Literature Review

Maanda Raudzingana
Supervised by: Dr. Karen Bradshaw

November 3, 2014

Abstract

Much research in the field of artificial intelligence (AI) has focused on the development and use of expert systems. Consequently, great advances have been observed in the development of these systems. This can be seen in the sophistication of the representation of knowledge and application of underlying knowledge bases to arrive at powerful inferences. However, despite the success of expert systems in the field of AI, research continues with a focus on the knowledge representation process, in an attempt to address what is commonly referred to as the expert system development bottleneck. This survey highlights notable advances with regard to knowledge representation. Furthermore, an overview of expert systems, examples of these, and established knowledge representation formats are given. The survey describes and evaluates available tools and techniques that can be used to assist developers in the construction of knowledge bases.

1 Introduction

The aim of this chapter is to review issues around the development of expert systems. With emphasis given to the knowledge representation process. An overview of expert systems highlights the foundation and necessity of improvements in the representation of knowledge. It has been argued that failure to represent effectively appropriate amounts of knowledge in knowledge based systems results in limited performance during consultation [25]. The discussion highlights the implications, and research and development trends in knowledge acquisition. Knowledge representation schemes and their characteristics are presented. Research efforts have influenced the development and refinement of automated knowledge engineering tools. A few of these are reviewed. Finally, the tools and techniques available for the development of these knowledge engineering tools are explored.

2 Expert Systems

An expert system [15] is an example of a knowledge based system whose primary purpose is to mimic human reasoning. This is achieved by emulating the decision-making ability of a human expert in a specific domain. A structured organization of knowledge, facts and reasoning techniques is used to solve problems that would normally require human expertise. The organized collection of facts and knowledge is known as the knowledge base. This essentially houses the knowledge transferable to a user during consultation. The expert system applies the facts stored in the knowledge base to arrive at a conclusion. This is the purpose of the inference engine. Facts together with information provided by the user are interpreted and evaluated to infer new knowledge. A representation language is used to express facts [11], with common representations including predicate calculus, rules, graphs and decision trees . Expert systems are considered to be successful because they are successful at transferring expert knowledge in a given problem domain. For this reason, expert systems have seen widespread use in several domains. Expert system technology is currently used to solve many industrial and commercial problems.

2.1 Applications of Expert Systems

Applications of the technology can be clustered into classes [12] such as diagnosis, troubleshooting, prediction, configuration, decision making, knowledge publishing, monitoring and control, design and manufacturing as well as debugging and instruction. Systems in these classes often employ highly structured rules to enable them to achieve the decision-making ability similar to that of a human. Successful examples include medical diagnostic systems that are in use today in several medical facilities, computer network diagnostic systems used by network administrators for network fault diagnosis, mission control systems used by military personnel, and the training and crisis management systems used by different organizations. Recent literature has highlighted the apparent interest in integrating expert systems technology with existing applications in various domains [11]. These include multimedia applications, accounting and business management applications, and applications in the medical and educational domains. This widespread use of expert systems technology highlights the relevance of expert systems research and development in this day and age.

2.2 Components of an Expert System

Three basic components make up an expert system: a knowledge base, an inference engine, and a user interface. Some systems have a knowledge acquisition module to allow domain experts to add new knowledge into the knowledge base. Shown in Figure 2.1 are the different components of an expert system.

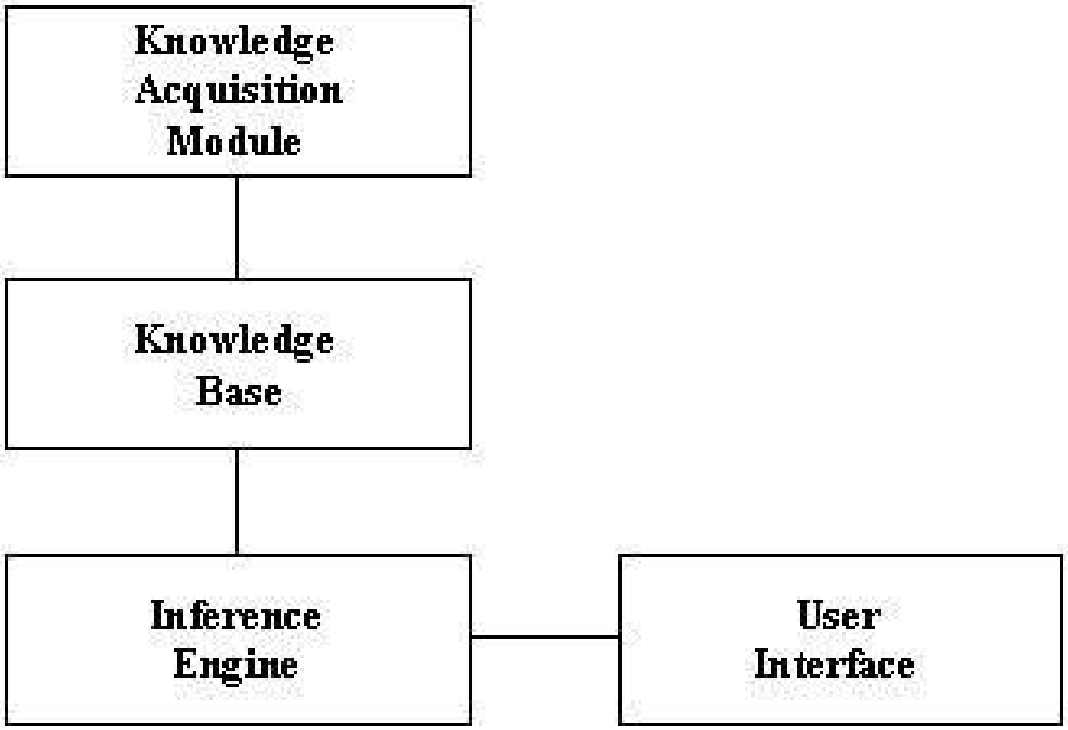

Figure 1: Components of an expert system

These components work together to effectively transfer declarative and procedural knowledge to the user upon consultation [31].

2.2.1 The Knowledge Base

The knowledge base is the store of facts and rules. It contains the domain knowledge vital for problem solving, i.e., declarative knowledge. Knowledge is organized as facts, objects, attributes and conditions [15]. During inference, the knowledge base is searched and manipulated according to predefined rules of logic. The knowledge contained in the knowledge base is commonly acquired through transfer from a human expert using appropriate methods. The system thus represents the expert's knowledge and expertise in the specific domain. A human expert is known to possess extensive knowledge on a given class of problems. Embedding such limited knowledge in a computer system has proven to be an easier task compared to embedding large amounts of information from different classes of problems [7].

Expert systems transfer human expertise. This has been described as intensive knowledge specific to a domain [11]. According to Iancu et al. [11], the knowledge can be categorized into knowledge types: facts and theories about the domain, rules and procedures, global strategies on solving problem types and finally, meta-knowledge.
Part of the contents of the knowledge base is facts. These are known items of information of a given subject. In a medical scenario, the statement: "Streptococcus pneumoniae causes Pneumonia" can be considered a fact. This is information that is known and can be proven. A collection of facts is housed in the knowledge base, which describe a given subject in detail.
Besides facts, contained in the knowledge base are objects [15]. Objects are representations of real world entities, perceptible by one or more senses. In a medical context, Pneumonia can be classified as an object. These objects have attributes that describe them. The use of the facts and objects by the inference engine is governed by a set of conditions or rules.

