CHAPTER VII

PROLOG

7.1 Introduction

This is one of the simplified but a powerful programming language whose logic follows the first order predicate principle. PROLOG name is from a French word ‘PROgramma-tion en LOGique’. In the year 1970, A. Colmerauer and P. Roussel own the credit for designing PROLOG at the University of Marseille by, which resulted from studies of R.A. Kowalski. Since from that time PROLOG follows logic programming which is a new style of programming (I. Bratko 2001).

PROLOG was only used by academicians till 70's end. Only after the PROLOG interpreter and compiler which was created at the University of Edinburgh by “D.H.D. Warren and F.C.N. Pereira” thus entering outside world from research institutes. The interest shown to this language has increased progressively. Nevertheless, researchers prefer PROLOG, serious and extensive programs can be developed, in small interval of time there was a need for implementation of other programs with similar functionality. Their applications in large number of fields justify its success. It is mostly applicable in Artificial Intelligence. For symbol manipulation PROLOG is used in several areas (Peter Lucas 1991).

Areas for application of PROLOG are:

- Processing for Natural-language (NLP)
- Construction of Complier
- development of experts systems
- Working in areas of computer algebra
- The development of Database systems and architecture for (parallel) computers.

PROLOG has a good potential in problem solving whose category is requirement of complex symbols. Comparing with other important programs, problems are more and impassable. Equivalent programs in PROLOG are simpler to work and understand. The benefits of its principle are that a formal specification is given to the coder to set
programs. Along with that PROLOG gives refinement in every step for program development as it has modularity nature. Hence PROLOG is a perfect language which can be used for prototype development (M. Balduccini 2000: 63-72).

### 7.2 The Logic Programming and PROLOG

The Logic Programming and PROLOG is now a principal language for programming when the “Japanese Fifth Generation Computer Systems” (FGCS) was introduced as Logic programming became the subject of improvement (J. Lloyd 1987).

Construction of predicate logic machine as a hardware and software framework of computer was done after the success of FGCS project. Their achievements are:

- A hierarchy of languages for logic programming,
- Logic programming was implemented in operating system and a family unit of machines for logic programming.

United States stimulated projects on FGCS which created lot of interest all over the world in logic programming.

It is a programming prototype that is simplified and logic of first-order however the influential gave for the implementation to advanced applications. In early 1970's Kowalski and Colmerauer, this idea was explored. Prolog is the most accepted logic programming language.

This language is predominantly suitable for programming in symbols in analyzing natural language, applications in database and artificial intelligence such as compiler manipulation in Algebra, writing and proving of theorem. Successful examples of Prolog's application in understanding natural language, knowledge representation, expert systems, the automatic generation of plans, expert database systems, database design.

Comparing with traditional programming languages, Prolog programming is prominent. It works on the relationships and facts existing in the domain of the system and explains the question acceptable for the given issue. Coders have the flexibility for task in prescribing the precise steps of calculations needed to solve a given problem as compared to other languages.
For computing a program in Prolog, sequence is calculated by inference mechanisms of Prolog system. Solving a problem in prolog, it first searches original set of facts and relationships of the given data. There is a usage of resolution to subdivide the original definition of the problem into smaller sub-problems. These sub-problems are further sub divided into still smallest sub-problems till computer develops a section of sub-problems whose solutions can be calculated directly from the given information and facts by set of programmer.

The process of Unification in which higher resolution is achieved by using generalization of pattern-matching mechanism in formulae set to make formulae of higher precision. In this process formulae are matched and are constructed in a similar format with instant variables and terms of particular standards (James Lu 2010).

Prolog receives higher preferences compared to procedural languages because of unique segregation between control and logical components. Hence it gives nature of Prolog being declarative and results in two important program developing concerns are:

- **Correctness**
- **Program Efficiency**.

In Prolog system, to assure the efficiency programmer has to improve the linkage between correct and verified codes. In unification the focus over scheme of pattern-matching makes PROLOG capable of easy creation, searching, manipulating complex data and comparing of different data structures.

Combination of random arbitrary data structures, results to a highly leveled coder languages in which the programmer is capable of giving suggestions with respect to entities and objects in the coding language within the domain of problem. The succeeding implementation and program secrecy can be achieved by relative freedom of codes supplied with relationships and facts in the Prolog. The above feature makes Prolog an ideal language for prototype. Overall Prolog is an important foundation for upcoming languages (George F. Lugev 2009).
7.3 Programming in PROLOG and SWI PROLOG

As Prolog is one of the declarative language hence Programmes in Prolog is varies from programmes in traditional languages. For further specification related to activity in Prolog is increase in knowledge and logic parameters within the defined domain of problem solving. Prolog allows us for a declarative style programming and is essentially logic combined with control. The problem solving statement forms the logic part in Prolog whereas the method of solving it, is its Control part. Fig 7.3.1 indicates the relationship between Prolog and logic programming.

The recent edition of SWI-Prolog (5.6.52) is used in the thesis. This software is not chargeable and Amsterdam University from 1990 to 2008 is its developer. Though this software is not chargeable it has productive and helpful features having a wide range of application in commercial and educational fields.

Algorithm = logic + control

Fig 7.3.1: The relationship between PROLOG and logic programming.

There is a twofold reason for using SWI-Prolog for our research. One is that SWI-Prolog provides an environment of a suitable and easy learning and the other is that many required tools and accessories might be absent in the latter which other editions of Prolog with GUI environments have. Keeping our research in consideration straightforward, simple software is sufficient. SWI-Prolog also provides a powerful extension for
programming based on Object-Orientation. For large-scale applications, programming with Object-Orientation acts like a simple and common sense. In different commercial or not Prolog’s editions, among them SWI-Prolog provides chances and opportunity for incorporation of OOP constructs. Hence this provides a well organised scope of application in pseudo-OOP (Ulle Endriss 2014).

For estimation of annual mean and monthly speed of wind Fig 7.3.2. demonstrates the basic Prolog flow chart used in research the given sites

7.4 Prolog in our study

For complicated and diversified problem solving multiple layers of perceptions has been tried by supervised training with highly affective algorithm called as algorithm for the error back-propagation. Algorithm of Back-propagation follows the correcting error and learning rule. This rule is based on (LM) algorithm. There are two passes in the Error back-propagation learning method namely:

- A forward pass and
- A backward pass.

![Fig 7.3.2: Basic prolog Flow Chart](image)
In forward pass, a vector of an input is given at network nodes where effect propagates layer by layer throughout the network. Finally the actual output is produced in the form of set of outputs. Weights of the networks throughout the process do not vary. In backward pass, according to correction rule of error, the weights are adjusted. Hence error signal is obtained when the actual response is deducted from response that is required in the network. Propagation of this error signal is done in backward direction against the synaptic connections. Then adjustment of weights is done according to the network's actual response to get nearer to the response that is required.

Analyzing our work delivers a base for future development Prolog systems. Compared to any other work of which we are aware, this is much more. We evaluate our Prolog profiles measurements and studies of other languages given by others. It helps in efficient implementation of Prolog when comparisons are made.

In all the sites the back prorogation algorithm model will be implemented using PROLOG. PROLOG follows predicate logic of first order as its principle hence defined as logic programming. PROLOG is meant for solving problems characterized with the help of complex symbolic computations. These programs are short and are easier to understand and grasp than conventional imperative programs. PROLOG is supportive to refinement in programs development step by step due to characteristics of modularity. Thus, for direct and suitable execution of program on computer, the coder just has to provide formal specification in Prolog language. The result for the program is generated easily.