Implementing a Prolog Interpreter in Flex and Bison

Aven Bross

5/8/2014

1 Introduction

In this project I created a simple prolog interpreter. The interpreter handles atoms, variables and complex terms and performs unification to solve queries. It does not provide more in depth prolog features such as arithmetic unification, lists or grammars.

2 Difficulties

The most difficult part of the project was by far the underlying unification algorithm. I unfortunately wrote a large portion of the final version of the code without researching good unification algorithms. Because of this, the algorithm is a bit odd and follows a very slow approach to unification. But it does unify!

3 Outline

The compiler works by reading clauses into the knowledgebase. Most of the code was written based off of how I knew prolog worked and the BNF grammar. Due to this, I made my main data structures Clauses and Terms, from their titles in the BNF. It constructs each clause by storing its name, type and then a vector of subclauses as well. Underneath Clauses are terms, which function the same way, with name, type and vector of subterms if necessary.

As clauses are read into the knowledgebase, all of their variables are renamed to unique names. In this way, we ensure that no user entered clauses can cause substitution errors later on.

Queries are made by attempting to unify the users list of clauses with each clause in the knowledgebase. Each set of substitutions that works is recorded, and then at the end the ultimate values for each of the users's variables are printed or simply "yes." if the user submitted no variables.