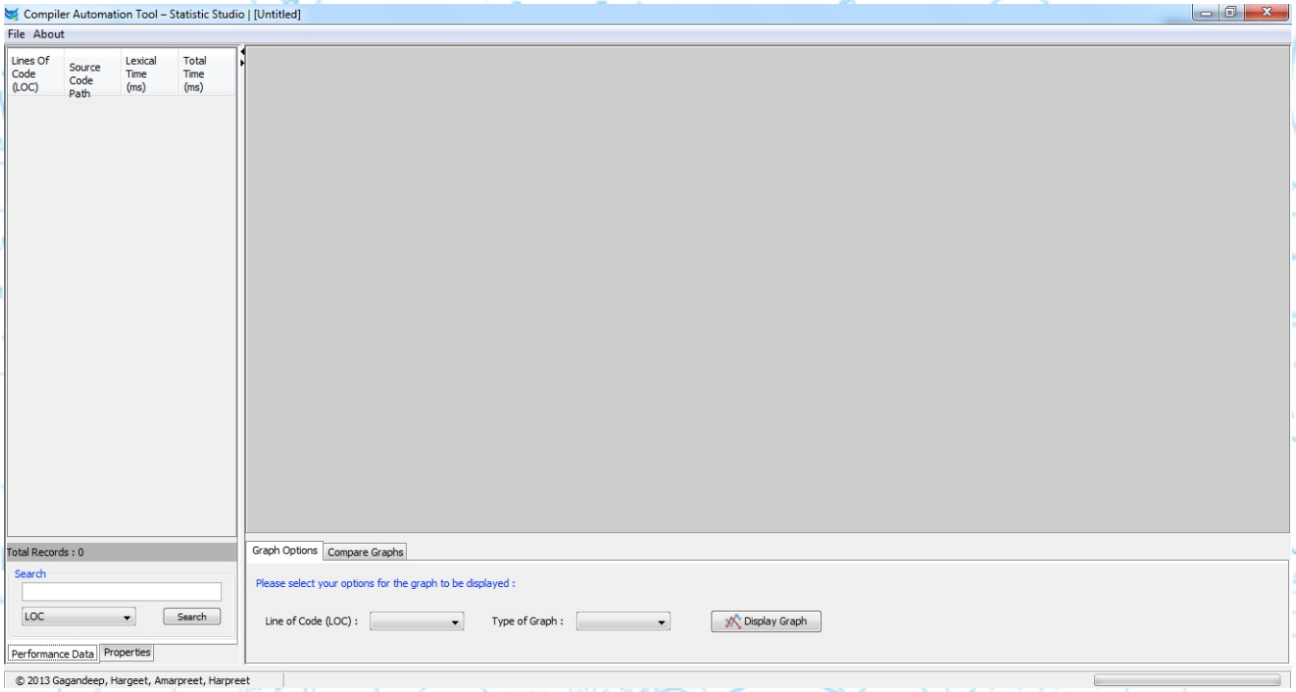
ScreenShot 22 - Source code loaded to be tested.



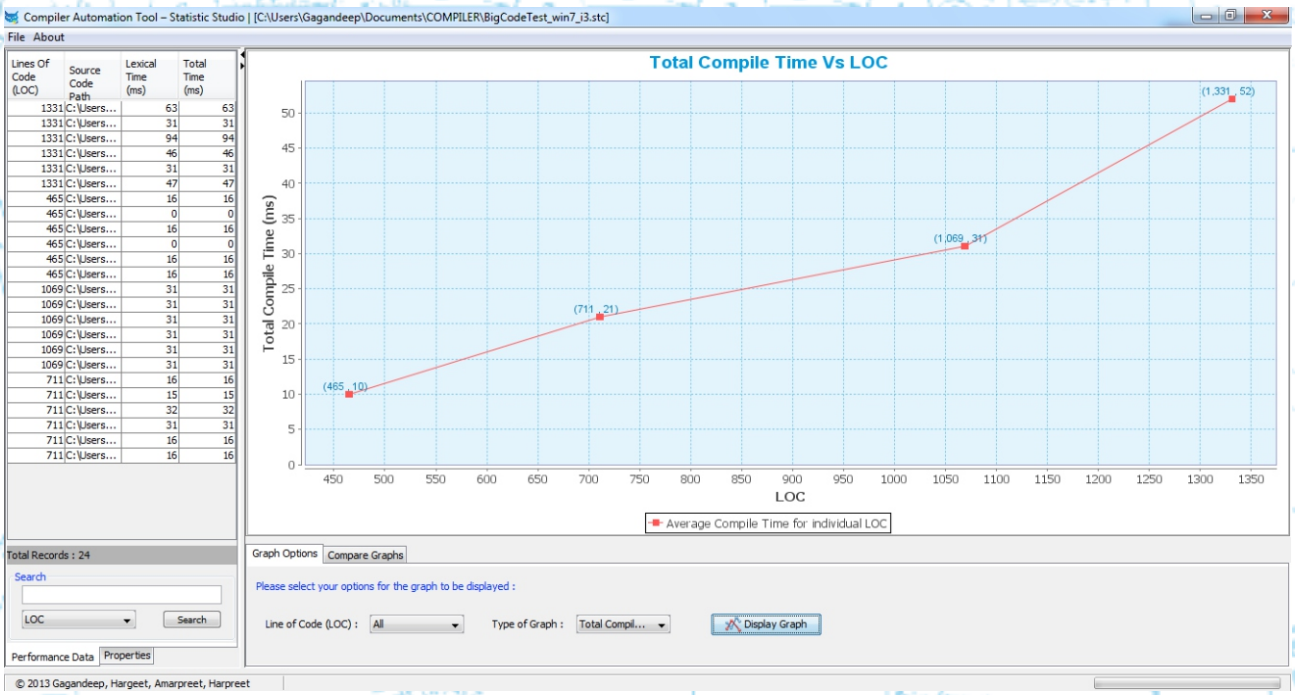
ScreenShot 23 - Source code being complied and intermediates results being loaded.



ScreenShot 24 - This screen shows the various token recognizes by the compiler using the model. On right side is shown the Symbol table and error if any.

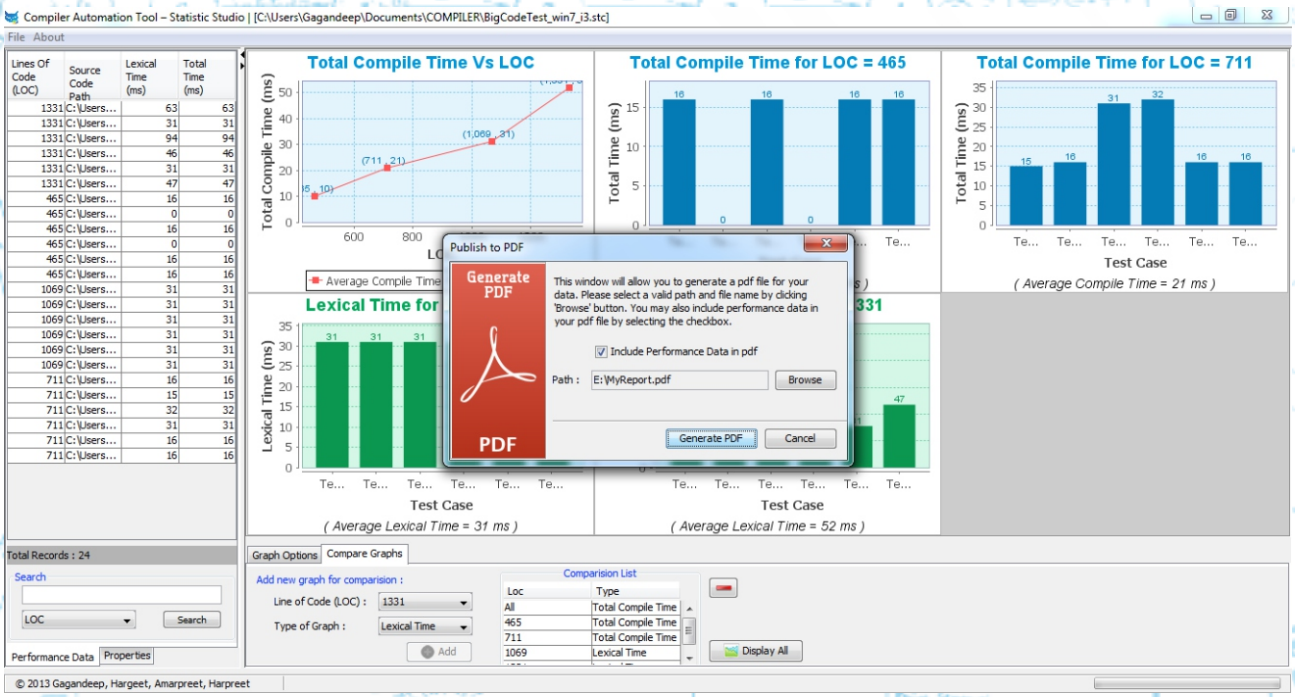


ScreenShot 25 - Graphical User Interfaces for CAT Statistic Studio. The left side of the screen displays various performance data and model properties.