Uncertainty

It is rare that knowledge is certain. The incomplete, ambiguous and uncertain nature of information introduces difficulties during the development of intelligent systems [25]. This introduces the need for expert systems to handle uncertain knowledge. In the early days of expert systems development, marking a milestone in the development thereof, was the introduction of a level of certainty in accumulation of evidence and confidence of the hypothesis. These came to be known as certainty factors [5]. Certainty factors allow an expert system to express a relative level of confidence, with a high factor denoting an absolute level of confidence and a low factor denoting the least level of confidence. A common approach to estimation of certainty is the use of probability theory. Sub-options include objective probability, which makes use of frequency refraining from any interpretation or degrees of confidence, and subjective probability, in which the lack of certainty is quantified, hence reflecting a degree of belief. Common practice is to use either approach in conjunction with other estimation mechanisms customized to a suitable extent. The PROSPECTOR system [7] handles uncertainty by assigning probabilities to conclusions using established statistical rules and theories. However, the use of statistical rules and theories introduces additional complexities with regard to statistical independence and prior probabilities. In an attempt to avoid such complexities, a novel calculus of certainty values was used to handle uncertainty in the MYCIN system. A statistical approach remains the most widely adopted approach to handling uncertainty during expert systems development. Despite successes of most implementations in handling uncertain knowledge, a point is yet to be reached where reasoning in the presence of vagueness and ignorance as well as recognition of the limits of knowledge bases by systems is possible [7].

2.2.2 The Inference Engine

The inference engine [15] is the heart of an expert system. Invoked whenever a user performs a query, the inference engine's main purpose is to draw inferences from the supplied information and the information contained in the knowledge base. Logical rules are applied to the knowledge base in order to infer new knowledge [5]. This involves effective comparison of available information, searching of goals and causal relationships and finally the evaluation of the relative certainty of facts [15]. The inference engine makes use of forward chaining and backward chaining strategies to go about inference.

Forward chaining

Forward chaining [5; 15] is one of the techniques used by the inference engine during inference. This method starts from known data and proceeds with the data effectively using inference rules to extract more data until the goal is reached. For this reason, it is also known as data-driven reasoning. The engine searches the rule set for a rule whose condition is satisfied and then infers the action effectively adding a new fact to the data store. This method is favored for its ability to trigger new inferences as a result of the reception of new data [9]. However, as a result of using forward chaining, more rules than necessary may be executed resulting in an inefficient inference process.

Backward chaining

Backward chaining [5; 15], often used as an alternative or in conjunction with forward chaining, is another technique for inference. Backward chaining starts from the goal and attempts to find the evidence to prove it. This is also referred to as goal-driven reasoning. The rule set is searched to find rules whose action parts or consequents match the goal. When found, the rule is fired and the goal is proved. This process involves adding rules to a list of goals that will have to be proven in order to prove the initial goal. The engine sets aside the rule it is currently working with and then establishes a new goal to prove the condition part of the rule. Searching takes place for rules that can prove the sub goal. The process continues until no rules can be found that prove the current sub goal.
The choice of method between forward and backward chaining is arguably dependent on the problem domain in which the resulting system will be used.

2.2.3 The User Interface

To achieve effective transfer of knowledge from the system to a user during consultation, an interface needs to be present. The user interface is the component that allows for communication between the user and the system [15]. This makes use of a natural language interface to simulate causal conversation [27]. It enables the input of facts about a given situation that the engine can use together with information stored in the knowledge base to arrive at conclusions. Furthermore, the interface allows the user to query the system for any information the user needs, like why a specific question has been asked. It is crucial that the interface is simple and easy to use. This stems from the assumption that expert systems are used by non-technical users who may not have an in-depth understanding of the inner workings of the system. It is for this reason that the interface should resemble a human as far as possible to guarantee effectiveness and ease of communication. Duda et al. [7] refers to this as a characteristic of effective consultants, that is, the ability to maintain a model of the user and effectively assess what the user does, does not know, and what he or she is trying to achieve.

2.2.4 Knowledge Acquisition

The knowledge contained in the knowledge base is acquired from human experts. The process of acquiring knowledge and transfer thereof into the knowledge base is commonly referred to as the knowledge acquisition process [6]. Most expert systems have a knowledge acquisition component that facilitates the transfer of knowledge from human experts to the expert system. This is achieved through the presentation of an interface that makes possible a dialogue between the system and the expert [15]. The knowledge acquisition interface is sometimes the user interface a non-expert user would use to query and supply information to the system during consultation, however, specific to the task of knowledge acquisition.

Alternative ways of knowledge acquisition include interviewing domain experts. The knowledge is collected and then translated into appropriate symbolic representations by a knowledge engineer. This was the common approach in the early days in expert systems development. However, research and development of new technologies has introduced tools and techniques that aid the automation of the knowledge acquisition process. Such tools are discussed in subsequent sections.

2.2.5 Explanation Facility

Besides the three basic elements discussed above, more sophisticated implementations include additional elements to improve the intended functionality of an expert system. One of these elements is the explanation facility, which most expert systems have [15; 5]. This enables the user to query the system on certain information. Examples include explanations on why certain information is needed, and how a particular conclusion was reached. An expert system should then be able to explain the reasoning behind its analysis and conclusion. This further enables the user to monitor the system as it processes rules [15]. Special commands are defined that cause the system to invoke this explanation functionality. In most expert systems, during consultation as the user supplies information to the expert system, it is possible to request an explanation on why a given question is being asked. The expected response to this is a display of where and how the piece of information is used. In most cases, this is an argument required for a given rule to be fired. An explanation of how a conclusion was reached is often required. The response takes the form of a chain of rules the system has applied to arrive at the reached conclusion. The necessity and importance of the ability of a system to explain its processes justifies the need for multiple levels of representation [5]. Such behavior further highlights the success of expert systems in mimicking human behavior in decision making; human experts are often required to provide explanations of their reasoning.

3 Knowledge Representation

Knowledge representation [5], considered by many be to the key issue at this point in the development of AI [22] is the area of AI that involves the transformation and translation of facts and information about a given subject into a structured symbolic form that allows manipulation in an automated way by reasoning programs. These facts are stored as knowledge in the system's knowledge base. The representation process is concerned with the encoding of the acquired knowledge into the knowledge base. For this reason it is a critical process in expert systems development. Research has focused on trying to find ways to represent knowledge that require less effort but maximize the system's performance. The representation of knowledge in notable systems was reported to be a major contributing factor to a large proportion of reported errors [7]. An effective representation has to address certain issues. Duda et al.

[7] demonstrates that a knowledge representation formalism should effectively represent the concepts and intentions of the expert, and allow correct interpretation by reasoning programs. Furthermore, it should support explanations that reflect human reasoning and be capable of identifying gaps in the knowledge as well as allow the separation of domain knowledge from the interpretation program to allow independent alteration of either. Such demands introduce difficulties in the process of knowledge representation, which if addressed would make way for improvements in the development of expert systems. Among these are issues on knowledge acquisition and the ensuing representation of the acquired knowledge.

3.1 Knowledge Acquisition

Knowledge acquisition is concerned with the transfer of knowledge from the expert into an intermediate store that can subsequently be translated into a formalized representation. Acquisition usually takes places in one of two ways: manually through interactions between an expert and a knowledge engineer, or automatically through the use of appropriate automated systems. Several methods are used to acquire knowledge from an expert. These include interviews, questionnaires, surveys and web-based systems that present an interface allowing experts to input knowledge into the system. However, the most common method of acquisition involves communication between a knowledge engineer and human experts. The knowledge engineer's responsibility is to program the knowledge into the expert system. This requires use of different knowledge representation schemes that present an executable representation to the inference engine or interpreter for execution.

Knowledge acquisition in its own right is time consuming and consequently poses costs during development of expert systems. Automating the knowledge acquisition process is a commonly proposed solution [8]. Automated knowledge acquisition tools have seen widespread use in recent years, with most of these under continuous refinement. These are discussed in a later section. Automated acquisition tools encourage interaction between experts and the system. Use of these requires that the experts receive training on how to use the system. Automated knowledge acquisition tools enable the knowledge engineer and domain experts to build and maintain the resulting knowledge systems [25]. Notable features used in the classification of such tools include the degree of automation, the dependency of the domain, problem type and user type.

