CLIPS: An Expert System Tool for Medical Applications

[1]Nkuma-Udah K.I.*, [2]Chukwudebe G.A. and [3]Ekwonwune E.

1. Department of Biomedical Technology, Federal University of Technology, Owerri, Nigeria
2. Department of Electrical / Electronic Engineering, Federal University of Technology, Owerri, Nigeria
3. Department of Computer Science, Imo State University, Owerri, Nigeria

(Received November 12, 2019; Revised December 14, 2019; Accepted December 19, 2019)

Abstract

CLIPS stands for C Language Integrated Production System. It is a programming tool, designed to facilitate the development of software to model human knowledge or expertise for an application, say in a medical diagnosis. CLIPS program is usually used by reason of the flexibility, the expandability and the low cost. The idea of expert systems came from the fact that human experts are able to solve problems at a high level because they exploit knowledge about their area of expertise. So an *expert system* is a computer system that emulates the decision making ability of a human expert. It uses human knowledge to solve problems that would require human intelligence. The first successful and largest area of applications of artificial intelligence was the expert systems. And one of the fields where the expert systems had these successful and largest applications to solving real-world problems was in medicine. Expert systems in medical practice function as decision-support systems and are currently being applied to a number of medical domains, mainly in diagnosis and treatment plan. CLIPS, developed at NASA's Johnson Space Center, 1985 is useful for creating expert systems and other programs where a heuristic solution is easier to implement and maintain than an algorithmic solution. The key features of CLIPS are *Knowledge Representation*, *Portability, Integration/Extensibility*, *Interactive Development*, *Verification/Validation*, *Fully Documented and* low cost. Some of the medical applications that already apply the CLIPS include those applied for diagnosing eye diseases. Diabetes, skin diseases and those for reducing medical errors.

***Keywords**: CLIPS, Expert systems, Artificial intelligence, Medical, Inference engine, Knowledgebase, User interface*

1. Introduction

CLIPS stands for C Language Integrated Production System. It is a productive development and delivery computer expert system tool which provides a complete environment for the construction of rule and/or object based expert systems. CLIPS programming tool is designed to facilitate the development of software to model human knowledge or expertise for an application, say in a medical diagnosis. CLIPS program is used by reason of the flexibility, the expandability and the low cost [1].

Unlike conventional programming languages, such as FORTRAN and C, designed and optimized for the procedural manipulation of data (such as numbers and arrays), CLIPS technology allows the modeling of information at higher levels of abstraction [2]. And humans often solve complex problems using very abstract, symbolic approaches which are not well suited for implemen0tation in conventional languages. Therefore because, CLIPS and other like programming languages emulate human expertise in well defined problem domains, they are called expert systems.

Now, an *expert system* is a computer system that emulates the decision making ability of a human expert [3]. By so doing, it acts in all respects like a human expert. It uses human knowledge to solve problems that would require human intelligence. The expert system represents an expertise as data or rule

*Corresponding author email: kenneth.nkumaudah@futo.edu.ng within the computer. These rules and data can be called upon when needed to solve problems. Expert system tools allow programs to be built that closely resemble human logic in their implementation and are therefore easier and cheaper to develop and maintain.

Expert System is itself a major aspect of *artificial intelligence,* which is the branch of computer science concerned with making computers behave like humans [4], i.e., concerned with automation of intelligent behavior. Artificial intelligence (AI) is the intelligence of machines and the branch of computer science concerned with making computers behave like humans [5][6][7]. The field of artificial intelligence attempts to understand intelligent entities. Common areas of applications of Artificial Intelligence include the following [8]: Computer Vision, Game Playing, Speech recognition, Natural Language Processing, Robotics and Expert systems.

As an expert system tool, CLIPS was developed using the rule-based programming, which is one of the most commonly used techniques for developing expert systems [3]. In this programming paradigm, rules are used to represent heuristics, or "rules of thumb," which specify a set of actions to be performed for a given situation. A rule is composed of an *if* portion and a *then* portion. The *if* portion of a rule is a series of patterns which specify the facts (or data) which cause the rule to be applicable. The process of matching facts to patterns is called pattern matching. The expert system tool provides a mechanism, called the inference engine, which automatically matches facts against patterns and determines which rules are applicable.