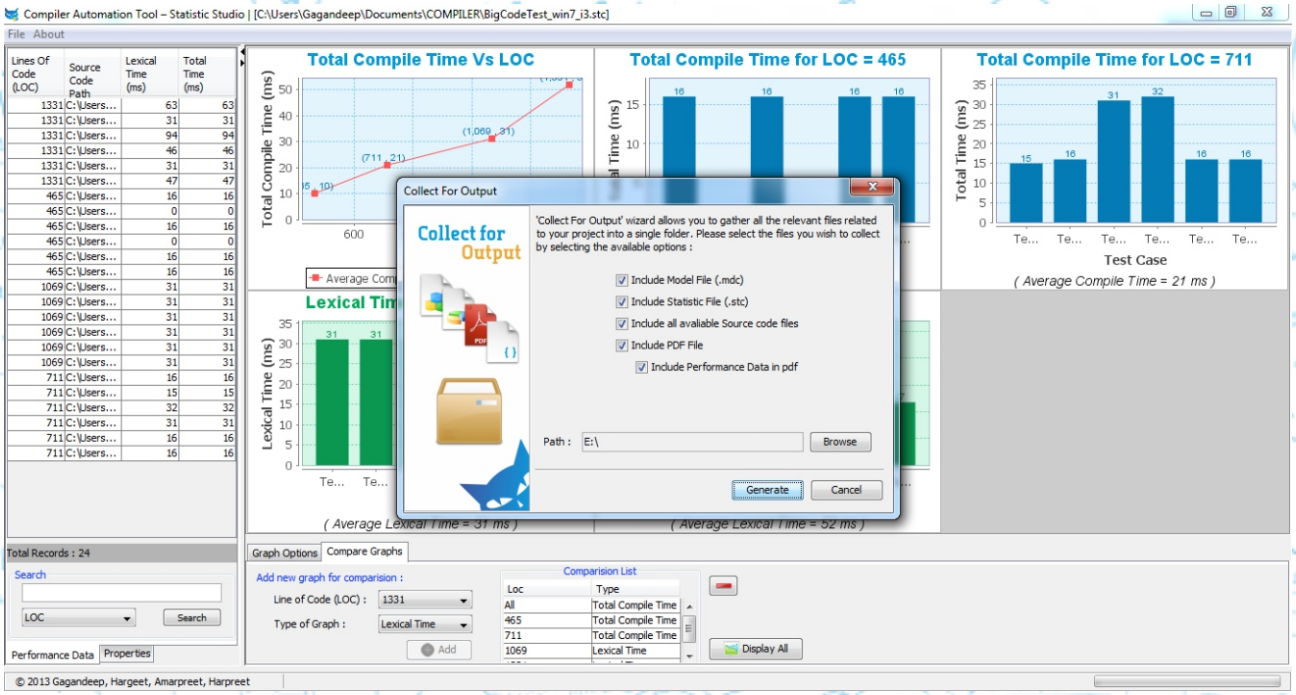


ScreenShot 26 - Total Compile Time Vs LOC graph generated as per the performance data obtained.

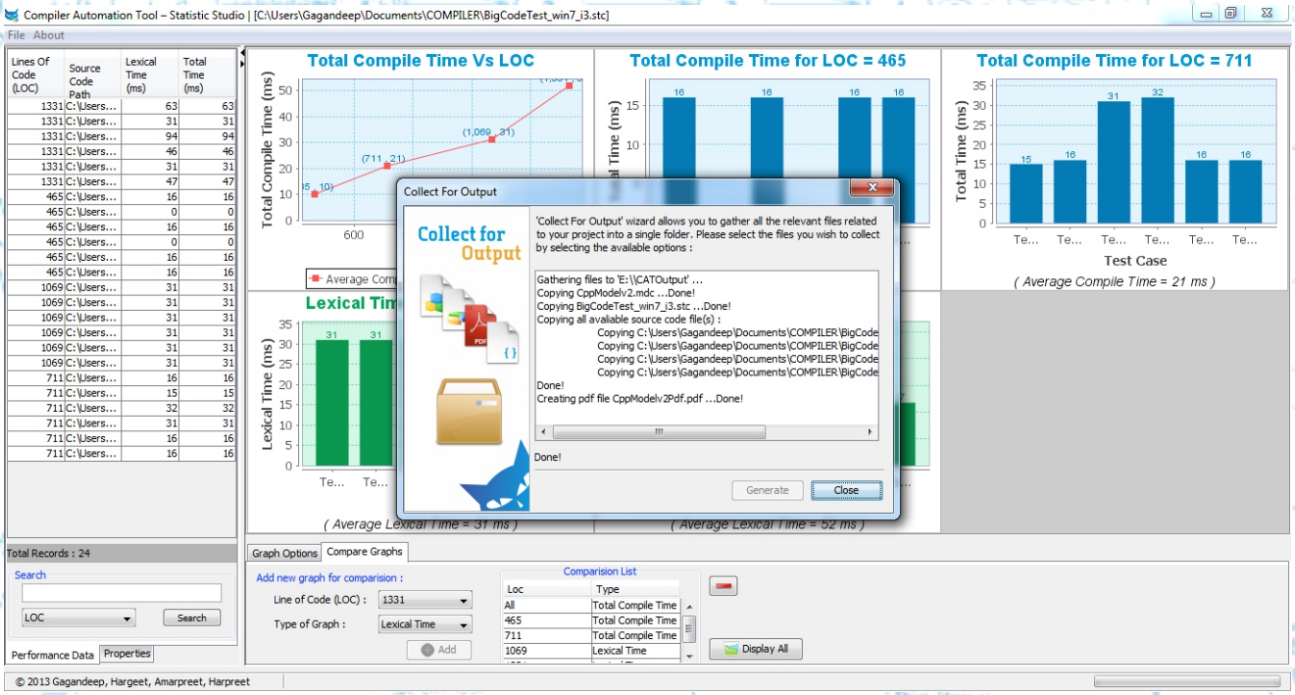
ScreenShot 27 - Comparisons of various graphs as selected by the user.



