ICS 2405 Knowledge Based Systems

LAST REVISION ON March 16, 2015

EDWARD KARIUKI
(e.kariuki@jkuat.ac.ke)

P.O. Box 62000, 00200
Nairobi, Kenya
ICS 2405 KNOWLEDGE BASED SYSTEMS

Course description

This course covers the overview and definitions of artificial intelligence and knowledge based systems and development of programs containing a significant amount of knowledge about their application domain. The course includes a brief review of relevant AI techniques; case studies from a number of application domains, chosen to illustrate principles of system development; a discussion of technical issues encountered in building a system, including selection of knowledge representation, knowledge acquisition, etc.; and a discussion of current and future research. The course also provides hands-on experience in building an expert system (term project), knowledge representation, resolution, quantification, unification, frames; semantic nets; introduction to AI programming languages such as prolog.

Course aims

To construct computer programs that perform at high levels of competence in cognitive tasks in Knowledge-Based Systems. To understand and develop computational models of human intelligence in Cognitive Science.

Learning outcomes

Upon completion of this course you should be able to:

1. Be able to understand the knowledge-based systems representation
2. Be able to understand automatic reasoning
3. Be able to understand inductive and deductive learning
4. Be able to implement a small knowledge based system

Instruction methodology

- Lectures and tutorials
- Case studies
- Group discussions
- Assignments and learning activities
**Assessment information**

The module will be assessed as follows;

- 20% of marks from two (2) assignments to be submitted online
- 20% of marks from one written CAT to be administered at JKCUT main campus or one of the approved centres
- 60% of marks from written Examination to be administered at JKCUT main campus or one of the approved centres
Contents

1 Knowledge based systems 1
  1.1 Introduction ....................................................... 2
    1.1.1 AI Concepts ............................................. 2
  1.2 Intelligence ....................................................... 2
    1.2.1 Dictionary definition .................................... 2
    1.2.2 Types of Intelligence ..................................... 2
  1.3 What is AI? ....................................................... 3
    Important commercial advantages of AI are:... 6

2 Understanding kbs 11
  2.1 Problem Solving Techniques in AI .............................. 11
  2.2 Knowledge Based Systems-Overview ............................ 11
  2.3 KBS-Tasks ....................................................... 12
  2.4 What is Knowledge? ............................................. 12
    2.4.1 Classification of Knowledge ............................... 12
      • Criteria 1 ................................................. 12
      • Criteria 2 ................................................. 13
    2.4.2 Conceptual structure of a kbs (conceptual schema) ...... 13
    2.4.3 Components of A Knowledge Based System ............... 14
      •.1 The database/working memory includes a set of facts used to
         match against the IF (condition) parts of rules stored in the
         knowledge base. ......................................... 14
      •.2 The user interface is the means of communication between a
         user seeking a solution to the problem and a KBS. ......... 15
      •.3 Diagram Depicting components of KBS ... 15
2.4.4 Criteria for choosing to implement a KBS .......................... 15
- 4 Only certain types of applications are suited to knowledge based implementation. The following criterion is used in identifying suitable domains. The fields under study: ......................... 15

2.4.5 Characteristics of a KBSs ............................................. 16
- 5 High-quality performance. ......................... 16
- 6 No matter how fast the system can solve a problem, the user will not be satisfied if the result is wrong! ......................... 16
- 7 On the other hand, the speed of reaching a solution is very important. Even the most accurate decision or diagnosis may not be useful if it is too late to apply, for instance, in an emergency, when a patient dies or a nuclear power plant explodes. ......................... 16
- 8 KBSs apply heuristics to guide the reasoning and thus reduce the search area for a solution. ......................... 16
- 9 A unique feature of a KBS is its explanation capability. It enables the system to review its own reasoning and explain its decisions. ......................... 16
- 10 KBSs employ symbolic reasoning when solving a problem. Symbols are used to represent different types of knowledge such as facts, concepts and rules. .......................... 16
- 11 We should be aware that an expert is only a human and thus can make mistakes, and therefore, a KBS built to perform at a human expert level also should be “allowed” to make mistakes. .......................... 16
•.12 In KBSs, knowledge is separated from its processing (knowledge base and inference engine are split up). A conventional program is a mixture of knowledge and the control structure to process this knowledge. .................. 17

