Prolog and Inductive Reasoning: A Logic Programming Language

A project submitted in partial fulfillment of the requirement for the degree of

Master of Science in Computer Science

By

Kumeel Alsmail

December 2012
The graduate project of Kumeel Alsmail is approved:

_________________________            ___________________________
Richard Covington, Ph.D.            Date:

_________________________            ___________________________
William Ford, Ph.D.            Date:

_________________________            ___________________________
Peter Gabrovsky, Ph.D., Chair            Date:
ACKNOWLEDGEMENT

I would like to express my gratitude to my dearest parents who gave me the possibility to complete this research paper. I am deeply indebted to my research supervisor Dr. Peter Gabrovsky whose help, stimulating suggestions and encouragement makes me able to do this project. This paper cannot come into being without his guidance and suggestions.
## Table of Contents

Signature page ............................................................................................................. ii
Acknowledgment ........................................................................................................ iii
Abstract ....................................................................................................................... vii

1. Section
   1.1 Introduction ........................................................................................................... 1
   1.2 Background/Rationale .......................................................................................... 3
   1.3 Research Problem ................................................................................................ 5
   1.4 Research Methodology ....................................................................................... 5
   1.5 Resources ............................................................................................................. 6
   1.6 Scope and Limitations ........................................................................................... 6
   1.7 The Problem Statement ...................................................................................... 7
       1.7.1 A clear formulation of the research problem ............................................... 7
       1.7.2 Identification of Research Topic ..................................................................... 8
       1.7.3 Refining the research topic into a research problem .................................... 9
       1.7.4 Influence of Problem Statement on Organizations ...................................... 10
   1.8 Professional Significance of the Study ............................................................... 10
       1.8.1 A Clarification of the Professional Significance ......................................... 10
       1.8.2 Motivation behind Research Topic ............................................................... 12
       1.8.3 Professional Value of the Study ................................................................... 12
       1.8.4 Contribution of study towards Professional Knowledge .......................... 13
           1.8.4.1 Detailed Professional Significance ....................................................... 13

2. Section
   2.1 Introduction ........................................................................................................... 14
   2.2 Diagnosis System and Medical Field ................................................................... 14
       2.2.1 Diabetes Expert System .............................................................................. 14
       2.2.2 A Novel System to analyze Multiple Skin Syndromes ............................... 16
       2.2.3 JESS based System for Endocrine Treatment and Diagnosis .................... 17
3. Section

3.1 Introduction ........................................................................................................................................19
3.2 Hierarchy of Development Process .................................................................................................20
3.3 Research Approach ..........................................................................................................................21
3.4 Quantitative Research ......................................................................................................................21
  3.4.1 Deductive Reasoning .....................................................................................................................22
3.5 Qualitative Research ..........................................................................................................................23
  3.5.1 Inductive Reasoning .....................................................................................................................24
3.6 Design of the Study .............................................................................................................................25

4. Section

4.1 Introduction ........................................................................................................................................27
4.2 How Inductive Reasoning Works? .......................................................................................................27
  4.2.1 Stage I – Gathering and Refinement of the Preliminary Data .......................................................27
  4.2.2 Stage II- Define an initial set of Hypothesis ..................................................................................28
  4.2.3 Stage III- Additional information derived by Preliminary hypothesis ........................................28
  4.2.4 Stage IV- Using Inductive Reasoning ........................................................................................28
  4.2.5 Stage V- Define and Test the Final Diagnosis .............................................................................29
4.3 Inductive Reasoning and Prolog .........................................................................................................29
4.4 Proof Theory .......................................................................................................................................30
4.5 Application ..........................................................................................................................................30
4.6 Conclusion ...........................................................................................................................................37

5. Section

5.1 Introduction ........................................................................................................................................38
5.2 What is Prolog? ...................................................................................................................................38
5.3 Why using Prolog? .............................................................................................................................40
5.4 Prolog vs. other languages ................................................................................................................42
  5.4.1 Prolog vs. C++ ..............................................................................................................................42
  5.4.2 Prolog vs. Java ...............................................................................................................................43
  5.4.3 Prolog vs. Lisp ...............................................................................................................................43
Abstract

PROLOG AND INDUCTIVE REASONING
A LOGIC PROGRAMMING LANGUAGE

By

Kumeel Alsmail

Master of Science in Computer Science

This following research presents the design and development of a diagnostic medical system. The proposed system provides an interactive and adaptive environment through which symptoms are linked with illness. Each new patient is asked a yes/no question concerning his or her symptoms, and responses are recorded. The diagnosis works by matching symptoms of known illness with recorded responses in decreasing order of complexity. Inductive reasoning (in the form of generalization) is used to update illness symptoms. The system is implemented in Prolog SWI- Version 6.0.0.

Key Words: Inductive Reasoning, Medical Diagnoses, Prolog, Logic Programming.
Section 1

1.1 Introduction

Research in the area of computer science has given birth to a comparatively novel and swiftly rising technology referred to as *expert systems*. An expert system is a computer program which encapsulates the intelligence of a human expert, and utilizes this intelligence to solve issues in a manner analogous to the expert. Such a system is able to help the expert by investigating, analyzing or operating in the place of the expert under circumstances in which proficiency is deficient. Expert systems have been progressive in such diverse fields as medicine, engineering, and business. In each case, expert systems have enhanced the worth, competence, and economic power of individuals, corporations, and intelligence-driven entities. Despite modern advancement, much research is still required to achieve optimal application of these technologies.

Specifically, the medical industry is in need of accurate, yet adaptive, medical diagnostic program system. Based on the nature of illness, traditional, imperative languages have proven insufficient to meet this demand. Prolog Inductive Logic Programming has a unique potential to return ongoing, progressive responses to patient queries. This inductive approach to logic programming in medicine explores potential for successful implementation of Prolog Medical Diagnosis interface.

In hospitals, a diverse array of monitoring tools has been built using computer programming. Medical imaging is now used to produce images of the human body. Several contemporary imaging and scanning techniques are also primarily based on computer technology. Rapid advancement in computer sciences has allowed us to execute numerous highly
developed medical imaging procedures without which many lives would most certainly have been lost. Moreover, these programs have proven critical to the development of magnetic resonance imaging systems used to diagnose several central nervous system disorders (Edelman and Warach, 1993, p. 708). In order to acquire 3-dimentional images, computed tomography employs digital geometry processing methods. Highly developed computer systems and infrared cameras are utilized to generate high-resolution images. Computer systems are extensively utilized for the creation of 3-D images. Today, computer software is embedded in several modern medical tools. Also, many of these medical tools work by pre-programmed command. In nearly all medical tools, the circuitry as well as logic is mainly a computer. Furthermore, the working of hospital bed tooting structure, X-ray machines, emergency alarm arrangements, and numerous other types of medical equipment is based on computer logic.

Recently, computer engineers have enlisted the help of computer software to diagnose and treat disease. Medical examinations are now performed with the assistance of highly developed computer-based systems. Computer-assisted surgery (CAS) is also a rapidly growing area. Complex surgeries can now be executed with fewer risks and greater accuracy. To provide swift, efficient and more precise surgical outcomes, CAS merges medical proficiency with computer intelligence.

Medicine is based on a set of guidelines rooted in statistical theory. Each day, physicians use scientific reasoning to diagnose the circumstances affecting patients and render the appropriate cure. Physicians apply their expertise as well as prior experience with diverse disease processes to form a differential analysis which they subsequently use to find out which syndrome or illness the patient likely has. This is the basis of scientific reasoning.
There are two types of scientific reasoning: deductive and inductive. Physicians rely upon these methods to conduct medical research and respond to queries concerning particular groups of patients. In general, deductive reasoning is conducted when physicians utilize general theory to reach a specific conclusion, whereas inductive reasoning involves collecting observations to form a broader theory.

1.2 Background/Rationale

For expert systems, inductive learning is an eminent channel to automatic knowledge achievement – far prevailing over the Feigenbaum bottleneck of knowledge obtained from human experts (Michie D., 1985) (Steels L., 1985). Under the Feigenbaum model, experts will input knowledge into a machine. This knowledge is articulated through the “Feigenbaum bottleneck” to produce a set of instructions to be stored in a database. This human “say-how” is then programmed and compiled to endow the machine with the ability to evoke and convey this expertise (Figure 1.1).

![Figure 1.1 Feigenbaum Model](image-url)
Rather than engaging in the lengthy procedure of extracting knowledge from professionals and proficient literature, an inductive learning system can be employed to produce this knowledge (generally in the vein of decision rules) through observation and data assembly. For medical diagnosis of specific analytic problems, it appears that this technology is compatible. Often, data regarding accurate diagnoses are presented in the form of medical annals in particular hospitals or their units. For inductive learning, one must type all the data into the computer in the proper manner and execute the system. The resultant set of decision rules can be employed to disclose the fundamental associations and rules of the problem domain in an unambiguous and understandable form and, certainly, can be applied toward patient diagnostics.

1.3 Research Problem

In order to meet a growing demand for computer diagnostic aids, appropriate and efficient programs must be developed. Under traditional (imperative) programming languages, declarative specifications for solving problems can be generated, but inferred causes of problems may not (Flach, 2007, p. 15). This limitation is especially confining to the field of medical diagnosis, which is, by nature, inferential. Thus, doctors will require a different programming approach to medical diagnosis.

1.4 Research Methodology

This project has been developed through widespread secondary research of accredited manuscripts, standard papers, business journals, white papers, analysts' information, and conference reviews.

The inductive logic program works by having knowledge of all symptoms but no illness. Each new patient is asked yes/no questions concerning his symptoms, and returned responses are recorded. The diagnosis works by matching symptoms of known illness with these recorded responses in a decreasing order of complexity. Inductive reasoning (in the form of generalization) is used to update illness symptoms.