The medical field was one of the first testing grounds for the Expert System technology. A classic expert system in the medical field is the MYCIN, which is actually a great breakthrough in Expert Systems [9]. Other notable examples of medical expert systems include NURSEx.

Given the high cost of human experts, the computer expert systems can be taken to an advantage. So dependence on the human expert can be minimized if his/her expertise can be transferred into a computer system. The expert system seeks and utilizes relevant information from their human users and from available knowledge bases in order to make recommendations [6].

2. Overview of the Expert Systems

The idea of expert systems came from the fact that human experts are able to solve problems at a high level because they exploit knowledge about their area of expertise [10]. Therefore, the design of programs with *expert-based* problem solving capabilities that use specific knowledge about a specialty area in order to obtain competence comparable to that of a human expert became an innovative. The specific knowledge may be accumulated in form of a database obtained by interviewing one or more experts in the area in question.

An *expert system* is a computer system that emulates the decision making ability of a human expert [11]. By so doing, it acts in all respects like a human expert. It uses human knowledge to solve problems that would require human intelligence. The expert system represents an expertise as data or rule within the computer. These rules and data can be called upon when needed to solve problems. Expert System is itself a major aspect of *artificial intelligence,* which is the branch of computer science concerned with making computers behave like humans [12], i.e., concerned with automation of intelligent behavior.

2.1 Expert Systems and Artificial Intelligence

Artificial intelligence (AI) is the intelligence of machines and the branch of computer science that aims to create it. In other words, AI is the branch of computer science concerned with making computers behave like humans [13][14][15]. The field of artificial intelligence attempts to understand intelligent entities. Thus, one reason to study it is to learn more about ourselves. But unlike philosophy and psychology, which are also concerned with intelligence, AI strives to *build* intelligent entities as well as understand them [16].

Common areas of applications of Artificial Intelligence include the following [17]: Computer Vision, Expert systems, Game Playing, Speech recognition, Natural Language Processing and Robotics. *Computer vision* is the artificial intelligence field that includes methods for acquiring, processing, analyzing, and understanding images [18][19][20].

Expert system as an application area of artificial intelligence is a computer system that emulates the decision-making ability of a human expert [1]. Artificial intelligence *game playing* refers to techniques used in computer games to produce the illusion of intelligence in the behaviour of non-player character (NPCs). Speech recognition in artificial intelligence is the translation of spoken words into text [21]. *Natural Language Processing*, NPL is the area of application of artificial intelligence focused on developing systems that allow computers to communicate with people using everyday language [22]. Artificial Intelligence *robot* is a mechanical creature which can function autonomously.

Artificial Intelligence may be subdivided into two main branches or aspects. The first branch, *cognitive science*, has a strong affiliation with psychology. The goal is to construct programs for testing theories that describe and explain human intelligence. The second branch, *machine intelligence*, is more computer science oriented and studies how to make computers behave intelligently. Expert systems fall in the later branch [23].

2.2 Expert Systems and Human Experts

Expert systems were the first successful and largest area of applications of Artificial Intelligence to real-world solving problems in medicine, chemistry, finance and even in space (examples, space shuttle, robots on other planets) [24]. In the simplest sense, Artificial Intelligence is the study of developing computer programs which exhibit human-like intelligence [25]. Early artificial intelligence researchers focused on such problems as game theory, robotic control, and vision systems [43].

Common to each of these problems was research into ways of representing and reasoning with knowledge, in a computer, in a fashion similar to humans. Thus artificial intelligence researchers used human experts for their source of problem-solving knowledge. They reasoned that by virtue of being an expert, the human possesses unique talents, made possible by the human's knowledge and problem solving skills on a particular subject. They developed intelligent computer programs as a result. And because of the nature of these intelligent computer programs, they were aptly called expert systems [27].

The expert system program models the following characteristics of the human expert [25]: knowledge, reasoning, conclusions and explanations. The expert system models the knowledge of the human expert, both in terms of content and structure. Reasoning is modeled by using procedures and control structures which process the knowledge in a manner similar to the expert. Conclusions given by the system must be consistent with the findings of the human expert. The expert system also provides explanations similar to the human expert.