•.13 When an expert system shell is used, a knowledge engineer or an expert simply enters rules in the knowledge base. Each new rule adds some new knowledge and makes the KBS smarter. .... 17

2.4.6 KBS Comparison .......................................................... 17

2.5 Knowledge engineering ...................................................... 17

•.14 Practitioners of Knowledge Engineering are called Knowledge Engineers. .... 17

•.15 Thus the development of a Knowledge based system should follow a professional approach with respect to: ............... 17

2.5.1 Process of Knowledge Engineering ................................. 18

2.5.2 Tasks of knowledge engineers ......................................... 18

2.6 Types of knowledge based systems .................................... 18

•.16 In general, knowledge based systems are classified according to the human behavior they attempt to mimic. On the basis of this, we have: ................. 18

•.17 Expert Systems: They model the higher order cognitive functions of the human mind They are used to mimic the decision making process of the human mind. 18

•.18 Neural Networks: They model the brain at the biological level They are adept at pattern recognition and introduce the concept of learning into computing. . . . 18
2.7 Knowledge acquisition

•.19 Case Based Reasoning: Models the human ability to learn from past experience. They borrow from the legal system where past cases are used as a basis for making decisions in the present cases. 19

•.20 Fuzzy Logic systems: Provide mechanisms for handling uncertain knowledge. 19

•.21 Knowledge acquisition is the process of extracting knowledge (facts, procedures, rules) from human experts, books, documents, sensors or computer files and converting it into a form that can be stored and manipulated by the computer for purposes of problem solving. 19

•.22 It occurs throughout the entire development process for a KBS. 19

2.7.1 Acquiring Knowledge

•.23 The important characteristics of knowledge are that it is experiential, descriptive, qualitative, largely undocumented and constantly changing. 19

•.24 There are certain domains where all these properties are found and some where there are only a few. 19

•.25 The lack of documentation and the fact that experts carry a lot of information in their heads, makes it difficult to gain access to their knowledge for developing information systems in general and expert systems in particular. 19
<table>
<thead>
<tr>
<th>Section</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.7.2</td>
<td>Knowledge Acquisition Process</td>
</tr>
<tr>
<td>2.7.2.27</td>
<td>Identification: Identify the problem including data, criteria for solutions to meet, available resources, etc.</td>
</tr>
<tr>
<td>2.7.2.28</td>
<td>Conceptualization: Determine the key concepts and relationships by characterizing the data, flow of information, the domain structure, etc.</td>
</tr>
<tr>
<td>2.7.2.29</td>
<td>Formalization: Understand the underlying search space, uncertainty issues, etc.</td>
</tr>
<tr>
<td>2.7.2.30</td>
<td>Implementation: Translate acquired knowledge into the program.</td>
</tr>
<tr>
<td>2.7.2.31</td>
<td>Testing: Validate and verify.</td>
</tr>
<tr>
<td>2.7.3</td>
<td>Knowledge elicitation</td>
</tr>
<tr>
<td>2.7.3.32</td>
<td>It implies that knowledge acquisition is accomplished from a human expert.</td>
</tr>
<tr>
<td>2.7.3.33</td>
<td>Interaction between the expert and a program or knowledge engineer where the purpose of the program is to:</td>
</tr>
<tr>
<td>2.7.4</td>
<td>Issues with Knowledge Acquisition (Problems)</td>
</tr>
<tr>
<td>2.7.5</td>
<td>Characteristics of Knowledge Acquisition</td>
</tr>
<tr>
<td>2.8</td>
<td>Players in the Development Team</td>
</tr>
<tr>
<td>2.8.1</td>
<td>Domain Expert</td>
</tr>
<tr>
<td>2.8.2</td>
<td>Knowledge Engineer</td>
</tr>
</tbody>
</table>
•.1 The knowledge engineer is someone who is capable of designing, building and testing a KBS. ... 23