Below is a list of important predicates involved in this research methodology:

- new_patient/1: Used to start the examination/diagnosis of a new patient
- fix_diagnosis/2: Used to update an illness symptoms based on an existing patient’s list of symptoms
- rediagnose/1: Rediagnose the illness of a patient based on the update symptoms
show_patient_records/0: Show all existing patients, their symptoms, and illness

illness/2: This can be used to match all illnesses with their symptoms

1.5 Resources

Significant resources are required to achieve an efficacious completion of this research study. The following prospectus details a list of resources that will play a primary role in the successful execution of our project:

- A properly functioning workstation (PC, laptop, net-books etc.) to carry out desired research and collect relevant content.
- Unlimited internet access.
- Unrestricted access to the university lab in order to gather a variety of literature including academic resources (for e.g. Prolog tutorials, online programming examples, bulletins, publications, e-books, journals etc.), technical manuscripts, etc.
- Prolog development kit in order to program the desired system and other related software that will be required to perform our research.

1.6 Scope and Limitations

The proposed system will undoubtedly elicit criticism from those who fear the usurping of human intelligence by computer technology in medical science. Although it is quite difficult to examine the real facts of this claim, it should be noted that the Prolog programming tool executes in collaboration with human intelligence in order to serve with sufficient rules and outcomes. Moreover, medical practitioners and physicians will remain active participants in the
maintenance of this system. Conversely, some critics may assert that the inductive model overloads physicians with an imperative to catch inefficient outcomes and prominently maintain the system. Some critics may also feel that a physician who must remain involved in maintaining the system might issue the diagnosis himself. This problem may occur during the initial stages of program development, but it cannot remain true for long. While the Prolog database is updated and perfected over time, diminishing human interference will be needed in order to retrieve precise outcomes in an effective manner.

Inductive reasoning in programming is also criticized for the limited information it contains at the start of usage. When the Prolog Medical project is first introduced, for example, only some data is contained within the program. This data is stored in the form of questions which may be posed to the user in an effort to gather more data to be categorized for later use. For example, the program will know to ask the patient if he is experiencing a headache, but it will not have a preprogrammed diagnosis for the patient who indicates he does or does not have a headache. While this absence of information undoubtedly hinders data return upon first usage, the computer’s ability to grow and adapt with incoming data is net beneficial. Not only does the program quickly build a wealth of stored data, it also maintains the ability to interpret that data in newly accurate ways. Thus, the data base can add new discoveries upon demand, and these new discoveries can, in turn, perfect the existing data base.

1.7 The Problem Statement
1.7.1 A Clear Formulation of the Research Problem

According to (Grolier, 1994), inductive reasoning is defined as,
"Induction is a major kind of reasoning process in which a conclusion is drawn from particular cases. It is usually contrasted with deduction, the reasoning process in which the conclusion logically follows from the premises, and in which the conclusion has to be true if the premises are true. In inductive reasoning, on the contrary, there is no logical movement from premises to conclusion. The premises constitute good reasons for accepting the conclusion. The premises in inductive reasoning are usually based on facts or observations. There is always a possibility, though, that the premises may be true while the conclusion is false, since there is not necessarily a logical relationship between premises and conclusion."

This kind of reasoning has great potential to develop productive characteristics of human beings (De Koning, Sijtsma and Hamers, 2003). However, its applications in the field of medical science are somewhat underrepresented in comparison to deductive and abdicative reasoning. Our study intends to develop a novel system that will utilize inductive reasoning in the domain of medicine, providing a platform from which physicians and other concerned individuals may gain constructive outcomes diagnosing disease.

1.7.2 Identification of Research Topic

At present, the key to any health-based profession is clinical practice. The ability of a clinician to achieve consistent results during the treatment of patients is the only way to judge a professional’s growth and credibility. It may be possible for logic programming to achieve similar results. This research proposes to explore the following topics:

- Differences between Inductive and Deductive Reasoning
- Utilization of Inductive reasoning in the domain of medicine
- Utilization of Prolog to attain effective results for the determination of an ailment
This research possesses the capacity to provide significant benefit to science and health professionals by obtaining real time results for diseases. Moreover, this system is intended to make use of Prolog, an optimal programming paradigm in the field of artificial intelligence. Prolog will facilitate multidirectional reasoning in which connections between inputs and outputs are observed to form distinct generalizations. This will allow the system to reuse programmed rules in a manner analogous to human reasoning.

1.7.3 Refining the Research Topic into a Research Problem

Many researchers and analysts have demonstrated interest in developing logic programming for the medical sciences. This is, in part, due to the fact that a demand for medical development will persist insofar as there are humans on this earth. Laura E. Hardin (2002) is just one example of the many researchers committed to improving the medical sciences through logic programming.

Hardin provides a comprehensive review of the problem-solving methods currently considered within the realm of medicine. The analyzed models intend to define and assess the rational processes triggered in medicinal domain problem-solving and to utilize this information to further estimate expert–novice differences.

Hardin discusses the hypothetic–deductive, illness scripts and probabilistic principle models and highlights their respective reasoning strategies. These strategies, forward–backward and broad-narrow, are analyzed as indicators of expert versus beginner problem-solving in the field of medicine. Finally, medical problem-solving methods (simulated patients, stimulated recall, think-aloud protocol, Pathfinder algorithm) are discussed.
1.7.4 Influence of Problem Statement on Organizations

Mark A. Jones summarizes findings from early medical education research of clinical reasoning, highlighting universal components of sound clinical reasoning. Above all else, Jones finds that “knowledge organization” (or the means by which one organizes his or her knowledge and expertise) plays a significant role in differentiating between expert clinicians and novices (1992). Early medical education studies analyzed clinicians' thoughts (e.g., perceptions, interpretations, plans), either retrospectively as the clinicians thought aloud while being prompted by a video or audio playback of a previously completed patient examination, or concurrently as the clinicians read a patient's unfolding clinical history.

1.8 Professional Significance of the Study
1.8.1 A Clarification of the Professional Significance

The medical field elicits ample opportunity to explore the differences between inductive and deductive reasoning. Practically, patients do not just walk into hospitals and declare the disease “X” that is afflicting them. Rather, they describe the symptoms “Y” they are experiencing and await information and advice from their practicing physician. Our study is designed to develop an expert system to aid physicians in this formerly underserved area of computer programming.

Chow et al (2010) of Queen University’s Nursing and Midwifery department claims that proper assessment of decision making ability in the field of medicine is a significant feature utilized by all doctors. Not only must clinicians have an accurate perception of the logical processes through which they arrive at their decisions, so to must those working within the
general medical climate develop acumen for identifying and developing good decision-making skills. Pritchard (2006) discusses the related demand for accurate medical and scientific decisions in the highly specialized field of nursing and midwifery.

According Mc Caffery et al., (2000), medical practitioners’ personal opinions of their patients have a significant effect on the conclusions they draw concerning any given course of pain treatment. The results and standards of care for patients are frequently determined by a medical practitioner’s ability to recognize a way to react to the symptoms of developing disease (Jones et al 2009). This kind of critical thinking can greatly influence the problem solving and decision making that medical professionals tackle on a regular basis. The skills needed to develop this critical thinking include inductive and deductive reasoning.

All doctors must engage in inductive reasoning when attempting to diagnose a patient. Unlike the deductive reasoning invoked when treating a patient with disease X, inductive reasoning must initially be used to confirm that the patient indeed has disease X. This inductive reasoning is executed through an interview conducted by the practicing physician of his patients. Here, the doctor asks his patient yes or no questions to determine the patient’s symptoms, and compares those symptoms to the known rules he has amassed over his professional. This comparison process will return one of two results: that the doctor does not know what afflicts the patient and more tests/exams will have to be executed to learn more, or that the patient suffers from disease X, and the appropriate treatment will be given to remedy the affliction.
1.8.2 Motivation behind Research Topic

In recent years, people have become highly circumspect toward their health. In response to a growing number of distinct diseases, people want to count on each and every one of their body’s functioning cells and resulting behaviors. Many websites and forums are intended to provide meaningful information about health related issues for their visitors. However, only a few of them provide authentic details about the symptoms of a certain disease.

Additionally, many physicians and doctors use predictive tools and methods to evaluate and diagnose an evolving disease. The proposed system is intended to service those individuals who wish to determine and diagnose emerging disease against expressed indications by means of concise yet accurate questionnaires.

1.8.3 Professional Value of the Study

Many valuable studies have recently been carried out with a corresponding rise of logic programming intensive trends within the field of medical science. Researchers intend to find more developed patterns in the desired field in order to make accurate assumptions for the newly evolved diseases and assign specific cures for them.

Cynthia A. Thompson and Raymond J. Mooney (1994) proposed a new inductive learning system - LAB (Learning for Abduction) - which acquired adductive rules from a set of training examples. Thompson and Mooney’s primary objective was to determine the minimum threshold for learning – the point at which there is sufficient foundation for accurate assumption generation. The proposed system contrasted with conventional systems in that it used inductive
reasoning to form these predictions. Moreover, it had been experimentally examined and matched against the other learning systems, subsequently providing an expert knowledge domain of brain damage detection following stroke.

