JFORLAN TOOL

By

SRINIVASA ADITYA UPPU

B.Tech, JAWAHARLAL NEHRU INSTITUTE OF TECHNOLOGICAL SCIENCES, INDIA, 2007

A REPORT

Submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2009

Approved by:

Major Professor
Dr. Alley Stoughton

Abstract

Forlan is a computer toolset for experimenting with formal languages. Forlan is implemented as a set of Standard ML (a functional programming language) modules. Forlan presently includes a tool named 'JFA' (Java Finite Automata Editor) which is a Java GUI tool for creating and editing 'Finite Automata' and a tool named 'JTR' (Java Trees Graphical Editor) which is used for creating and editing 'Parse Trees' or 'Regular Expression Trees'.

The JForlan tool is an attempt to unify the 'JFA' and the 'JTR' tools into one single tool so as to make it more robust, efficient and easy to use. Apart from integrating the tools a lot more functionality like creating and editing 'Regular Expression Finite Automata' and 'Program Trees' (special kinds of Forlan trees which are used to represent Programs in Forlan) has been added to the tool which were not present in the 'JFA' and the 'JTR' tools. As the 'Automata' and the 'Trees' are closely related concepts it is highly beneficial to combine the tools into one single tool.

The JForlan tool can be invoked either from Forlan or can be run as a standalone Application. Due to the integration of the 'JFA' and the 'JTR' tools the user can now view the regular expression which was entered as a transition label in the 'Automata' mode as a tree structure in the 'Tree' mode. Another important feature which is added to the tool is that during the creation of the trees the user need not follow the top down approach only (i.e. creating first the root and then adding children to it) but can create the nodes of the tree in any order the user wishes. An important feature of the tool is that after drawing the necessary automaton or tree the user can save it directly as an 'image' or a JForlan project or can select the option of saving it in Forlan syntax, which translates the figures drawn into the Forlan code automatically. The main purpose of developing this tool is to provide a user friendly, easy to use tool which would be ideal for students as educational software which would help them to learn and understand the basic concepts of automata and tree structure.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my Major Professor Dr. Alley Stoughton for her constant help, encouragement and guidance throughout the project.

I would also like to thank Dr. Gurdip Singh and Dr. Torben Amtoft for serving in my committee and for their valuable cooperation during the project.

Finally, I wish to thank my family and friends for all their support and encouragement.

CHAPTER 1 - INTRODUCTION

1.1 Forlan

The conventional way of students learning the concepts of formal languages is with the use of a pencil and a paper. Forlan attempts to create a set of tools which can be used by the students to learn these concepts in an innovative and an interactive way. Forlan supports software tools which can generate visual diagrams of the already existing Forlan code which can be modified as necessary with ease just with a few mouse clicks. The tools can also be used to generate the Forlan code automatically by drawing different kinds of automata and tree structures by using the interactive user interface which is provided by the tool. Presently the Forlan tool set has got separate tools one for drawing 'Automata' called the 'Java Finite Automata Graphical Editor' (JFA) and another for drawing 'Parse Trees' and 'Regular Expression Trees' called the 'Java Trees Graphical Editor' (JTR) .

1.2 JFA (Java Finite Automata Graphical Editor)

JFA is a Java program for creating and editing Forlan Finite Automata. The JFA user interface provides buttons which allow the user to perform actions like creating the nodes of the Automata, creating the transitions between the nodes. Buttons are also provided to create a new JFA project, open an already existing JFA project and to save the current project. The JFA project can be saved either in the JFA's original file format which stores the Finite Automata's spatial data as well as its states and transitions or can be saved in the Forlan Syntax format which is a text file in Forlan's syntax for Finite Automata or can be saved as an image file of the format PNG (Portable Network Graphics). JFA can be invoked either from the Forlan command prompt or can be started as a standalone application. In the stand-alone mode the user can open more than one Finite Automaton simultaneously. Each Finite Automaton is opened in a separate tab whereas in the Forlan mode we need to pass a file name as the parameter while starting JFA so

that only that particular file is loaded into the JFA. A Screenshot of the JFA tool is shown below in Figure 1.1

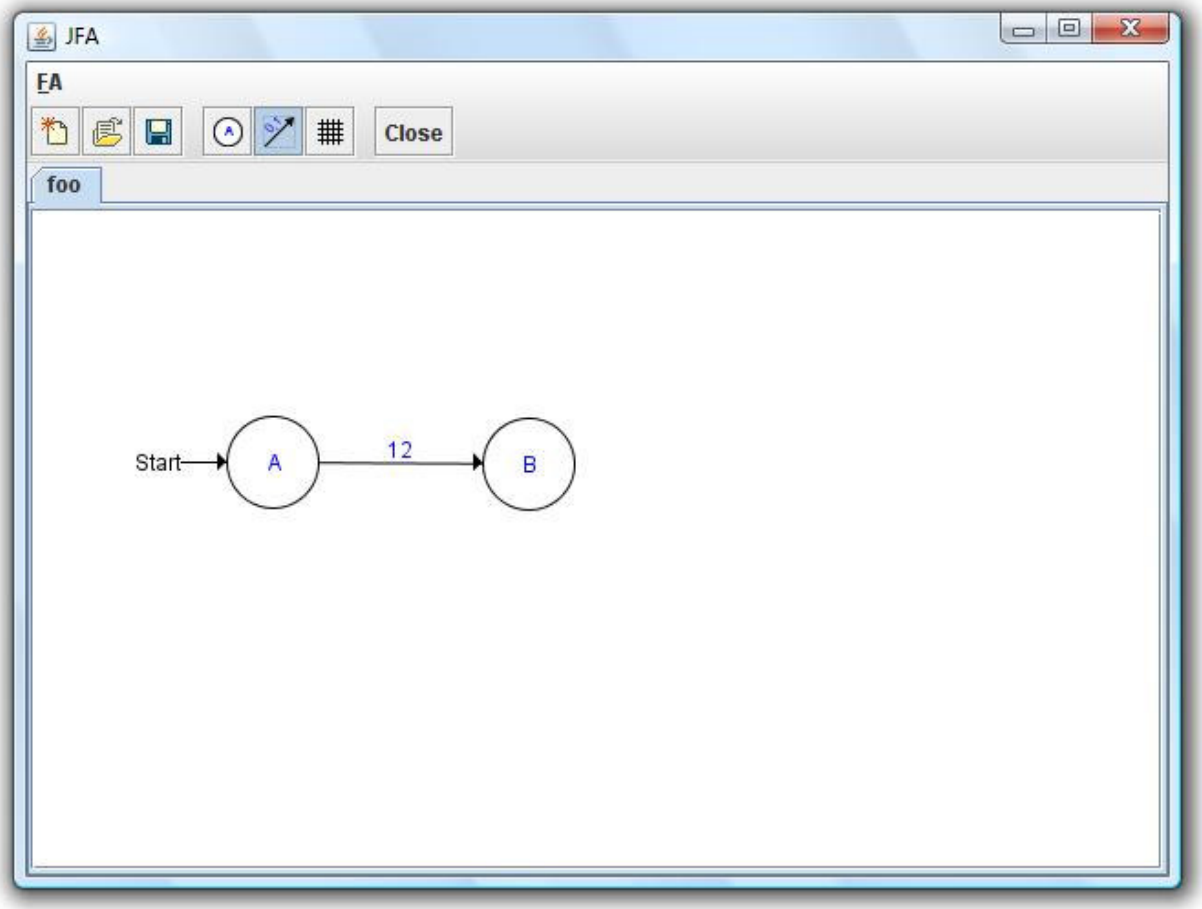

Figure 1.1 A screenshot of a Finite Automaton drawn using the JFA tool

1.3 JTR (Java Trees Graphical Editor)

JTR is another tool of the Forlan toolset which is used for creating and editing Forlan Regular Expression and Parse Trees. The JTR user interface like the JFA interface provides buttons which allow the user to perform actions like creating the nodes of the tree, deleting the nodes of the tree. Buttons are also provided to create a new JTR project, open an already existing JTR project and to save the current project. JTR allows the user to select the type of tree the user

wants to create or open while creating a new JTR project or opening an already existing project respectively. The JTR project can be saved in the JTR's original file format which stores the data of the tree including its spatial data. The JTR project can also be saved in the Forlan Syntax format which is a text file in Forlan's syntax for the tree or the JTR project can be saved as a PNG (Portable Network Graphics) file similar to that of JFA. There is a special button provided by JTR which can be used to check the tree for any errors after you have finished creating your drawing. Depending upon whether you selected the parse tree mode or the regular expression mode the tree will be checked and errors reported accordingly. JTR also can be invoked either from the Forlan command prompt or can be started as a standalone application depending on which the user can either work with more than one JTR project simultaneously or not. A screenshot of the JTR tool is shown below in Figure 1.2