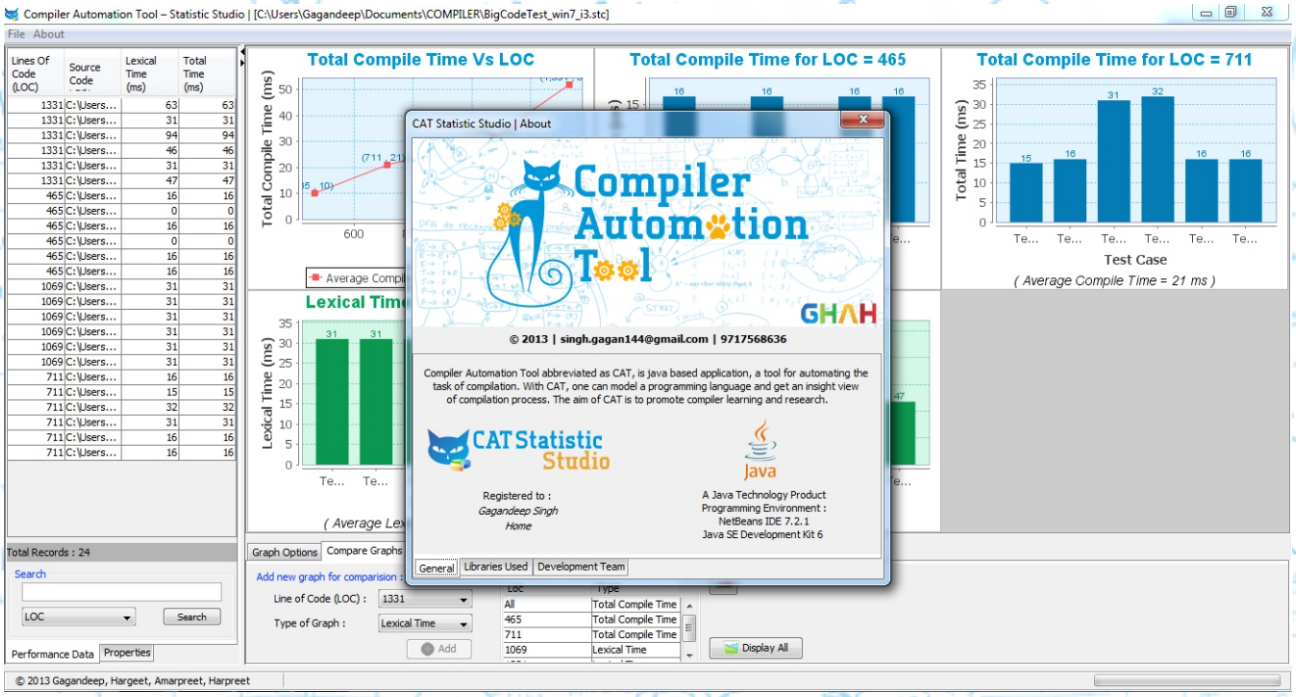
ScreenShot 28 - User generating statistic report of format PDF.



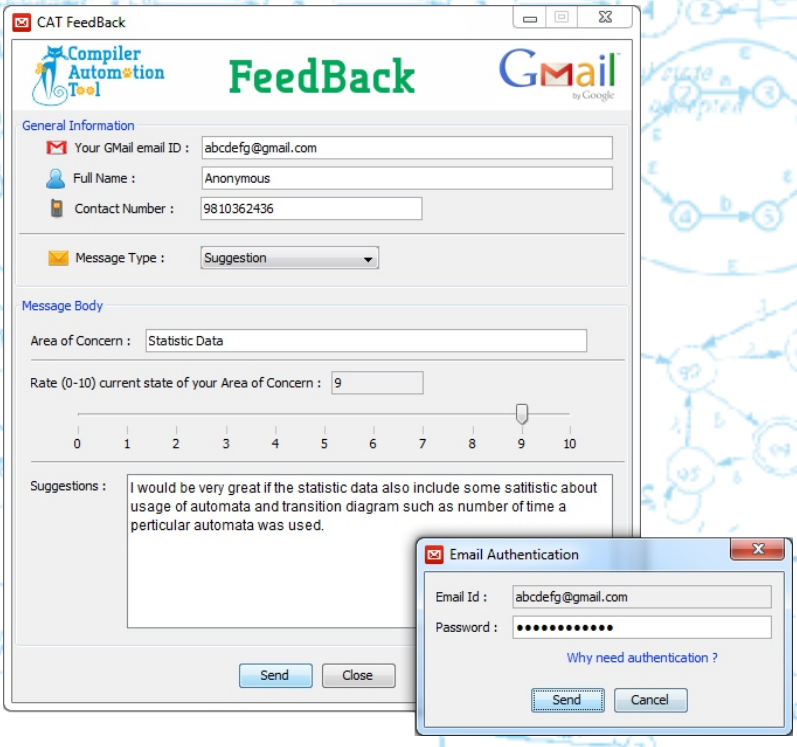
ScreenShot 29 - A smart feature of CAT Statistic Studio using which a user can gather all data files include model file, statistic , source code, pfd file into a single folder.



ScreenShot 30 - Summary of the files gathered.



ScreenShot 31 - About screen of CAT Statistic Studio.



ScreenShot 32 - CAT Feedbak. A utility using which the user can contact the support team with error reports, suggestions etc through automatically generated email.

Compiler Automation Tool

Model Performance Report

Report generated by :

CAT Statistic Studio

NAME	
DESIGNATION	
DATE	

Development Team : Gagandeep Singh, Hargeet Kaur, Amarpreet Singh, Harpreet Kaur

1. INTRODUCTION

Compiler Automation Tool (CAT) is a java based application, a tool for automating the task of compilation process. With CAT, a designer can design compiler for any language using user-friendly graphical interface, by specifying various language specification and then using it to compile source codes showing the intermediate results and performance information. CAT is compiler field specific tool and is useful for anyone related to compiler design, development as well as learning.

This report is the result of user's usage of the Compiler Automation Tool that describes his/her modelling specification regarding the programming language under consideration along with the working and the information generated by him/her. The complete report describes the summary of various actions and results that the user has obtained while using this software. Moreover, this report is software generated report whose structure and format are predefined by the CAT development team.

1.1 HOW WAS THIS REPORT GENERATED ?

As mentioned above this report has been automatically generated by CAT with respect to user specification and usage however, it is not just an outcome of a single click of a button. CAT uses three-tier architecture in which each phase forms a consumer-producer relationship with each other. This strategy not only allow user to model the language but also put it into test and generate statistical information. The below figure describes the basic architecture of CAT :

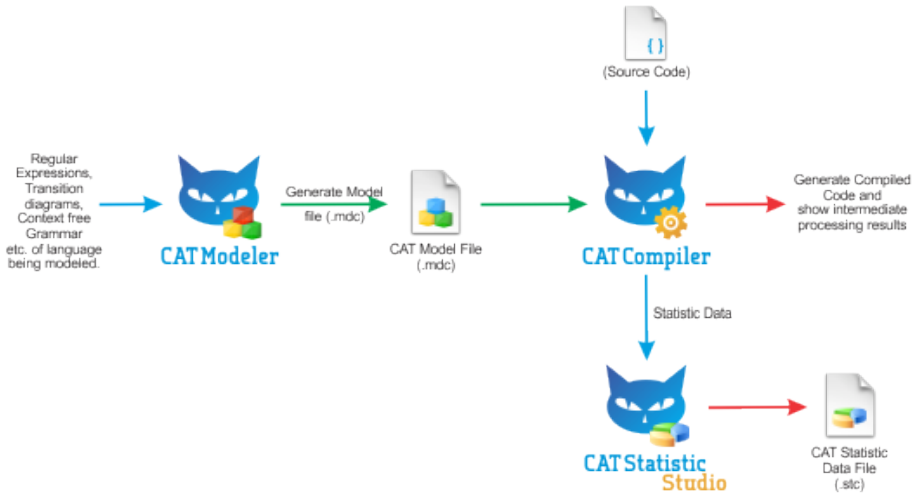


Fig 1.1 - CAT Architecture

Step 1: CAT Modeler - This is the first and most basic step in which the user decides upon a programming language and tries to model it by specifying various language related information such as regular expressions, transition diagram, context free grammar etc. Once finalised, all these information is saved in a CAT Model file with extension .mdc that is fed into the next phase.

Step 2: CAT Compiler - With the model of the language ready, the user may now use it to compile various test source code to verify the correctness of his model. He may choose any arbitrary source code of the extension supported and view intermediate actions of the compiler such as lexical analysis, symbol table etc. Besides this, the user may instruct the CAT Compiler to keep a track of the source codes compiled along with their performance information.