1.8.4 Contribution of study to Professional Knowledge

1.8.4.1.1 Detailed Professional Significance

Medical reasoning is defined as qualitative evaluation and investigation of reasoning procedures included in developing medical decisions (Patel, V. L., Arocha, J. F., & Zhang, J., 2004). The relative terms used for medical reasoning (clinical reasoning, diagnostic reasoning, medical problem solving etc.) are frequently used in the emerging body of literature that evaluates the process by which physicians form medical conclusions. To add to this, the authors have clarified the concept of medical cognition by referring to it as the study of cognitive procedures (comprehension, decision making, and problem solving) that occur during medical practices. Many researchers have investigated this subject and have provided valuable conclusions concerning artificial intelligence in the field of medicine.
Section 2

2.1 Introduction

Modern development in the field of computer science has led to the advent of intelligent systems and computational equipment designed to encapsulate and provide expert knowledge. The fundamental technology presented is derived from initial research on biomedical systems in the 1970s. Medical researchers continue to pursue advancements in such vital domains as knowledge acquisition, inductive based reasoning, and system incorporation for scientific environments. Hence, it is significant for physicians to realize the existing shape of such research and the theoretical and logical hurdles that confront those who wish to make these systems accessible.

2.2 Diagnosis System and Medical Field

2.2.1 ES DIABETES (An Expert system In Diabetes)

*ES Diabetes* is an expert system which was implemented by Computer Science graduate students at Texas University, Corpus Christi. Their research was built upon articles published by the Consortium for Computing Sciences in Colleges, and the Journal of Computing Sciences in Colleges (2001). ES diabetes is the upshot of two different research papers. Both first as well as second phase were carried out by the students of the same university. The first phase was done in 1999. From this, the team offered a small model in CLIPS 6.0. The second part was carried out in 2000.
Justification

*ES Diabetes* is specifically made to help patients diagnosed with Type 2 diabetes control their blood glucose levels. Through ES, diabetic people are able to enter information at any time regarding their condition and blood sugar level and receive immediate treatment recommendations. There is a built in interactive mechanism between *ES Diabetes* and its user Expert system. Thus, questions are posed by *ES Diabetes* and the user must give answers to those queries. *ES Diabetes* will continue this process until it is unable to extract sufficient data to generate a suggestion. Various types of questions are posed by the system in an effort to determine who the patient is, what symptoms the patient is experiencing, if those symptoms are diabetic expressions, what food the patient has consumed, its type and quantity, the user’s level of blood sugar and if the individual is taking insulin injection, its quantity, who administered the injection, etc. After gathering enough information, *ES Diabetes* will provide the user with an effectual suggestion to keep his level of blood glucose within normal range.

Restriction

Patients diagnosed with Type 2 Diabetes are the target users of this expert system. Therefore, this expert system can only be used by those people who have been diagnosed with Type 2 Diabetes and are aware of their condition. Responses to the system queries regarding present signs of illness, blood glucose level, food, and medication will eventually lead the system to a conclusion, and on its basis, a suggestion for treatment. For instance, the system may, upon gathering sufficient information, recommend that the patient intake 5 units of insulin. This system neither provides treatment for Diabetes type 1 or gestational diabetes nor makes
diagnoses of the diseases. Diabetes Monitoring instruments, such as a glucometer, are easily accessible in the market. ES Diabetes just provides help to keep blood sugar levels under control.

2.2.2 A Novel System to analyze Multiple Skin Syndromes

S. Samy, Abu Naser, and Alaa N. Akkila (2008) suggested a proficient system to diagnose multiple skin syndromes. The proposed system is intended to assist professional Dermatologists and other related individuals by analyzing nine kinds of skin diseases. The prominent skin ailments include Icthyosis, Insect Bites, Acne, Measles, Scarlet Fever, Psoriasis, Eczema, Stings, Warts and Meningitis. The proposed methodology not only highlights these diseases but also suggests the suitable cure for them respectively. The system has been developed under CLIPS environment.

The expert skin syndrome diagnostic system is built upon a menu based interface. The targeted nine skin diseases are organized under three classifications:

- Skin Infections
- Skin Rashes (with Fever)
- Skin Rashes (without Fever)

According to the developers, the system will require the user to enter some related information needed to process the correct ailment. The system is also programmed to offer an overview of the analyzed disease. In order to incorporate this feature, however, the system will need to be efficient enough to prescribe the related treatment and medical advice.
The system’s primary sources of knowledge and information are expert Dermatologists and expert skin disease websites. The information was then processed under CLIPS environment to develop this intelligent system. In order to manage the treatment of all nine diseases, the system follows approximately 60 rules to process the user’s information.

2.2.3 JESS based System for Endocrine Treatment and Diagnosis

A JESS based intelligent system to treat and analyze Endocrine Syndromes was developed by the faculty of Engineering and Information Technology, Al-Azhar University, Gaza, Palestine under the publication of the Asian Network for Scientific Information (2010) in the Journal of Artificial Intelligences 3(4): 239-251, ISSN 1994-5450.

System justification:

The novel methodology has inaugurated the concept of an expert system that will be able to treat and analyze the syndromes related to the Pancreas, Parathyroid Glands, and Thyroid, specifically. It should be significantly noted that the suggested system does not aim to replace the human efforts, accuracy, or speed of processing data. Rather, it is meant to assist the endocrine specialists and clinicians with their work.

The Human Endocrine System is critical for life. It is comprised of the pituitary gland, parathyroid gland, thyroid gland, stomach, pancreas, intestine, ovaries in females, testes in male, heart, adrenal, kidney, hypothalamus etc. The glands and hormones in an endocrine system not only affect the essential organs of the human body, but also influence each and every cell behavior of the body. In short, endocrine system can be considered the primary agent through which growth and cell development takes place.
The proposed JESS is subsequently highly complex, and highly comprehensive. It is intended to perform the following features:

- Assist beginner clinicians in analyzing the Patient’s history and case so that they can understand the proper treatment and learn from it.

- Analyze the basic endocrine syndromes such as parathyroid glands, pancreas and thyroids and prescribe the suitable cure for them.

- Provide essential information to users who wish to engage in precautionary activities or who must quickly react in emergency cases.

This research based system has been developed under Java Expert System Shell (JESS). The chosen development kit works as a rule based engine for Java language platform which is a superset of CLIPS programming language. Moreover, it provides rule based programming which is suitable for making the system intelligent and automated. The version for development is kept as Java 1.6.0 or 1.5.0 and is compatible with Windows Vista Home Premium or Windows XP.
3.1 Introduction

The program works by having knowledge of all symptoms but no illness. Each new patient is asked yes/no questions concerning his symptoms, and his responses are subsequently recorded. The diagnosis works by matching symptoms of known illness with these recorded responses in a decreasing order of complexity. Inductive reasoning (in the form of generalization) is used to update illness symptoms.

Below is a list of important predicates which are involved in the research methodology:

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>new_patient/1</td>
<td>Used to start the examination/diagnosis of a new patient</td>
</tr>
<tr>
<td>fix_diagnosis/2</td>
<td>Used to update an illness symptoms based on an existing patient’s list of symptoms</td>
</tr>
<tr>
<td>rediagnose/1</td>
<td>Re-diagnose the illness of a patient based on the update symptoms</td>
</tr>
<tr>
<td>show_patient_records/0</td>
<td>Show all existing patients, their symptoms, and illness</td>
</tr>
<tr>
<td>illness/2</td>
<td>This can be used to match all illnesses with their symptoms</td>
</tr>
</tbody>
</table>
3.2 Hierarchy of Development Process

The class structure of our development process is clearly shown in Figure 3.1.
Under this model, background information is gathered and conceptualized to give rise to the identification and selection of a problem. At this stage, more knowledge is acquired to satisfy that problem and, ultimately, represent related knowledge in a user-friendly way.

3.3 Research Approach

There are two primary types of research approaches (Figure 3.2):

![Figure 3.2: Types of Research Approaches](image)

3.4 Quantitative Research

Generally, quantitative research requires gathering and transformation of data into numerical form compatible with statistical computations and resulting conclusions. Researchers will hold one or more suppositions. These suppositions must be converted to queries, a forecast for probably associations, and the things researchers will need to scrutinize (variables). To determine answers to these queries, researchers will draw upon a range of tools as well as
materials (for example: paper or computer tests, inspection check lists, and so forth.) and an unambiguously identified plan of action.

Data is gathered through a variety of means following a precise process and prepared for statistical analysis. Today, this process is achieved with the help of highly developed statistical computer packages. These computers aid the researchers in determining the degree to which there is an association among two or more variables. This could be a straightforward relationship (for example, inhabitants who exercise regularly have lower blood pressure) or a causal association (e.g. regular exercise essentially leads to lower blood pressure). Statistical analysis allows researchers to determine complex causal associations and to learn the degree to which one variable has an effect on another.

3.4.1 Deductive Reasoning

Conclusions deduced from a set of laws and standard principles are the result of Deductive Reasoning. A deductive approach to reasoning derives specific facts from existent theory (OUCL, 1999). A deductive reasoned will engage in a top-down approach in which he or she moves from general to specific information. Figure 3.3 illustrates this approach.
For example, a doctor engaging in deductive reasoning may consider the theory that high blood pressure and cholesterol are directly correlated with risk of heart disease. He will thus hypothesize that high blood pressure and cholesterol cause heart disease. When he observes a patient with high blood pressure and cholesterol, he will subsequently confirm that this patient has a higher than average risk of contracting heart disease.

Deductive reasoning is an excellent tool to utilize when theoretical information is present. In the aforementioned example of the doctor, the data was high blood pressure and cholesterol, and the theory was a strong correlation between those factors and heart disease. Thus, high blood pressure and cholesterol was the cause, and heart disease was the deduced result. When knowledge of the cause is absent, however, a different reasoning tool must be employed.

3.5 Qualitative Research

Qualitative research is often related to the social constructivist model, which emphasizes socially constructed authenticity. This type of research endeavors to record, analyze, and reveal deeper meaning and importance of human actions and experience, together with ambiguous thinking, behaviors, and feelings. Qualitative researchers are more concerned with obtaining a rich and intricate understanding of the forces that construct the human being than in gaining information which can be applied to larger units.