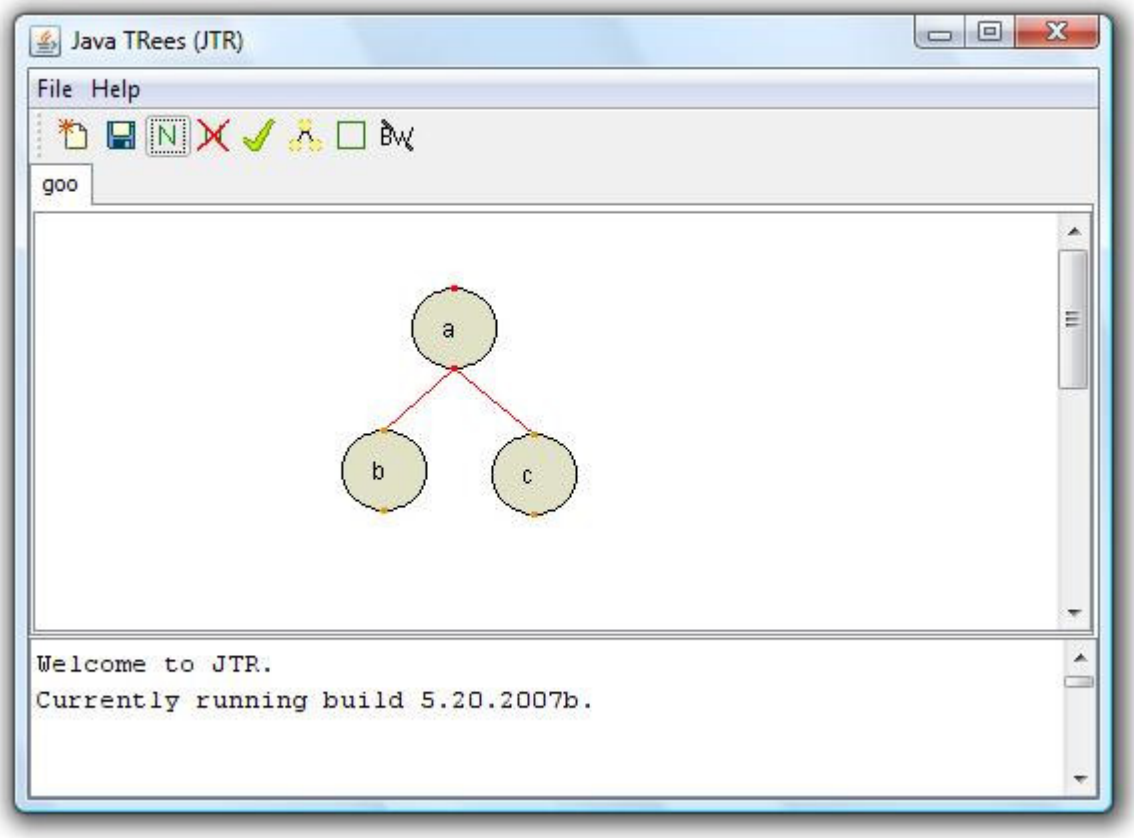

Figure 1.2 A screenshot of a tree drawn using the JTR tool

1.4 Problems with the Existing System

The present tools JFA and JTR does not support a few Forlan automata types and tree types and contain some bugs which need to be corrected. The following list gives details of the functionality which needs to be added and it also lists the major errors which need to be corrected.

- In JFA there is no option of creating and editing Forlan's 'Regular Expression Finite Automata' (RFA). RFA are a special kind of Forlan Finite Automata in which the labels of the transitions between the nodes can take 'Regular Expressions' as their values.

- The transitions in the 'automata' mode are straight lines and cannot be converted into arc's because of which there is no flexibility in the layout of the Automata.

- There is no option of checking whether the Finite Automaton drawn is either valid or invalid as there is with that of JTR.

- In JTR there is no option of creating and editing Forlan's 'Program Trees' which are special kind of Forlan trees which are used to represent Forlan Programs.

- While translating the figures drawn in JTR to Forlan Syntax the output depends on the order in which the children are added to the node. Hence if the children are moved around in JTR then the output is not consistent with the relative positioning of the nodes.

- While drawing trees in JTR always a top down approach needs to be followed (i.e. the parent should be created first and then it's children) because of which it is not robust and flexible.

1.5 Need for Integrating JFA and JTR tools

As the Finite Automata, Regular Expressions, Parse Trees are all different aspects of formal language theory it would be highly beneficial to merge these two tools into one single tool which can be used either in the ‘Automata’ mode to draw the desired Forlan automata or can be used in the ‘Tree’ mode to draw Forlan trees. By doing this a bridge is created between the automata and the trees which would help the students a lot in learning the principles of formal language theory. After the user creates a Finite Automata in the ‘Automata’ mode by creating nodes and drawing transitions between them, the user can view the regular expression transition labels that he entered in the form a tree structure. This helps the user to visual the regular expressions not just as a string of symbols but also in the form of a tree structure which helps them to understand regular expressions and formal language theory in a better way.

CHAPTER 2 - REQUIREMENT SPECIFICATION

2.1 Goal

The idea behind the 'JForlan Tool' Project is to provide a single unified tool which provides all the features of the 'JFA' and the 'JTR' tools and to remove all the problems which are present in the existing system. As the main goal of the 'JForlan Tool' is to provide an interactive, easy to use interface for the users, extensive research has been done to gain an insight into the various features which will be beneficial to the user and the user's behavior to the various scenario's which can occur while using the tool. As this is a learning tool for the students extensive care has been taken in order to convey the mistakes made by the user while creating and editing the Automata and Trees drawn, and to guide the user in the correct direction. This is achieved with the help of pop up messages prompted by the tool in case the user does any mistake and using the color coding scheme the tool indicates to the user as to where the user went wrong. The working of the tool is made convenient and easy to use for the end user. Dr. Stoughton, Associate Professor, CIS, provided regular feedback on the project which helped a lot in making the tool more robust.

2.2 Scope

This tool can be used for either teaching or learning the various concepts of automata and trees in an innovative way. The students will find the tool very useful because of its ease of use feature and the various options which are provided by the tool. It is always beneficial to learn the concepts of formal languages in a pictorial and a graphical way rather than the conventional way of paper and pen. The tool helps the students to visualize the Forlan code of Automata and trees and understand it in a better way.

2.3 Platform Specifications - Deployment

2.3.1 Hardware Specifications

Processor P IV

RAM 128 MB

Minimum Space Required 10 MB

Display 16 bit color

2.3.2 Software Specification

Operation Environment Win2000/XP/Vista

Java SE 6

Forlan toolset

SML/NJ (Version 110.65 or greater)

CHAPTER – 3 SYSTEM ANALYSIS

3.1 Reverse Engineering

The first step in order to accomplish integrating the JFA and the JTR tools was that to analyze the structure and behavior of each of the tool separately and to obtain the class diagrams using the concepts of reverse engineering. The class diagram for the JFA tool is shown below