One of the principal attractions of expert systems is that they enable computers to assist humans in many fields of endeavor with the processes of analyzing and solving complex problems. This is accomplished by encoding in the expert system the knowledge and problem-solving skills of a human expert. This expert computer program can then be used by others to obtain and use this expertise for solving a current problem that would have previously required the expert to be present.

2.3 Expert Systems and Medical Practice

As mentioned previously, the first successful and largest area of applications of artificial intelligence was the expert systems. And one of the fields where the expert systems had these successful and largest applications to solving real-world problems was in medicine [24]. In the medical practice, expert system seeks to exploit the specialized skills or information held by a group of people on specific areas, in this case the medical areas. By so doing, an expert system can be thought of as a computerized consulting service. Such a system, for example, can be used for prospecting medical diagnosis or as educational aids.

Expert systems in medical practice function as decision-support systems. They are currently being applied to a number of medical domains, mainly in diagnosis and treatment plan. In this situation, they function to assist the medical practitioner in making his final decision by giving ready access to the levels of skill shown by experts in a particular medical sub-field. Enormous research effort has been put into expert systems for medical applications, although only a few of this effort have been felt in routine medical use [28].

2.4 Expert Systems Architecture

An expert system, as a computer program, is able to draw conclusions and make decisions, based on knowledge, represented as the database, it has. The computer expert system has two main advantages: one, there is a need for systems that perform their functions faster and better than a human is able to do, and working in real-time. Two, computer programs are much more cheaper than human experts, especially in terms of maintenance, costs of education, salaries etc. If there is a way to duplicate a part of knowledge a human expert has, it is economical to do that using a computer program [29].

Expert systems usually consist of two core parts in its kernel: a knowledge base and an inference engine [25]. A knowledge base represents knowledge in certain domain and stores all the facts and rules about a particular problem in that domain. An inference engine is a set of algorithms, which perform judgment and reasoning. For a user to interact with the expert system, there has to be an additional part: the user interface, which is a means of communication between a user and the expert system problem-solving processes. Thus, the entire expert system architecture in the most simplified version actually consists of three parts: a knowledge base, an inference engine and a user interface (Figure 1).

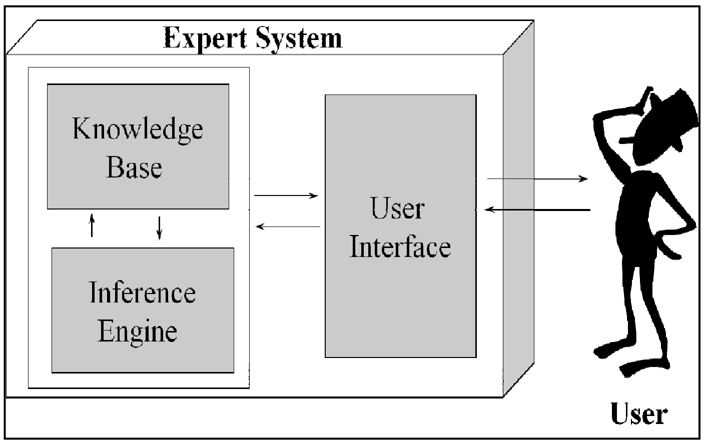

Figure 1: Expert System Architecture

3. The CLIPS Programming Tool

CLIPS is a multiparadigm rule-based programming language that provides support for Rule-based, Object-oriented and Procedural programming [2]. It was developed at NASA's Johnson Space Center from 1985 to 1996. It is useful for creating expert systems and other programs where a heuristic solution is easier to implement and maintain than an algorithmic solution. Written in C for portability, CLIPS can be installed and used on a wide variety of platforms. Since 1996, CLIPS has been available as a public domain software.

3.1 Main Features of CLIPS

The key features of CLIPS are [2]:

• *Knowledge Representation*: CLIPS provides a cohesive tool for handling a wide variety of knowledge with support for three different programming paradigms: rule-based, object-oriented and procedural. Rule-based programming allows knowledge to be represented as heuristics, or "rules of thumb," which specify a set of actions to be performed for a given situation. Object-oriented programming allows complex systems to be modeled as modular components (which can be easily reused to model other systems or to create new components). The procedural programming capabilities provided by CLIPS are similar to capabilities found in languages such as C, Java, Ada, and LISP.