Typically, qualitative analyzers prefer to gather data and observe patterns prior to developing a theory. Thus, qualitative research will often present a theory constructed from specific observations in a bottom-up approach. Nevertheless, a large number of qualitative
research projects also entail some degree of deductive reasoning. Qualitative researchers unambiguously recognize an issue or topic that they need to look at and possibly will be directed by a hypothetical lens - a type of overarching theory which offers a structure for their research.

3.5.1 Inductive Reasoning

An inductive approach to reasoning surmises general beliefs from specific facts. An inductive thinker will thus engage in a bottom-up approach in which he or she moves from specific to general information. Figure 3.4 depicts a logical inductive pathway.

![Figure 3.4: Our Descriptive Approach](image)

For example, a doctor employing inductive reasoning may observe patients who have had heart attacks. As he observes these patients and collects data on their respective conditions, he will begin to detect a pattern. This pattern will show that every patient who has had a heart attack also had high blood pressure and cholesterol. Based on this pattern, the doctor will develop the tentative hypothesis that all patients who have had a heart attack also have high blood pressure and cholesterol. If this hypothesis is repeatedly confirmed, he will theorize that high blood pressure and cholesterol cause heart attacks.
Inductive reasoning is necessary when theoretical information is absent. In the case of medical diagnoses, patients can only bring evidence of their specific observations. For example, a patient may know that he or she has experienced chest pain, but will not know what is causing this pain. Only inductive reasoning can build on this clue. Therefore, inductive reasoning is the preferred method to approach symptom diagnosis.

3.6 Design of the Study

The design for our study is built on an inductive reasoning approach. The following sample session will illustrate how our study will progress:

- A new patient p1 has the symptom: [runny_nose]. He/she is diagnosed with unknown_disease. The user fixes the learning process by issuing a fix_diagnosis (p1, cold) goal. And thus, the system “learns” that [runny_nose] is the list of symptoms for cold.

- A new patient p2 comes with the same symptoms. Since the symptoms match “cold”, the system diagnoses cold for p2.

- Another patient p3 comes with slightly different symptoms: [fever, runny_nose]. The system works first by attempting to find an illness with all of these symptoms. Since there is none, the system moves to match a partial list of symptoms, and succeeds in matching cold. So the system proposes that the best match is cold. The user of the system fixes the learning process by issuing a fix_diagnosis (p2, flu) goal. And thus, the system “learns” that [fever, runny_nose] is the list of symptoms for flu.

- A new patient p4 comes with the symptoms [fever, runny_nose]. This time the system diagnoses flu.
Another patient p5 comes with different symptoms: [fever, cough]. The system tries all possible matches but fails to diagnose anything and says that the diagnosis is unknown_disease. The user of the system fixes the learning process by issuing a fix_diagnosis (p5, chest_infection) goal.

Another patient p6 comes with a subset of the symptoms, just [cough]. The system tries all possible matches but fails to diagnose anything. The user of the system fixes the learning process by issuing a fix_diagnosis (p6, chest_infection) goal. The inductive learning process relates the old symptoms [fever, cough] with new symptoms just [cough] and generalizes that the symptoms for check_infection are just [cough].

The system is working to identify patterns in symptoms and disease. In order to achieve effective inductive reasoning, however, its operative physician must intervene to ensure accurate pairing of symptom and disease. In the beginning, much oversight will be required to ensure no inaccurate data is recorded by the system. However, as time progresses and patterns are accepted, the system can lend an increasing amount of assistance to the physician.
4.1 Introduction

This chapter describes the process by which Prolog implements and practices inductive reasoning. The proposed methodology is built on Proof Theory - a branch of mathematics that describes the concepts of mathematical proof and mathematical provability. Our System is meant to gather the relevant data (i.e. syndromes of a disease) through forms and match it with stored variables on the basis of proof theory. This section of the report will provide an explanation of each transaction, drawing upon examples to clarify application work flow in its entirety.

4.2 How Inductive Reasoning Works

Inductive reasoning starts with preliminary observation and proceeds towards the discovery of a specific pattern. This enables us to make an uncertain prediction that leads to a general assumption about the behavior of a certain entity. Eva KW (2005) describes the stages through which inductive reasoning gives rise to a logical program.

4.2.1 Stage I – Gathering and Refinement of the Preliminary Data

During Stage I, data is obtained from any viable source: visual images, reading, listening, questionnaires or other sensory input. In our case, it has been collected through user forms provided by the system to its end users. Users are presented with Yes/No questions to which they respond with data. This data is taken into the system and stored accordingly.

4.2.2 Stage II - Define an initial set of Hypotheses
The refined set of information is then formulated in the domain of identified queries and issues in the existing case as well as the knowledge base of former cases by means of schemas and pattern identification. Experts swiftly develop a small set of hypotheses with nominal medical data to show the issue to be resolved. Temporary memory can actively handle only about 5 items at once. Therefore, experts will usually have the concluding diagnosis for this set of hypotheses within 5 minutes of opening whereas novices and beginners will take longer to construct a set of hypotheses.

4.2.3 Stage III - Additional information derived from Preliminary hypothesis

This stage enables us to develop a framework for gathering additional information derived from the initial hypothesis. The process is recursive by nature and continually refines itself. Beginners and learners have more iterations of this process.

4.2.4 Stage IV - Using Inductive Reasoning

In this phase of implementation, inductive reasoning begins working from specific to general. This means that the observed information is matched with a proven pattern (i.e. the algorithm to achieve a hypothesis). The hypothesis is then matched against the problem in the case.

Inductive reasoning gives effective results when an expert, accurately derived algorithm is followed. The algorithms have been developed with statistical significance to real-time cases. It should be noticed that the human body is very complex; it is impossible to fetch all the information we want. Therefore, we can only estimate the “most likely” diagnosis and can never actually prove or disprove most hypotheses.
4.2.5 Stage V - Define and Test the Final Diagnosis

After achieving the most likely diagnosis through inductive reasoning, the expert system will define and test it against the positive and negative findings and standard criteria for description of a disease process. To reduce the possibility of early closure, we can assume that the working diagnosis is false and then consider the alternative diagnosis.

4.3 Inductive Reasoning and Prolog

The logical programming language, Prolog, possesses inductive facets. In contrast to purely declarative languages, such as C/C++ or Java, Prolog is based on proof theory enlisting the use of logical inference to progressively create, revise, and sustain rules. Prolog enlists the use of Resolution Theorem Proving (figure 3) to draw conclusions (OUCL, 1999). Specifically, Prolog inverts this process, developing inductive inference rules by observing correlations and assuming causation.

![Figure 4.1 Inductive Process using Prolog](image)

4.4 Proof Theory
Proof theory states that an event can be proven a consequence of a problem if every model of the problem is a model of the consequence. For example, if the patient has both high blood pressure and high cholesterol, and these conditions are shown to be members of the set heart attack, then proof theory proves the heart attack is the consequence of high blood pressure and cholesterol.

Proof Theory in Prolog is only efficient in the presence of a complete set of data. When information is missing, must be verified, or added, a degree of flexibility should be exercised to achieve greater efficiency and accuracy. Through inductive proof generalization, Prolog can build and maintain a dynamic, adaptive, medical database.

Using inductive proof generalization, Prolog draws conclusions based on user input and monitoring. Through a process of inductive reasoning, general rules are subsequently drawn from specific cases. For example, the rule “All masters students are smart”, coupled with the input data, “I am a master student” would result in Prolog’s conclusion that the student is smart. In this experiment, questions posed will seek to determine the cause of ailments and illnesses. Inquiries such as “Do you have headache?” and “Do you have a running noise?” will be posed in an effort to gather responses which can be meaningfully matched with programmed rules to generate reasonable conclusions.

4.5 Application

The application of Prolog Inductive Reasoning Language into the world of medical diagnosis is both interactive and dynamic. As such, the program, itself, begins with no established rules or information. Instead, the program is equipped with Yes or No questions it
may pose to the patient in order to induct enough clues to draw an inference. When a new patient enters the program, the program recognizes this new patient as

4 ?- new_patient(p1).

Prolog will begin an examination in which it poses the following questions:

Starting examination...

Symptom: fever? no.
symptom: cough? no.
symptom: shivering? no.
symptom: runny_nose? yes.

If the patient responds that he or she has a runny nose, but no fever, cough, or shivering, the program will surmise that the patient’s confirmed observed symptom is a runny nose.

Confirmed symptoms are [runny_nose]

Next, the program will attempt to determine the illness that is causing the runny nose.

Determining illness...

However, when the program searches the new database, it will find no existent rules to satisfy this query, instead returning the diagnosis

AI diagnosed that you have unknown_disease

ture.

At this point, a doctor, medical practitioner, or human monitor will intervene to correct the diagnosis.

5 ?- fix_diagnosis(p1,cold).

Confirmed patient symptoms [runny_nose] will be related to cold

There was no earlier definition of cold
New definition is \texttt{[runny\_nose]}

true.

The program is now equipped with an existing rule: cold is defined by runny nose. Thus, all future positive responses to the query runny nose will search the cold rule for a match. Once the rule is applied, the system will rediagnose the patient and return the following result:

6 ?- rediagnose(p1).

Confirmed symptoms were \texttt{[runny\_nose]}

Re-diagnosing...

Updated diagnosis is that Patient p1 has a cold

true.

A new patient accesses the program and engages in a similar interview process.

7 ?- new\_patient(p2).

Starting examination...

symptom:fever? no.

symptom:cough? no.

symptom:shivering? no.

symptom:runny\_nose? yes.