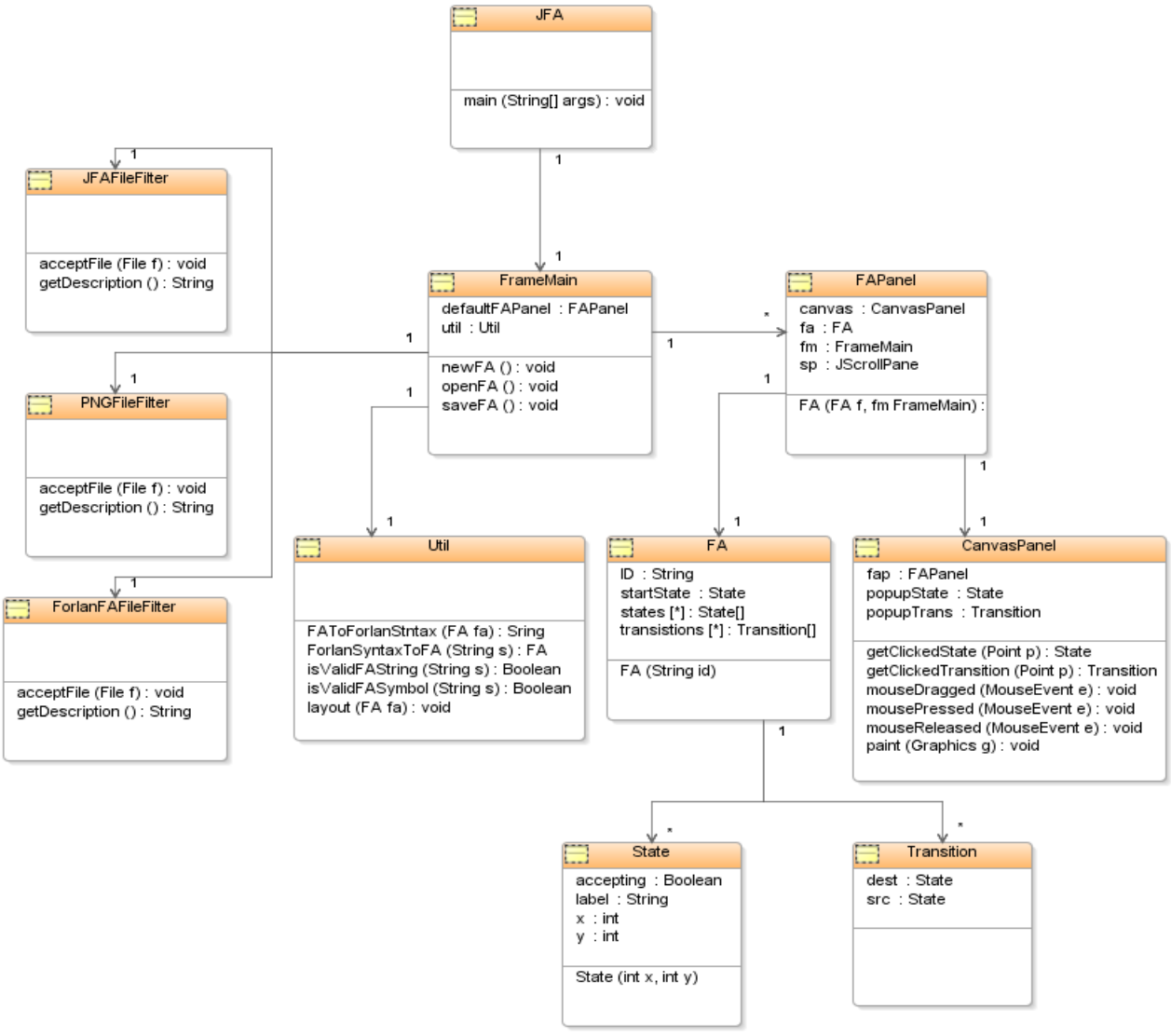

Figure 3.1 Class Diagram of JFA

The class diagram for the JTR tool is as shown below

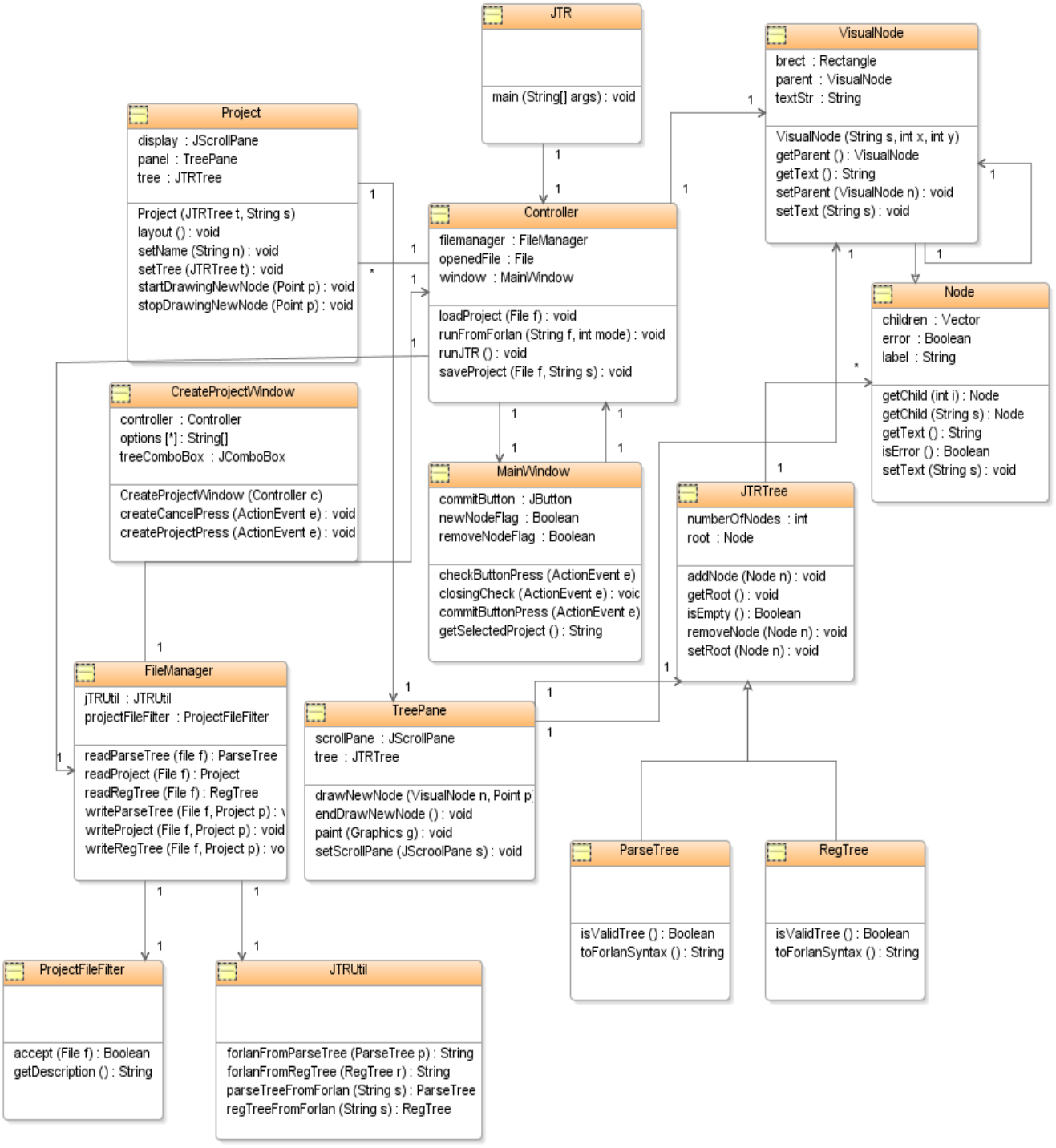

Figure 3.2 Class Diagram of JTR

These diagrams helped a lot to understand the different entities (classes) which are present in the tools, their functionality and the relationship among them. These diagrams also helped to figure out the similarities and differences which existed in the JFA and the JTR tools. The process of reverse engineering is a very important part in the project because in order to achieve integration of the tools one needs to know the structure and the way the data flow occurs in the existing system.

The next step in the Analysis process was to co-relate the entities (classes) and the attributes in the entities of JFA tool with those of the JTR tool on the basis of their functionality. This was an important task which helped to identify which classes need to be merged together and modified in order to achieve the necessary integration.

3.2 Design for adding New Features

The next step in the design process was to decide the classes and the functionality which is to be added to the existing system in order to incorporate the new features such as support for Forlan Regular Finite Automata (RFA), program trees (PT). The Design process also involved identifying the classes which need to be modified from the existing set of classes in order to incorporate functionality like providing more flexibility while drawing the transitions in an Automaton in the form of arcs which can be moved easily by dragging the mouse instead of straight lines which is how presently the transitions are being implemented while drawing the automata.

