Developing an Interpreter for the Expert Systems Language, Flex

05017742 - Thomas Cowell
05017742@glam.ac.uk

August 31, 2010

Abstract

The aim of this project is to create a simple interpreter for the language Flex. Flex is an expert systems language that focuses on using plain English to make it accessible for non-technical users.

Contents

Chapter 1

Introduction

1.1 Why This Project?

Before this paper begins, it is worth noting the motivation behind this project. During particular modules on the masters course, students were using a particular piece of software that allowed them to run Flex programs. Whilst it achieved results, it was felt that the debugging features were sub-par and were often inaccurate, whilst giving no helpful messages for a beginner to use to correct their mistakes.

Chapter 2

Research

2.1 Research

Before any implementation can begin, research needs to be conducted in order to gain better knowledge about expert systems, the Flex language and various toolkits for developing interpreters.

2.2 Expert Systems

In the modern world where information is highly valuable and where time is of the esence, many experts have turned to computing to help provide answers for them that may otherwise take a while to reach through their own expertese. Expert Systems provide an easier way for professionals to reach conclusions through a series of questions, usually linking to a knowledge base. For example, a doctor may use an expert system to reach a diagnosis that may not be readily obvious. The system would ask a question, where the doctor would answer, and then the expert system would continue asking more questions, based on the previous answers, until a conclusion is reached.

Expert system development entials the conversion of information about a bounded problem domain into knowledge, which is then represented in a format suitable for computer manipulation. The created depository of knowledge, known as the knowledge base, can then by used by various deductive reasoning techniques to derive solutions [Nikolopoulos, 1997].

From this, it is obvious that the usefulness of the expert system is only as good as the underlying data - the knowledge base. Using the doctors example from the previous paragraph, the system would be of no use if the knowledge base was minimal, whilst in the reverse situation, it would be extremely useful to have a large knowledge base.

2.2.1 Flex

The 'Flex Tutorial' describes Flex as "Flex is a software system specifically designed to aid the development and delivery of Expert Systems."

To appreciate both the power and limitations of Expert System approaches to reaching expert conclusions, it is necessary to construct and experiment with expert systems. In this module, the Flex Expert System Shell will be used. Flex describes knowledge in terms of *production rules* (that is, *if-then* statements), which has proved the most popular approach to encapsulation expert knowledge. Such rules, despite appearing simple, enable relatively complex connections to be made between individual pieces of 'knowledge', thereby solving apparently difficult problems.[Roach, 2009].

2.3 Development

2.3.1 Environment

One of the main aims of this project was to create an open-source alternative to the propietary package "WinProlog", which provides the Flex toolkit. This aim was to provide the project on an open-source platform, such as Linux, which would allow lecturers to tailor the package to their needs, allowing them to fix any bugs or add more advanced features, without the fear of violating any licences. Therefore, Linux was the platform of choice to develop on, where the distribution of choice was Ubuntu, as it has a great wealth of development applications in its repositories, such as `flex`, `bison` and `gcc`. Ubuntu provides a package, `build-essential`, to help developers write and compile applications easily, which contains many useful tools that should please a large majority of software developers.

Installing these packages in Ubuntu simply required the following command:

```
$ sudo apt-get install flex bison build-essential
```

Flex and Bison shall be discussed in the next chapter. `build-essential` is a Debian meta-package, that is, it is simply a list of packages that are essential to building applications in a Debian based distribution. The package satifisies the needs for most developers wishing to develop applications on Linux, and includes packages such as `gcc` and `g++` - a C and C++ compiler, respectively.

Ubuntu provides a wealth of choice when it comes to development tools, however only the simple tools are required, such as a text editor such as `gedit`, or console based text editor, `nano`, whilst the terminal is perfectly acceptable for running commands to compile the tokens, grammar and C program together. `make` comes part of the `build-essential` package, which allows a developer to write a series of commands into a makefile, and then simply run the command:

```
$ make
```

in the directory where the makefile is located.

2.3.2 Source Control

Source control allows the source code to be backed up on a remote repository and ensures that several users are working on the same code, rather than each working on several different versions. Whilst this won't be an issue for this project, it will be a useful time to research and learn how to use source control, and will also provide a useful back up tool. The advantage of using source control, over traditional back-ups, or using removable memory, is that it's unlikely it will get lost, and the same, up to date work can be accessed from other machines if the situation requires it. For example, a developer has a desktop PC and a laptop, and requires to work on the code using the laptop for several days. Source control allows the developer to easily pull the latest version of the code from the repository, code away, and then push the changes back to the repository, so that when the developer comes to use their PC, they can pull the latest code and be up to date, without having to work out which files are more recent on a USB stick.

There are several different source control services available for free, that do roughly the same job.

Chapter 3

Technical Research

This chapter will cover the research into the technical research involved into the implementation process.

3.1 Lexical Analysis

Flex and Bison are tools for building programs that handle structured input. They were originally tools for building compilers, but they have proven to be useful in many other areas.[Levine, 2009].

Lexical analysis is the process of taking an input, and splitting it up into meaningful parts, or *tokens*. For instance, `answer = 5 + 4;` would be split into six tokens: *answer, equals, five, plus, four* and *semi-colon*. From this, the parser can work out that `5 + 4` is an expression, in which the answer is assigned (`=`) to `answer`. By using this approach, it makes it easier to analyse and input sources that will be interpreted.

3.2 Parsing

What is parsing? derp derp

3.2.1 Top-Down Parsing

Parses input string of tokens by tracing out the steps of the leftmost derivation. Called top down because the implied traversal of the parse tree is a preorder traversal and thus occurs from the root to the leaves (think BST's).

Two Types: Backtracing parsers & predictive parsers. Predictive attempts to predict the next construction in the input string using one or more lookahead tokens, while a background parser will try different possibilities for a parse of the input, backing up an arbitrary amount in the input if one possiblity fails. Backtracing is more powerful, but slower than predictive - useful point to make when running an interpreter.[Louden, 1997]

3.2.2 Bottom-Up Parsing

Bottom up parsing is parsing that goes from bottom to up.

Chapter 4

Conclusion

I conclude that the project was poo.

Bibliography

GNU. Flex. http://www.gnu.org/software/flex/.

John R. Levine. *Flex & Bison*. O'Reilly Media, 2009.

Kenneth C. Louden. *Compiler Construction - Principles and Practice*. PWS Publishing Company, 1997.

Chris Nikolopoulos. *Expert Systems*. CRC Press; 1 edition, 1997.

Paul Roach. Techniques handout 3. Module CS4S22, Lecture Note, 2009.