Step 3: CAT Statistic Studio - The information regarding source codes as collected by CAT Compiler is passed to CAT Statistic Studio where the data is analysed and presented in graphical form generating various statistical information useful in judging the performance of the model. The user may further use this application to generate

a report of format .pdf that contains all discussed information. Thus, this report is the result of this application that allows user to summarize his/her work in a well-structured formatted report.

1.2 WHAT DOES THIS REPORT CONTAIN ?

This reports contains rich collection of user usage about the product that ranges from the model he had created to the statistical information. The following report consists of the model properties followed by the environment details in which the compilation occurred, the performance data and statistical analysis including comparisons and analysis of all the phases of compilation.

2. MODEL PROPERTIES

A model in terms of Compiler Automation Tool (CAT) is a set of programming language specifications such as regular expressions, transition diagram, context free grammar etc. corresponding to a particular programming language that user wishes to model and test. Better the specifications, better the compiler performs. Besides, specifications related to the language, there are certain properties that go hand in hand with such specifications, distinguishing it from other models. These may be described in terms of language name, version, source code extensions and many more. These properties are essential part of the model which not only distinguishes them but also provide light over the language that has been modelled into it. The following table describes the properties of the model:

Table 2.1 - Model Properties

Model Attribute	Value
Model file	CppModelv2.mdc
Programming Language	C++
Model Version	2.0
Author	gagandeep Singh
Source Code extension(s)	cpp, h
Description	This model describes the compiler specifications of the programming language C++.

3. COMPILATION ENVIRONMENT

This section describes various attributes of the system on which the compilation was performed using various test source codes. Various system attributes such as memory, speed etc. has significant impact on the performance of the compiler. In fact, better the system, better is the compilation speed and response time. Of course, a system with higher memory and speed configuration is likely to compile and respond much earlier than a system with comparably lower configuration. Thus, it is necessary to specify the system attributes as the part of this report. The following table describes various system attributes on which the compilation were performed by the user:

Table 3.1 - Compilation Environment

System Attribute	Value
User Name	Gagandeep

Computer Name	Desktop-i3
OS Name	Windows 7
OS Version	6.1
System Manufacturer	INTEL_
System Model	DH55TC__
System Type	x86
CPU Model	Core(TM) i3 CPU 540 @ 3.07GHz
CPU Speed	3059
Total CPU Cores	4
CPU Manufacturer	Intel
RAM	3.2G

4. PERFORMANCE DATA

The performance data describes a tabular representation of the data related to the performance of various source codes that were compiled using the model described in above section under specified system environment. This data representation can be regarded analogous to a standard relational database (to some extent) that uses tables with attributes and tuples. As the source code gets compiled, certain information about the performance is generated if carefully observed. This can extend from overall compile time to memory usage as well as to those in individual compiler phases such as lexical, syntax and so on. So, as the source code(s) gets compiled, the user may instruct CAT Compiler to keep track of the performance. The following table describes the performance data collected by the user

Table 4.1 - Performance Data (Total no. of records :42)

S.No	Lines of Code (LOC)	Source Code Path	Lexical Time (ms)	Total Time (ms)
1	465	E:\Test Source Codes (CPP)\ADMIN.CPP	42	42
2	465	E:\Test Source Codes (CPP)\ADMIN.CPP	29	29
3	465	E:\Test Source Codes (CPP)\ADMIN.CPP	27	27
4	465	E:\Test Source Codes (CPP)\ADMIN.CPP	27	27
5	465	E:\Test Source Codes (CPP)\ADMIN.CPP	19	19
6	465	E:\Test Source Codes (CPP)\ADMIN.CPP	17	17
7	1069	E:\Test Source Codes (CPP)\AIEEEC.CPP	45	45
8	1069	E:\Test Source Codes (CPP)\AIEEEC.CPP	35	35
9	1069	E:\Test Source Codes (CPP)\AIEEEC.CPP	41	41
10	1069	E:\Test Source Codes (CPP)\AIEEEC.CPP	39	39
11	1069	E:\Test Source Codes (CPP)\AIEEEC.CPP	38	38
12	1069	E:\Test Source Codes (CPP)\AIEEEC.CPP	40	40

13	711	E:\Test Source Codes (CPP)\Brainteaser.CPP	30	30
14	711	E:\Test Source Codes (CPP)\Brainteaser.CPP	16	16
15	711	E:\Test Source Codes (CPP)\Brainteaser.CPP	24	24
16	711	E:\Test Source Codes (CPP)\Brainteaser.CPP	24	24
17	711	E:\Test Source Codes (CPP)\Brainteaser.CPP	32	32
18	711	E:\Test Source Codes (CPP)\Brainteaser.CPP	28	28
19	659	E:\Test Source Codes (CPP)\PROJECTR.CPP	38	38
20	659	E:\Test Source Codes (CPP)\PROJECTR.CPP	23	23
21	659	E:\Test Source Codes (CPP)\PROJECTR.CPP	31	31
22	659	E:\Test Source Codes (CPP)\PROJECTR.CPP	27	27
23	659	E:\Test Source Codes (CPP)\PROJECTR.CPP	30	30
24	659	E:\Test Source Codes (CPP)\PROJECTR.CPP	28	28
25	1331	E:\Test Source Codes (CPP)\REGIST.CPP	56	56
26	1331	E:\Test Source Codes (CPP)\REGIST.CPP	29	29
27	1331	E:\Test Source Codes (CPP)\REGIST.CPP	46	46
28	1331	E:\Test Source Codes (CPP)\REGIST.CPP	42	42
29	1331	E:\Test Source Codes (CPP)\REGIST.CPP	56	56
30	1331	E:\Test Source Codes (CPP)\REGIST.CPP	47	47
31	241	E:\Test Source Codes (CPP)\SUM.H	31	31
32	241	E:\Test Source Codes (CPP)\SUM.H	23	23
33	241	E:\Test Source Codes (CPP)\SUM.H	18	18
34	241	E:\Test Source Codes (CPP)\SUM.H	16	16
35	241	E:\Test Source Codes (CPP)\SUM.H	18	18
36	241	E:\Test Source Codes (CPP)\SUM.H	13	13
37	346	E:\Test Source Codes (CPP)\VERBAL.H	68	68
38	346	E:\Test Source Codes (CPP)\VERBAL.H	19	19
39	346	E:\Test Source Codes (CPP)\VERBAL.H	24	24
40	346	E:\Test Source Codes (CPP)\VERBAL.H	29	29
41	346	E:\Test Source Codes (CPP)\VERBAL.H	28	28
42	346	E:\Test Source Codes (CPP)\VERBAL.H	35	35

5. STATISTICAL ANALYSIS

The Statistical Analysis is the major part of this report for which all efforts have been made in previous section from modelling to compiling to generating data and finally analysing it. It is here where the performance data is processed and represented in graphical form to describe the efficiency of the model that the user has created. Of course, this directly describes the efficiency of the language that user intend to model. In terms of automation where CAT Compiler describes decisions made by individual phases, CAT Statistic Studio goes one step deeper into details by generating statistics. The whole idea behind statistic is to allow user to choose the best decision regarding the model on the basis of space and time complexity by comparing the data depicted by the graphs generated.

This section has been further divided into several sub sections each focusing on particular phase of compilation for instance lexical analysis, syntax analysis and so on and then finally summing up with compilation in totality.

5.1 LEXICAL ANALYSIS

This subsection of the statistical analysis targets the most basic step of any compiler i.e. the lexical analysis where various types of token are identified and recorded in symbol table. Out of all performance data collected, the information regarding lexical analysis is extracted and presented as follows.

