**GROUP ASSIGNMENT 2**

Expert Systems with Uncertainty

**ARTIFICIAL INTELLIGENCE**

CSC 3206

**TUESDAY P3 1**

Muhammad Awad Luckhoo (15042278)

Choong Kai Wern (15053648)

Teh Cuok Syen (15061567)

Ong Li Shen (15071863)

Mu Chun Khang (13079272)

Mah Qi Hao (16080111)

# TABLE OF CONTENTS

[TABLE OF CONTENTS 2](#_Toc485898682)

[QUESTION 1 3](#_Toc485898683)

[Literature Review 3](#_Toc485898684)

[Comparative Analysis 3](#_Toc485898685)

[Justification 3](#_Toc485898686)

[Expert System Design 3](#_Toc485898687)

[QUESTION 2 4](#_Toc485898688)

[Literature Review 4](#_Toc485898689)

[Quality of Recommendations 4](#_Toc485898690)

[QUESTION 3 5](#_Toc485898691)

[Contributions of Members 5](#_Toc485898692)

[Gantt Chart 5](#_Toc485898693)

[REFERENCES 6](#_Toc485898694)

# QUESTION 1

## Literature Review

Literature review about expert system

* What is expert system
* How it works
* Applications of expert system
* Downfall

Expert systems are one area of artificial intelligence. Expert systems are computer software that have been designed in order to replicate human experts in both acts and activities, by the system’s capability to find new facts from available facts and give advice, to demonstrate and carry out intelligent tasks.

Various choices are provided when designing an expert system, the methodology, the mode of knowledge representation within the model (production rules, frames, logic programming, or some combination of these methods), the software development package, the hardware upon which the expert system is to be implemented.

## Comparative Analysis

Compare various expert system development tools

* Expert system shell: Jess
* Agent-based development environment: Netlogo
* Programming language: Prolog

For the purpose of building an expert system, there are three main development tools available: expert system shell, agent-based development environment and programming language. To illustrate the differences between all three expert system development tools, this section of the paper shall provide an exhaustive comparative analysis. In this context, an example is given for each respective expert system development tool. The three tools are first discussed in terms of their common purpose. Subsequently, their distinct features and differences are deliberated individually. This should provide readers with a good overall view of the various expert system development tools.

It is imperative to note that expert systems use reason of knowledge to solve complicated problems. The systems are represented predominantly with if-then rules, and not the conventional procedural code. At this juncture, the development tools for expert systems must be specialized. An expert system development tool is simply a software development environment that contains the basic components of an expert system: knowledge base and reasoning engine. Typically, expert system development tools come with prescribe methods of building applications through configuration and instantiation of these components. Besides that, developers are also offered numerous choices when designing an expert system. The methodology, mode of knowledge representation, software development package and hardware to be implemented on are among the factors for consideration. Herein, all expert system development tools, be it expert system shell, agent-based development environment or programming language, are crucial in aiding developers construct expert systems.

Expert System Shell

Expert system shells consist mainly of a user interface, inference engine and an editor to assist developers in building their knowledge base for expert systems. A great example of an expert system shell is Jess which stands for The Java Expert System Shell. Jess is an expert system shell written in Java entirely, driven by a Lisp-styled scripting language. Java provides the external mechanisms that generate and control the rules. To use Jess, the data must first be converted into text before it is handled by the interpreter.

Jess can be utilized in two ways, namely as a rule engine or as a general-purpose programming language. A rule engine is a special program that applies rules to large sets of data in an efficient way. A rule-based program can have up to thousands of rules, but Jess will have no problem applying them as data in the form of a knowledge base. In particular domains, the rules are represented as the heuristic knowledge of human experts, while evolving situations are represented by the knowledge base. On the other hand, as a general-purpose programming language, Jess can be extended easily as it directly accesses all Java classes and libraries. New commands can be written in Java or Jess to be integrated into Jess. Thus, Jess is very customizable when it comes to building applications.

The advantages of Jess include the ease of working with the code builder because it is an independent scripting language. Jess releases the burden from developers because they are not required to declare each rule as a set of nested class instances. On top of that, the programming effort for building the user interface and inference engine is also greatly reduced. Projects can be completed faster and cheaper in an efficient manner. However, Jess has the disadvantage of Java being disconnected from the rule engine. Normal Java syntax cannot be used to debug the syntax after external files and strings have been used to specify the rules. Besides that, Jess implements the Rete algorithm to match rules against the knowledge base. Since the algorithm trades space for speed, Jess requires relatively-large memory usage for moderately-sized programs.

* High-level programming languages - an expert system can be implemented using general purpose programming language. However, the programming languages such as LISP and PROLOG are the most commonly used in expert system implementation due to their high capabilities in handling symbolic data efficiently.
* Multiple-paradigm programming environments - these systems expand the capabilities of shells in various directions. They run on engineering workstations, minicomputers, or mainframes; offer tight integration with large databases; and support the building of large expert systems.