CHAPTER 4-IMPLEMENTATION

4.1 User Interface Design and Implementation

The user interface of the application has been designed and implemented in Java using the Eclipse IDE. Since the JForlan tool involved working with Windows, Graphics and text, various Swing components were used in developing it. Swing encompasses a set of features for building graphical user interfaces (GUI's) and adding rich functionality and interactivity to Java Applications. Swing is built on top of AWT (Abstract Window Toolkit) which supports Graphical User Interface (GUI) programming. AWT features the core foundation of Java Window Applications. It includes a robust event-handling model, graphics and imaging tools including shape, color and font classes. AWT also incorporates layout managers for flexible window layouts. It also incorporates a basic set of user interface components such as windows, buttons, etc. However there are various advantages of using Swing components over the AWT (Abstract Windows Toolkit) components such as Swing components are more powerful and flexible components than the AWT components. In addition to the AWT components such as buttons, check boxes Swing provides several additional components like tabbed panes, scroll panes etc. Another important advantage of Swing Components over AWT components is that they are lightweight components and are platform independent. Apart from Swing the other major API which is used in the project is the Java 2D API. It consists of a set of classes for advanced 2D graphics and imaging, encompassing line art, text, and images in a single comprehensive model. A Screen shot of the JForlan tool is as shown in figure 4.1

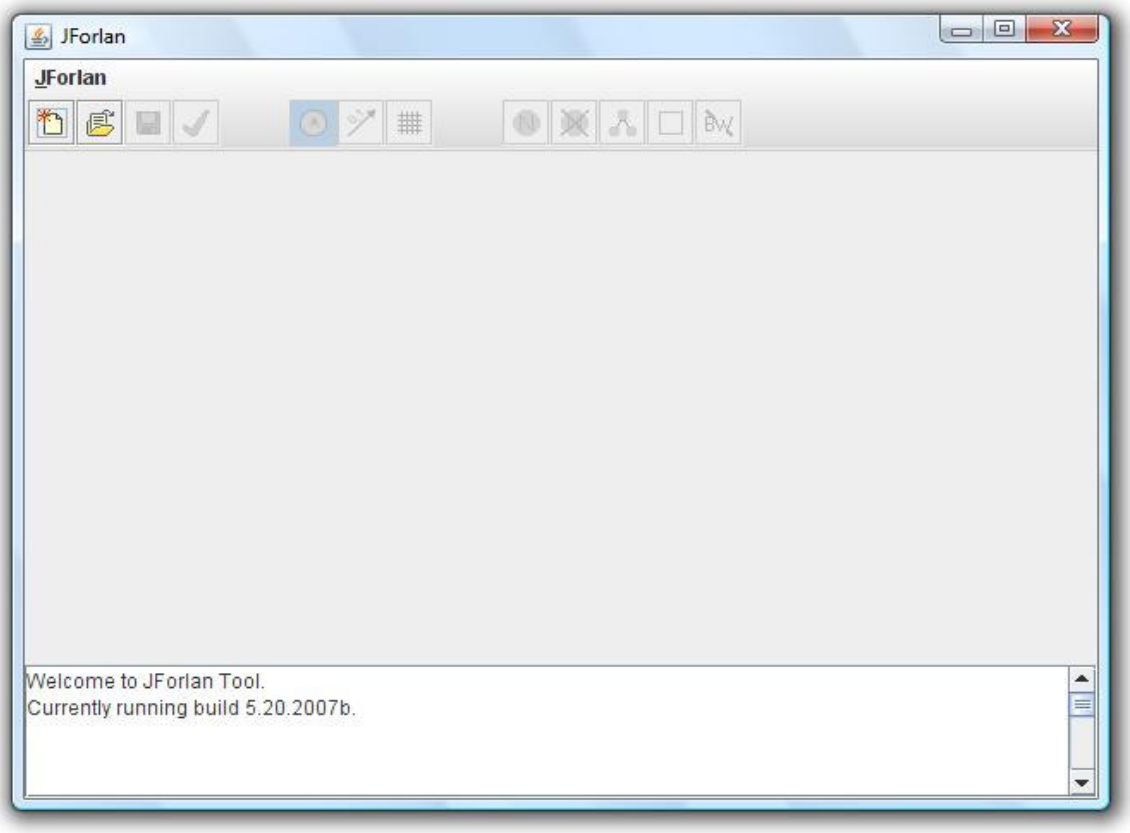

Figure 4.1 A Screen Shot of the JForlan Tool

As the JForlan tool is an integration of the JFA and the JTR tools, the JForlan toolbar at the top has buttons of both the JFA tool and the JTR tool. The buttons on the toolbar can be divided into three categories depending upon their functionality. The first category consists of general buttons which can be used both in the 'Automaton' mode and the 'Tree' mode. These include buttons for creating new 'Automata' or 'Trees', opening already existing 'Automata' or 'Trees', saving the 'Automata' or 'Trees' and for checking whether an 'Automaton' or 'Tree' is valid or not. The second category of buttons is those which are helpful to the user when working in the 'Automata' mode. It consists of three buttons of which the first one is used to draw the states of the Automaton and the second button is used for drawing transitions between the different states of the Automaton and the third one is the snap to grid button which when activated moves the states of the automaton by a fixed distance every time the user moves tries to move the states. The third category of buttons is those which are helpful to the user when working in the 'Tree' mode. In consists of five buttons one for creating the nodes of a tree. Second for deleting the already existing nodes of the tree and third is the layout button which

when clicked uses a default layout algorithm and format's the nodes of the tree so that the tree appears in a proportionate manner. The fourth button is the Bounding Boxes button using which the visibility of the Bounding Boxes can be turned on/off. The last button is used to toggle the coloring of the lines and Bounding Boxes. Care has been taken that when the user is using the tool in the 'Automaton' mode then only the buttons which are used in creating and editing the Automata are enabled and the rest are disabled. Similarly when the user is using the tool in the 'Tree' mode then only the buttons which are needed for creating and editing the tree will be enabled. Apart from the buttons on the toolbar these features are also provided using the JForlan menu as shown in the figure below

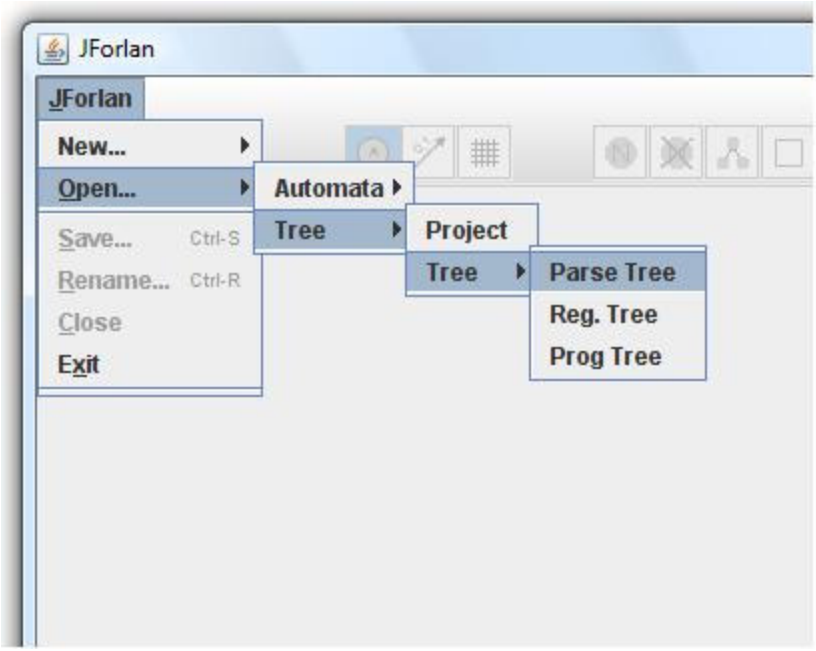

Figure 4.2 A Screen Shot of JForlan Menu

When the user clicks on the 'new' button then the user is presented with a window where JForlan allows the user to select the type of diagram the user wants to create i.e. an 'Automaton' or a 'Tree' as shown below in Figure 4.3.

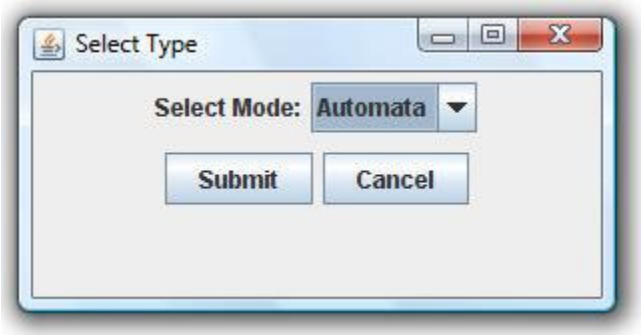

Figure 4.3 Mode Selecting Window

If the user selects to create an Automaton then JForlan presents the user with a window to select whether the user wants to create a ‘Finite Automaton’ (FA) or a ‘Regular Expression Finite Automaton’ (RFA) as shown in figure 4.4.

Figure 4.4 Window to select type of Automaton

If the user selects to create a tree then JForlan presents the user with a window where the user needs to enter the name of the project and select the type of tree the user wants to create i.e. a ’Parse Tree’, ‘Regular Expression Tree’ or a ‘Program Tree’ as shown in the figure 4.5.

