Rhodes University
Computer Science Department

Honours year project

Project Proposal

Author: Etienne Stalmans
Supervisor: Dr. Karen Bradshaw
19 February 2010

1 Principle Investigator

Etienne Raymond Stalmans
29 African Street
Grahamstown, 6140
0798748161
g07s0924@campus.ru.ac.za
Supervised by: Dr. K. Bradshaw

2 Project Title

The project is proposed under the title:
VISUALIZATION OF ABSTRACT SYNTAX TREES AND COMPILER GENERATION WITH COCO/R

3 Statement of the Problem

Modern programming languages are high-level in nature [5], while compiler generation remains a low level operation [6, p.4-p.9]. Coco/R, a popular compiler generator tool, allows for the generation of compilers from any grammar. This is achieved through high-level abstraction, allowing for simplification of the compilation process. Currently no integrated development environment (IDE) exists for Coco/R. This poses a problem for young programmers who are use to modern development environments with syntax highlighting and code visualization. Abstract Syntax Trees (ASTs) can be useful in the analysis and comparison of programs [6, p.20-p.23]. The purpose of this project is to produce an IDE for Coco/R.Furthermore the project proposes to help programmers, new to the field of compilers, understand the compiler generation process and visualize the output of syntax analysis. This will be done in the form of ASTs. The user should be able to use a tree walker to step through the generated ASTs and gain an understanding of the underlying program. This implementation will aim to improve on work done in compiler visualization by implementations such as VCOCO and ANTLR. VCOCO was developed for use with Coco/R and provides step-by-step information about the compiler generation process [4]. VCOCO however does not provide visualization of ASTs.Thus it is necessary to produce a new tool, capable of visualizing the compilation process.

4 Objective of Research

Based on the above problem statement, the objectives of this research are:

Primary objectives

- Produce an Integrated Development Environment for Coco/R
- Visualize abstract syntax trees produced during syntax and constraint analysis
- Increase understanding of the compilation process for new users

Secondary objectives or future extensions

- Provide step by step analysis of the compilation process
- Allow for comparison of abstract syntax trees
- Include JAVA code generation

The development of an IDE for Coco/R will allow for easier development of grammars for compiler generation. Visualization will assit in the understanding of compilers and will aid in the optimization of code generation. The extension of Coco/R should not affect performance or ease of use negatively and should not change the core functionality in any way.

5 History and Background

High-level languages provide an advantage over low-level languages by reducing complexity and being easier to learn for new programmers [6, p.4-p.9]. Compiler generation has traditionally been a task performed at a low-level, with the use of compiler generator tools such as Coco/R, it is possible to abstract to a higher-level [6, p.4-p.9]. Coco/R allows for the formal description of a language through a grammar. The supplied grammar provides specification of the syntax as well as the static and dynamic semantics of the language [6, p.20]. Coco/R is a table-driven parser generator which was developed by Hanspeter Mossenbock at ETH Zurich in 1989. Coco/R is open source and has been ported to many different languages such as C#, Java and Delphi. Grammars used in Coco/R are in the EBNF form and are processed by a recursive descent parser, output is in the form of a scanner and parser, which are able to process the intended grammar. The syntax of the language can be represented in the form of ASTs. ASTs represents the components of a language which can convey the ultimate meaning of a program. Due to the nature of ASTs it is possible to use tree differencing to determine the difference between two programs. Tree differencing can furthermore be used for source control and the extraction of source code change [1].

Related work has been done by Terence Parr who produced ANTLR, ANother Tool for Language Recognition [3]. ANTLR is similar Coco/R as it also provides a framework for constructing compilers from a user specified grammar [3]. ANTLR provides a tool for generating visual representations of programs in the form of ASTs. Furthermore ANTLR allows for the walking of produced ASTs which assists in increasing the understanding of a program. Other tools for visualizing compilers are VCOCO and VYACC [4]. Both of which allow for step by step visualization of the compiler generator process. The limitation of existing visualization tools for Coco/R is that they do not provide visualization of ASTs and lack a mechanism for walking the ASTs.