• *Portability:* CLIPS is written in C for portability and speed and has been installed on many different operating systems without code changes. Operating systems on which CLIPS has been tested include Windows, Macintosh, and Unix. CLIPS can be ported to any system which has an ANSI compliant C or C++ compiler. CLIPS comes with all source code which can be modified or tailored to meet a user's specific needs [30].

• *Integration/Extensibility*: CLIPS can be embedded within procedural code, called as a subroutine, and integrated with languages such as C, Java, FORTRAN and ADA. CLIPS can be easily extended by a user through the use of several well-defined protocols.

• *Interactive Development*: The standard version of CLIPS provides an interactive, text oriented development environment, including debugging aids, on-line help, and an integrated editor.

•*Verification/Validation*: CLIPS includes a number of features to support the verification and validation of expert systems including support for modular design and partitioning of a knowledge base, static and dynamic constraint checking of slot values and function arguments, and semantic analysis of rule patterns to determine if inconsistencies could prevent a rule from firing or generate an error.

•*Fully Documented*: CLIPS comes with extensive documentation including a Reference Manual and a User's Guide.

• *Low Cost*: CLIPS is maintained as public domain software.

3.2. The Expert System Tool in CLIPS

Characteristically, CLIPS expert systems may be executed in three ways: interactively using a simple, text-oriented, command prompt interface; interactively using a window/menu/mouse interface on certain machines; or as embedded expert systems in which the user provides a main program and controls execution of the expert system [3].

The generic CLIPS interface is a simple, interactive, text-oriented, command prompt interface for high portability. The standard usage is to create or edit a knowledge base using any standard text editor, save the knowledge base as one or more text files, exit the editor and execute CLIPS, then load the knowledge base into CLIPS [4]. The interface provides commands for viewing the current state of the system, tracing execution, adding or removing information, and clearing CLIPS. CLIPS is called an expert system tool because it is a complete environment for developing expert systems which includes features such as an integrated editor and a debugging tool [5]. The CLIPS shell provides the basic elements of an expert system, performing *inferences* or reasoning.

The basic elements of a CLIPS expert system include [26] (figure 2):

1. fact-list, and instance-list: these are global memory for data
2. knowledge-base: contains all the rules, the rule-base
3. inference engine: controls overall execution of rules

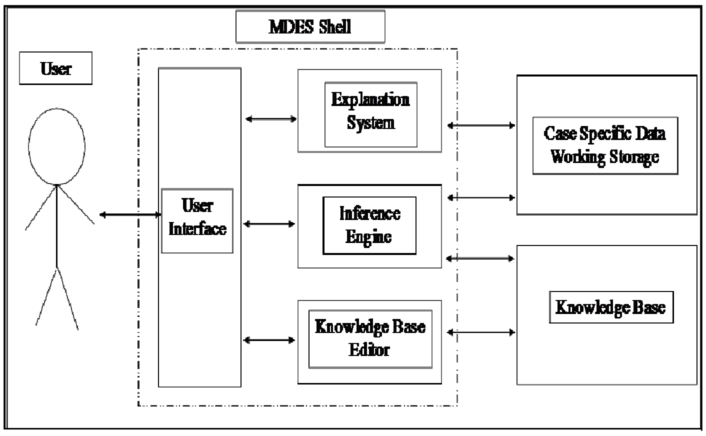

Figure 2: Basic Elements of a CLIPS Expert System

A program written in CLIPS may consist of *rules, facts*, and *objects*. The inference engine decides which rules should be executed and when. A rule‑based expert system written in CLIPS is a data-driven program where the facts, and objects if desired, are the data that stimulate execution via the inference engine [27].

This is one example of how CLIPS differs from procedural lan-guages such as Pascal, Ada, BASIC, FORTRAN, and C. In procedural languages, execution can proceed without data. That is, the statements are suffi-cient in those languages to cause execution. For example, a statement such as PRINT 2 + 2 could be immediately executed in BASIC. This is a com-plete statement that does not require any additional data to cause its execution [28]. However, in CLIPS, data are required to cause the execution of rules.