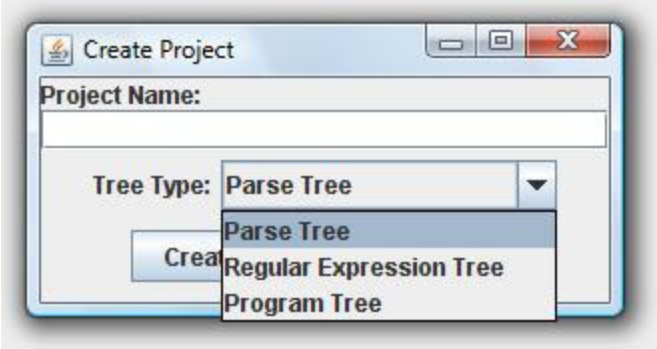

Figure 4.5 Window to select type of Tree

Once the user selects the type of diagram a 'Tabbed Pane' is added to the tool where the user can create the desired diagram using the buttons which are available on the JForlan toolbar. The major difference in drawing the Automaton using the JFA tool and the JForlan tool is that in JFA tool the transitions are just straight lines but in JForlan tool more flexibility is provided for the transitions i.e. the user can click on the midpoint of the transition (represented by a small black dot) and drag it so as to convert the straight line into an arc as shown in figure below

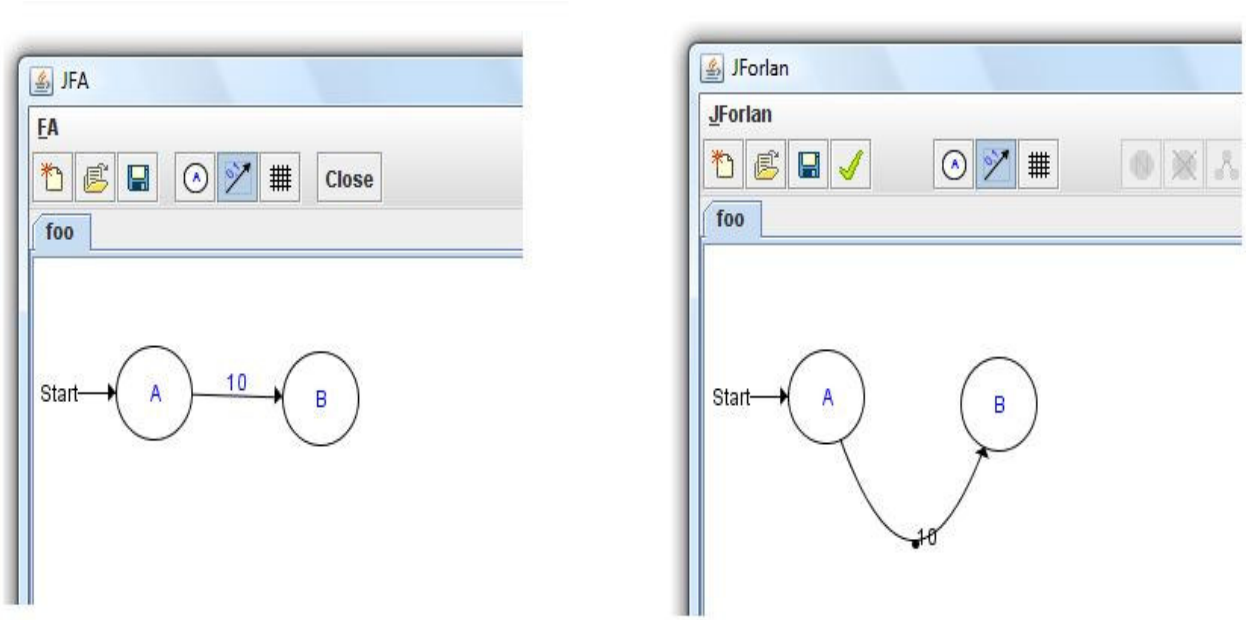

Figure 4.6 Difference between the JFA tool and the JForlan tool while drawing an Automaton

The JForlan tool also allows the user to create a 'Regular Expression Finite Automaton' in which the transitions between the nodes can take regular expressions as their labels which is not possible in JFA tool. After entering the transition label the user can view it in the form of a tree structure by just right clicking on the transition and selecting the "View Tree" option which opens a new tab next to the existing one which shows the transition label in the tree structure as shown in the figure below.

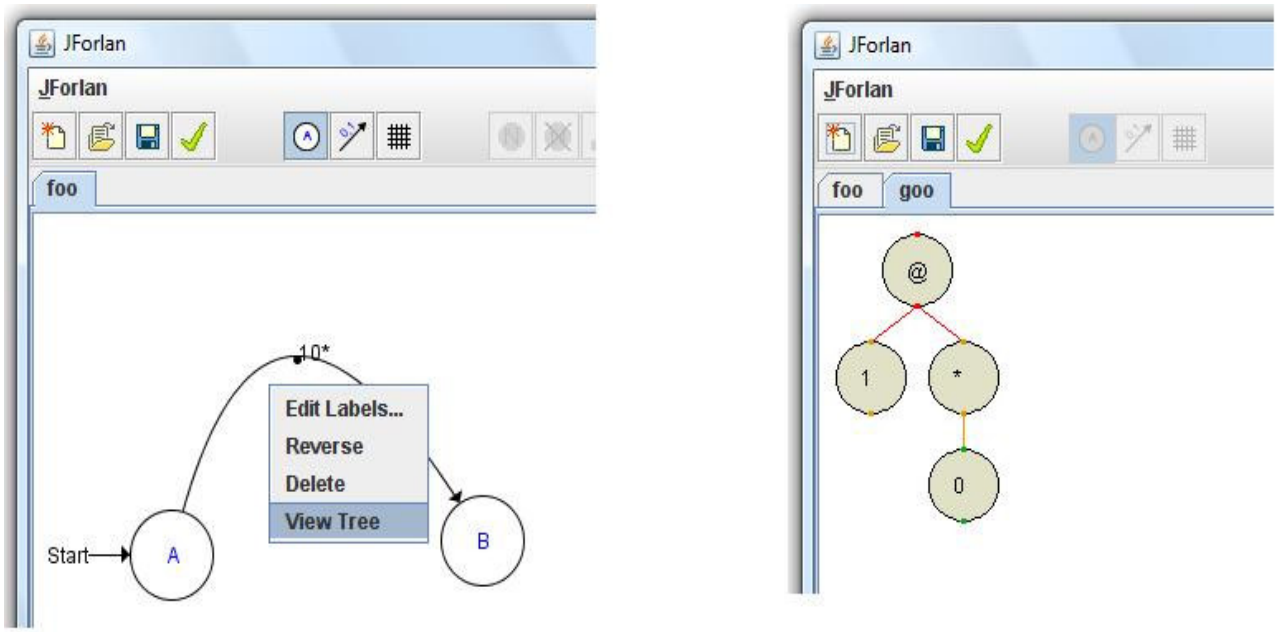

Figure 4.7 Viewing the tree structure of the Transition label drawn in Automaton

The JForlan tool also allows the user to draw "Program Trees" (special kind of Forlan trees which are used to represent Programs in Forlan) apart from "Parse Trees" and the "Regular Expression Trees" which can be drawn even by the JTR tool. In order to draw a program tree the user has to click on create a new project button and select the tree mode, then the user has to enter the project name and select program tree as the type of tree the user wants to create which adds a 'Tabbed Pane' to the tool where the user can create the desired Program Tree. A sample program tree is as shown in the figure 4.8.