•.2 The knowledge engineer’s main tasks are: ... 23

2.8.3 Programmer ... 23

•.3 The programmer is the person responsible for the actual programming, describing the domain knowledge in terms that a computer can understand. ... 23

•.4 The programmer needs to have skills in symbolic programming in such AI languages as LISP, Prolog and OPS5 and also some experience in the application of different types of expert system shells. ... 24

•.5 In addition, the programmer should know conventional programming languages like Java, C, Pascal, FORTRAN and Basic. ... 24

2.8.4 Project Manager ... 24

•.6 The project manager is the leader of the KBS development team, responsible for keeping the project on track. ... 24

•.7 The project manager makes sure that all deliverable and milestones are met, interacts with the expert, knowledge engineer, programmer and end-user. ... 24

2.8.5 End-User ... 24

•.8 The end-user, often called just the user, is a person who uses the KBS when it is developed. ... 24

•.9 The user must not only be confident in the KBS performance but also feel comfortable using it. ... 24
• 10 Therefore, the design of the user interface of the KBS is also vital for the project’s success; the end-user’s contribution here can be crucial. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
4.1.3 Discrimination function

•.1 They are used in many pattern recognition techniques.

4.1.4 Classification techniques

•.2 The numeric methods are classification techniques that use measurements of geometric patterns.

•.3 They can be deterministic or statistical.

•.4 The non-numeric methods are classification techniques that depend on symbolic processing such as those dealing with fuzzy sets.

•.5 Our scope of treatment will only cover numeric-methods of classification include deterministic and statistical techniques.

5 Logic programming and prolog

5.1 Introduction

5.2 Prolog

•.6 Prolog is the most widely used language to have been inspired by logic programming research. Some features:

•.7 Prolog uses logical variables. These are not the same as variables in other languages. Programmers can use them as ‘holes’ in data structures that are gradually filled in as computation proceeds.

•.8 Unification is a built-in term-manipulation method that passes parameters, returns results, selects and constructs data structures.

•.9 Basic control flow model is backtracking.
•.10 Program clauses and data have the same form. ........................................ 39
•.11 The relational form of procedures makes it possible to define ‘reversible’ procedures. ........................................ 39
•.12 Clauses provide a convenient way to express case analysis and nondeterminism. ........................................ 39
•.13 Sometimes it is necessary to use control features that are not part of ‘logic’. ........................................ 39
•.14 A Prolog program can also be seen as a relational database containing rules as well as facts. ........................................ 39

5.2.1 Prolog as a ‘declarative’ language ........................................ 39
•.15 Clauses are statements about what is true about a problem, instead of instructions how to accomplish the solution. ........................................ 39
•.16 The Prolog system uses the clauses to work out how to accomplish the solution by searching through the space of possible solutions. ........................................ 39
•.17 Not all problems have pure declarative specifications. Sometimes extralogical statements are needed. ........................................ 39
•.18 Example: Concatenate lists a and b ........................................ 39

5.2.2 Structure of a Prolog program ........................................ 40
•.19 example ........................................ 40
• A Fact ........................................ 40
• A Rule ........................................ 41
•.1 A rule has the following form: ........................................ 41
•.2 The BODY may be a CONJUNCTION or a DISJUNCTION of predicates ........................................ 41
•.3 For instance: ........................................ 41

5.2.3 Querying a Prolog Database ........................................ 41
Once a prolog program is written, the user may ask different questions.

A question to Prolog is always a sequence of one or more predicates (called goals) as, for instance:

If the question consists of a predicate with no variables like, for instance

?- parent(bob, pat).% Is Bob a parent of Pat?

Prolog uses the closed world assumption which states that all relevant, true assertions are explicitly represented or are derivable from those explicitly represented.

If the question consists of one or more predicates with variables such as:

Prolog will look for all the instances of the variables from the question (X and Y in the above example) such that the predicates in the question logically follow from the facts and the rules in the program.