Problems with knowledge acquisition have contributed significantly to the deterred improvement of expert systems over the years. Duda et al. [7] notes the incompleteness of the knowledge bases of the MYCIN and PROSPECTOR systems as stemming from problems in knowledge acquisition. Often the construction of knowledge bases, involving the identification and encoding of knowledge, reveals gaps and weakness in both the understanding of the problem domain and the representation techniques [7]. Rafea et al. [25] emphasizes the importance of a well functioning knowledge acquisition tool in expert systems development endeavors. Often, consultation performance is limited as a result of failure to acquire and encode relevant knowledge, which warrants diligence during the construction of these tools. Ford et al. [8] maintain that the knowledge acquisition process extends for the life cycle of the system. It is stated that at any given instance, an acquisition tool should support directly both the domain expert and the knowledge engineer. With this, Ford et al. [8] note the different facets of the knowledge acquisition process, which encompasses the elicitation and modeling of human expertise, testing of the systems efficacy, refinement of the knowledge base, maintenance of the resulting system, and elaboration of an explanation capability. These facets highlight the importance of the knowledge acquisition process in the development of consultants.

3.2 Knowledge Representation Schemes

During development, a knowledge engineer must choose the format to be used in representing the knowledge. A number of different formats are available with different structures and characteristics and as result fare differently in representing different kinds of knowledge.

Semantic Networks

A semantic network [15] is a model of associative memory, a graphical knowledge representation method that uses patterns of interconnected nodes and arcs. The nodes effectively represent objects, with the arcs representing the relationships between the objects. The idea behind semantic networks is that knowledge can be stored in the form of a graph. Semantic networks are widely used as a knowledge representation technique owing to their robustness and guarantee of efficiency in searching and manipulation of the data structure. The relationship with other connecting objects or nodes gives meaning to an object, and hence allows for activation of an object on a given node. A simple semantic network in the LISP [17] language could be defined as:

```
(defun *database* ()(
(human  (has-a house) (color white) (size big))
   (house   (has-a kitchen))
    (kitchen    (is-a room)(has-part stove))
    )
 )
```

In the example above, humans, house and kitchen are objects. The arcs are represented by the association functions: have-a, is-a, and has-part. The attributes of the *house* object in this case are color and size. New nodes and arcs can be added to a network at any time during the development process. These data structures have been widely used in a variety of ways. Tanwar et al. [30] demonstrates that there are six common kinds of networks used in various applications, namely, definitional networks, assertional networks, implicational networks, executable networks, learning networks and hybrid networks.

Definitional networks represent relations between a concept type and a newly defined subtype, like the relation: an apple is-a fruit. These are known to support the rule of inheritance. Assertional networks are used for the assertion of propositions, like, if a man owns a car, then he drives it. Implicational networks are commonly used to represent patterns of beliefs, causality, or inferences. They use implication as the relation to connect nodes. An example would be: if the grass is wet, then it is slippery. Executable networks present mechanisms to perform inferences and or search for patterns and associations. Mechanism include attached procedures, graph transformations, and message passing. Yet another example is learning networks. These use acquired knowledge to build their representations. Knowledge is acquired from examples. When new knowledge is introduced, extension or deletion of the old contained knowledge takes place. Hybrid networks are made up of a combination of any of the networks discussed above, either in a single network or in separate networks [29].

The presence of links between objects in semantic networks, synonymous to mental links humans form between real world objects, presents semantic networks as a representation that closely resembles human knowledge structuring. Meaning is derived from the presence of and connection to other objects. Semantic networks are favored for their ability to facilitate the

inheritance of data that prevents duplication of data, as well as their ability to allow efficient manipulation. Argued to be the most useful form of inference, inheritance allows elements of a class to inherit characteristics and behavior from a class higher in the hierarchy. Multiple inheritance introduces additional benefits, that is, objects can belong to more than one class, and in turn, a class can be a subset of more than one class. Additionally, semantic networks fare well in representing knowledge involving related terms, and in conveying meaning in a transparent manner. Their simple and easily understood structure makes it easy to translate them into a formalized symbolic representation that can be executed by programs.

Frames

A frame [5] is an AI data structure used for the representation of data. Two types of frames exist: individual frames, used for the representation of single objects, and generic frames, for the representation of classes of objects [5]. Each individual frame represents an object or situation and consists of a group of attributes that describe an object or situation. A frame consists of slots in which the attributes are stored. These attributes contain values and procedures that act on the values. Illustrated below is the schematic representation of a frame.

```
(Frame-name
<slot-name1 attribute1>
<slot-name2 attirbute2>
..)
```

The attribute values can be numbers, strings, or identifiers of other frames. Frames can be linked together, which is called inheritance. Upon retrieving a frame, an agent changes some of the information in the frame. This information can be declarative or procedural. A declarative representation, representing objects, facts and relations asserts the validity of an attribute whereas a procedural representation, representing the actions performed by objects, contains instructions to be executed in a given slot. With this, the frame provides information storage and instruction on how to act in a given situation. Information types stored in frames include: relationships between the frame and other frames, information for choosing frames, blank slots and procedures to be executed after various slots have been filled. Frames are favored for their structure, which allows them to be programmed and manipulated using object-oriented programming tools.

Frames, like semantic networks have the advantage of allowing reuse of related information through inheritance, thus eliminating the need to develop repetitive blocks of knowledge data. Frames fare well in representing knowledge that can be stored in chunks [15].

First-order logic

First-order logic [15], also known as first-order predicate calculus, propositional calculus, and predicate logic, is a method used for capturing declarative knowledge.

In first-order logic, a sentence is made up of two parts, a subject and a predicate. A predicate, which defines the properties of the subject can only refer to a single subject. Sentences take the form P(x), where P is the predicate and x is the subject, represented as a variable [26]. Manipulation takes places on logically combined complete sentences according to the same rules as those used in Boolean algebra. Universal (e.g. for all) and existential (e.g. for some) quantifiers are used to structure sentences. For example, given a variable x and predicates A and B, first-order logic allows the construction of sentences like:

$\forall x$: Ax Bx

The above translates to: "For all x, if x is A , then x is B. The level of ease with which sentences can be constructed using such syntax has resulted in AI researchers taking interest in first-order logic as a means of representing knowledge. Subsequently, computer programs have been developed using first-order logic [26].

First-order logic is favored for its declarative nature which allows facts to be represented within the syntax. Furthermore, it allows operations on information unlike most data structures. Examples of operations include dis-junction and negation. It further allows for the assignment of context-independent meaning, unlike natural languages where meaning is context-dependent. However, first order logic falls short in that it has limited expressive power [2]. A famous argument in logic can be used to illustrate this.

```
All A are B.
All B are C.
Therefore, all A are C.
```

It is argued that no good way exists to express the above using propositional calculus [23].

Rules

Rules or production rules [15] are perhaps the most prominent method for representing knowledge. Systems that employ rules for representation are commonly referred to as rule-based systems or production systems. Production systems maintain an ongoing store of assertions in working memory [5]. A production rule is made up of condition-action or antecedent-consequent pairs. A given pair represents an IF-THEN structure that links supplied information in the IF part to an action in the THEN part. The simple structure and syntax of rules makes them human readable and easy to implement. A rule in its own right provides some description of how to solve a given problem. It is possible to extend the antecedent part of a rule by adding more of these. This is achieved by adding keywords like AND for conjunction and OR for disjunction, or a combinations of these. An antecedent contains an object linked to a value by an operator. Mathematical operations are often used with rules on objects and values that permit mathematical manipulations. Examples include comparisons, additions, logical operations and boolean operations. Rules can be used to represent relations, recommendations, directives, strategies and heuristics. Depending on the amount of knowledge to be represented, rule structures range from simple antecedent-consequent pairs to a large complex set of complicated rules. An example of a rule is given below.

```
if<antecedent> then <consequent>

IF day_of_week = 'Sunday'
THEN lectures = False
```

The given example shows how a rule can be used to represent relations. According to this specification, whenever the day of the week is Sunday, lectures should be set to false, which would simply imply no lecture attendance, as is the case in most academic institutions. In this case, *day_of_week* is a variable whose value would have been supplied by a user. The system

uses the value to determine whether there should be lecture attendance. During interpretation, the rule set is searched for a rule whose condition is satisfied. When found, the action is invoked. Although simple, this example highlights the basic procedure of rule interpretation by a rule-based system. More complex rules are common in large systems. For example, the MYCIN system had a rule-based knowledge base consisting of five hundred rules [28]. This can be argued to be a reasonable amount considering MYCIN's performance during consultation.