5.1.1 LEXICAL TIME VS LOC GRAGH

This representation describes a graph plotted between lexical time and LOC (lines of Codes). On x-axis are various LOC for which codes were compiled while on y-axis lies the lexical compile time that relates to the average time taken to identify tokens for particular LOC. The purpose of this graph is to represent the average time spend in lexical analysis. As the LOC grows, one expect that the lexical time must also increase. However, it is not always the case. The reasoning regarding such cases can be interpreted from the graph as the slop rises and falls. The graph is as follows:

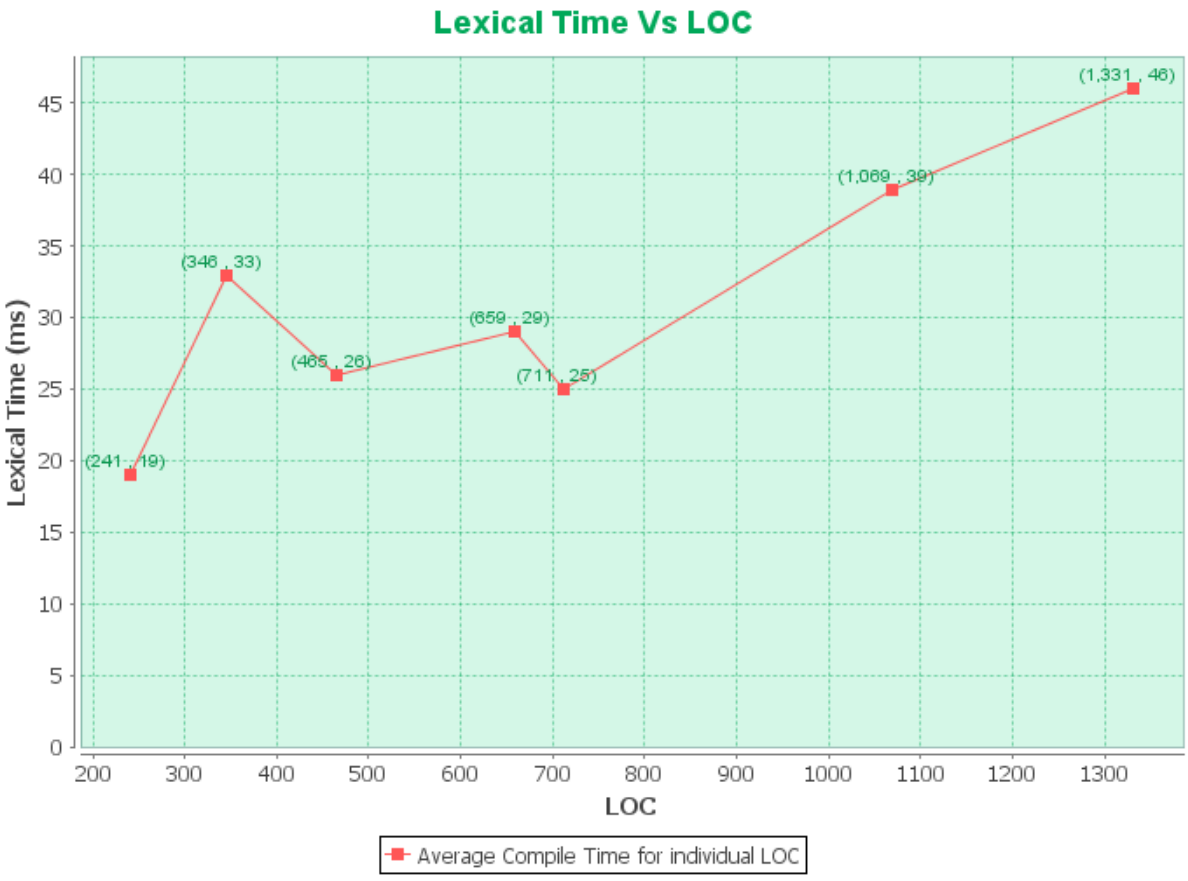
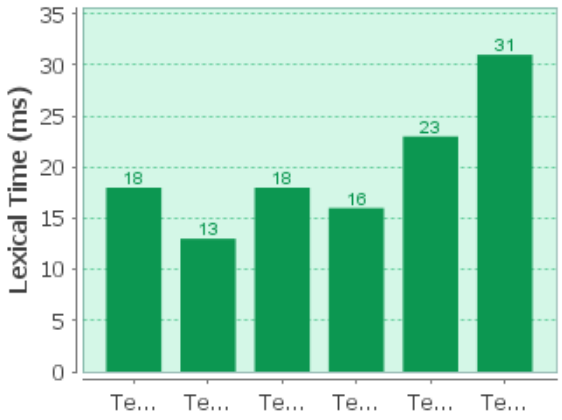


Fig 5.1 - Lexical Time vs LOC graph

5.1.2 LEXICAL TIME GRAGH FOR INDIVIDUAL LOC

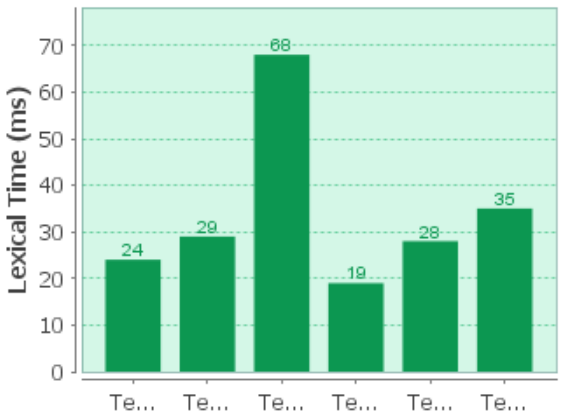
The user may compile a code more than once or perhaps many codes more than once. In such a case it necessary to take the average of the time spend describing the performance corresponding to a particular LOC. This section consider each individual LOC and plots a bar graph of the lexical time spend in each case as well as specifies the average time taken.

Lexical Time for LOC = 241



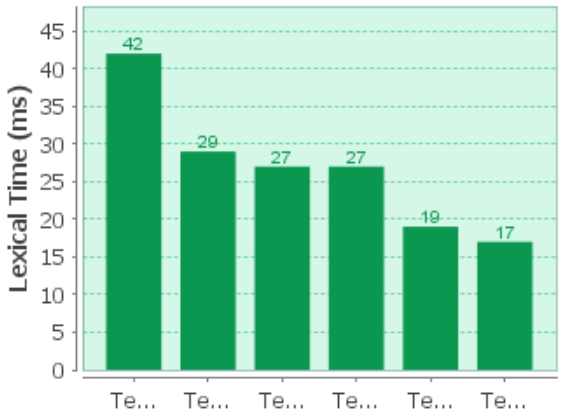
(Average Lexical Time = 19 ms)

Lexical Time for LOC = 346



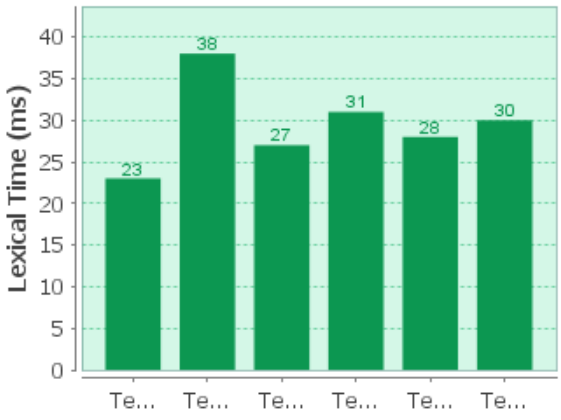
(Average Lexical Time = 33 ms)

Lexical Time for LOC = 465



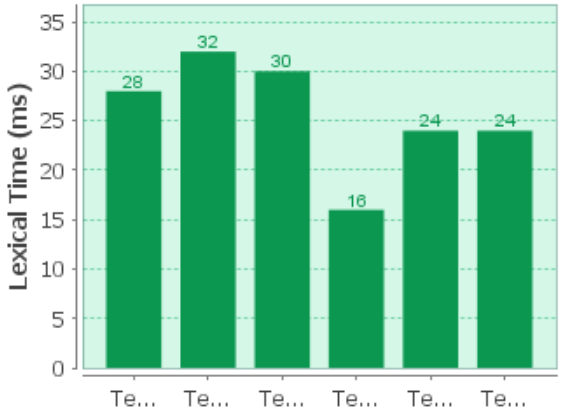
(Average Lexical Time = 26 ms)