Prolog on the other hand is considered as a high-level programming language. This program is made up of a set of clauses. A clause is either a fact or a rule which is usually used to indicate a relationship between elements. The orders of database entries are extremely important because the efficiency of the program will mainly depend on the arrangement as it can influence the number of search required to satisfy the goal. There might be situations where Prolog may not find a solution even if a solution might be easily be found from the given information because of the exact nature of the search process. The process of deriving a reasoning for Prolog programming is as follows:

* A goal is given, Prolog searches into the database form the top to the bottom for a fact that matches the goal.
* A pointer is left where the match is found and prolog instantiates the suitable variables.
* When a goal matches the head of a rule rather than a fact, the atoms within the body of the rule are treated as sub goals that must all be satisfied to prove that the head is satisfied.

Advantages:

* Prolog is a powerful language which deduces the desired supplementary facts with built in powers of deduction from statements and principles that has been set to the system which one wishes to reason.
* The programmer only has to define what is required rather than indicating how it should be computed and thus, removing the burden of worrying about implementation details from the programmers.
* Prolog encourages incremental system development and makes program tracing and debugging easier.
* With Prolog, it is easy to maintain program.
* Modular programming and testing is possible with Prolog.
* Tracing and debugging programs in Prolog is much easier.
* Enables implementation of complex data structure.

Disadvantages:

* It can be very difficult to design a database that accurately represents relationships.
* Prolog is not best suited to solving complex arithmetical computations.

Prolog vs Jess:

1. Prolog and Jess which is a Rete-based system are very different. Prolog is backward chaining meaning that it starts with the desired conclusion(s), work backwards to find supporting facts and is goal-directed. If one forgets the result and ask for it again, Prolog has to compute it all over again. Jess on the other hand includes both a kind of backwards chaining and a construct called defquery which lets one make direct queries of the knowledge base.
2. Prolog is meant to be used from the console whereas in Jess, the command-line is not intended for end-users.
3. Prolog focuses more on answering queries whereas Jess focusses more on acting in response to input.

One more difference is that Prolog is really meant to be used from the console; i.e., you’re actually supposed to sit down and type mortal(Socrates). In Jess, only developers do this; the command line is not intended for end-users. Prolog is really about answering queries, while Jess is about acting in response to inputs. Jess is different than some Rete-based systems in that it includes both a kind of backwards chaining and a construct called defquery which lets you make direct queries of the knowledge base. Both of these help Jess a better fit for some Prolog applications, but they don’t make Jess into a Prolog-like system. Prolog is optimized, in a sense, for space, at the cost of speed. Jess (and its Rete algorithm) is optimized for speed at the cost of space. The Rete algorithm is all about computing things -once- so they never need to be recomputed, and then reusing them. Prolog’s approach is targeted at exploring large numbers of possibilities once, while Rete is aimed at exploring medium-sized numbers of possibilities repeatedly.

**Agent-based development environment: NetLogo**

**Multi-paradigm programming** environments is when the systems expand the capabilities of the shells in various directions. One of the example of the multi-paradigm programming languages is Pascal. Pascal are also known as the typical block-structured algorithmic language. Due to many cases, it may be implemented very efficiently on the conventional computer hardware. Conventional programming language such as Pascal are aimed at creation of simple language that will be suitable for scientific computation and construction of system software.

Current Expert System programming language vs Very high-level languages

|  |  |  |
| --- | --- | --- |
| **Language Features** | **High Level** | **Very High Level** |
| Data structure | Lists | collections |
| Decomposition | Selectors | Pattern match |
| Database | Property list | Association assertions |
| Program specification | functions | Pattern-action rules |
| Control structure | Recursion | Non-determinism |

Pascal does not have the programming features as mentioned above but it does have some features for the design and also the constructions of the expert system. Example of typical high level language is LISP and PROLOG, in some sense may regard as something close to very high level Expert System language.

The main drawback of Pascal like languages will be their inflexible control structures besides they have very limited or almost none inference power.

Compare LISP with Prolog

**DATA STRUCTURE**

LISP : list

List can be explained as 1D array with the following properties which is having a dynamically alterable length of array, the elements may have heterogenous types, composed by means of constructor functions to carry elements and decomposed by selectors functions rather that position Index

Prolog: Collections

In a very high level languages, other data structures other than list also available

**DECOMPOSITION METHOD**

Lisp: By function CAR and CDR

Very simple mainly by the means of functions

Prolog: pattern matching

AI language usually have one critical property which is pattern matching because it will be needed when there is explicit structure decomposition that will also be used for associative retrieval and pattern action rules.