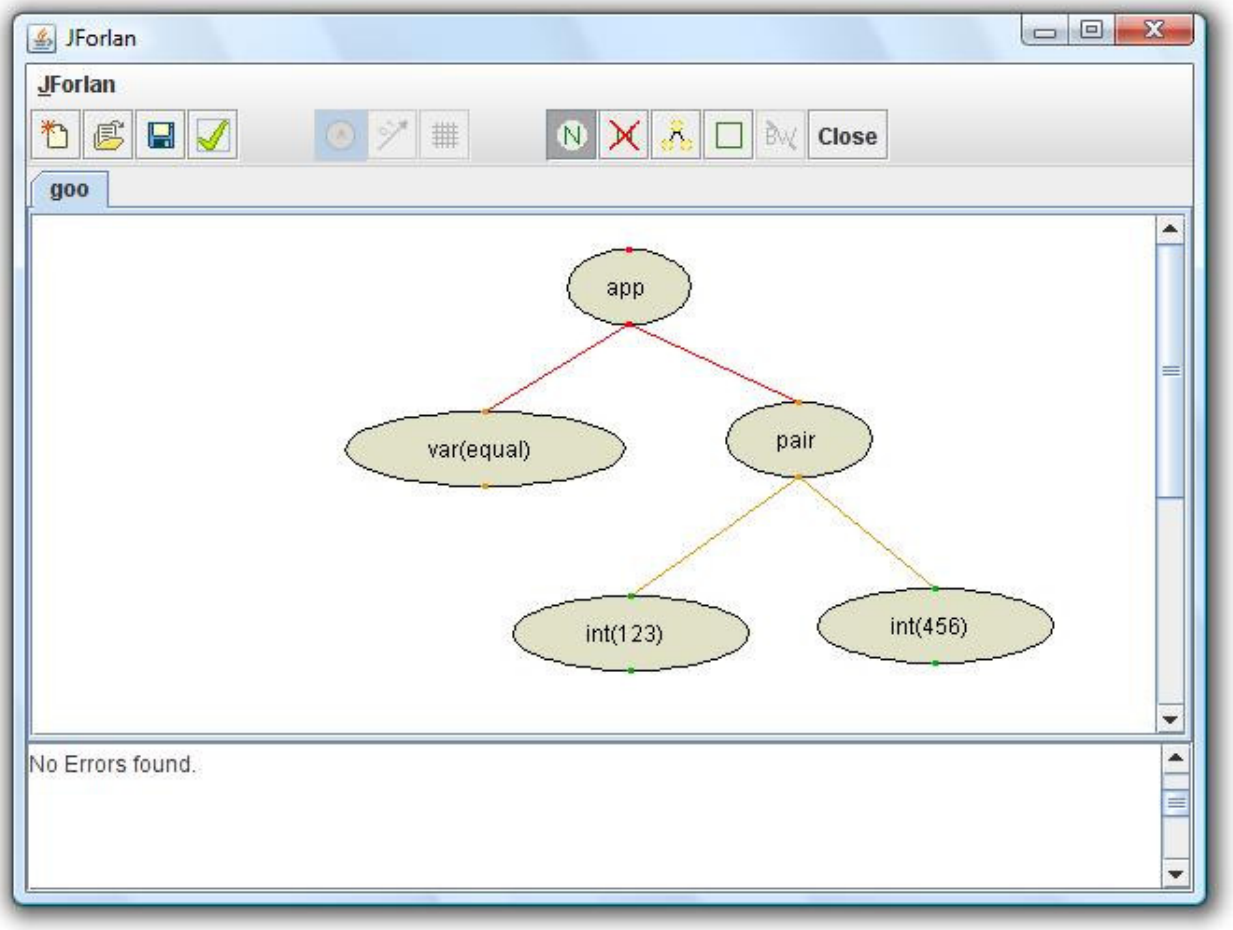

Figure 4.8 A Sample Program Tree drawn using the JForlan Tool

Another advantage of the JForlan tool over the JTR tool is that in JForlan tool the user can create a forest of trees which was not allowed in the JTR tool. The advantage of this feature is that the user need not follow always a top down approach (i.e. creating the parent first and then creating the children) while creating the tree but can create the tree nodes randomly in any order and can then attach them together which gives the user more flexibility as shown in the figure below

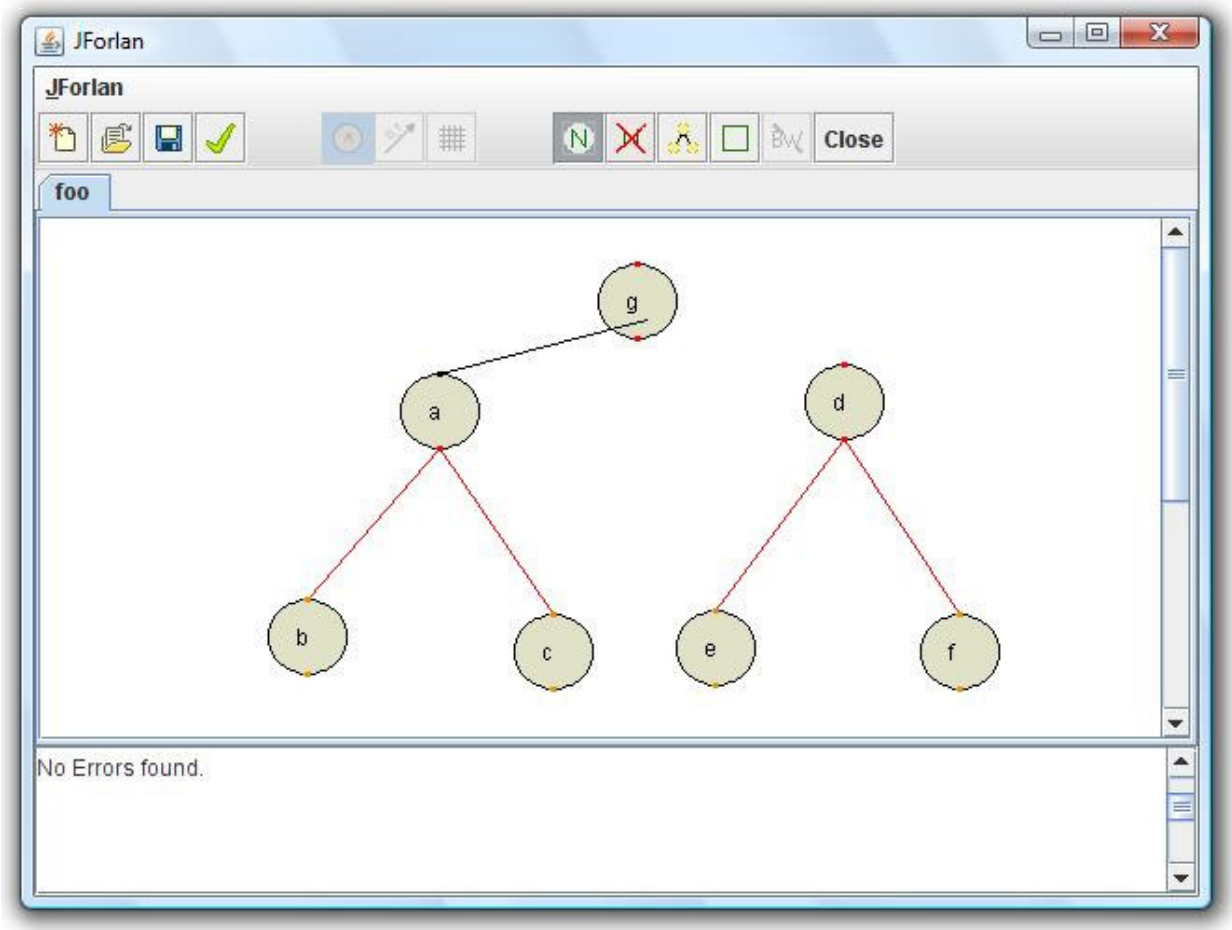

Figure 4.9 Drawing a Forest using the JForlan Tool

Apart from running the JForlan tool as a standalone java application it can also be run from the Forlan command prompt. The look and feel of the JForlan tool changes when it is invoked from the Forlan command prompt. Forlan has functions which can be used to invoke the JForlan tool from the command prompt. For example figure 4.10 below shows how to create a new 'Program Tree' using the JForlan tool from the command prompt

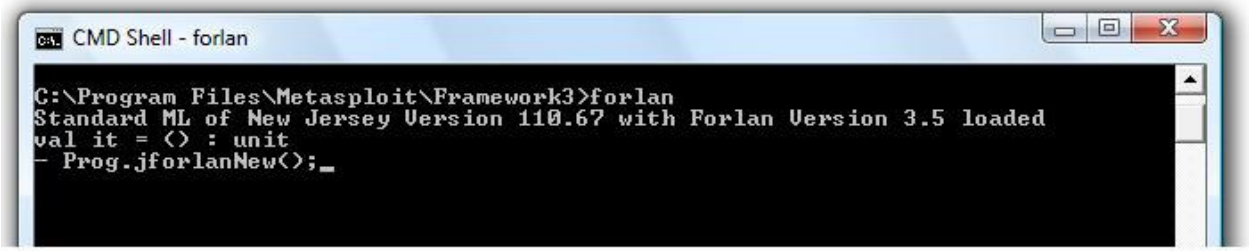

Figure 4.10 Invoking JForlan from the Forlan Command Prompt

Similarly command ‘Prog.jforlanEdit ()’ can be used to edit an already existing program tree using the JForlan tool. Similar functions are available in Forlan for creating and editing parse trees, regular expression trees, Finite Automata, Regular Expression Finite Automata. Once the user gives the command as shown in figure 4.9 the user JForlan interface as shown in figure below

