CS421 Unit Project: Prolog in OCaml

Haoyu Wang Xiaohong Chen

November 26, 2016

1 Overview

This project fulfills our extra-credit workload in CS421 – Programming Languages and Compilers. Among all other projects listed in Elsa's website, we found this one the most interesting and meaningful.

We aim for a Prolog implementation in OCaml. The Prolog that we implemented in this project is a fragment of the real Prolog language, so we refer it to SimProlog. SimProlog is able to reason with facts and rules about (potentially nested) Prolog terms, but do not support data reasoning (such as integers) and data structures (such as lists).

2 Implementation

The SimProlog implementation contains four important aspects.

1. A lexer (`lexer.mll`)
2. A parser (`parser.mly`)
3. Main functionality of SimProlog (`main.ml`) including
   - Abstract data structures
   - Unification and substitution
   - Backtracking
4. An interactive interface (`simp.ml`)

We fully covered all aspects that were proposed in our original proposal (which we enclosed in Appendix for reference). In addition, we provide an interactive interface for users to teach SimProlog engine facts and rules and ask her questions.

3 Tests

3.1 Ordering reasoning

3.2 Piano arithmetic reasoning

Appendix

Original proposal

```
We propose to implement a simple version of Prolog in OCaml.
A program of such a simple version of Prolog language that
we concern in this unit project, as SimProlog we call it ,
consists of the following components.

(1) A number of *declarations* that declare predicates.
(2) A number of *base clauses* that state fact.
(3) One inquiry where free variables are allowed to show up.

A sample of such a SimProlog language looks the follows.

 ----------- SimProlog Program Sample Begins -----------

% This is a simple SimProlog program.
% Everything following '%' is a comment.


mother(X, Y) :- parent(X, Y), femail(X)
parent(john, bill)
parent(jane, bill)
femail(jane)
| ?- mother(M, bill)

  ----------- SimProlog Program Sample Ends -----------

Our unit project will consist of five basic building blocks,
which are divided into either syntax category or semantics
category.  Each of us will be responsible for one category.
The five basic building blocks are (where [xc] stands for
Xiaohong and [wh] stands for Haoyu):
(1)[xc] A grammar for SimProlog language.
(2)[xc] A lexer that consumes streams of characters and recognizes
tokens.
(3)[xc] A parser that consumes streams of tokens and build
ASTs.
(4)[wh] An evaluator that takes ASTs, understands predicate
declarations and facts, and answers the inquiry.  This may
include:  a unification process and a backtracking mechanism.
```

Code

```
open Parser (* The type token is defined in parser.mli *)
exception Eof
   let digit = ['0'-'9'] let lcase = ['a'-'z'] let ucase = ['A'-'Z']
let wchar = digit | lcase | ucase | '_' let lword = lcase wchar* let
uword = ucase wchar* let space = [' ' ''' '
n']
   rule token = parse | space  token lexbuf  (* skip over whitespace
*) | ":-"  CDASH  | "?-"  QDASH  | '('  LPAREN  | ')'  RPAREN  | '.'
DOT  | ','  COMMA  | lword as w  LWORD w  | uword as w  UWORD w  |
eof  raise Eof
```