This patient enters responses identical to those of Patient 1, and the system returns a positive diagnosis: cold.

Confirmed symptoms are \texttt{[runny\_nose]}

Determining illness...

AI diagnosed that you have cold

true.
A third patient enters the program.

8 ?- new_patient(p3).

This patient also indicates that he or she is experiencing a runny nose, only this time, the patient also answers yes to the query “Do you have a fever?”

Starting examination...

symptom:fever? yes.

symptom:cough? no.

symptom:shivering? no.

symptom:runny_nose? yes.

Because no known rule exists to address the symptom fever, the system will only run a check against runny nose and confirm that the patient has a cold.

Confirmed symptoms are [runny_nose,fever]

Determining illness...

AI diagnosed that you have cold

ture.

At this point, the doctor, medical practitioner, or monitor will intervene to correct the diagnosis, programming the additional rule: flu is defined by runny nose and fever.

9 ?- fix_diagnosis(p3,flu).

Confirmed patient symptoms [runny_nose,fever] will be related to flu

There was no earlier definition of flu

New definition is [runny_nose,fever]

ture.

A fourth patient enters the program.
This patient returns responses identical to those of Patient 3, and an identical result is subsequently generated.

Starting examination...

symptom: fever? yes.
symptom: cough? no.
symptom: shivering? no.
symptom: runny nose? yes.

Confirmed symptoms are [runny nose, fever]

Determining illness...

AI diagnosed that you have flu

true.

A fifth patient enters the program. This time, he or she indicates that they have a fever and cough, but no shivering or runny nose.

11 ?- new_patient(p5).

Starting examination...

symptom: fever? yes.
symptom: cough? yes.
symptom: shivering? no.
symptom: runny nose? no.

When the program runs a check against fever, it finds that there is a mismatch between rules known rules for fever, and the symptom combination exhibited by this patient. Under proof
theory, this constitutes insufficient data to draw a reasonable conclusion, and unknown disease is subsequently identified.

Confirmed symptoms are [cough, fever]

Determining illness...

AI diagnosed that you have unknown_disease true.

This is yet another opportunity for a doctor to intervene and correct the problem. The doctor recognizes cough and fever as symptoms of a chest infection, and adds a new rule to the database.

12 ?- fix_diagnosis (p5, chest_infection).

Confirmed patient symptoms [cough, fever] will be related to chest_infection

There was no earlier definition of chest_infection

New definition is [cough, fever] true.

This rule will prove sufficient to return a positive diagnosis for chest infection if and when another patient enters symptoms identical to those of patient five. A sixth patient accesses the system, this time returning another unique set of data: cough. In the absence of a fever, the cough is insufficient to indicate a chest infection. No known rule exists for only a cough, and the system subsequently identifies the patient’s disease as unknown.

13 ?- new_patient(p6).

Starting examination...

symptom: fever? no.

symptom: cough? yes.
symptom:shivering? no.
symptom:runny_nose? no.
Confirmed symptoms are [cough]

Determining illness...

AI diagnosed that you have unknown_disease
true.

Here, the doctor may intervene to correct the misinformation that a chest infection must be characterized by both a cough and a fever. New data will return to indicate that cough, alone, is sufficient to return a positive diagnosis for chest infection.

14 ?- fix_diagnosis (p6, chest_infection).

Confirmed patient symptoms [cough] will be related to chest_infection

Earlier definition of chest_infection was [cough,fever]

New definition is [cough]
true.

After gathering and processing only a few patient inputs, the inductive prolog tool has compiled a rather extensive list of rules by which patient-indicated symptoms may be matched with known causal diseases.

15 ?- show_patient_records.

p1 has symptoms [runny_nose] and diagnosed cold
p2 has symptoms [runny_nose] and diagnosed cold
p3 has symptoms [runny_nose,fever] and diagnosed flu
p4 has symptoms [runny_nose,fever] and diagnosed flu
p5 has symptoms [cough,fever] and diagnosed chest_infection
p6 has symptoms [cough] and diagnosed chest_infection true.

4.6 Conclusion

From the above discussion, we have concluded that proof theory is syntactic as it is concerned with the rules and standards used for developing or transforming the symbols and words of a language. To add this, we have analyzed the complete work flow of the proposed system in terms of Prolog syntax and the expected outputs to gain a better understanding of the system’s functional areas. The results of the experiment are highlighted in the following section.
5.1 Introduction

The following research has been compiled in an effort to inform the practical as well as theoretical background of Prolog Programming language as a development tool for our proposed system. Moreover, this section sheds light on the root cause for preferring Prolog over other languages. In order to justify this preference, we have highlighted the main differences between Prolog and other well-known programming languages (C++, Java and LISP) that were included as an alternative in the development phase of the system.

5.2 What is Prolog?

Prolog stands for “Programming in Logic” and was developed by Colmerauer and his colleagues in 1970’s. It is among the highest level general purpose languages in broad use today. The factor that primarily differentiates this language from other popular programming languages is that it is based on declarative programming technique (i.e. it tells the computer what to something), whereas the others are based on procedural programming (i.e. they tell the computer how to do something).

Prolog’s primary theme is coding compact and crisp rules. The Derivations and Deductions prompted through user-entered queries are the products of a built in mechanism of Prolog known as Backtracking. Initially, Prolog was designed for non-numeric data processing, but the current versions of Prolog mainly provide mathematical extensions.

Prolog is worth considering because of its logical, declarative aspects. Its explanatory nature, intrinsic modularity, and compactness make it a dominant choice over the other
languages while developing intelligent and expert systems. The following list presents the types of applications that can provide meaningful, desired outcomes when developed under Prolog environment:

- **Adept Systems** The novel yet automated systems that are able to reproduce decision making similar to human expertise.
- **Intelligent Applications** Programs that can be able to carry out meaningful tasks by making use of Artificial Intelligence (AI) rules and techniques.
- **Natural Language Processing** The applications that are intended to analyze and respond to the queries made in simple language contrary to the approved keywords or menu based selections.
- **Relational Database Systems** The systems that rely on relation based data linked together by means of tables and that contain a set of constraints on each relation. Huge data processing is involved as the volume of data is high in such applications and is stored in tables.

Prolog doesn’t work like the traditional procedural languages do. It emphasizes logical relationships between objects or entities related to a specific problem rather than simply following steps to solve a problem. The system decides the way to solve an assigned task and may involve a series of instructions for a computer to solve a problem. Hence, users may state what they wish the program to accomplish, and leave the workstation to complete the task. In order to accomplish this task, Prolog utilizes a unique and helpful prototyping feature that makes use of parallel architectures. It resolves an issue by searching a knowledge base which would be broadly enhanced if many processors were made to search distinct parts of the database.

5.3 Why use Prolog?
Prolog is highly preferable to other programming languages because it has an immense collection of data structures in its language, a robust notation for encoding end-user applications, and acumen for adopting and implementing human reasoning.

When we decided to work with an expert system during the initial phases of our project, we started to gather information from books on the subject of adept applications. We were anticipating thorough discussions of the nature of intelligence and the ability of a computer to replicate, or even adopt, that type of capability. Rather, we were able to find large collections of search and pattern-matching algorithms.

Since then, we observed that the main theme behind developing such an expert system is pattern-matching, and for this reason we found Prolog to be a suitable option for designing the suggested methodology into a working system.

Prolog is a good source of pattern matching and search. It possesses both qualities proven to significantly facilitate this process:

- Dynamic and robust memory management enabling the developer to use symbols without concern for issues of memory allocation.

- Great support for symbols as a primitive data type that can be employed without need for the nominal distinction of functions.

The aforementioned features constitute the primary reasons Prolog is categorized as a Symbolic Language. David Poole, Randy Goebel and Romas Aleliumas (1985) proposed an expert system Theorist that used a uniform deductive reasoning mechanism to construct explanations of observations with respect to some facts and their hypothesis. The system was developed in Prolog to meet the actual requirement of the methodology. Observations, facts and possible hypothesis were treated as sets of formulas they used to portray a group of observations
on a partial domain, a set of facts for which that domain was a model, and a cluster of uncertain hypotheses which may be needed to provide a constant explanation of observations.

The proposed system was able to replace a conventional, rule-based system by expressing its knowledge in terms of problems and the observations that consequently occurred. The recognition of these patterns gave rise to a system capable of illness diagnosis. It used an alternative formulation of rules and expressed them in the following terms:

Problem $\rightarrow$ observation

Similarly, while diagnosing a disease, the theorist was able to use the rules of the following form in order to encode the observable symptoms of a certain disease:

Disease $\rightarrow$ Symptoms

The authors found this way of representing rules more suitable for expressing text book knowledge of disease as it records what is known without any requirement of heuristic methods like inevitability issues. According to the authors, Prolog is a query answering system in which statements and bodies of clauses are handled as questions to be answered by heads of clauses. Here, the authors have referred to clauses as assertions according to which the program is able “to respond to a query matching the head”.

Max J. Egenhofer and Andrew U. Frank (1990) suggested a research model they used to incorporate the robust logic-based concept of Prolog with a database suitable for spatial, three-dimensional data controlling in order to develop a database query language that is more reliable and powerful than SQL. This implementation, coined LOBSTER, allowed various researchers to explore many areas of Geographic Information and Analysis.
Clocksin and Mellish (1981) explore a Prolog system primarily known as a programming language, but that also contains specific aspects of a database management system. They combined Prolog with a database management system in order to implement a persistent Prolog that enabled its users to store data in a structured manner as defined by the database schema with prolog facts and rules in the same database portraying unstructured data.