6 Approach

In order to develop a front end for Coco/R is its imperative to gain an understanding of how the different functions of Coco/R function. To achieve this the documentation for Coco/R will be studied and sample programs will be generated using different grammars. Furthermore, mechanisms for exchanging information between the existing Coco/R code and the visual front-end will need to be investigated.

The second phase of the project will require the investigation of abstract syntax tree generation and the creation of efficient algorithms to achieve this. The visualization of ASTs will require the creation an efficient mechanism to display them in an easy to understand and user friendly manner. This interface should provides as much information as possible without becoming overwhelming or difficult to read. This phase will also require the production of an abstract syntax tree walker able of parsing the ASTs and providing information about each node to the user. The current Coco/R parser generator will be extended to produce a 2-Dimensional AST. The resultant AST will then be parsed by the visualization function.

The third phase will necessitate the creation of the front-end for Coco/R which the user will interact with. This phase will involve design decisions as to provide the most user friendly manner of manipulating grammars and displaying the ASTs. The front-end will be created in C# which will interface with the C# version of Coco/R. During this phase possible extensions will be investigated for their validity and possibility of implementation.

The final phase is the testing phase. The project aims to have a working prototype ready for evaluation by third year computer science students. The test group are all currently taking Prof. P. Terry's compilers course at Rhodes University or have taken the course in the past. The testing will be divided into three categories; performance evaluation of the IDE, ease of use and user awareness. The performance evaluation will require students to evaluate the performance of the Coco/R IDE and the generation of ASTs. Ease of use will be evaluated by students to determine if any changes need to be made to the front-end. The students will also be asked to evaluate how well the visualization of ASTs helps with their understanding of compiler generation and affects their awareness of the steps involved in arriving at a final solution.

7 Requirements

The hardware requirements for this project are as follows:

- Access to a x86 based personal computer

The software requirements are as follows:

- Windows XP Professional SP2
- Microsoft Visual Studio 2008
- .Net 3.5 SP1
- Java compilers
- Coco/R C# sources and Coco/R Java sources

8 Project Progression Time-line

Deadline	Activity
22 February 2010	Formal written Project Proposal
23 February 2010	Presentation of Project Proposal
25 February 2010	Launch research website
10 March 2010	Aquire understanding of Coco/R
15 March 2010	Complete prototype design of IDE
25 March 2010	Finilise message passing between IDE and Coco-R
01 April 2010	Produce ASTs algorithm
10 April 2010	Create ASTs display mechanism
21 May 2010	Literature review and plan of action complete
15 July 2010	Initial prototype of Coco/R IDE
20 July 2010	Optimisation of prototype
27 July 2010	Oral presentation of completed work
01 August 2010	Initial testing by third years
13 September 2010	Short paper hand-in
20 September 2010	First draft of paper
01 October 201025 October 2010	Final oral presentation
01 November 2010	Final project hand in

References

[1] Fluri, B., Wursch, M., Pinzger, M., and Gall, H. C. Change distilling: Tree differencing for fine-grained source code change extraction. *IEEE Transactions on software engineering 33*, 11 (November 2007), 725–743.

[2] Linsen, L., Karis, B. J., Gregory, E., and Hamann, M. B. Abstract tree growth visualization.

[3] Parr, T. *ANTLR Reference Manual*, 2.7.4 ed. University of San Francisco, May 2004.

[4] Resler, D. Vcoco: A visualisation tool for teaching compilers.

[5] Stansifer, R. The study of programming languages, 1995.

[6] Terry, P. *Compiling with C# and JAVA*. Pearson Education Limited, 2005.

[7] Von, S., Garcia, M., and harburg (germany, T. U. H. Eclipse corner article how to process ocl abstract syntax trees, 2007.