One says that Prolog tries to satisfy the goals "parent(Y, kenny)" and "parent(X, Y)".

Prolog returns the found pairs of the values of X and Y, or No if no such pair is found.

In all the queries the goals are satisfied by matching the questions with the facts.
If a goal cannot be satisfied in such a way, Prolog will try to use the rules from the program.

Prolog backtracks and tries any applicable rules if one goal fails.

5.3 Unification

5.3.1 Order of Clauses

5.3.2 Data objects

5.3.3 Variables

5.3.4 Compound Terms

Some atoms have built-in operator declarations so they may be written in a syntactically convenient form. The meaning is not affected. This example looks like an arithmetic expression, but might not be. It is just a term.

Constants are simply compound terms of arity 0.

owino means the same as owino().

Structured objects (structures) are objects that have several components.

A structured object corresponds to a function (or functional term) in logic.

All structured objects in Prolog are trees, represented in the program by terms.

It should be noted that the functions of Prolog are seen as compound data structures similar to the records of Pascal: they can be constructed and analyzed.

A list is a data structure that is either empty or consists of two parts: a head and a tail. The tail itself has to be a list.
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3.7</td>
<td>Some operations on lists-Membership</td>
<td>48</td>
</tr>
<tr>
<td>5.3.8</td>
<td>Concatenation</td>
<td>48</td>
</tr>
<tr>
<td>6.1.1.1</td>
<td>Types of Reasoning:</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>Reasoning from signs:</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>Cause and effect:</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>Reasoning by analogy:</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>Reasoning by example:</td>
<td>52</td>
</tr>
</tbody>
</table>
6.2 Reasoning Methods ............................................. 53
  6.2.1 Deductive Reasoning .................................... 53
    • Deductive Reasoning: Example Axiomatic System 53
    • Deductive Reasoning: Axiomatic Systems .... 54
    • Process of Deductive Reasoning .............. 54
      •.1 Multiple Inference .............................. 54
    • Inference in Propositional Logic and Calculus . 54
    • Inference in Predicate Calculus .............. 55
  6.2.2 Rules of Inference ....................................... 55
    • Deductive Reasoning: Axiomatic Systems .... 56
    • Resolution and Clausal Form ................ 56
  6.2.3 Resolution in FOL ...................................... 56
  6.2.4 Unification ............................................. 57
    • Unification Example .............................. 58
  6.2.5 Multiple inference in deductive reasoning .... 58
  6.2.6 Graphs ................................................ 58
    • Directed Graph: ................................... 59
    • Tree ............................................... 59
    • And/Or Graph ................................. 59
      •.1 Examples .................................... 60
  6.2.7 Backward and Forward Chaining .................. 60
  6.2.8 Inferencing .......................................... 60
  6.2.9 Criteria for choosing inferences ............... 60
  6.2.10 Forward Chaining .................................. 61
    • The Forward Chaining algorithm ............. 61
    • Reasons for forward chaining ............. 62
• Characteristics of forward chaining: 62

6.2.11 Backward Chaining 62
• The Backward Chaining algorithm 62
• Backward Chaining example (see diagram in the exercise) 63
• Reasons for Backward Chaining 64
• Characteristics of backward chaining: 64
• Backward vs. Forward chaining 64

6.2.12 Inductive Reasoning 65
6.2.13 Methods for inductive reasoning in expert systems include 65
• Decision trees 65
• ID3 65
• Case-based approach: reasoning by analogy 66

7 Uncertainty 69
7.1 Motivation 69
  • while traditional logic in principle is capable of capturing and expressing these aspects, it is not very intuitive or practical 69
7.2 Objectives 70
7.3 Sources of Uncertainty and Inexactness in Reasoning 70
7.4 Dealing with Uncertainty 70
7.5 Sources of Uncertainty 71
7.6 Uncertainty in Individual Rules 71
7.7 Uncertainty and Multiple Rules 71
7.8 Probability Theory 71
  7.8.1 Basics of Probability Theory 71
  7.8.2 Compound Probabilities 72
  7.8.3 Conditional Probabilities 72
  7.8.4 Advantages and Problems: Probabilities 72
  7.8.5 Bayesian Approaches 73
  • Advantages and Problems of Bayesian Reasoning 73
7.9 Certainty Factors Belief and Disbelief 74
7.9.1 Certainty Factors .............................................. 74

