Prolog in Ocaml

He Xiao, Shijiao Yuwen

December 7, 2014

Abstract

This report presents the design and implementation of the application 'Prolog-Ocaml'. Basically, this project used Prolog as a template, and designed a variant of Prolog in Ocaml; it covers the most basic phases of implementing a new programming paradigm: lexing, parsing, interpreting. Given the source code in predefined grammar, the delivered product can parse it to an internal AST and execute it. After testing it with some well-known Prolog programs and comparing the results with the official Prolog implementation, some strengths and weaknesses of our implementation have been found.

0.1 Grammar

0.2 Lexing and Parsing

0.3 Interpreting

0.4 Backtracking

Rigorously speaking, the backtracking algorithm implemented in this project is not the same as the classical ones. It will backtrack silently and try to gather all the results, but it will not print the result immediately when find some; instead, it will output the results to the terminal after collecting all the results it obtains. In oder to simulate the behavior of standard Prolog, when multiple results are available, after presenting the first result, it will wait for the user's instruction and then respond accordingly.

Currently, in the application which uses backtracking algorithm (**play_all.exe**), only Horn clause is supported: i.e. the body of the rules as well as the query must be predicates connected via logical and operators (in Prolog, ',').
Furthermore, in order to avoid entering infinite loops, a list is maintained which records, for each rule, all the queries that have matched the head predicate of that rule in the past. It has both advantages and disadvantages after adding this feature: the good thing is it will not get lost in one branch while some other results can be found easily in other branch; the bad thing is that it may have false alarm so that it will miss some true results. In the evaluation part, we will demonstrate that in certain area, the advantage of our approach outweigh its drawbacks.

Method for collecting all the results

0.5 Testing Results and Evaluation

In order to pass some certain tests, several built-in functions (such as 'write', 'nl' and most arithmetic and comparing functions) in Prolog are implemented, order than that, no built-in functions in Prolog are supported.

0.5.1

0.6 Possible future work

- Currently, not many built-in functions are supported due to the limited implementation time. However, an idea is that, given the path to official Prolog's library, if the built-in functions are also written in

normal syntax of Prolog program, then our implementation should be able to make use of those built-in rules and obtain the results as if all the rules are defined locally in one file.

-

0.7 Bibliography

0.8 Appendix