Lexical Time for LOC = 659



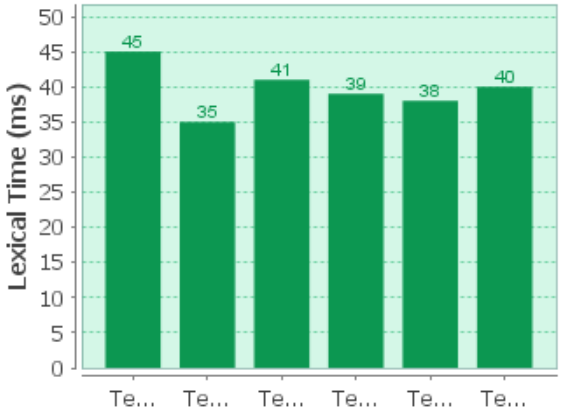
(Average Lexical Time = 29 ms)

Lexical Time for LOC = 711



(Average Lexical Time = 25 ms)

Lexical Time for LOC = 1069



(Average Lexical Time = 39 ms)

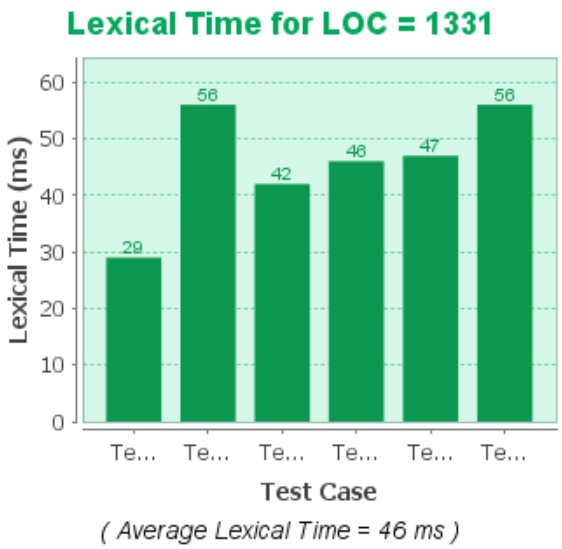


Fig 5.2 - Lexical Time graph for individual LOC

5.2 TOTAL COMPILATION ANALYSIS

This section provides over all statistical information regarding total time required to completely compile source codes covering all phases. This section describes the combined performance of individual phases and is useful in judging the performance of the compilation in totality.

5.2.1 TOTAL COMPILE TIME VS LOC GRAGH

As discussed in previous subsections corresponding to individual phases, this section provides the same information by representing a graph plotted between total compile time and LOC (lines of Codes) rather than just considering each phase. The graph is as follows:

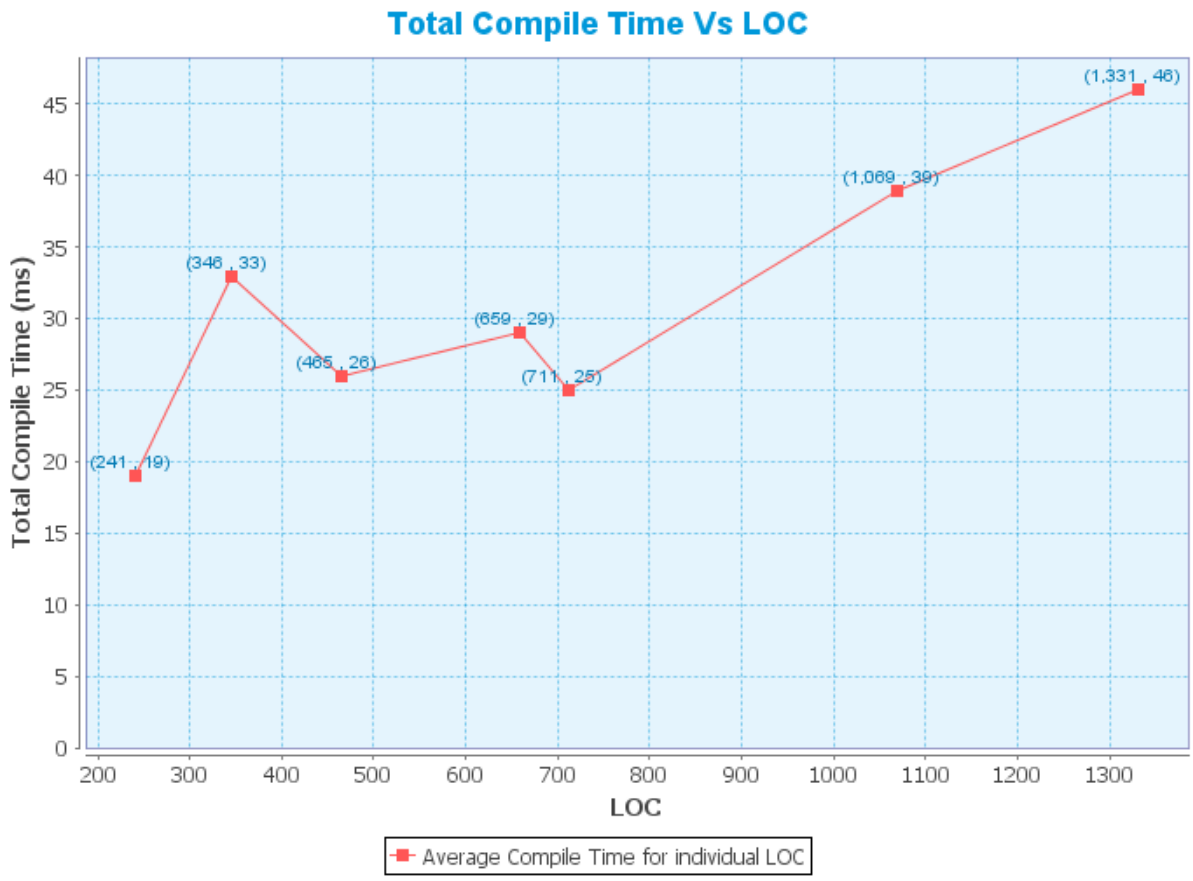
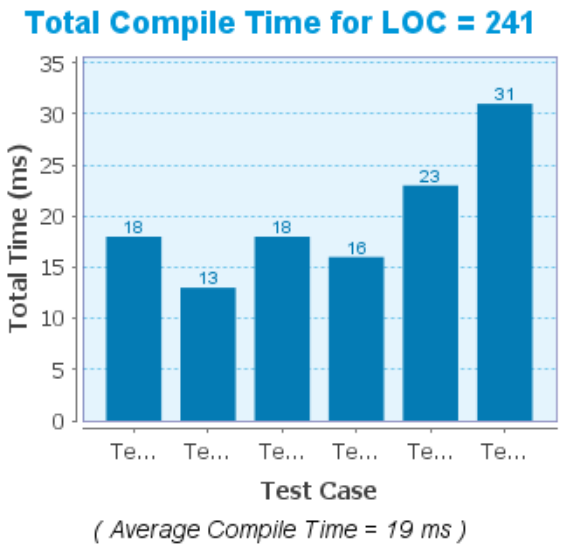


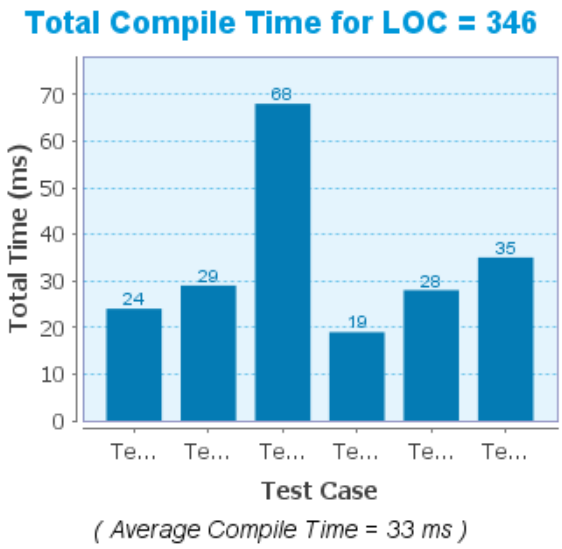
Fig 5.3 - Total Compile Time vs LOC graph

5.2.2 TOTAL COMPILE TIME GRAGH FOR INDIVIDUAL LOC

Below are the bar graphs for individual LOC describing total time taken to compile. The average total time is also specified along with each graph.