7.9.2 Advantages and Problems of Certainty Factors ............... 74

7.10 Dempster-Shafer Theory Evidential Reasoning ................... 75

7.10.1 Combination of Mass Probabilities ............................ 75

7.10.2 Evidential Reasoning ........................................ 75

7.10.3 Advantages and Problems of Dempster-Shafer ................. 76

• advantages ............................................... 76

• problems ............................................... 76

7.11 Important Concepts and Terms ................................... 76

7.12 Summary ...................................................... 77

8 Approximate Reasoning and intelligent agents ................. 79

8.0.1 Approximate Reasoning ....................................... 79

•.1 inference of a possibly imprecise conclusion from possibly imprecise premises ............................................. 79

•.2 useful in many real-world situations ........................... 79

•.3 often used synonymously with fuzzy reasoning .................. 79

•.4 although formal foundations have been developed, some problems remain ............................................. 79

• Approaches to Approximate Reasoning .......................... 79

• Advantages of Approximate Reasoning ......................... 80

• Problems of Approximate Reasoning ............................ 80

8.0.2 Fuzzy Logic .................................................... 80

•.1 relies on quantifying and reasoning through natural language ............................................. 80

• Advantages and Problems of Fuzzy Logic ...................... 81

8.0.3 Important Concepts and Terms ................................ 81

8.1 Intelligent agents (IA) ............................................ 81

8.1.1 Definition of intelligent agent ................................ 81

8.1.2 Agent and conventional programs ............................ 82

8.1.3 Components of agents ..................................... 82

8.1.4 Characteristics of intelligent agents .......................... 83
8.1.5 The need for agents ........................................ 83
8.1.6 TYPES OF AGENTS ......................................... 84
  •.1 1. Mobility: static or mobile; .................. 84
  •.2 2. Reasoning model: deliberative or re-
      active; ........................................... 84
  •.3 3. Ideal attributes: autonomy, learning
      and cooperation; .............................. 84
  •.4 4. Role: information, management; ........ 84
  •.5 5. Hybrid: combination of the above. ..... 84
      • Classifying agents from organizational and per-
        sonal view ...................................... 84
      • Classifying agents from characteristics point of
        view ............................................ 85

9 Expert Systems ............................................ 88
  9.1 Definitions: Expert Systems ......................... 88
     9.1.1 Expert Systems VS. Conventional Programs .... 88
     9.1.2 Expert Systems VS. Other AI Programs ........ 88
     9.1.3 Expert Systems VS. KBS ......................... 89
     9.1.4 Expertise ........................................ 90
          • Examples of expertise: ..................... 90
          • Facts about expertise: ..................... 90
     9.1.5 Experts .......................................... 90
     9.1.6 Knowledge Acquisition ........................... 91
     9.1.7 Inferencing ..................................... 92
     9.1.8 Knowledge Representation ...................... 92
  9.2 Types of Knowledge ...................................... 92
      • Expert System Problem Domains ............... 95
     9.2.1 Advantages of ES ............................... 96
     9.2.2 Disadvantages of ES ........................... 97
     9.2.3 Elements of Successful ES .................... 98
     9.2.4 Software for building Expert Systems ....... 99

10 Software Lifecycles ....................................... 101
   10.0.5 KADS Methodology ............................... 112
• Why Different Models? .................. 112
• KBS approach: .............................. 113
Solutions to Exercises ......................... 115
LESSON 1

Knowledge based systems

Learning outcomes

Upon completing this topic, you should be able to:

- Identify and explain knowledge based systems and artificial intelligence
1.1. Introduction