Uncertainties can be expressed in rules by attaching certainty factors to the antecedent, consequent, or both [15]. Implementations often make use of custom methods to handle uncertain knowledge. These methods mostly include assigning a level of confidence to a given rule. This requires customization of the knowledge base to allow expression of the confidence of the expert system.

Hybrids

A common approach to knowledge representation is the use of more than one knowledge representation scheme. The systems are customized to leverage the functionality and benefits of two or more representations formats. This has been known to introduce improvements in knowledge representation and in most cases, inference. Tanwar et al. [30] highlights the strength of a hybrid knowledge representation technique. A technique is described, which uses semantic networks and scripts to represent knowledge. In this experiment, scripts complement the representation of non-event based knowledge by filling the gaps resulting from incomplete or disjoint observations. Despite improvements introduced by hybrid representation techniques, the use of two representation techniques introduces complexities. For this reason, it is encouraged to adopt a hybrid approach with extreme caution and complete understanding of the intended functionality.

Comparison of schemes

Niwa et al. [22] presents a comparison of four knowledge representation schemes: simple production systems, structured production systems, frame systems and logic systems, based on the difficulty of the implementation and runtime efficiency. The applications domain used was the risk management of large construction projects. Implementations of the representations would be tested on four systems developed using a modified version of the LISP programming language. Common components of all the systems were an inference engine, and a knowledge maintenance function to facilitate alteration of the knowledge base. The systems were capable of both forward and backward chaining reasoning. Forward chaining was used to inform the user of consequent risks that could follow specific causes, whereas upon input of a risk hypothesis by the user backward chaining was used to query the user about conditions until the hypothesis, in this case the risk, was proven. After testing, findings revealed that the level of difficulty of implementing a knowledge base was directly proportional to the degree of structuredness. However, runtime efficiency was found to increase with increased structuredness. Structuredness was also found to decrease sensitivity to the size of the knowledge base [22]. Findings further revealed that inference was slow with logic systems used as representations. This lead to the conclusion that logic systems, compared to the other representations are less efficient and difficult to implement owing to mathematical completeness. Use of modular knowledge, that is, simple production systems and logic systems, is encouraged in poorly understood

domains with poorly structured knowledge, although this has the cost of low runtime efficiency. Structured knowledge representations guarantee maximum runtime efficiency at the cost of difficulties during implementation.

Summary

The process of knowledge representation has proven to be the bottleneck of expert systems development[6; 16]. Iancu et al. [11] argues that the problem in knowledge problem is the introduction of information in systems that permits easy access and usability in solving problems. Duda et al [7] predicts significant advances in the capabilities of expert systems when the unsolved problems in areas such as knowledge representation and learning are solved. Efforts have been directed toward establishing alternative ways of acquiring and representing knowledge using the least resources yet still guaranteeing effectiveness [6]. This has lead to the emergence of knowledge acquisition and representation tools. Achieved functionality includes automation to varying degrees, of the sub-processes involved.

4 Examples of Expert Systems

As discussed, the main purpose of an expert system is to transfer human expertise within a specific domain to users who may not have such knowledge. This can be done in a number of ways. Notable examples are: rule based expert systemsÂÿ which use knowledge bases containing expert knowledge in the form of IF-THEN rules; case based reasoning systems, which aim to solve new problems by reusing already established solutions used to solved previous problems; fuzzy expert systems, which aim to model uncertainty through the use of fuzzy membership functions and rules to reason about data; and neural network systems, which solve problems by simulating the biological structure of the human brain and nervous system [**?**]. Among the the first truly successful forms of AI software [32] and despite their limited knowledge and versatility, expert systems have achieved remarkable levels of performance in designated domains [7]. The expert systems domain has a wide area of application [11]. In an attempt to highlight the successes in various domains, a description of various expert systems, their domains, and characteristics are given below.

The MYCIN System

MYCIN [28] is an expert system developed at Stanford in 1972 for the purpose of treating blood infections. Reported to have operated at the same level of competence as specialists in blood infections, it was capable of identifying the bacteria causing infections, proceeding to recommend antibiotics and prescribing a dosage taking into account the patient's body weight. It used reported symptoms and medical results in an attempt to diagnose patients. MYCIN could request further information on a patient and was capable of suggesting additional tests in an attempt to arrive at a diagnosis. It could, at request, explain the reasoning behind a specific diagnosis. It gathered information from the user by asking a series of questions. At the end, it would give a list of possible answers, i.e., its guess on the bacteria causing the infection, together with the certainty factor associated with each diagnosis. MYCIN used a goal-oriented strategy to reason from an initial goal. With a fairly simple inference engine, the system's success was due to its effective use of certainty factors together with the implementation of a sophisticated knowledge representation and reasoning scheme, which influenced the development of subsequent rule-based systems following MYCIN's introduction of the approach [28]. MYCIN was

never used in practice for ethical reasons, not including poor performance.

The R1 System

R1 [18], also known as XCON is a rule-based expert system for configuring VAX-11 computer systems. Developed at Carnegie Mellon and DEC in the late 70's, it has the ability to display diagrams showing relationships between the components of a computer system. It has been used by the Digital Equipment Corporation's manufacturing organization since January 1980. R1 is said to have sufficient knowledge of the configuration domain thus it requires little search in order to configure a computer system. Reported errors were due to lack of information on new products [7]. R1 at the time of publication was in routine use. The acceptance of R1 in practice highlights the usefulness of expert systems technology.

PROSPECTOR

PROSPECTOR's [7] main purpose was the evaluation of the mineral potential of a geological region. It was used by geologists working in mineral exploration. Tasks included ore deposit identification and drilling site selection. In tests, PROSPECTOR was reported to have repeatedly produced results closely agreeing with those of geologist consultants [7]. PROSPECTOR's knowledge base contains models of different types of ore deposits. The success of the system in its domain justified efforts to extend the system's knowledge base.

DRILLING ADVISOR

DRILLING ADVISOR [14] was a rule-based system designed to diagnose oil drilling problems. It offered assistance to oil-well drillers on possible causes of problems encountered as well as solutions. It used backward chaining with certainty for inference, and frames as a knowledge representation scheme.

COMIX

COMIX [**?**], developed at Central Laboratories in New Zealand, is an expert system designed to give advice on the design of concrete mixes. COMIX is used in engineering and consulting disciplines, as well as by concrete technologists. Using the water over cement ratio, the system is able to calculate the amount of cement, course aggregate and sand required. Using mix designs based on the New Zealand code specification for concrete construction, the system refers the type of structure to the consistency and placement method [**? ?**]. The system houses a rule and frame based knowledge base, with a resident authority as the information source. Development is on-going aimed at the extension of the knowledge base to include revisions of cement types and the strength factors thereof.

BETVAL

BETVAL [**?**], another expert system in the construction industry, is a rule-based system designed to give advice on the selection of ready-mix concrete at a job site. It facilitates the selection of the type of concrete ordered from a ready-mix concrete plant. BETVAL was developed by the Technical Research Center using Insight2+ [**?**] and an IBM PC/XT or AT computer. BETVAL is seen as a demonstration prototype, primarily used as a learning tool. It was noted at the time of publication that an extension of BETVALâĂŹs knowledge would have to take place in order for BETVAL to be used as a production system [**?**].

Sports Injury Clinic

Sports Injury Clinic [34] is a web-based system that helps a user diagnose sports injuries. It presents a graphical user interface displaying different human body parts, and allows the user to point to problematic areas using the mouse cursor. The system gathers information from the user in an attempt to arrive at a conclusion. Like many other systems, Sport Injury Clinic offers a vast array of injuries, treatments and recommendations. The systems is limited in the sense that only muscular skeletal sports injuries are catered for. Besides this, the functionality and effectiveness of the system has lead to reviews of appraisal and great acceptance by experts in the medical field.

