Experiment No: 06

Aim: Tutorial to learn SWI-Prolog installation and use

Lab Objective: Design and implement declarative programs in functional and logic programming languages.

Lab Outcomes: Design and Develop solution based on declarative programming paradigm using functional and logic programming (LO2)

Requirements: swi-pl interactive command line tool installed on local machine.

Theory:

Prolog Stands for logic programminglanguage. It has important role in Artificial Intelligence. Unlike many other programming languages, Prolog is intended primarily as a declarative programming language. In prolog, logic is expressed as relations (called as Facts and Rules). Core heart of prolog lies at the mathematical logic being applied. Formulation or Computation is carried out by running a query over these relations.

The language was developed and implemented in Marseille, France, in 1972 by Alain Colmerauer with Philippe Roussel, based on Robert Kowalski's procedural interpretation of **Horn clauses** at University of Edinburgh.

Prolog was one of the first logic programming languages and remains the most popular such language today, with several free and commercial implementations available. The language has been used for theorem proving, expert systems, term rewriting, type systems, and automated planning, as well as its original intended field of use, natural language processing.
Modern Prolog environments support the creation of graphical user interfaces, as well as administrative and networked applications. Prolog is well-suited for specific tasks that benefit from rule-based logical queries such as searching databases, voice control systems, and filling templates.

Data Types in Prolog:
Prolog uses single data type called the *term*. *Terms* are either *atoms, numbers, variables or compound terms*.

- An *atom* is a general-purpose name with no inherent meaning. Examples of atoms include x, red, 'Taco', and 'some atom'.
- *Numbers* can be floats or integers. ISO standard compatible Prolog systems can check the Prolog flag "bounded". Most of the major Prolog systems support arbitrary length integer numbers.
- *Variables* are denoted by a string consisting of letters, numbers and underscore characters, and beginning with an upper-case letter or underscore. Variables closely resemble variables in logic in that they are placeholders for arbitrary terms.

- *A compound term* is composed of an atom called a "*functor*" (mathematical representation of relation/function) and a number of "*arguments*", which are again *terms*. Compound terms are ordinarily written as a functor followed by a comma-separated list of argument terms, which is contained in parentheses. The number of arguments is called the term's arity. *An atom can be regarded as a compound term with arity zero.* An example of a compound term is person_friends(zelda, [tom, jim]).
- Special cases of compound terms:
  - *A List* is an ordered collection of terms. It is denoted by square brackets with the terms separated by commas, or in the case of the empty list, by []. For example, [1,2,3] or [red,green,blue].
  - *Strings:* A sequence of characters surrounded by quotes is equivalent to either a list of (numeric) character codes, a list of characters (atoms of length 1), or an atom depending on the value of the Prolog flag double_quotes. For example, "to be, or not to be".

Rules and Facts:

Prolog programs describe relations, defined by means of clauses. Pure Prolog is restricted to Horn clauses. There are two types of clauses in a prolog code: facts and rules.

A rule is of the form *Head :- Body.* It is read as "*Head is true if Body is true*". A rule's body consists of calls to predicates, which are called the rule's goals.

Clauses with empty bodies are called facts. An example of a fact is: *cat(tom).* which is equivalent to the rule: *cat(tom) :- true.* The built-in predicate true/0 (true functor name and arity zero) is always true.

Given the above fact, one can ask:

- is tom a cat? *?- cat(tom).* Yes
- what things are cats? *?- cat(X).* X = tom

Clauses with bodies are called rules. An example of a rule is: *animal(X) :- cat(X).* If we add that rule and ask what things are animals?

?- animal(X).
X = tom

Execution : Execution of a Prolog program is initiated by the user's posting of a single goal, called the query. Logically, the Prolog engine tries to find a *resolution refutation of the negated query*. (resolution refutation is formal logical proof technique that is much like proof by contradiction used in geo,etric proofs) The resolution method used by Prolog is called SLD

resolution. If the negated query can be refuted (Proven False logically), it follows that the query, with the appropriate variable bindings in place, is a logical consequence of the program. In that case, all generated variable bindings are reported to the user, and the query is said to have succeeded.

Operationally, Prolog's execution strategy can be thought of as a generalization of function calls in other languages, one difference being that multiple clause heads can match a given call. In that case, the system creates a choice-point, unifies the goal with the clause head of the first alternative, and continues with the goals of that first alternative.

If any goal fails in the course of executing the program, *all variable bindings that were made since the most recent choice-point was created are undone*, and *execution continues with the next alternative of that choice-point*. This execution strategy is called *chronological backtracking or simply backtracking*.

Performance: Students are expected to perform the tutorial from swi-prolog web site under the quick start section on the local swi-prolog machine. Students must write all steps with output in the write up. Write up should be written on both side ruled pages. The link for the Tuotorial is : https://www.swi-prolog.org/pldoc/man?section=quickstart
Student need to follow steps mentioned under 2.1 i.e. starting with 2.1.1 upto 2.1.5 to complete this write up.

Conclusion: Thus we have understood how to create functional solution to programming problems using Haskell.

Reference:

1. swi-prolog getting started online tutorial, https://www.swi-prolog.org/pldoc/man?section=quickstart, 01 Jan 2023.
2. Wikipedia Article: Prolog, https://en.wikipedia.org/wiki/Prolog, 01 Jan 2023.