5.4 Prolog vs. other languages

5.4.1 Prolog vs. C++

In object oriented programming, an object is a data structure that can inherit fields and executable methods from a class hierarchy to which the object belongs. Clocksin, Mellish (2003) Although there were various parallel efforts to develop an object oriented language based on C, the most successful was C++, a design developed in the early 1980s by BjarneStroustrup at At&T Bell Laboratories. C++ enabled vast communities of programmers to implement the object oriented paradigm in a gradual, evolutionary way. C++ is highly efficient in nature and can be used as a developing source in systems when efficiency is required at highest levels. Prolog, in contrast, is more modular and flexible. It is preferred by the expert systems developers when decision making and logical reasoning are the main themes of the application.

The Prolog program can modify itself and consists of a set of specific facts and rules, whereas C++ programs specify an algorithm and cannot modify.

5.4.2 Prolog vs. Java

Unlike Prolog’s declarative programming paradigm, Java is primarily derived from object oriented programming. Both languages can be preferred, depending on the need for any
given application. SWI-Prolog is free, open source, and very well-maintained. Therefore, it is notably easier to distribute SWI-Prolog applications than Java ones. A Java program needs a Java Virtual Machine to be compiled. With SWI, the compiler can easily generate stand-alone .exe files.

Not only is Prolog easier to distribute, it is also more compatible with words or collections of symbolic characters (e.g. >>, +, /, // etc.) as an infix, postfix or prefix operator. Therefore, it is relatively easy to estimate traditional mathematical notation in any program for numerical operations or multiple multiplication, vector addition and so on. In Java or other domineering languages, the main issue that arises is “identity”. When programmers define their own data structures to represent mathematical entities (e.g. sets, matrices, vectors etc.), Java would handle distinct copies of a structure as different entities. This leads to failure of an equality test of structure and its copy, as they present indistinct parts of store and equality is by pointer.

5.4.3 Prolog vs. Lisp

Prolog and Lisp are the two most commonly used and recognized languages in the domain of AI applications, today. They are developed with two distinct programming paradigms: Prolog is a Declarative language, while Lisp is based on Functional Programming technique. Prolog is mostly used for logic and reasoning problems while Lisp is used for the problem with fast prototyping requirements.

Although both of these languages are considered suitable for AI programming languages, they possess multiple differences. Prolog is ideal for the systems that require symbolic reasoning - database and language parsing - whereas Lisp is the better choice for implementing graphic
oriented applications requiring heavy user interfaces. Lisp is quite flexible because of its fast prototyping and macro characteristics. These attributes enable the language to extensively match problem with solution. Conversely, Prolog is more appropriate for the programs that need logical reasoning and decision making abilities. Making choices between these two languages depends entirely upon the type of AI problem being considered. In this case, we found Prolog to be a more suitable selection than Lisp because we required the use of logical reasoning features.

5.5 Prolog Logic

Logic Programming and Prolog are directly related to one another. This is evidenced by their uses of identical terminology such as “true”, ”predicates”, ”proof”, etc. Not only is Prolog applicable to the systems that inherit logical reasoning, its entire query resolution process is the product of a logical deduction system.

Programming in Prolog is primarily based on three objectives (Clocksin, Mellish, 2003):

- Identifying “Facts” about objects and their relationships
- Defining certain “Rules” about objects and their relationships
- Requesting “Queries” about objects and their relationships

In logic programming, a program is comprised of a set of statements defined in terms of formulae in symbolic logic. The rules of inference cause new formulae to be generated when cross-referenced against old ones. Expressing these rules of inference in terms of symbols is said to have symbolic manipulation processed by computer. This is the main methodology working behind execution of certain logic by a computer (Spivey, 2008). A computer processes new formulae from old ones to provide a solution to the proposed problem. If the old formulae in the program behave as “true”, then the derived formulae will replicate them, resulting in accurate and precise results. In order to verify if the program produces correct outcomes, a developer
checks that the running program contains only true statements and that if it is being executed in the correct manner. The developer may also be concerned with the response time of the program to ensure that the results can be fetched quickly.

Logic programming and Prolog are closely related as logic programming is the backbone of Prolog. The major benefit of Prolog is the way in which it can provide a clearer knowledge of the underlying concepts and act as a medium to develop new methods as defined by the prototypes.

5.6 Prolog Interpretation

The major entity in Prolog that is associated with the operations and procedures in programming languages and with the relational databases domain is known as Predicate. A Predicate usually takes a number as arguments which is known as “arity” of the predicate. The notation q/2 indicates the name ‘q’ and arity ‘2’ and q (a, b) is represented as an atom.

A Prolog predicate implies the predicate symbols in logic and the terms inside a predicate relate to the functional terms occurring as arguments of logic predicates. These terms are usually based on three pillars: constants, variables and function symbols. These major elements in Prolog are defined as Prolog atoms, variables, and functors, respectively.

Syntactically, logic formulae can be defined as finite sequences of symbols such as variables, functors and predicate symbols (Nillson, Maaluszynski, 2000). All these appear in a variety; hence, the symbols are generally presented in finite strings based on primitive characters. The representation of each of these symbols is described as follows:

- Variables: These are required to be written as alphanumeric identifiers starting with capital letters or subscripted letters in some cases. Examples include, X, Xs, Y, X_7, etc.
- Constants: These are represented as numerals or alphanumeric identifiers starting with small-letters. Example are, y, x, none, alf, 14 etc.

- Functors: These are followed by alphanumeric identifiers starting with small-letters along with arity>0. To define the arity “n” of a functor “f”, it is sometimes followed in the form f/n.

- Quantifiers: ∀ for Universal Quantifier and ∃ for Existential Quantifier.

- Logical Connectives: ¬ for negation, ∧ for conjunction, ∨ for disjunction, ⊃ for implication and ⇔ for logical equivalence.

- Auxiliary Symbols: Parenthesis () and Commas “,”.

It should be noted that syntactic distinction will not be imposed between constants, predicate symbols, and functors. Conversely, we can use a, b, c, d, etc. to indicate constants and X, Y, Z to represent variables. Characters like, f, g, h can be used to denote Functors, whereas, p, q, r can be read as predicates.

Prolog employs declarative and procedural semantics. Declarative semantics refers to a logical description of what results the Prolog program should ideally produce. It is dependent upon knowledge of clauses as first-order formulae. On the other hand, procedural semantics can be defined as the abstract specification of how, and in which order, the program executes simpler operations to achieve an outcome. Both of these semantics are described by means of a central concept named “substitution”. A substitution is defined as a mapping of variables to the defined terms and can be used for the purpose of specializing clauses so that they meet the specified goals.

In Prolog, every variable can be invoked with any of the Prolog terms. The translation of Prolog clauses can be clearly understood due to the strong mapping from Prolog predicates to
atomic first-order formulae. For instance, in Prolog, ",-" can be interpreted as “if” meaning the implication from left to right. Moreover, the sub-goals separated with commas in the clause consist of Conjunctions. The empty head denoted by \(|\) (known as falsum) indicates the Prolog rules derived from Prolog queries. While transforming a clause for every variable \(X\) existing in the clause, one has to put \(\forall x\) in front of the resulting formula. We can consider the following example provided by Endriss (2007) to understand the formulation of Prolog clauses.

\[
\text{bigger (elephant, horse).}
\]

\[
\text{bigger (horse, donkey).}
\]

\[
\text{is\_bigger (X,Y) : - bigger (X,Y).}
\]

\[
\text{is\_bigger (X,Y) : - bigger (X,Z), is\_bigger (Z,Y).}
\]

Endriss (2007) provides the formulated version of the above example as,

\[
\{\text{bigger (elephant, horse),}
\]

\[
\text{Bigger (horse, donkey),}
\]

\[
\forall \ x. \ \forall y \ \text{bigger (x,y) } \rightarrow \text{is\_bigger (x,y),}
\]

\[
\forall \ x. \ \forall y. \ \forall z \ (\text{bigger (x,y) } \land \text{is\_bigger (z,y) } \rightarrow \text{is\_bigger (x,y)})\}
\]

5.7 Inductive Logic Programming

Inductive Logic programming (ILP) can be defined as the connection of logic programming and inductive learning. Subsequently, ILP involves both machine learning and logic programming techniques. ILP has two primary objectives (Muggleton and Raedt, 1994):

- To construct tools and techniques in order to form a hypothesis from observations.
- Derive new knowledge from understanding.
Inductive logic Programming covers the model and training of computational logic by means of exploring induction rather than deduction as the elementary mode of implication. While the present computational logic explains deductive inference from logic formulae given by the user, inductive logic programming theory explains the inductive implication of logic programs by making use of background knowledge. In this way, inductive logical programming may be effective for building and verifying programs.

5.7.1 Example of Inductive Logic Programming

Inductive implication is quite common in every-day reasoning. The following example reflects the inductive reasoning process through which relationships within a family can be determined. Consider a relationship of people in your family circle. Given that your grandfather is the father of one of your parents, and without knowing what a parent is, the following set of rules can be derived from this scenario:

\[ A = \{ \text{grandfather} (X,Y) \leftarrow \text{father} (X,Y), \text{parent} (Z,Y) \} \]

\[ A = \{ \text{father} (henry, jane) \leftarrow \} \]

\[ A = \{ \text{mother} (jane, john) \leftarrow \} \]

\[ A = \{ \text{mother} (jane,alice) \leftarrow \} \]

Following are the facts pointing towards the relationship between particular grandfather and his grandchildren i.e. positive fact.

\[ E^+ = \{ \text{grandfather} (henry, john) \leftarrow \} \]
\[ E^+ = \{\text{grandfather (henry, alice)} \}
\]

Similarly, below is a set of facts depicting what does not exist i.e. negative fact.

\[ E^- = \{\text{\textless grandfather (john, henry)}\} \]

\[ E^- = \{\text{\textless grandfather (alice, john)}\} \]

Now we are able to guess the following relationship on the basis of the facts given by \( E^+ \), and \( E^- \) and the information provided to A:

\[ H = \text{parent (X, Y)} \text{\textless mother (X, Y).} \]

5.8 Conclusion

Prolog is formulated in terms of relation, and its computations are performed by executing a set of queries over a relation. Prolog is preferred by many analysts because of its modular and flexible approach to tasks and its ability to greatly facilitate applications involving AI problems. The primary difference between Prolog and other programming languages is logical reasoning. While Prolog works logically in the presence of only some rules, others don’t. The conventional programming languages like C++ and Java are object oriented by nature while Prolog works on a declarative programming paradigm.
Section 6

Reasoning can be defined as a way to derive an inference on the basis of certain rules and principles by making use of a given approach or method. Induction, abduction, confirmation, deduction, and model-based reasoning are some of the popular methods of reasoning.