Expert System for Diagnosis of Pests and Diseases In Fruit Plants

Dewanto et al. [**?**] discusses the development of an expert system to diagnosis pests and diseases in fruit plants. The intended use of the system is to help users easily identify the type of disease in fruit plants. The design features primary components of an expert system, that is, a knowledge base, an inference engine, and user interface. The system employs backward chaining and utilizes a rule-based knowledge base to arrive at conclusions. Corvid Exsys [**?**] software, developed by Exsys company [1] was used to develop the system. The Exsys software tool features a rule editor with a visual interface of decision trees and an inference engine. It is possible to run applications developed using the Exsys software tool online. To handle uncertainty, a formula is used to calculate confidence values in the system. During consultation, the traditional approach of interactively asking the user for values to be used in the rules is used. The user selects relevant answers to questions posed, based on observed symptoms. The system uses the answers provided together with information stored in the knowledge base to arrive at a diagnosis. Experiments showed that the output provided by the system is in accordance with the knowledge obtained from knowledge sources, hence proving the credibility of the developed expert system.

Summary

The discussion of systems above confirms the usefulness and relevance of expert systems in society. Expert systems in general have notable limits. These include the inability to reason based on intuition and common sense as an expert would. Furthermore, they have limited knowledge and inherently make it difficult to integrate knowledge from other domains. Often, the learning process of an expert system is not automated and thus requires intervention which is not always possible [11]. It is argued that providing tools that exploit new ways to represent knowledge for use in solving problems, and not the duplication of human behavior in all aspects, is the goal of expert systems research [7]. A vast array of systems exist that perform exceptionally well in their domains. Although problems exist that are believed to be holding back the improvement of such programs, research continues in an attempt to address these issues.

5 Attempts at Knowledge Base Automation

In an attempt to address problems in knowledge acquisition, research and development efforts have been directed at creating tools to enable the automated construction of knowledge bases.

[1] http://www.exsys.com/

EXPERT SYSTEM CREATOR

Expert System Creator [24] is one of the tools available for the development of expert systems. It is a portable development and integration environment for knowledge management, expert systems construction and validation, and database integration. Over and above the tasks mentioned, Expert System Creator makes it possible to integrate the system with external projects. Representation of domain knowledge is achieved through the use of production rules, decision tables or classification trees. The Java programming language [1] is used to implement the tool. Expert System Creator uses CLIPS and JESS expert system shells for the reasoning process, and generates C/C++ and Java code for the decision tables and trees [24]. The use of such established tools contributes to the reliability of the output as well as the familiarity of the system to users. Pop et al.[24] note that the current status warrants further development, with respect to the representation, and support for automatic project documentation generation. Through the use of advanced widgets to support the knowledge acquisition phase and integration of fuzzy logic engines, enhancements in Expert System Creator are expected to address the current problems in knowledge representation. It is also noted that the integration of intelligent reporting tools will improve the performance of Expert System Creator. There is an apparent focus on integration with existing tools to achieve maximum efficiency. Although this may introduce complexities, when done correctly, significant improvement can be achieved. It is important to note the limitations in terms of flexibility and customizability that are inherent in robust systems like Expert System Creator. Providing a wide array of capabilities, Expert System Creator has been the tool of choice for various development endeavors [24], confirming its inherent strengths.

PROTEGE-2000

PROTEGE-2000, a knowledge management tool from Standford Medical Informatics [24], facilitates the construction of a domain ontology. In addition it customizes electronic knowledge acquisition forms used by experts and knowledge engineers during the knowledge acquisition process. Additionally, it encodes the knowledge into a database. However, in order to leverage the full power of the tool, it is necessary to integrate it with other existing knowledge acquisition tools.

KRITON

Diederich et al. [6] proposed a hybrid system, called KRITON, for automatic knowledge acquisition. KRITON [6; 4] combines cognitive science and AI methods to create knowledge bases using different representations. Declarative knowledge is acquired through automated interview methods whereas protocol analysis is used for the acquisition of procedural knowledge. KRITON uses the knowledge acquired to construct and present an intermediate representation language that is subsequently used by frame, rule and constraint generators to build up the final executable knowledge base. A hybrid approach was taken in an attempt to address the knowledge-acquisition bottleneck. Through the use of hybrid knowledge acquisition tools, KRITON captures different kinds of knowledge thereby filling the gaps resulting from the limitations of using a single tool for representation.

KRITON uses three knowledge acquisition methods [6]. From AI, KRITON borrows the knowledge engineering strategy of interviews. KRITON relies on a dialogue between the expert and knowledge engineer to acquire declarative knowledge. From cognitive science, protocol analysis is used to acquire procedural knowledge. This involves the processing of texts acquired through the transcription of protocols of loud thinking during a problem solving

process [6]. KRITON also makes use of incremental text analysis that enables acquisition of valuable knowledge from different sources. The acquired knowledge goes through an intermediate processing phase in which consistency checks and completion by inference takes place. The output of this phase is an intermediate representation language consisting of descriptive language for functional and physical objects and a propositional calculus. The introduction of an intermediate representation has proven to be an advantage in most implementations [4]. Intermediate output allows for monitoring and intervention where applicable. To complete the engineering phase, frame, rule and constraint generators build up the final representation using the intermediate representation language as input. The knowledge already in the knowledge base is used to guide subsequent elicitations thus aiding the incremental development and completion of the knowledge base.

KRITON presents an open, hybrid, and modular approach to the acquisition of declarative and procedural expert knowledge. An open system in this context provides facilities to enable extension and elaboration, while modularity guarantees ease of change and debugging. It is stated that the use of different acquisition tools and information from different sources makes the system hybrid. Hybrid systems have been noted to introduce inherent benefits [30]. Such features could arguably be the difference between an effective representation and an ineffective one. Diederich et al. [6] note however, the shortcomings of the KRITON system, namely, inaccurate and erroneous working of the protocol and text analysis. This has been addressed by the reliance on editors to aid employment and testing of the system. Furthermore, the lack of facilities to enable integration with other systems is a notable limitation. The emphasis on integration in the Expert System Creator project [24] presented a flexible and extensible system. In this domain, the ability to extend a given tool is one that is crucial to the success of a project.

KELVIN

KELVIN [19] is a tool designed for the automated construction of knowledge bases. This is achieved through the processing of a text corpus and then refinement of a knowledge base about people, locations, and organizations. Unlike most systems in its domain, which facilitate the acquisition of knowledge through interviews, KELVIN makes use of natural language processing software. Knowledge is extracted from various sources with care. KELVIN uses asserted facts and information obtained from sources other than documents obtained from web-searches and various online collaboration communities. The knowledge extraction process includes detection of named entities, relation extraction, intra-document co-reference resolution and entity disambiguation [19]. KELVIN, as with KRITON [6], makes use of external tools to aid the construction of knowledge bases. These include tools to enable document annotations, add-on packages to enable annotations about identified entities, querying tools, as well as tools to enable the smooth integration with other tools. When tested, KELVIN showed evidence of learning. The system presented correct facts that it supposedly extracted from the unstructured knowledge source, suggesting a significant and acceptable level of success. The use text analysis and natural language processing is common in the knowledge engineering domain. Adoption is encouraged in a suitable environment and knowledge domain. Ongoing work on the KELVIN system as conveyed includes scaling to large corpora, detection of contradictions, expansion of inferences, exploitation of information extraction confidence and branching out to various genres of text.