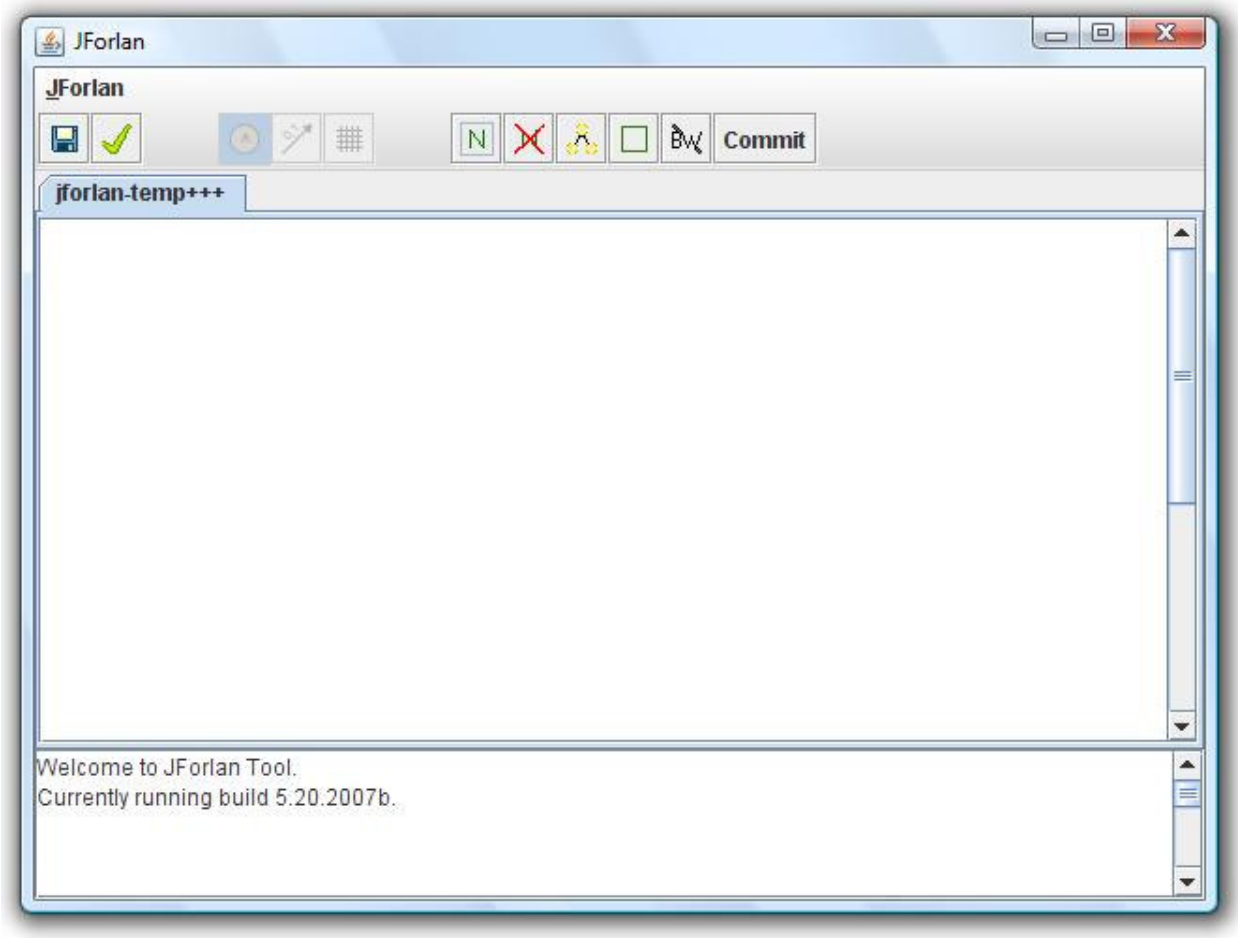

Figure 4.11 JForlan tool when invoked from the Forlan Command Prompt

The JForlan interface when the JForlan tool is invoked from the Forlan command prompt is specific to the command that the user has used to invoke it. For example if the user has used the command ‘Prog.jforlanNew ()’ then the user can create only program tree and has no option of creating any other diagram i.e. the user will not find the ‘create new project’ and ‘open new project window’ to create and open other kinds of tree’s or automata. Since we have used the Forlan command to draw a Program Tree using the JForlan tool during saving the tree it is

obvious that the user is saving a 'Program Tree' and hence the JForlan tool does not ask for any other option during saving as it was the case with the JForlan tool when invoked as a standalone application. The user can either 'commit' or 'abort' the operation which the user has done which returns a zero value to the Forlan command prompt which indicates that there was no error. In case of any error when using the JForlan tool from the Forlan command prompt the JForlan tool returns a non zero value to the command prompt indicating that there was an error.

4.2 Technical Discussions

As the JForlan tool is a Graphical User Interface tool written in Java its implementation involved using various classes and interfaces of Java's AWT (Abstract Window Toolkit) and swing features. The functionalities of various buttons and Menus which are used in the tool are dynamic in nature i.e. the functionality changes depending on the type of drawing that the user is creating in the current tab of the JForlan Tool. Swing components have been used where ever possible as they are lightweight and are platform independent and have various advantages over the AWT components. As the JForlan tool is an integration of the JFA and the JTR tools there were lot of changes made in order to make both the 'Automata' mode and 'Tree' modes look symmetrical so that it will be easy for the students while working on the tool. For example changes were made to the open/save dialog boxes to make them look symmetrical in both 'Automata' and 'Tree' mode. Functionalities like checking the diagram for correctness, saving the diagrams drawn directly into the image format were present only in the JTR tool but not in the JFA tool, these features were added to the Automata mode as well so as to make the tool symmetrical. As the tool is for learning purposes, sufficient functionality (in the form of a color coding scheme) has been added to the tool so as to give feedback to the users of their errors and mistakes so that they can correct and learn on their own. The code has been made robust so that the JForlan tool is platform independent and behaves the same on Windows, Mac and Linux platforms.

CHAPTER 5 – TESTING

Software testing is a process of running with intent of finding errors in software. It is an activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. The purpose of testing is to ensure quality assurance, verification and validation and reliability estimation.

This chapter discusses the various tests performed on the JForlan Tool. Unit testing, Integration Testing and White Box Testing were performed on the tool. Testing is done on a machine with the following configurations.

Processor	Intel® Core™ Duo
Processor Speed	2.00 GHz
Physical Memory	2.00 GB of RAM
Operating System	Windows Vista™ Home Premium

Table 5.1: System Configuration 1

Processor	Intel® Core™ 2 Duo
Processor Speed	2.40 GHz
Physical Memory	2.00 GB of RAM
Operating System	Mac OS X v 10.5 Leopard

Table 5.2: System Configuration 2

5.1 Unit Testing

The foremost goal of unit testing is to take the smallest piece of testable software in the application, isolate it from the remainder of the code, determine whether it behaves exactly as you expect. Each unit is tested separately before integrating them into modules to test the interfaces between modules. It is proved fact that large percentages of defects are identified when Unit Testing is performed. Unit testing is performed in parallel with the coding Phase. Unit testing tests units or modules not the whole software.

As the JForlan tool was being developed for the students to understand the concepts of formal languages in a better way, testing the tool and making sure that everything was working as expected was a major role in the project. Every time I made any change to the tool I tried to test it with each and every kind of input and checking the corresponding output until it is working correctly. The functionality of each and every feature of the tool was also tested as separate units.

In the 'Automaton' Mode the JForlan tool allowed the user to draw a new kind of automaton called 'Regular Expression Finite Automaton' in which the transition labels of the Automaton can take regular expressions which satisfy Forlan Syntax as their value. It has been made sure to test the various combinations in which the user can enter the regular expression so that the JForlan tool correctly displays to the user as to whether the entered regular expression has a valid Forlan Syntax or not and prompts the user accordingly.

Now that there are two modes in the 'Automaton' mode it has been made sure that the tool prompts an error message when the user is trying to open a 'Regular Expression Finite Automaton' as a 'Finite Automaton' as it is an illegal operation. Another feature which was added to the 'Automaton' mode was the flexibility provided for the transitions i.e. the user can drag them and convert the transitions into arcs. This feature was also tested by dragging the transitions and nodes in all possible directions to test whether the code was robust. It was also made sure to prompt the user if the user tries to save a file with an already existing file name to confirm the operation of the user.

In the ‘Tree’ mode the JForlan tool allows the user to create a new type of Forlan Trees called the Program Trees in which the nodes of the tree can take Forlan key words like var, int calc, etc. It has been made sure to test the various combinations in which the user can enter the node names so that the JForlan tool correctly displays to the user as to whether the created Program Tree has a valid Forlan Syntax or not and prompts the user accordingly.