6.1 Monotonic and Non-Monotonic Logic

A set of clear and well-versed rules and principles for deriving conclusions is known as formal logic. Such conclusions can be made through mathematical logic by means of rules that can enable a computer to be programmed, to evaluate the accuracy of an argument, and to represent objects and relationships emblematically.

6.2 Examples

The following is an example of predicate logic and its derived conclusions:

All humans are mortal. Plato is a human.

Consequently, Plato is mortal

Predicate logic and the derivation of its related conclusions reflect the monotonic type of reasoning. In monotonic reasoning, we cannot withdraw any present claims and statements once we extend the group of statements.

Humans are not shown to follow this monotonic structure while reasoning:

- If we want to plan, we need to make inferences.
  
  One cannot expect all the possible outcomes of our plan.
  
  We need to prepare rules about something that we don’t know about.

The majority of formal logics have a monotonic nature, which indicates that one cannot reduce the set of derived conclusions by adding a formula to the concept. Briefly, logic is said to have
monotonic nature when the reality of a preposition or a statement does not tend to change with
the addition of new knowledge or statements. Traditional logic is of a monotonic classification.

In the mid-1970’s, Marvin Minsky and John McCarthy suggested that classical logic is
not sufficient to reflect the rationality of human reasoning. The underlying cause of their
discovery is that human reasoning is, by nature, non-monotonic. This clearly indicates that we
can make an inference on the basis of particular rules and principles which is rendered
impossible if certain other principles are added to our rules.

This non-monotonic reasoning results from the fact that our information about the real-
world remains incomplete. Hence, we are compelled to provide reason in the absence of
complete knowledge. For this purpose, we tend to review earlier inferences as soon as we gain a
new set of knowledge.

Listed below are a few shortcomings of monotonic logic for reasoning.

- Monotonic cannot handle abductive reasoning as the outcomes are only realized as the
  possibility of certain theory.

- Monotonic lacks the ability to review certainties as new knowledge may oppose old
  beliefs.

- Monotonic fails in reasoning by default because results can only be achieved in the
  absence of proof.

These drawbacks led to the birth of non-monotonic reasoning in the domain of AI along with its
various enactments.

A formal logic whose result is not monotonic by nature is known as non-monotonic logic.
If the reality of a preposition or statement tends to change with the addition of new knowledge,
then the relation is said to have non-monotonic nature. Generally, non-monotonic is a kind of
reasoning that draws conclusions on the basis of incomplete knowledge. With the availability of more information, one can revise prior conclusions. A classic example of this phenomenon is shown below:

If we know Tom is a cat, then consequently we would reach the conclusion that he can climb. Conversely, if we come to know that Tom is a bird, then we would definitely withdraw the previously made conclusion. This kind of relation is said to have non-monotonic nature as the set of possible conclusions cannot be extended monotonically with the increase of information. It should be noted that all non-monotonic natured reasoning is associated with constancy and contradiction can be overcome by eliminating the related outcomes that occur from the defined rules and principles. Consider the following example:

The truth value of the statement “Tom is a cat” can be taken as “true” by default as we know that “Cats naturally can climb”. This fact would result in the conclusion that “Tom can climb”. This shows that, if a contradiction is found in a preposition, then only the truth value of the last type would be transformed.

6.3 Default Reasoning

One of the most common types of non-monotonic reasoning is default reasoning. In this, we want to draw conclusions based on what is almost certainly true.

A novel inference rule is initiated by default logic:

\[ \frac{X \rightarrow Y}{Z} \]

which asserts if X is deducible as well as uniform, we may suppose Y, then conclude Z.

This method is analogous to non-monotonic logic; however, there are certain differences:

- In order to compute the set of plausible extensions, novel inference rules are employed
In default logic, whichever non-monotonic expressions are rules of inference instead of expressions.

6.4 Extensions to Fundamental Logic

Generally, logics use *Monotonic Reasoning*. In this type of logic expression operation, once a truth is derived or a value assigned to a variable, it cannot be taken back or changed. Non-monotonic reasoning is more intricate and executes processes in order to maintain track of the dependency which expressions possess on each variable with the intention that alterations in value can be spread by means of the decision space.

6.5 Extension of First Order Predicate Logic

This is fundamentally an augmentation of 1st order predicate logic to incorporate a modal operator, $H$. The intention of this is to permit constancy. Let's observe an example:

$y: \text{plays	extunderscore tool}(y) \land H \text{improvises}(y) \rightarrow \text{jazz	extunderscore instrumentalist}(y)$

The aforementioned example asserts that for all $y$, $y$ plays a tool. If it is also stated that $y$ can improvise, and that statement is factually supported by a related, complete knowledge set, then we can conclude that $y$ is a jazz instrumentalist.

6.6 Defining Consistency

One general solution (compatible with PROLOG notation) is to present that fact $Z$ is true in an effort to verify $\neg Z$. If we do not succeed, we possibly will articulate that $Z$ is constant (in view of the fact that $\neg Z$ is false).

Nevertheless, think about the renowned set of assertions associating to President Nixon.

$y: \text{Republican}(y) \land H \neg \text{Pacifist}(y)$

$y: \text{Quaker}(y) \land H \text{Pacifist}(y)$

53
At this time, this asserts that Quakers are likely to be pacifists and Republicans are likely not to be pacifists.

However, Nixon was both a Quaker and a Republican; thus we could state:

- Quaker (Nixon)
- Republican (Nixon)

Now, our entire set of knowledge has been rendered incompatible.

### 6.7 Abstract Consequence Relations

Conventional 1\textsuperscript{st} order logic is monotonic; if a sentence $\varphi$ can be deduced in 1\textsuperscript{st} order logic from a set $\Gamma$ of premises, after that it can also be deduced from any set $\Delta$ of premises expanding $\Gamma$. In other words, 1\textsuperscript{st} order logic gives a relation $\models$ of logical outcome flanked by sets of premises as well as particular sentences together with the property that if $\Gamma \models \varphi$ and $\Gamma \subseteq \Delta$ subsequently $\Delta \not\models \varphi$. This follows instantaneously from the type of the relation $\models$, for $\Gamma \models \varphi$ holds exactly when $\varphi$ is true for every analysis on which the entire sentences in $\Gamma$ are true.

From an abstract point of view, take into account the formal properties of a consequence relation. Let $\models$ be any relation among sets of premises and particular sentences. The given properties are all fulfilled through the consequence relation $\models$ of 1\textsuperscript{st} order logic:

- **Supraclassicality**: if $\Gamma \models \varphi$ then $\Gamma \models \varphi$
- **Reflexivity**: if $\varphi \in \Gamma$ then $\Gamma \models \varphi$
- **Cut**: If $\Gamma \models \varphi$ and $\Gamma, \varphi \models \psi$ then $\Gamma \models \psi$
- **Monotony**: If $\Gamma \models \varphi$ and $\Gamma \subseteq \Delta$ then $\Delta \models \varphi$

The following two properties have been considered in the literature, where $\models$ is a random consequence relation:

- **Cautious Monotony**: If $\Gamma \models \varphi$ and $\Gamma \models \psi$, then $\Gamma, \varphi \models \psi$
Rational Monotony: If it is not the case that $\Gamma \vdash \neg \varphi$, and furthermore $\Gamma \vdash \neg \psi$, then $\Gamma, \varphi \vdash \psi$

A non-monotonic consequence relation is a relation that fulfills Supraclassicality, Reflexivity, Cut, and Cautious Monotony. According to Gabbay (1985), generally it has been identified that Reflexivity, Cut as well as Cautious Monotony are crucial properties for any good non-monotonic consequence relation.

6.8 Skeptical

A discrete concern from the conventional properties of a non-monotonic consequence relation, however one i.e. strictly entangled with it, is the problem of how conflicts between possible infeasible conclusions are to be dealt. Within a given non-monotonic structure, there are two distinct types of conflicts that can take place:

- Conflicts flanked by infeasible conclusions as well as “hard facts,” some of which maybe recently learned

- Conflicts flanked by one probable infeasible conclusion and another (several formalisms, for example, offer some kind of infeasible inference rules, and such rules possibly will hold contradictory conclusions)

When a conflict (of any type) comes up, steps need to be taken in order to reinstate evenness. All non-monotonic logics deal with conflicts of the first type in the similar manner: without a doubt, it is the essence of infeasible reasoning that conclusions can be withdrawn when novel facts are learned. However, conflicts of the second type can be dealt in two diverse modes: one can derive inferences either in a cautious manner (also referred to as skeptical). These two choices represent considerably diverse modes to interpret a given body of infeasible knowledge, and generate diverse outcome.
6.9 Conclusion

With respect to the issues of conventional logic, non-monotonic logic seems to be persistently problematic. Besides default logic, other types of non-monotonic reasoning are far-away from immune to the combinatorial sources of intractability. However, some imperative work has been made, with a lot more progress required.
Section 7

7.1 Result

Initially, the software returns symptoms diagnoses. For example, when a patient submits that he or she is experiencing a cough, a result for chest infection returns. In the same way, a symptom which does not match a rule for the program registers as an unknown disease. After that, decisions are made by the program which is based solely on the patient’s symptoms. In order to generate a logical conclusion, no outside information is used. As a final point, the diagnosis is made based on the response of the user to symptom related inquiries. For example, it asks the patient, “Do you have a cough?” and the patient provides the answer “Yes or No”. A ‘Yes’ response generates the disease chest infection. In our research method, the diagnosis is based solely on the patient’s yes/no responses and not on any other scheme. We have analyzed and verified that the results obtained from this diagnosis system are both correct and constructive.