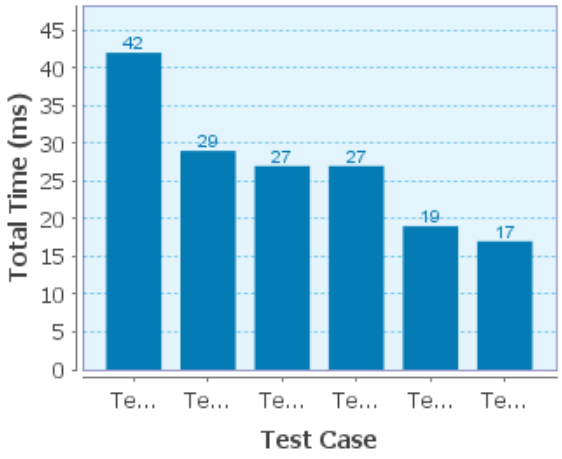


(Average Compile Time = 19 ms)



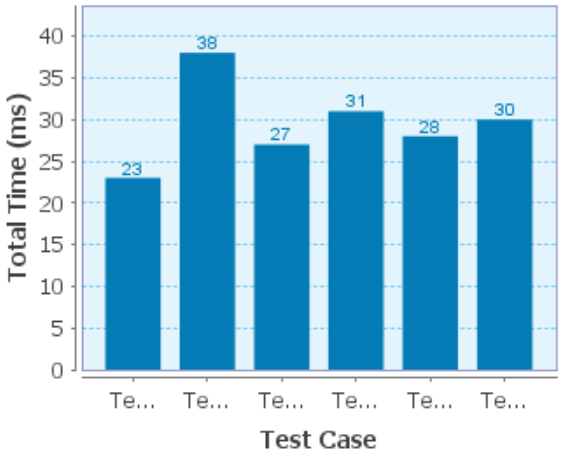
(Average Compile Time = 33 ms)

Total Compile Time for LOC = 465



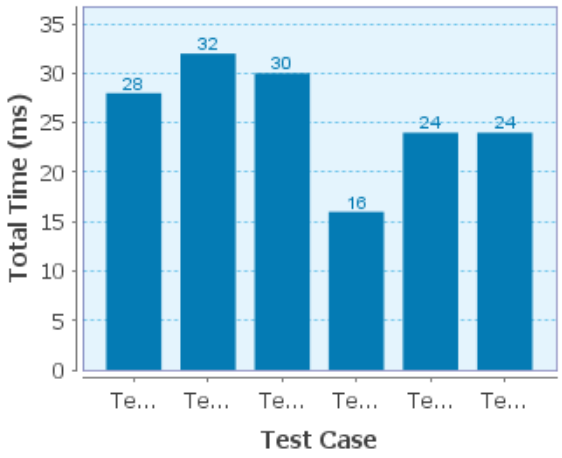
(Average Compile Time = 26 ms)

Total Compile Time for LOC = 659



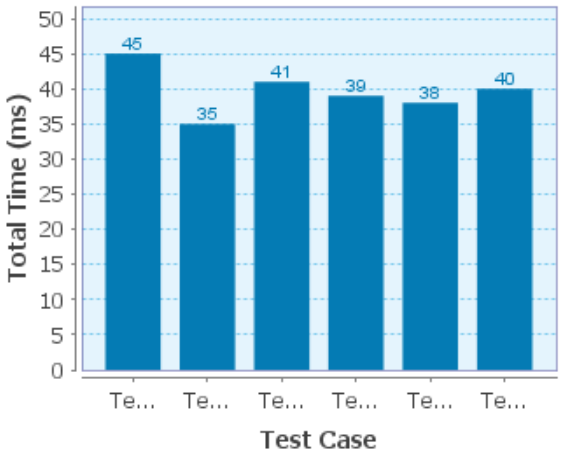
(Average Compile Time = 29 ms)

Total Compile Time for LOC = 711



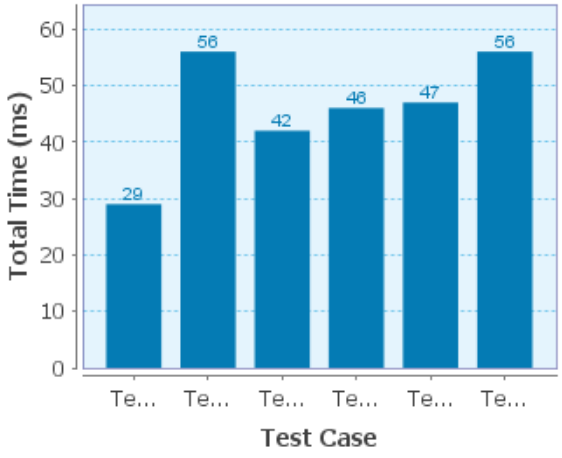
(Average Compile Time = 25 ms)

Total Compile Time for LOC = 1069



(Average Compile Time = 39 ms)

Total Compile Time for LOC = 1331



(Average Compile Time = 46 ms)

Fig 5.4 - Total Compile Time graph for individual LOC

DtransNode.java

```
package commanLib;

import java.util.HashSet;

public class DtransNode
{
   class DtLnkNode
   {
      DtransNode addr;
      String value;
      DtLnkNode next;

      public DtLnkNode()
      {
         addr=null;
         value="";
         next=null;
      }
   }

   public String lb;
   public HashSet<String> set;
   boolean mark;
   public DtransNode nextSET;
   public DtLnkNode links;


   public DtransNode()
   {
      lb=null;
      set=null;
      mark=false;
      nextSET=null;
      links=null;
   }

   static DtransNode addDtransStateToLast(DtransNode st,String l,HashSet<String> s)
   {
      DtransNode ptr=null,newState;

      ptr=st;
      while(ptr.nextSET!=null)
      {
         ptr=ptr.nextSET;
      }

      newState=new DtransNode();
      newState.lb=l;
      newState.set=s;

      ptr.nextSET=newState;

      return st;
   }
```

```
void addDtransLink(DtransNode st,HashSet<String> s, String v)  //on the basis of set U
{
   DtransNode dest=null,ptr=null;
   DtLnkNode ptrLnk=null;

   //find dest
   ptr=st;
   while(ptr!=null)
   {
      if(ptr.set.equals(s))
      { dest=ptr; break; }
      else
      { ptr=ptr.nextSET; }
   }

   //move to last of the links
   ptrLnk=links;
   if(ptrLnk==null)
   {
      ptrLnk=new DtLnkNode();
      ptrLnk.addr=dest;
      ptrLnk.value=v;
      ptrLnk.next=null;
      links=ptrLnk;
   }
   else
   {
      //travse to last
      while(ptrLnk.next!=null)
      {
         ptrLnk=ptrLnk.next;
      }

      ptrLnk.next = new DtLnkNode();
      ptrLnk=ptrLnk.next;
      ptrLnk.addr=dest;
      ptrLnk.value=v;
      ptrLnk.next=null;
   }
}


public void showLinks()
{
   DtLnkNode ptr=links;

   if(ptr==null)
   {
      System.out.print("No links");
   }
   while(ptr!=null)
   {
      System.out.print("|"+ptr.addr.lb+"|"+ptr.value+"| -> ");
      ptr=ptr.next;
   }
```

```
}

public static GraphNode DtransToGraphNode(GraphNode dfaSt, DtransNode dtrans, String lastStLb)
{
   GraphNode ptrDfa=null;
   DtransNode ptrDtrans=dtrans;

   //(1) copy All states lb
   //(a) copy first state
   ptrDfa= new GraphNode();
   ptrDfa.lb=ptrDtrans.lb;
   ptrDfa.state='i';
   if(ptrDtrans.set.contains(lastStLb))
   {  ptrDfa.state='F';  }
   dfaSt=ptrDfa;
   ptrDtrans=ptrDtrans.nextSET;

   //(b) copy remaining
   while(ptrDtrans!=null)
   {
      ptrDfa.nextSt= new GraphNode();
      ptrDfa=ptrDfa.nextSt;
      ptrDfa.lb=ptrDtrans.lb;
      if(ptrDtrans.set.contains(lastStLb))
      {  ptrDfa.state='F'; }
      ptrDtrans=ptrDtrans.nextSET;
   }

   //(2) Copy list
   DtLnkNode ptrDtLnk=null;

   ptrDtrans=dtrans;
   ptrDfa=dfaSt;
   while(ptrDtrans!=null)
   {
      if(ptrDtrans.links!=null)
      {
         ptrDtLnk=ptrDtrans.links;

         while(ptrDtLnk!=null)
         {
            ptrDfa.addLink(dfaSt,ptrDtLnk.addr.lb , ptrDtLnk.value);

            ptrDtLnk=ptrDtLnk.next;
         }

      }


      ptrDtrans=ptrDtrans.nextSET;
      ptrDfa=ptrDfa.nextSt;
   }


   return dfaSt;
}
```

```
    public Object[] createTableRow(int c,Character[] inp)
    {
        Object data[]=new Object[c];

        data[0]=lb;
        data[1]=set;

        DtLnkNode ptr=links;
        int i=2;
        while(ptr!=null)
        {
            if(ptr.value.equals(String.valueOf(inp[i-2])))
            { data[i++]=ptr.addr.lb; ptr=ptr.next; }
            else
            { data[i++]=""; }

        }

        return data;
    }


}
```