3.3 CLIPS Fields and Notations

In CLIPS, a *field* is a placeholder (named or unnamed) that may have a value associated with it. As a simple analogy, you can think of a field as a picture frame. The frame can hold a picture, perhaps a picture of a pet duck. A fact such as (duck) or (quack) is said to consist of a single *field* [31]. The (duck) fact has a single, unnamed placeholder for the value duck. The *order* of unnamed fields is significant. For example, if a fact was defined: (Brian duck) and interpreted by a rule as the hunter Brian shot a duck, then the fact: (duck Brian) would mean that the hunter duck shot a Brian. In contrast, the order of named fields is not significant.

A *list* is a group of items with no implied order. An *ordered* list means that the position in the list is significant. A *multifield* is a sequence of fields, each of which may have a value. The examples of (duck Brian) and (Brian duck) are multifield facts [32]. If a field has no value, the special symbol *nil*, which means "nothing" may be used for an empty field as a placeholder. For example:(duck nil) would mean that the killer duck bagged no trophies today.

There are about eight (8) different types of fields in CLIPS: *float, integer, symbol, string, external address, fact address, instance name and instance address* [33]. The type of each field is determined by the type of value stored in the field. In an unnamed field, the type is determined implicitly by what type you put in the field. In deftemplates, you can explicitly declare the type of value that a field can contain. A symbol is one type of field that starts with a printable ASCII character and is followed optionally by zero or more printable characters. Fields are commonly delimited or bounded, by one or more spaces or parentheses.

CLIPS is case-sensitive [34] and certain characters have special meaning to CLIPS: " ( ) & | < ~ ; ? $. The "&", "|", and "~" may not be used as stand-alone symbols or as *any* part of a symbol. Some characters act as delimiters by *ending* a symbol. These include:

- any non-printable ASCII character, including spaces, carriage returns, tabs, and linefeeds
- double quotes, **"**
- opening and closing parentheses, ( )
- ampersand, **&**
- vertical bar, **|**
- less than, **<**. Note that this may be the *first* character of a symbol
- tilde, **~**
- semicolon, **;** indicates start of a comment, a carriage return ends it
- ? and $? may not begin a symbol but may be inside it.

3.4 Facts and Templates in CLIPS

In CLIPS, a ***fact*** consists of one or more fields enclosed in matching left and right parentheses. Exceptions are certain functions such as assert and retract which only apply to facts, not objects. A fact may be *ordered* or *unordered.* Example of ordered facts:

```
f-5    (coordinates 1 2 3)
f-6    (coordinates 1 3 2)
```

Ordered facts *must* use field position to define data [35]. As an example, the ordered fact (duck Brian) has two fields and so does (Brian duck). –However, these are considered as two separate facts by CLIPS because the order of field values is different. In contrast, the fact (duck-Brian) has only one field because of the "-" concatenating the two values.

Deftemplate facts are unordered because they use named fields to define data [36]. This is analogous to the use of records in Pascal and other languages. Multiple fields normally are separated by *white space* consisting of one or more spaces, tabs, carriage returns, or linefeeds. For example, if the following examples are entered as shown and it will be seen that each stored fact is the same.

```
CLIPS> (clear)
CLIPS> (assert (The duck says "Quack."))
<Fact-0>
CLIPS> (facts)
f-0    (The duck says "Quack.")
For a total of 1 fact.
CLIPS> (clear)
CLIPS> (assert (The duck says "Quack." ))
<Fact-0>
CLIPS> (facts)
f-0    (The duck says "Quack.")
For a total of 1 fact.
CLIPS>
```

Carriage returns may also be used to improve readability [37]. In the following example, a carriage return is typed after every field and the asserted fact is the same as before when the fact was entered on one line.

```
    CLIPS> (clear)
CLIPS> (assert (The
duck
says
"Quack"))
<Fact-0>
CLIPS> (facts)
f-0     (The duck says "Quack")
For a total of 1 fact.
CLIPS>
```