Similar to the ‘Automaton’ mode care has been taken to prompt the user with an error message when the user is trying to open a ‘Program Tree’ as a ‘Parse Tree’ or a ‘Regular Expression Tree’ as it is an illegal operation. Checks were made whether the Program trees drawn using the JForlan tool when saved in Forlan Syntax get translated correctly into a valid Forlan Program in Forlan Syntax using numerous input’s and vice versa i.e. when the user tries to open a Forlan Syntax file as a program tree in JForlan tool whether the JForlan tool is displaying it correctly or not. As the JForlan tool allows the user to add nodes to the tree in any order the user desires while building the tree, tests were made by building the tree in a top down, bottom up and in a haphazard manner to test whether the tree was forming correctly. As the tool is being developed for the students, care has been taken at every step to prompt the user as to the errors they are doing while drawing the diagrams, using the color coding scheme i.e. the node with error has a different color from other nodes.

5.2 Integration Testing

In integration testing a system consisting of different modules is tested for problems arising from component interaction. Integration testing should be developed from the system specification. Firstly, a minimum configuration must be integrated and tested.

The integration testing on the JForlan tool has been done in a bottom up fashion i.e. in the project after performing the unit testing of each and every functionality of the tool the modules of the project are integrated and then tested for problems arising from component interaction.

Since the JForlan tool is an integration of the JFA and the JTR tools it was made sure that after integrating both the tools all the features were preserved which were provided by the JTR and the JFA tool. Checks were are also made using various inputs to check the 'View Tree' option of the tool using which the user can view the 'Regular Expression' entered as a transition label in the Finite Automaton in the form of a tree structure. The Graphical User Interface of the JFA and the JTR tools are slightly different. It is taken care that the JForlan tool is symmetrical both in the 'Automata' mode and the 'Tree' mode so that the JForlan tool appears as a one single tool.

5.3 White Box Testing

In white box testing (Structural testing) tests are conducted to ensure that the internal operations are performed according to specification and all internal components have been adequately exercised. In white box testing, logical paths through the software are tested by providing test routines that exercise specific sets of conditions and loops.

Using white box testing software developers can derive test cases that guarantee that all independent paths within a module have been exercised at least once; exercise all logical decisions on their true and false side; Exercise all loops at their boundaries and within their operational bound; Exercise internal data structure to ensure their validity. White box testing requires programming skills to identify all paths through the software. The test case inputs are chosen to exercise paths through the code and determine the appropriate output.

At every stage of the project development white box testing was done to test the logics of the program by supplying not only valid inputs but also invalid inputs and generating respective error messages. All the loops and conditional statements are tested to the boundary conditions and validated properly.

CHAPTER 6 – PROJECT METRICS AND EXPERIENCE

This chapter presents the project metrics showing the number of hours spent completing each phase of the project. It also summarizes the experiences gained during the entire life-cycle of the project.

6.1 Project Metrics

Project Metrics indicate the progress of the ongoing project. The project metrics discussed in this document are source lines of code and the amount of time spent during the entire project span. The following table shows the project metric source lines of code of the project.

Changes made to the JFA tool	700 lines of code approx
Changes made to the JTR tool	1200 lines of code approx
Integration code	800 lines of code approx

Table 6.1: Project Lines of Code

The following table shows the project phases and their durations respectively

Analyzing and Understanding already existing code	3 Weeks
Requirement Gathering and Design	2 Weeks
Implementation	5 Weeks
Testing	2 Weeks
Documentation	2 Weeks

Table 6.2: Project Phases and Duration

6.2 Overall Experience

Dr. Stoughton had been working on the Forlan Project which consists of a toolset (called Forlan) for experimenting with Forlan Languages. The JFA and the JTR tools were developed as a part of the Forlan toolset and are Java programs for creating and editing Automata and Trees. The JForlan project has been started with an intention to unify these two tools into one single, more advanced and effective Graphical User Interface tool. Dr. Stoughton's continuous input and support aided a lot in making the tool more user-friendly and easy to use. The intention of building JForlan tool was to design something new and innovative and to include some cool features which would help the students to understand the concepts of Automata and Tree in a better and an easy way. The biggest challenge involved in the project was to understand the already existing code of the JFA and the JTR Tools. It is always easy to start a project from scratch rather than to understand the already existing code written by others and to extend it because one does not know the coding standards and principles followed by the person who has written the code. In the present project I had to analyze the code written by not one but two different individuals and had to correlate the components used in both the codes so as to unify the code into one single tool which has all the features offered by the two tools. The design and implementation part involved thinking about new features which could be incorporated in the tool which would improve the robustness, functionality and ease of use of the tool. Understanding the structure of the application was the biggest problem at the start of the project. Finally, the scope of the project was defined which greatly helped in understanding what all features have to be included in the project. As the project involved translating code into design in the earlier stages and then designing and adding new features to the tool, it helped me a lot to get a real time experience of both Reverse Engineering as well as the Forward Engineering concepts.

After the initial design phase, learning Java's support for developing interactive Graphical User Applications was the second phase which gave me a great learning experience. Even though I had the basic knowledge of Java I never had the opportunity to work on a real time Java GUI Application. I spent a lot of time learning and understanding the various classes and interfaces of Java's AWT package and Swing with the help of sample applications and

online tutorials. This learning brought me in a very comfortable position to think about what AWT and swing features I can use to make the application more robust.

Another problem which I faced in my project was the implementation of the flexible transitions in the Automaton mode for drawing the Automata because it involved a lot of Math which included lot of formulae for calculating the angle by which the arrow mark of the transition has to be rotated and translated as and when the transition or the states are moved around by the user. Another problem was to make the code robust so that the JForlan tool would behave the same on any platform. The JFA code would work properly on Windows platform but was causing problems when trying to run the tool on a Mac System with Leopard OS. I had to see the JFA code and track down the problem by dividing the JFA code into chunks of smaller pieces and test the code separately so as to find out where the problem was and had to rectify it to make the JForlan tool platform independent.

I have also learnt a lot of new things while developing this dynamic Java GUI Application using Eclipse IDE and Java Swing Concepts. This project gave me a lot of new concepts in programming and Application development which would help me throughout my career.

CHAPTER 7 – RESULTS AND CONCLUSION

The JForlan Tool Application was designed to provide users with a user friendly interface using which the user can create and edit various kinds of Forlan Automata and Trees, save them directly into Forlan syntax or as an image file or as a JForlan file. The tool can be used either as a teaching tool to teach the concepts of formal languages or it can be used by students to learn the concepts of Formal Languages in a graphical way. The application is easy to use and interactive which makes learning the concepts of Automata and Trees a fun filled experience. The performance of this application is evaluated by rigorously testing it against various test scenarios. Efficiency and correctness of the application is evaluated with the help of various test cases. The Limitations and the ways in which the present tool can be enhanced with additional functionalities are discussed below.

7.1 Limitations

A concern with the application is the default layout of the Finite Automata when the user tries to open a Forlan Syntax file with the JForlan tool. The present default layout does not incorporate any arcs for the transitions in the Automaton mode which can make the diagram cumbersome when the user opens a Forlan Syntax Automaton in JForlan. After opening the Automaton the user has to then modify the spatial positions of the transitions and states in order to make the diagram look neat. Another limitation of the tool is that the JForlan tool allows the user to just draw the required Automaton but does not give any option to the user to enter a string and check whether that string is accepted by the grammar of that Automaton which would be a useful functionality.

7.2 Scope for Future Work

The following things can be done in the future.

- A more advanced default layout algorithm can be implemented by integrating the JForlan tool with tools like Graphviz which allow the user to draw graphs specified in DOT language scripts.
- The feature of allowing the user to check the validity of a string entered as to whether it belongs to the grammar of the automaton or not can be added to the tool
- Functionality can be added such as when a user opens a transition label of an Automaton in the form of a tree structure, the changes made in the tree structure get reflected even in the automaton.
- The application can be modified so that it can be used by other students/researchers to make the JForlan tool more robust.

REFERENCES

- The Forlan Project
  http://people.cis.ksu.edu/~stough/forlan/index.html

- Java AWT and Swing Tutorial
  http://java.sun.com/docs/books/tutorial/uiswing/index.html

- Java 2D Graphics Tutorial
  http://www.java2s.com/Tutorial/Java/0300__SWT-2D-Graphics/Catalog0300__SWT-2D-Graphics.htm

- Wikipedia for various diagrams and testing methods
  http://www.wikipedia.org/

- K-State Research Exchange for samples in report writing
  http://krex.k-state.edu/dspace/