Dialog_ExceptionHandler.java

```
package commanLib;

import commanLib.CATFeedback.CATFeedback;
import java.awt.Component;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.UIManager;
import javax.swing.border.Border;

public class Dialog_ExceptionHandler extends javax.swing.JDialog
{

    public Dialog_ExceptionHandler(java.awt.Frame parent, boolean modal,String title,String
label,Throwable t)
    //public Dialog_ExceptionHandler(Component parent, boolean modal,String label, String errDescptn)
    {
        super(parent,title, modal);
        initComponents();

        this.setLocationRelativeTo(null);

        textA_lbl.setText(label);

        StackTraceElement stackTrace[]=t.getStackTrace();
        String message=t.toString()+"\n";
```

```
    for(int i=0;i<stackTrace.length;i++)
    { message+="\tat "+stackTrace[i].toString()+"\n"; }
    textA.setText(message);
}


/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
    jScrollPane1 = new javax.swing.JScrollPane();
    textA = new javax.swing.JTextArea();
    jButton1 = new javax.swing.JButton();
    textA_lbl = new javax.swing.JTextArea();
    jLabel2 = new javax.swing.JLabel();

    setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
    setMaximumSize(new java.awt.Dimension(470, 347));
    setMinimumSize(new java.awt.Dimension(470, 347));
    setResizable(false);

    jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    jLabel1.setIcon(UIManager.getIcon("OptionPane.errorIcon"));

    textA.setEditable(false);
    textA.setColumns(20);
    textA.setFont(new java.awt.Font("Courier New", 0, 12)); // NOI18N
    textA.setRows(5);
    jScrollPane1.setViewportView(textA);

    jButton1.setText("OK");
    jButton1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton1ActionPerformed(evt);
        }
    });

    textA_lbl.setEditable(false);
    textA_lbl.setBackground(new java.awt.Color(240, 240, 240));
    textA_lbl.setColumns(20);
    textA_lbl.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
    textA_lbl.setLineWrap(true);
    textA_lbl.setRows(5);
    textA_lbl.setWrapStyleWord(true);
    textA_lbl.setOpaque(false);

    jLabel2.setFont(new java.awt.Font("Tahoma", 2, 11)); // NOI18N
    jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    jLabel2.setText("Contact us at singh.gagan144@gmail.com for any queries.");
```

```
        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(198, 198, 198)
                        .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 75,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jScrollPane1)
                            .addComponent(textA_lbl)
                            .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(textA_lbl, javax.swing.GroupLayout.PREFERRED_SIZE, 0,
Short.MAX_VALUE)
                    .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE, 40,
Short.MAX_VALUE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 224,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel2)
                .addGap(12, 12, 12)
                .addComponent(jButton1)
                .addContainerGap())
        );

        pack();
    }// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        dispose();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
```

```
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Dialog_ExceptionHandler.class.getName()).log(java.util.logging.Leve
l.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Dialog_ExceptionHandler.class.getName()).log(java.util.logging.Leve
l.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Dialog_ExceptionHandler.class.getName()).log(java.util.logging.Leve
l.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Dialog_ExceptionHandler.class.getName()).log(java.util.logging.Leve
l.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the dialog */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                Dialog_ExceptionHandler dialog = new Dialog_ExceptionHandler(new javax.swing.JFrame(),
true,"<Title>","<Label Text>",new Exception("<description>"));
                dialog.addWindowListener(new java.awt.event.WindowAdapter() {
                    @Override
                    public void windowClosing(java.awt.event.WindowEvent e) {
                        System.exit(0);
                    }
                });
                dialog.setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextArea textA;
    private javax.swing.JTextArea textA_lbl;
    // End of variables declaration

    public static void autoPopJOptionPane(Component parentComponent,String title,String label,Exception
e, int messageType)
    {
```

```
        /*
        JLabel jlabel = new JLabel(label);
        jlabel.setSize(400, 25);
        jlabel.setFont(new java.awt.Font("Tahoma", 1, 11));
        */

        JTextArea textA_lbl = new JTextArea(label);
        textA_lbl.setFont(new java.awt.Font("Tahoma", 1, 11));
        textA_lbl.setBackground(null);
        textA_lbl.setEditable(false);

        StackTraceElement stackTrace[]=e.getStackTrace();
        String message=e.toString()+"\n";
        for(int i=0;i<stackTrace.length;i++)
        { message+="\tat "+stackTrace[i].toString()+"\n"; }
        JTextArea textA = new JTextArea(message);
        textA.setFont(new java.awt.Font("Courier New", 0, 11));
        textA.setEditable(false);
        JScrollPane scrollPn_textA = new JScrollPane();
        scrollPn_textA.setPreferredSize(new java.awt.Dimension(400,250));
        scrollPn_textA.setViewportView(textA);


        //JOptionPane.showMessageDialog(parentComponent, new Object[]{textA_lbl,scrollPn_textA}, title,
messageType);
        int choice =JOptionPane.showOptionDialog(parentComponent, new
Object[]{textA_lbl,scrollPn_textA}, title,JOptionPane.YES_NO_OPTION, messageType,null,new
String[]{"Close","Report Error"},null);
        if(choice==JOptionPane.NO_OPTION)
        {
           CATFeedback feedBkFrame = new CATFeedback(CATFeedback.REPORT,label,message);
           feedBkFrame.setVisible(true);
        }

    }


    public static void autoPopMessageJOptionPane(Component parentComponent,String title,String
label,String message, int messageType)
    {
        JTextArea textA_lbl = new JTextArea(label);
        textA_lbl.setFont(new java.awt.Font("Tahoma", 1, 11));
        textA_lbl.setBackground(null);
        textA_lbl.setEditable(false);

        JTextArea textA_msg = new JTextArea(message);
        textA_msg.setFont(new java.awt.Font("Tahoma", 0, 11));
        textA_msg.setEditable(false);
        textA_msg.setBackground(null);
        textA_msg.setBorder(null);

        JOptionPane.showMessageDialog(parentComponent, new Object[]{textA_lbl,textA_msg}, title,
messageType);

    }
```

}


GURU TEGH BAHADUR INSTITUTE OF TECHNOLOGY
DELHI
ਦੇਗ ਤੇਗ ਫ਼ਤਹਿ ॥