Although you can accomplish a lot with simple facts, in many cases it is desirable to link bits of information together. Consider, for example, an application which is to try to determine the general fitness of a number of people. Several indicators [38] will be used, so each person will have a different value for each one. We could express the information for two people as follows:

```
(age Andrew 20)
(weight Andrew 80)
(height Andrew 188)
(blood-pressure Andrew 130 80)
(age brenda 23)
(weight brenda 50)
(height brenda 140)
(blood-pressure brenda 120 60)
```

But this involves lots of separate facts, with nothing to link them together other than a single field bearing the person's name. A better way to do this is by using the *deftemplate* structure, thus:

```
(deftemplate personal-data
      (slot name)
      (slot age)
      (slot weight)
      (slot height)
      (multislot blood-pressure)
)
```

deftemplate does not create any facts, but rather the form which facts can take. Every time a fact of this type is created, it contains the slots specified in its definition, each of which can contain a value and can be accessed by name. Each person's data can now be asserted thus:

```
(assert (personal-data (name Andrew) (age 20) (weight 80)
      (height 188) (blood-pressure 130 80)))
```

or, in a deffacts structure thus:

```
(deffacts people
  (personal-data (name Andrew) (age 20) (weight 80)
           (height 188) (blood-pressure 130 80))
     (personal-data (name Cyril) (age 63) (weight 70)
           (height 1678) (blood-pressure 180 90)))
```

Although you don't have to specify all the information, so

```
(assert (personal-data (weight 150) (age 23) (name Brenda)))
```

is perfectly valid. The order in which you access the slots does not matter, as you are referring to them by name (this also allows you to set as few of them as you wish). Template facts can be altered without the need to retract them and assert a new version [39]. The function **modify** allows you to change the value of one or more slots on a fact. Suppose it's Andrew's birthday. If we define the rule

```
(defrule birthday
      ?birthday <- (birthday ?name)
      ?data-fact <- (personal-data (name ?name) (age ?age))
=>
      (modify ?data-fact (age (+ ?age 1)))
      (retract ?birthday)
)
```

then asserting the fact **(birthday Andrew)** will modify Andrew's age while leaving all his other personal data intact. Incidentally, the reason we retract the birthday fact is for the purposes of truth maintenance. It's only Andrew's birthday once a year, and if we left the fact lying around it would soon become false. Further, every time any other part of Andrew's personal data (weight, for example) was changed the birthday rule would be fired again, causing rapid ageing!

Such systems are often used as a support when a human cannot collect all vital information due to their amount or complexity.

4. Selected CLIPS Medical Applications

Some of the medical applications that already apply the CLIPS include the following:

- An Expert System for Diagnosing Eye Diseases
- An Expert System for Diabetes Diagnosis
- Design of Expert System for Search Allergy and Selection of the Skin Tests
- Development of an Expert System for Reducing Medical Errors

4.1 An Expert System for Diagnosing Eye Diseases

Here, the design of the expert system using CLIPS was done with the aim to provide the user with background for suitable diagnosis of some of the eye diseases [39]. There are many disease states that may produce symptoms from the eye. The CLIPS language is used as a tool for designing this expert system for diagnosing eye diseases. The system performs many functions. It will conclude the eye disease diagnosis based on answers of the user to specific question that the system asks. The questions provide the system for explanation for the symptoms of the patient that helps the expert system for diagnosing the disease by inference engine. It stores the facts and the conclusion of the inference of the system, and the user, for each case, in data base. It processes the data base in order to extract rules, which complete the knowledge base.

4.2 An Expert System for Diabetes Diagnosis

Diabetes is a serious disease that affects almost every organ in the body like heart, eyes, kidney, skin, nerves, blood vassals, foot etc [40]. If left the disease unchecked it will make serious complications including death. It is true that the disease can not possible be cured completely, early diabetes diagnosis can play a crucial role in its control, and prevent further medical complications so the person affected can lead a very healthy life.

Here, CLIPS expert system building tool version 6.3 in Windows/DOS environment was used to design and develop a medical expert system for diabetes mellitus disease. The system supports diagnosis, give information about complications and act as diabetes trainer. It used rule based approach to collect data and forward chaining inference technique.