Automatic Knowledge Acquisition Tool

Rafea et al. [25] presented a tool for the automatic acquisition of knowledge for irrigation and fertilization expert systems. The efficacy of the tool was measured based on the tool's flexibility, usability, accuracy, and exception handling. The tool's architecture comprises four main components: a knowledge elicitation module, a library, a knowledge base generator, and a verification knowledge base. The knowledge elicitation module is responsible for the creation, maintenance, and storage of information acquired from experts. Furthermore, it fetches information from the library and combines this with information elicited from the user to create an intermediate knowledge structure. The library contains control and domain knowledge about the problem domain. The knowledge structure created by the elicitation module is used by the knowledge base generator to generate an executable system. This system is then verified by the verification knowledge base. The function of the verification knowledge base is to help experts test and verify the knowledge structure.

The modular structure of the system guarantees greater control. It enables the engineer to separate concerns, allowing focus to be placed on individual modules. Pre-defining problem solving methods and domain knowledge schema presents the domain experts with an easier task of filling in the gaps, which is achieved through automated interviews. This relieves developers of the burden of manually engineering the knowledge. Unlike most systems in its domain, which rely solely on elicited information, this system takes advantage of a separate differently structured base of knowledge. This arguably introduces improvements in the resulting representation. The presence of a verification knowledge base helps automate the testing phase, which in different circumstances would require continuous human intervention. None of the systems discussed above, that is, Expert Systems Creator, PROTEGE-2000, KRITON, and KELVIN incorporated automated verification facilities. The final output of the system is an executable knowledge base system. Rafea et al. [25] reported success of the proposed tool in the structuring of acquired knowledge by the use of relevant predefined domain knowledge to accelerate the knowledge base construction process. It is important to note that external effort would be required to construct and define the domain models, control knowledge, and domain ontologies contained in the library.

MORE

MORE [13], an intelligent knowledge acquisition tool is another attempt at addressing problems in knowledge acquisition. Aimed at assisting with the elicitation of knowledge from domain experts, MORE adds information acquired from experts through interviews to domain models of qualitative causal relations. The domain models are then used to generate diagnostic rules. MORE has functionality to elicit further information allowing the generation of a stronger set of diagnostic rules. MORE, which has seen use in several domains, presents a dynamic approach to knowledge elicitation and representation, which guarantees continuous relevancy of the knowledge base. Kahn et al. [13] demonstrate how MORE is used in parts of the drilling fluids domain. Furthermore, MORE has been used successfully to build systems to diagnose computer disk faults, network problems and circuit board manufacturing problems. Successful use of MORE in these domains is argued to be an indication of the power of the strategies MORE employs [13]. Future work is directed at using MORE in the development of expert systems in various other domains. Although MORE is not as robust as some of the systems discussed, it presents itself as an solution to problems in its domain.

ICONKAT

ICONKAT [8] is an integrated knowledge acquisition system designed to facilitate the design, construction, testing, maintenance and explanation of knowledge bases. It comprises three subsystems: the knowledge elicitation subsystem, maintenance subsystem, and the explanation subsystem. The knowledge elicitation subsystem facilitates the construction of domain models from an expert's knowledge. The maintenance subsystem assists engineers and experts with testing the system's performance, refining the knowledge, and maintaining the system through the use of appropriate support tools. Concept maps and repertory grids [8] are utilized to provide various perspectives of the domain knowledge. The explanation subsystem makes use of constructed domain models as the foundation of the resulting consultant system's explanation capability. The use of subsystems enables the effective management of individual subsystems. Teams can work on different subsystems to improve performance. ICONKAT at the time of publication was in routine use for the purpose of designing and constructing systems for diagnosing first pass cardiac functional images [8]. ICONKAT, compared with the systems discussed, has plenty to offer. The robustness of the tool suggests exceptional performance, however, this is often at the cost of ease of use and implementation and integration.

ROGET

ROGET [3] is a knowledge-based system designed for the task of acquiring the conceptual structure of a diagnostic system. Through a dialogue with an expert, ROGET acquires the expert system's conceptual structure, which is then presented to developers to aid the design and development of the expert system. In addition, ROGET recommends system-building tools as well as identifying the scope of the resulting expert system. Although there is little emphasis on knowledge acquisition, ROGET presents excellent functionality in the domain of expert systems development.

Exsys Corvid Expert System Development Tool

The Exsys Company [2] has a reputation for providing expert system tools. Exsys provides a semi-automated tool for building and fielding interactive consultation systems online. With significant focus on ease of use and learning, the Exsys Corvid Expert system development tool aims to enable the conversion of domain expert knowledge and decision-making logic into a structured symbolic form capable of being dynamically executed by the Exsys inference engine during consultation. The development of Corvid has been assisted by organisations and business entities over a significant number of years, with the ultimate goal of developing a system that would allow fast and easy creation of simple and or complex systems online. The presence of online tutorials makes it easy for non-technical users to build systems over short periods of time. The Corvid system enables the development of expert systems by facilitating the capturing of decision-making logic of a domain expert, the generation of a suitable user interface based on the knowledge gathered, and finally, the integration of the system with other available IT resources. The uptake of Exsys services by users in various domains reflects potential credibility and reliability.

The aim of the discussion in this section was to highlight advances in the effort of exploiting new ways to effectively represent knowledge that can be used to solve problems. As noted, some of the systems excel under certain circumstances and fall short in others. This necessitates further research and development of such tools.

[2] http://www.exsys.com/

6 Tools and techniques

It has been argued that the success in consultation of an expert system is highly dependent on the representation and organization of knowledge in the knowledge base [8]. Consequently, it is believed that the presence of a variety of tools and techniques to aid the knowledge representation process would result in the development of highly sophisticated systems that are capable of unprecedented levels of inference. It is not hard to argue for the necessity of research and development of such tools and techniques. These tools directly and indirectly aid the development of systems to enable effective knowledge acquisition and representation.

6.1 Client Interface

As discussed, the approach to knowledge elicitation has recently been the use of elicitation environments or systems. These present an interface that allows for a dialogue between a domain expert and a system. These allow the acquisition tools to facilitate automated interviews in an attempt to gather domain knowledge and the expert's expertise. A common characteristic of these systems is that they are web-based, which guarantees accessibility and mobility. Several tools exist that can be used in the development of interfaces that allowing for data input.

HTML

HTML (Hyper Text Markup Language) [10] is a markup language used for the creation of websites. An HTML document consists of HTML elements. A collection of these and plain text are used to describe the structure, content and semantics of a web document. HTML documents are read by a web browser, which composes them into visible structured web pages. It is possible to embed objects into web pages using HTML. Objects can include images, video, audio and other documents. Below is an example of an HTML document.

```
<!DOCTYPE html>
<html>
  <head>
    <title>I am a title</title>
  </head>
  <body>
    <p>Hello world!</p>
  </body>
</html>
```

The tags have special meaning attached to them, which is understood by web browsers. For example, the <!DOCTYPE html> tag denotes the document type to allow correct display of the web page.

As an open format, HTML is available for use without the need to acquire licenses. With HTML5 as the most recent version, HTML is supported by every web browser, consequently, it is used in the development of most applications, including mobile, desktop and web applications. Additional benefits include its simplicity, ease of use and maintenance, and availability at no cost. One of the notable benefits is its ability to allow embedding of scripts to extend a websites functionality. HTML as a markup language presents itself as an excellent solution for creating web interfaces that can be adapted into a client interface for a knowledge acquisition tool.

JavaScript

JavaScript [21] is a programming language developed jointly by Netscape Communications Corporation and Mozilla Foundation. It is argued to be the most popular programming language, with JavaScript interpreters believed to be installed on every computing device capable of browsing the web. Its primary use is the creation of interactive effects in web browsers. It introduces interactivity within web pages and aids the development of web applications. JavaScript code is often embedded within HTML documents to extend the functionality thereof.

JavaScript code is interpreted, thus making it faster than traditionally compiled programming languages. However, modern implementations may be compiled in situations where a browser feels this will result in better performance. Like HTML, JavaScript is widely supported with most browsers capable of running JavaScript code. JavaScript is introduced in an HTML document using a special tag. An example is shown below.

```
<script type="text/javascript">
//JavaScript code
document.write("Hello World!")
</script>

<script src="/path-to-file/.js" name="name"></script>
```

The first example executes the JavaScript code enclosed between the script tags. In this case, a simple 'hello world' text is displayed on a web page. The second example imports a script from a defined location. The result will therefore be whatever the code in the input file does. Often invoked upon the click of a button or submission of a form, JavaScript code enables the creation and manipulation of objects. For interactivity, JavaScript is arguably the best development tool available.