7.2 Summary and Conclusion

Expert systems technology is a rising field of computer science which is used in diverse applications in numerous regions. Organizations are making use of expert systems to aid experts or complete their work under those circumstances in which the expert is not present.

Up until now, the most productive and useful application of expert systems has been in the field of medical science. A potential cause for the broad application in this field is that a large amount of medical applications have been analytic in nature. In analyzing or diagnosing medical issues of a patient or facilitating the analysis of a medical test, the expert system can help a
medical doctor. In addition, applications of expert systems in the medical sciences are expected to be more enhanced in the near future.

This paper has provided a concise conceptualization of an innovative technology and has explained its application in the domain of medical science. We have shown the design and development of a medical system for the diagnosis of a disease. Symptoms, as well as illness linked with patients, have been the origin of this research. In order to determine symptoms, each new patient is asked a yes/no question and his responses are recorded. The diagnosis works by matching symptoms of identified illness with previously recorded responses in a decreasing level of complexity. Inductive reasoning is used to update illness symptoms.

7.3 Future Work

This research can be considered a basis for future work. Other learners can extend this work as there are scores of potential to continue this research. This diagnostic system possesses a functional understanding of symptoms. However, the system is not capable of recommending treatment. This system can be extended by including diverse exercise control plans, diet charts, and so forth. Moreover, it can be developed into a full-fledged diagnosis expert system that can analyze whether the patient’s condition is stable or critical. Afterward, it would be an outstanding supplementary tool for analysis, treatment, and education. It can be made more accessible to users by transforming it from a desktop based application to a web based application. This diagnosis expert system tool can be incorporated with different high level languages like Java.

Because of time and space restrictions, we were unable to grant extensive consideration to alternative inductive learning models. Such models include automata induction, in which
knowledge is symbolized in the shape of nite-state automata (Parekh & Honavar, 1997). Furthermore, evolutionary algorithms may be applied toward the pursuit of target ideas for suitably expressed concept spaces (e.g. neural networks, decision trees, LISP programs, etc.) (Goldberg, 1989; Holland, 1992; Mitchell, 1996).
7.4 Code

:- dynamic known_answers/3.
:- dynamic patient_record/2.
:- dynamic illness/2.

%------------------
% symptom
% if you want to add more symptoms, just add more predicates below
%------------------
symptom(fever).
symptom(cough).
symptom(shivering).
symptom(runny_nose).

%------------------
% new_patient/1
% starts a new session
% It must be initiated with a patient name.
% This goal clears all known_answers,
% and starts the examine subgoal followed by diagnosis subgoal.

%------------------
new_patient(PatientName) :- not(patient_already_exists(PatientName)),
    retractall(known_answer(_,_,_)), write("Starting examination...

(examine, confirmed_symptoms(S), write("Confirmed symptoms are "),
    write_term(S, []), write(""), assert(patient_record(PatientName,S)), !,
    write("Determining illness...

(diagnose(S,I), !, write("AI diagnosed that you have "),
    write_term(I, []), !.

%%%%%%%%%%%%%%%%
% symptoms/1
% returns a list of all symptoms
symptoms(L) :- findall(X, symptom(X), L).

% examine/0
% starts the examination process
% by asking for a yes/ no question against each symptom
examine :- symptoms(L), check_symptoms(L).

diagnose(PS, I) :- length(PS, MustMatchCount), diagnose(PS, MustMatchCount, I).

diagnose(_, unknown_disease).

diagnose/3
% recursively matches the illness symptoms and patient symptoms with decreasing number of matches
% PS: Patient Symptoms (expected to be passed as a parameter)
% I: Illness (expected to be unified)
% MustMatchCount: The number of symptoms that should exist in Illness
diagnose(PS, MustMatchCount, I) :- (illness(I,S), length(S, MustMatchCount),
  subset(S,PS),! ); (MustMatchCount > 1, NewCount is MustMatchCount-1,
  diagnose(PS, NewCount, I) ).

%------------------
% check_symptoms/1
% given a list of symptoms, ask questions
%------------------

check_symptoms([]) :- !.
check_symptoms([H|T]) :- ask(symp,tom,H), check_symptoms(T).

%------------------
% confirmed_symptoms/1
% returns a list of symptoms for which the answer is yes
%------------------

confirmed_symptoms(C) :- findall(X,known_answer(yes,symptom,X),C).

%------------------
% ask/2
% given an attribute and a value, gets a yes/no answer from the user
% It works by writing a prompt and having a subgoal to assert the answer
%------------------

ask(Attr,Val) :- write(Attr:Val), write(‘?’), read(Y), asserta(known_answer(Y,Attr,Val)).

%------------------
% fix_diagnosis/2
% learns "actual" illness of a "patient" and improves the learning process
%------------------

fix_diagnosis(PatientName, ActualIllness) :- patient_record(PatientName, PS),
  write('Confirmed patient symptoms '), write_term(PS, []),
  write(' will be related to '), write_term(ActualIllness, []), !, update_definition(ActualIllness, PS, FS),!, write('\nNew definition is '), write_term(FS,[]).
% ------------------
% update_definition/3
% given an illness and new symptoms, returns the updated symptoms for that illness
% Following case is when the illness is not already defined.
% ------------------
update_definition(Illness, RelateSymptoms, RelateSymptoms) :- not(illness(Illness, _)),
write('nThere was no earlier definition of '), write_term(Illness, []), assert(illness(Illness, RelateSymptoms)),!.
% ------------------
% update_definition/3
% given an illness and new symptoms, returns the updated symptoms for that illness
% Following case is when the illness is already defined, and hence takes an intersection of old and new symptoms
% ------------------
update_definition(Illness, RelateSymptoms, FinalSymptoms) :- illness(Illness, OldSymptoms),
write('nEarlier definition of '), write(Illness), write(' was '),
illness(Illness, OldSymptoms), write_term(OldSymptoms, []),
intersection(OldSymptoms, RelateSymptoms, FinalSymptoms), retractall(illness(Illness, _)),
assert(illness(Illness, FinalSymptoms)).
% ----------------------
% rediagnose/1
% given a patient name, rechecks diagnosis based on existing symptoms
% This goal could be requested, for example, when illness predicates are updated
% ----------------------
rediagnose(PatientName) :- patient_record(PatientName, C), write('nConfirmed symptoms were '),
write_term(C, []), write('nRediagnosing...'), diagnose(C, NewIllness),!.
write('nUpdated diagnosis is that Patient '), write_term(PatientName, []),
63
write(' is having '),write_term(NewIllness, []).

%-------------------------------

% patient_already_exists/1
% true if given patient name is already in the patient records
%-------------------------------

patient_already_exists(PatientName) :- patient_record(PatientName, _),
   write('Patient '),write_term(PatientName, []), write(' already exists.
').

%-------------------------------

% show_patient_records/0
% shows all patient records
%-------------------------------

show_patient_records :- findall((P, S), patient_record(P, S), L), !, show_records(L).

%-------------------------------

% show_records/1
% Calls show_record for each (PatientName, Symptom) pair
%-------------------------------

show_records([]).

show_records([(P, S)|T]) :- show_record(P, S), show_records(T).

show_record(P, S) :- diagnose(S, I),!, write_term(P, []), write(' has symptoms '),
   write_term(S, []), write(' and diagnosed '),
   write_term(I, []), write('n').

%-------------------------------

% change_diagnosis/2
% Associate the symptoms from one illness to another for a given patient
%-------------------------------

change_diagnosis(Patient, NewIllness) :- patient_record(Patient, Symptoms),!.
write_term(Patient, []), write(' has symptoms '),
write_term(Symptoms, []), diagnose(Symptoms, OldIllness),
write(' and was diagnosed '), write_term(OldIllness, []),
write('
'), !, write('Changing it to '),
write_term(NewIllness, []), write('
'),
retractall(illness(NewIllness, _)), retractall(illness(OldIllness, _)),
assert(illness(NewIllness, Symptoms)).
References


Appendix

1 ?- new_patient(kumeel).
Starting examination...
symptom:fever? yes.
symptom:cough? no.
symptom:shivering? no.
symptom:runny_nose? no.

Confirmed symptoms are [fever]
Determining illness...
AI diagnosed that you have unknown_disease
true.

2 ?- fix_diagnosis(kumeel,celluities).
Confirmed patient symptoms [fever] will be related to celluities
There was no earlier definition of celluities
New definition is [fever]
true.

3 ?- new_patient(john).
Starting examination...
symptom:fever? yes.
symptom:cough? no.
symptom:shivering? yes.
symptom:runny_nose? no.

Confirmed symptoms are [shivering,fever]
Determining illness...
AI diagnosed that you have celluities
true.

4 ?- fix_diagnosis(john,bacteremia).
Confirmed patient symptoms [shivering,fever] will be related to bacteremia
There was no earlier definition of bacteremia
New definition is [shivering,fever]
true.