It provides a user interactive, menu driven environment. Symptoms and risk factors associated with diabetes are taken as the basis of this study. In case of diagnosis the system will ask a bunch of questions about the symptoms and risk factors to the expert system user and user should give yes or no answer. According to the answer the system will make judgment about the possibility of illness, how much severe it is like slight chance, moderate chance, high chance, very high chance, diabetic or not.

4.3 Design of Expert System for Search Allergy and Selection of the Skin Tests

This work is the design of an expert system using CLIPS that aims in the procurement of patient medial background and in the search for suitable skin test selections [17]. Skin testing is the tool used most widely to diagnose allergies. This present work is probably a first step of design of an expert system for the search of allergy in patient and the suitable selection skin tests. The system constitutes part of intelligent robotic system of diagnosis of allergies. Here in is presented the evaluation of expert system through concrete medical cases. The expert system was evaluated by the import of certain medical cases and the system produced with suitable successful skin tests.

4.4 Development of an Expert System for Reducing Medical Errors

Recent advances in patient safety have been hampered by the hard dealing with the development of a uniform classification of patient safety concepts in a systematic way [18]. A hierarchical structure of the medical errors expert system was developed here, which was written and complied in CLIPS. It has 225 rules, 52 parameters and 830 conditional paragraphs. The system prompts the user for response with suggested input formats. The system checks the user input for consistency within the given limits. In addition, the system was validated through numerous consultations with the experts in the field. The benefits that are gained from such types of expert systems are eliminating the fear from dealing with personal mistake, and providing the up-date information and helps medical staff as a learning tool.

5. Conclusions

The CLIPS Programming is really an expert system tool, which can be used in medical practice. This is because of the functional fundamental components of CLIPS, we know that Facts make up the first component of CLIPS, made up of fields – symbol, string, integer, or float. Rules make up the second component of a CLIPS system. Rules are divided into LHS – IF portion and the RHS – THEN portion. The third component is the inference engine – rules having their patterns satisfied by facts produce an activation that is placed on the agenda.

References

[1] Martin, Linda and Taylor, Wendy. A Booklet of CLIPS Applications, NASA, Johnson Space Center, Houston, TX, 1992.

[2] Giarranto J. C., 2007. CLIPS User Guide, Quicksilver Beta

[3] Giarranto J. C., 2007, CLIPS Basic Programming Guide

[4] Giarranto J. C., 2007, CLIPS Interfaces Guide

[5] Jackson & Peter, (1998) Introduction to Expert Systems, 3rd ed, Addison Wesley

[6] Harpreet K., et al., Artificial Intelligence: Bringing expert knowledge to computers,
Discovery, 2012, 2(4), 4-7, http://www.discovery.org.in/d.htm

[7] Bonn, Deutsche Gesellschaft fuer Luft- und Raumfahrt, 1986, p. 443-448

[8] Eric Tatro, Artificial Intelligence, Robotics, 2004, http://www.asimovlaws.com/articles/archives/2004/07/why_we_need_fri_1.html

[9] Russell, Norvig P. Artificial Intelligence A Modern Approach, 2011

[10] Nilsson N.J., 2010: The Quest for Artificial Intelligence, Cambridge University Press, UK

[11] Belavkin R.V, Introduction to Expert Systems , Lecture note 6, BIS4435, Middlesex University, London

[12] Harpreet K., et al., Artificial Intelligence: Bringing expert knowledge to computers,
Discovery, 2012, 2(4), 4-7, http://www.discovery.org.in/d.htm

[13] Bonn, Deutsche Gesellschaft fuer Luft- und Raumfahrt, 1986, p. 443-448

[14] Eric Tatro, Artificial Intelligence, Robotics, 2004, http://www.asimovlaws.com/articles/archives/2004/07/why_we_need_fri_1.html

[15]. Russell, Norvig P. Artificial Intelligence A Modern Approach, 2011

[16]. Weyand J. IN: Yearbook 1986 II; DGLR, Annual Meeting, Munich, West Germany, Oct. 8-10, 1986, Reports, A87-48154 21-01

[17] Nilsson N.J., 2010: The Quest for Artificial Intelligence, Cambridge University Press, UK

[18] Shapiro L.G. and Stockman G.C., 2001. Computer Vision. Prentice Hall. ISBN 0-13-030796-3

[19] Morris T., 2004. Computer Vision and Image Processing. Palgrave Macmillan. ISBN 0-333-99451-5.