**DATA BASE**

Lisp: property lists

Property lists used to characterize a sysmbol with arbitrary number of properties. The property list for the symbol S can be implemented as the even-length list associated with S by hash-coding.

Prolog: Associative assertions

For a very high ES language, the facts about the symbol can be written as assertions. The set of assertions will have the associative data base that will allow access with more ease and natural. They can also used to deduce new facts.

**PROGRAM SPECIFICATION**

**Lisp:** functinos

In lisp, function will be the only language concepts on which everything else must be build and there are three kind of function in LISP which is transformation functions, predicate and special functions.

Transformation functions are used to perform expressions into other expressions. Predicate is to test the expresions for their properties. Special function is used to evaluate expressions in the non-standard ways

**CONTROL STRUCTURE**

LISP: recursion

Most complex programs are written using recursion and it is very important in programming

PROLOG : non-determinism

Most important feature of high level ES language. Used to abstract points within unnecessary details and is one of the very few ways to cope with complex system. The existence quantifier of the logic non-determinism will permit the abstraction from the explicit disjuncts only one of which holds true, non-determinism control the permits an abstraction from a set of computation where only one of which need to succeed.

PROLOG is obviously better than LISP

Pascal is simple to write in program but no always a good choice to implement ES.

LISP and prolog are suitable languages to write in when building a ES but only a well trained programmer can build the system. LISP and PROLOG are both less famous compared to Pascal because one of the factor is the availability of the hardware. LISP and prolog were not available for mainframe until recently.

|  |  |  |  |
| --- | --- | --- | --- |
| property | Lisp | prolog | pascal |
| List |  |  |  |
| Decomposability |  |  |  |
| Flexible control structure |  |  |  |
| Interactiveness |  |  |  |
| Efficiency |  |  |  |
| Late binding time patter matching for data and control |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Justification

Justify Prolog

* Problem domain
* Knowledge domain

## Expert System Design

Describe and justify expert system design

* Approach / methodology
* System components
* Major design elements
* System strengths
* System weaknesses
* approach/methodology
  + This system will help the user to find out their interested subject or subject that suits them.
  + This system will filter out the subject that suitable for the user by asking them few preset questions.
  + What the user needs to do is answering the questions by choosing their answer from given choices.
  + A subject will only being suggested if all its pre-conditions or rules are achieved.
  + In order to prove a rule, the user should answer a question and the system will decide whether the user meets the rule.
* system components
  + questioning mechanism
  + answer parser
  + subject finder
  + degree finder
* major design elements
  + Agenda
    - Subject
    - degree
  + knowledge base
    - rules (subject rules, degree rules, logical thinking)
  + knowledge acquisition facility
    - question
    - answer
  + working memory
    - skills (science,logical thinking) after answering question
  + Inference engine
    - Backward chaining
  + Explanation facility
    - describe
* system strengths
  + Increased availability
    - Mass production of expertise since it can be made available on a computer
  + Reduced cost
    - Average cost of providing expertise is greatly lowered
  + Permanence
    - Expertise is permanent, lasts longer than the human expert.
  + Fast response
    - Sometimes real-time response is required, ES is more available than human expert
* weaknesses
  + no security
  + no error handling
  + does not include all the subjects available

# QUESTION 2

## Literature Review

Literature review about uncertainty in expert systems

* What is uncertainty
* Why should it be introduced in expert systems?
* Applications of expert systems with uncertainty

## Quality of Recommendations

Compare recommendations made by expert system without uncertainty and expert system with uncertainty

* Input
* Difference in output
* Justification
* Which one is better?

# QUESTION 3

## Contributions of Members

Muhammad Awad Luckhoo

Tasks here

Choong Kai Wern

Tasks here

Teh Cuok Syen

Tasks here

Ong Li Shen

Tasks here

Mu Chun Khang

Tasks here

Mah Qi Hao

Tasks here

## Gantt Chart

Gantt chart for planned and actual timeline

# REFERENCES

APA references here

[1] <http://www.jessrules.com/doc/61/intro.html>

[2] http://www.it.kmitl.ac.th/~pattarachai/ES/PDF4/Ch08-ESTools.pdf

[3] <http://developeriq.in/articles/2014/jul/14/expert-systems-programming-languages-tools-and-she/>

[4] <https://www.academia.edu/23352301/Tools_of_Development_of_Expert_Systems_A_comparative_study>

[6] <https://www.cs.unm.edu/~luger/ai-final2/JAVA/CH%2026_Case%20Studies%20-%20JESS%20and%20other%20Expert%20System%20Shells%20in%20Java.pdf>

[5] <http://www.iau.dtu.dk/teaching/31380/Jess/manual.pdf>

[7] http://http-server.carleton.ca/~aramirez/4406/Reviews/SRetchford.pdf