ASP.NET

ASP.NET [20] is a server side development framework developed by Microsoft for the creation of web sites, web applications and web services. Development models supported include web pages, model view controller, and web forms. It allows developers to create dynamic websites using a virtual interface.

Unlike notable alternatives, ASP.NET offers increased performance resulting from compiled code. Furthermore, it supports standard programming languages, hence eliminating the need to learn a new programming language. One other notable benefit of ASP.NET is its .NET framework foundation. This introduces .NET development tools that can be used to create Windows desktop applications as well as web applications. Developers can make use of the Visual Studio .NET tool. ASP.NET is widely used and favored as a framework for the development of web based interfaces. In addition, full technical support and in most cases, reliability is guaranteed, which may not be the case with open source alternatives.

A notable downside is the reliance of ASP.NET websites on ASP.NET compatible servers, that is, servers that support ASP.NET applications. At this point in time, ASP.NET is a viable and extremely capable tool for the development of interactive web interfaces.

Windows Forms

Windows Forms [**?**] is a smart client technology in the format of a graphical application programming interface (API) for the .NET Framework. It provides a platform for the development of robust applications with rich user interface elements by providing a set of managed libraries that simplify common applications. A form is a visual surface on which information is displayed to the user. These are created in a development environment like Visual Studio [3] by dragging and dropping user interface elements in a Windows Forms Designer. Forms can be used to request input from users, and communicate with remote computers over a network. Windows Forms applications are referred to as event-driven application. Actions are generated when the user interacts with the form, which are then processed through the execution of supplied code. This code, often referred to as code-behind, can be written in any of the languages supported by the .NET framework. Windows Forms has the benefit of allowing the creation of robust applications with relative ease. Most of the code is managed allowing the developer to focus solely on the domain logic. However, as a Microsoft technology, Windows Forms applications function best on machines running MicrosoftâĂŹs Windows operating system. This is seen by many as limitation and for this reason, attempts have been made to allow Windows form applications to run on non-Windows systems. As with ASP.NET, Windows Forms offers increased performance from compiled code. Furthermore, the underlying .NET framework introduces tools that can be used in conjunction with Windows Forms to create robust Windows applications.

6.2 Intermediate Representation

The benefits of an intermediate representation were highlighted in an earlier section. These include the ability to allow monitoring and management of the resulting representation. Several document formats exist that could allow the representation of somewhat intermediate data.

XML

XML [33] (eXtensible Markup Language) is a markup language designed for the store and transportation of data. Like HTML, XML is widely used in various domains and is favored for its simplicity, openness and extensibility. The structure of an XML representation is similar to that of HTML. As with HTML, it is possible to embed existing data within an XML representation. Markup languages present inherent benefits that render them excellent tools for storing data.

Comma Separated Values (CSV)

CSV, for Comma Separated Values, is a file format commonly used for the storage of delimited data in plain text format. Data elements are separated by comma characters with records terminated by newline characters. Although they are not used as much as notable alternatives, CSV files are used to exchange data between applications in various domains. Their structure permits the store of data representing sets or records. With little effort, CSV can be used to effectively represent a variety of structured data sets. For this reason, it is an option worth considering.

[3] http://msdn.microsoft.com/en-us/vstudio/aa718325.aspx

Plain text

This refers to textual data in ASCII format which is said by most to be the most portable format due to availability of machine independent applications that support it. Characters supported include numbers and symbols. The format is limited and as a result does not support any type of formatting. Plain text file sizes are usually small with documents taking up less than half the size of rich text documents containing the same number of characters. The simple structure plain text enables files to be easily processed by different applications owing to their immunity to computer architecture incompatibilities. A file is generally considered plain text if it maintains a human-readable forms. For this reason, files containing markup or meta data are considered plain text. Ultimately, plain text files are excellent for storing data likely to be processed by different applications.

6.3 Translation

To transform acquired knowledge into an executable representation, some sort of translation has to take place. This translation process takes as input an intermediate representation and produces as output a final executable knowledge base system to be used by reasoning programs. This warrants the creation of such a translator.

Java

The Java programming language [1] is an object-oriented programming language designed by Sun Microsystems. Java, which is a high-level or imperative programming language enables the development of computer programs using English based commands. Java source code is compiled into Java bytecode, which is verified and interpreted for a native architecture. Java offers great power to developers because of its extensive libraries, which makes it a first choice for skilled and non-skilled developers developing applications in various domains. Java is easy to use and extremely reliable. The widespread use of the language as well as the ubiquity of devices running Java applications is an indication of this. In addition, Java is secure, concurrent and platform independent. This makes it an attractive choice in the development of robust applications.

C#

The C# programming language [**?**] is one of the programming languages supported by the .NET framework. Derived from the C programming language, it is a simple and powerful type-safe object-oriented language that enables the development of secure and robust applications that run on the .NET Framework. Common applications include Windows client applications, XML Web services, distributed components, client-server applications and more. C# is specified as a common language infrastructure (CLI) language. The language provides features that make developing solutions faster and easier; these include type-safety, garbage collection, simplified type declarations, versioning and scalability support, and many more. The language boasts syntax similar to that of C, C++ and Java programming languages, but by introducing unique features such as delegates and lambda expressions, it stands out as better a choice. Microsoft has implemented Visual C# from the C# language, which is commonly used for developing Windows applications using the Visual studio development environment. As highlighted earlier, the .NET Framework class library provides inherent including access to many operating system services and other well-designed classes, which significantly improve the development process. As one of the languages supported by the .NET framework, C# can be used to implement the

different tiers of a Windows Forms application, that is, the presentation layer, domain layer, and data layer. Like Java, C# offers great power and flexibility to developers, which render it an attractive choice of a programming language for the development of robust applications in various domains, especially those intended to run on MicrosoftâĂŹs Windows operating system.

Other Options

A compiler is a program designed to translate source code expressed in one language into another language. A compiler generator is a tool that enables the efficient development of compilers. Compiler generators are available that can be used to create a compiler suitable for the translation of an intermediate representation into a final executable representation. The effectiveness of compiler generators and the efficiency with which they achieve their task makes them attractive tools worth considering. Rule generators are an option to consider. These facilitate the development of rule based knowledge bases.

6.4 Testing

VP-Expert

Much AI research has focused on improving the already existing technology and finding better ways to exploit it. As a result, tools have been developed to aid the efficient development of expert systems in various domains. A shell, which allows for the building and maintenance of knowledge based applications [**?**], is a notable example. VP-Expert [**?**] is an expert system shell developed by Paperback Software International. It is a powerful and easy-to-use tool that has seen use in educational and commercial applications. It contains everything needed to run an expert system. This includes an inference engine, a rule editor and a user interface. The inference engine uses specially structured knowledge base files to facilitate consultation. The editor provides a means for creating and editing the rules in a knowledge base, while the user interface allows for the asking of questions, presentation of traces and explanations of the flow of execution where needed. The VP-Expert shell is capable of backward and forward chaining. Furthermore, it applies the concept of confidence factors. An expert system shell like VP-Expert introduces efficiencies during the development of expert systems. Use of such a tool rids a developer of the effort needed to create the main components of the expert system like the: Inference engine, working memory and user interface. Furthermore, it has proven easier to build systems in various domains, since the only component required is the problem domain information. Using an expert system shell in development of expert systems has proven to significantly eliminate overhead. However, the input of new knowledge across systems can be inflexible and sometimes complicated. It is important to note that the sophistication of a shell has a direct influence on the overall performance of the expert system, that is, an expert system running a shell with a poorly implemented algorithm, irrespective of the quality and extensibility of the knowledge base, can yield an undesired inferior result. The VP-Expert software package comes in two genres: a student version or educational version, and a commercial one. These versions differ in the extent or depth of functionality they provide with the former providing limited functionality and the latter featuring fully-fledged features that significantly increase productivity and performance. An expert system shell like VP-Expert in this case provides sufficient functionality and capabilities, making it a credible tool for consultation. Considering the two main internal components of an expert system, that is, the knowledge base, and the inference engine, there is evidence to suggest that the use of an expert system shell like VP-Expert can notably improve the development and testing of an expert

system [**?**]. The shell as noted takes as input a knowledge base file. This can be fed into the system from an external source or created within the system itself using built-in templates and procedures for data management. These files make up the knowledge base used by the inference engine during consultation. The shell employs built-in routines for reasoning with the rules and facts supplied in the knowledge base. The system is easy to use and does not require exhaustive configuration effort to operate. The engineer is effectively left with the job of creating and structuring the domain knowledge into a knowledge base. The VP-Expert shell has been designed to accept knowledge base files with knowledge structured as IF-THEN rules. The structure of a VP-Expert rule is shown below:

```
RULE rulename
IF antecedent
THEN consequent;
```