[20] Jähne B. and Haußecker H., 2000. Computer Vision and Applications, A Guide for Students and Practitioners. Academic Press. ISBN 0-13-085198-1.

[21] Luger G.F., 2002: Artificial Intelligence: Structures and Strategies for Complex Problem Solving. Addison-Wesley Publishing Company , USA

[22] Mooney R.J., Artificial Intelligence: Natural Language Processing, University of Texas, Austin, USA accessed June 23, 2013.

[23] Helsgaun K. Ten Project Proposals in Artificial Intelligence. Roskilde University, Denmark. Assessed May 25, 2013

[24] Durkin, J., 1990, Research Review: Application of Expert Systems in the Sciences, The Ohio Journal of Science. 90, 5, pp 171-179 http://hdl.handle.net/1811/23417, Downloaded from the Knowledge Bank, The Ohio State University's institutional reposit

[25] Nilsson, N. J. 1980 Principles of Artificial Intelligence. Palo Alto, CA.Tioga.

[26] IOM (Institute of Medicine). 1990. Medicare: A strategy for quality assurance (2 vols.). Washington DC, USA: National Academy Press.

[27] An Expert System For Diabetes Diagnosis (2010) Submitted in Partial Fulfillment of the Requirements for the Award of the degree of Master Of Philosophy By Smitha V (Roll No: 0935014), Department Of Computer Science, Christ University Bangalore

[28] J. G. Holman & M. J. Cookson (1987) Expert systems for medical applications, Journal of Medical Engineering & Technology, 11:4, 151-159, DOI: 10.3109/03091908709008986

[29] Karagiannis S., Dounis A. I., Chalastras, T., Tiropanis P., and Papachristos D., 2007, Design of Expert System for Search Allergy and Selection of the Skin Tests using CLIPS, Journal of Information and Communication Engineering 3:6

[30] Riley G. (2019), CLIPS: An Expert System Building Tool, NASA. accessed on 21st Nov. 2019 @ <https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19920013450.pdf>

[31] Roberts J., Dehart D., Tolle K. and Heckerman D., 2009: Healthcare Delivery in Developing Countries: Challenges and Potential Solutions in The Fourth Paradigm Data-Intensive Scientific Discovery, Microsoft Corporation, Redmond, Washington, USA

[32] Buchanan, B.G.; Shortliffe, E.H. (1984). Rule Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project. Reading, MA: Addison-Wesley. ISBN 978-0-201-10172-0.

[33] IOM. 2004. Patient safety: Achieving a new standard for care. Washington, DC: The National Academies Press.
[34] The Millennium Development Goals Report. United Nations, 2008.
[35] Feigenbaum, E. A. 1977 The Art of Artificial Intelligence: Themes and Case Studies of Knowledge Engineering. IJCAI. 5: 1014-1029.
[36] Belavkin R.V, Introduction to Expert Systems , Lecture note 6, BIS4435, Middlesex University, London
[37] Dean T., 1994, Artificial Intelligence: Theory and Practice, Addison-Wesley Publishing Company , USA
[38] [1]Giarranto J. C., 1998. CLIPS User's Guide, Version 6.22, 1998.
[39] Naser S.S.A and Ola A.Z.A., 2008: An Expert System for Diagnosing Eye Diseases Using Clips, Journal of Theoretical and Applied Information Technology www.jatit.org/volumes/research-papers/Vol4No10/5Vol4No10.pdf
[40] An Expert System For Diabetes Diagnosis (2010) Submitted in Partial Fulfillment of the Requirements for the Award of the degree of Master Of Philosophy
By Smitha V (Roll No: 0935014), Department Of Computer Science, Christ University Bangalore
[41] Karagiannis S., Dounis A. I., Chalastras, T., Tiropanis P., and Papachristos D., 2007, Design of Expert System for Search Allergy and Selection of the Skin Tests using CLIPS, Journal of Information and Communication Engineering 3:6
[42] Ramadan M. and Al-Saleh K., 2013, Development of an Expert System for Reducing Medical Errors, International Journal of Software Engineering & Applications (IJSEA), 4 (6).