Simple Prolog Interpreter

Sivani Channamasetty(IMT2019020)
Manohar Suggula(IMT2019025)
Sri Harsha G(IMT2019030)
Satwik Samayamantry(IMT2019077)

May 11, 2022

Problem description

The aim of the project is to create an simple Prolog interpreter using Ocaml language. In this interpreter we will be implementing most of the core Prolog instructions which will be a subset of the whole Prolog instruction set. The main learning goals of this project are

- Explore the depth in both logical and functional programming at a time while comparing and contrasting between them.
- Understanding how a basic prolog interpreter works.
- How lexer and parser are useful in interpreting a program.
- Explore how Ocaml's strong static typing with type inference can be exploited to achieve the given task.

Solution outline

The solution can be briefly divided into sub parts as follows

- Recursive query input handler : Takes the queries from stdin in string format and sends it to lexer.
- Lexical Analyser : It is a synonym for tokenization. It ignores the white spaces in between and outputs the identified tokens. The tool used is Ocamllex.
- Syntax Analyser : It is adherence to the grammatical constructs of Prolog. It outputs the list of tokens in a known data type to the interpreter. The tool used is Ocamlyacc.
- Interpreter : The interpreter reads the database file and stores the rules and facts. We take the queries form the stdin. The interpreter checks the query with each fact/rule in the database and tries to unify the query with the predicates in the database and prints appropriate true./ false. It will search the whole database until there is a match possible.
- If the query has variables, it will create an environment and unify the variables, using the unification algorithm, with the atoms in the predicates of the database.

Unification Algorithm

We need to unify two clauses. Let them be $\Psi_i(x)$ and $\Psi_2(y)$. Start with an empty environment E. The Unification rules are as follows:

1. If Ψ_i and Ψ_2 are different, then false.
2. If Ψ_i and Ψ_2 are same then we need to unify x and y.
3. If x is a variable, y is a constant, then Bind x with y. $E = \{x \rightarrowtail y\}$
4. If y is a variable, x is a constant, then Bind y with x. $E = \{y \rightarrowtail x\}$
5. If both x and y are constants, then if x == y then true(E = {}) else false.
6. If both x and y are variables, then if x == y then E = {} else $E = \{x \rightarrowtail y\}$

Also, if there are multiple arguments in the clauses, then they should have equal number of arguments. The same unification rules will be applicable for every argument.

Deliverables of the project

The expected deliverables for the final submission are

- Source code for the interpreter. Contains interpreter.ml, env.ml and expression.ml and their respective .mli files.
- Report explaining the project.
- README file explaining how to run the code.

Presentation Slides

Google Slides

Github repository of the code

https://github.com/Satwik-Samayamantry/Simple-Prolog-Interpreter

References

1. Smith, Joshua. (2007). Ocamllex and Ocamlyacc. 10.1007/978-1-4302-0244-8_16.
2. Armand E. Prieditis and Jack Mostow. 1987. PROLEARN: towards a prolog interpreter that learns. In Proceedings of the sixth National conference on Artificial intelligence - Volume 2 (AAAI'87). AAAI Press, 494–498.
3. https://www.javatpoint.com/ai-unification-in-first-order-logic
4. https://www.sciencedirect.com/science/article/pii/S0304397596001156
5. https://nms.kcl.ac.uk/maribel.fernandez/papers/slides-TCS-SOUP.pdf
6. https://stackoverflow.com/questions/64638801/the-unification-algorithm-in-prolog
7. Unification algorithm in Prolog