The system requires that every rule has a unique name, an antecedent, and a consequent. An example of rule is given below:

```
RULE Diagnosis_of_measles
IF   Diagnosis = measles
THEN Treatment = penicillin;
```

VP-Expert permits the use of variables and values in rules, as shown above. The above rule states that upon the discovery of measles in a diagnosis, we may conclude that the treatment is *penicillin* by assigning variable `Treatment` the value `penicillin`.

7 Summary

A brief overview of expert systems was presented. This highlighted the applications, elements and examples of expert systems. The relevance of knowledge representation within the context of expert systems development was conveyed. Emphasis was on knowledge acquisition, noted to be the bottleneck of expert systems development, and the various knowledge representation schemes available. Systems were discussed that aimed to address problems observed in the process of knowledge acquisition and representation. Their strengths, weakness and differences were noted. Finally, tools and techniques that could aid the development of such systems were explored.

References

1. ARNOLD, K., AND GOSLING, J. *The Java Programming Language*. Addison-Wesley, 1998.

2. BECKERT, B. Introduction to Artificial Intelligence: First-order Logic. Online, 2005. Accessed on: May 20, 2014. Available from: `http://www.skit.edu.in/menu/CSE/r09/AI/08FirstOrderLogic.pdf`.

3. BENNETT, J. S. ROGET: a knowledge-based system for acquiring the conceptual structure of a diagnostic expert system. *Journal of Automated Reasoning 1* (1985), 49–74.

4. BOOSE, J. H. A survey of knowledge acquisition techniques and tools. *Knowledge Acquisition 1* (1989), 3–37.

5. BRACHMAN, R. J., AND LEVESQUE, H. J. *Knowledge Representation and Reasoning*. Morgan Kaufmann, 2004.

6. DIEDERICH, J., AND RUMANN, I. KRITON: a knowledge-acquisition tool for expert systems. *Int. J. Man-Machine Studies 26* (1987), 29–40.

7. DUDA, R. O., AND SHORTLIFE, E. H. Expert systems resarch. *Science 220* (1983), 261–268.

8. FORD, K., CANAS, A., JONES, J., STAHL, H., NOVAK, J., AND ADAMS-WEBBER, J. ICONKAT: an integrated constructivist knowledge acquisition tool. *Knowledge Acquisition 3*, 2 (1991), 215 – 236.

9. FREDERICK, H.-R., WATERMAN, D., AND LENAT, D. *Building Expert Systems*. Addison-Wesley Longman Publishing Co., Inc., 1983.

10. HTML.NET. HTML. Online. Accessed on: May 20, 2014. Available from: `http://html.net/`.

11. IANCU, E., MATES, D., AND VOICU, V. Considerations regarding the expert systems in the economy and the use method of the production systems based on rules. *Journal of Applied Computer Science & Mathematics 1* (2010), 63–65.

12. JAPANESE TECHNOLOGY EVALUATION CENTER. The Applications of Expert Systems. Online, May 1993. Accessed on: May 20, 2014. Available from: `http://www.wtec.org/loyola/kb/c1_s2.htm`.

13. KAHN, G., NOWLAN, S., AND MCDERMOTT, J. MORE: An intelligent knowledge acquisition tool. In *Proceedings of the Ninth International Joint Conference on Artificial* (1985).

14. KURMANGAZIYEVA, L. T., UTENOVA, B. E., AND MAILYBAYEVA, A. J. Expert systems and their use in oil and gas complex. *Life Science Journal 11* (2014), 392–395.

15. MARTIN, J., AND OXMAN, S. *Building Expert Systems: A tutorial*. Prentice-Hall, 1988.

16. MATSATSINIS, N., DOUMPOS, M., AND ZOPOUNIDIS, C. Knowledge acquisition and representation for expert systems in the field of financial analysis. *Expert Systems with Applications 12*, 2 (1997), 247 – 262.

17. MCCARTHY, J. *LISP 1.5 Programmer's Manual*. MIT Press, 1965.

18. MCDERMOTT, J. R1: A rule-based configurer of computer systems. *Artificial Intelligence 19* (1982), 39–88.

19. MCNAMEE, P., MAYFIELD, J., FININ, T., OATES, T., LAWRIE, D., XU, T., AND OARD, D. KELVIN: a tool for automated knowledge base construction. In *Proc. Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (June 2013), Association for Computational Linguistics. (demonstration paper).

20. MICROSOFT. ASP.NET. Online. Accessed on: May 20, 2014. Available from: `http://www.asp.net/`.

21. NETSCAPE COMMUNICATIONS CORPORATION. JavaScript. Online. Accessed on: May 20, 2014. Available from: `http://www.javascriptkit.com/`.

22. NIWA, K., SASAKI, K., AND IHARA, H. An experimental comparison of knowledge representation schemes. *The AI Magazine 5* (1984), 29–36.

23. ONCONTEXT. First Order Predicate Calculus. Online. Accessed on: May 30, 2014. Available from: http://www.ontotext.com/factforge/first-order-predicate-calculus.

24. POP, D., AND NEGRU, V. An extensible environment for expert system development. In *Knowledge-Based Intelligent Information and Engineering Systems*, V. Palade, R. Howlett, and L. Jain, Eds., vol. 2773 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2003, pp. 1016–1022.

25. RAFEA, A., AND HAZMAN, H. H. M. Automatic knowledge acquisition tool for irrigation and fertilization expert systems. *Expert Systems with Applications 24* (2003), 49–57.

26. ROUSE, M. First-order Logic. Online, September 2005. Accessed on: May 20, 2014. Available from: http://whatis.techtarget.com/definition/first-order-logic.

27. SAWYER, B., AND FOSTER, D. L. *Programming Expert Systems in Pascal*. Wiley Press, 1986.

28. SHORTLIFE, E. H. *Computer-Based Medical Consultations: MYCIN*. Elsevier, 1976.

29. SOWA, J. F. *Principles of Semantic Networks*. Morgan Kaufmann, 1991.

30. TANWAR, P., PRASAD, D. T., AND DATTA, D. K. Hybrid technique for effective knowledge representation & a comparative study. *International Journal of Computer Science & Engineering Survey 3* (2012), No. 4.

31. TEN BERGE, T., AND VAN HEZEWIJK, R. Procedural and declarative knowledge. *Theory & Psychology 9* (1999), 605–624.

32. VAN DE GEVEL, A. J. W., AND NOUSSAIR, C. N. *The Nexus Between Artificial Intelligence and Economics*. Springer, 2013.

33. W3SCHOOLS.COM. XML. Online. Accessed on: May 20, 2014. Available from: http://www.w3schools.com/xml/default.ASP.

34. WALDEN, M. Sports Injury Clinic. Online, 2000. Accessed on: May 20, 2014. Available from: http://www.sportsinjuryclinic.net/.