1.1.1. AI Concepts

- Intelligence and types of intelligence
- Understanding AI
- History of AI
- Characteristics of AI
- Modeling and AI System
- Main branches of AI
- Applications of AI:
  - Application areas
  - Application domains
- Problem solving techniques

1.2. Intelligence

1.2.1. Dictionary definition

- The ability to learn or understand or to deal with new situations
- The ability to apply knowledge to manipulate one’s environment or to think abstractly as measured by objective criteria (as tests)

1.2.2. Types of Intelligence

According to Howard Gardner’s multiple intelligence theory, there are various types of intelligence viz:

- **General intelligence**: Abilities that allow us to be flexible and adaptive thinkers, not necessarily tied to acquired knowledge.

- **Linguistic-verbal intelligence**: Use words and language in various forms / Ability to manipulate language to express oneself poetically
• Logical-Mathematical intelligence: - Ability to detect patterns / Approach problems logically / Reason deductively

• Musical intelligence: - Recognize nonverbal sounds: pitch, rhythm, and tonal patterns

• Spatial intelligence: - Typically thinks in images and pictures / Used in both arts and sciences

• Intrapersonal intelligence: - Ability to understand oneself, including feelings and motivations / Can discipline themselves to accomplish a wide variety of tasks

• Interpersonal intelligence: - Ability to "read people"—discriminate among other individuals especially their moods, intentions, motivations; / Adept at group work, typically assume a leadership role.

• Naturalist intelligence: - Ability to recognize and classify living things like plants, animals

• Bodily-Kinesthetic intelligence: - Use one’s mental abilities to coordinate one’s own bodily movements

Note: Understanding the various types of intelligence provides theoretical foundations for recognizing different talents and abilities in people

1.3. What is AI?

There is no agreed definition of the term artificial intelligence. However, there are various definitions that have been proposed. These are considered below.

• AI is a study in which computer systems are made that think like human beings. Haugeland, 1985 & Bellman, 1978.

• AI is a study in which computer systems are made that act like people. AI is the art of creating computers that perform functions that require intelligence when performed by people. Kurzweil, 1990.
• AI is the study of how to make computers do things, which at the moment people are better at. Rich & Knight. AI is the study of computations that make it possible to perceive, reason and act. Winston, 1992

• AI is considered to be a study that seeks to explain and emulate intelligent behaviour in terms of computational processes. Schalkeoff, 1990.

• AI is considered to be a branch of computer science that is concerned with the automation of intelligent behavior. Luger & Stubblefield, 1993.

Artificial Intelligence is the development of systems that exhibit the characteristics we associate with intelligence in human behavior:

• perception,

• natural language processing,

• reasoning, planning,

• problem solving,

• learning and adaptation, etc.

Development of AI: History
1943 McCulloch & Pitts: Boolean circuit model of brain
1950 Turing’s “Computing Machinery and Intelligence”
1950s Early AI programs, including Samuel’s checkers (draughts) program Newell & Simon’s Logic Theorist, Gelernter’s Geometry Engine
1956 Dartmouth meeting: “Artificial Intelligence” adopted
1966–74 AI discovers computational complexity, Neural network research almost disappears
1969–79 Early development of knowledge-based systems
1980–88 Expert systems industry booms
1985–95 Neural networks return to popularity (Return of ANN)
1988– Resurgence of probabilistic and decision-theoretic methods Rapid increase in technical depth of mainstream AI, “Nouvelle AI”: ALife, GAs, soft computing
Turing Test for Intelligence  Tests the ability of a computer system to act humanly
The aim is to determine if the human interrogator thinks he/she is communicating with a human.
To pass Turing Test the computer must:

- Process natural language;
- Represent knowledge;
- Reason;
- Learn and adapt to the new situations.

Total Turing test included vision & robotics.

Characteristics of AI

- **Symbolic Processing:** AI emphasizes manipulation of symbols rather than numbers. The manner in which symbols are processed is non-algorithmic since most human reasoning process do not necessarily follow a step by step approach (algorithmic approach).

- **Heuristics:** Are similar to rules of thumb where you need not rethink completely what to do every time a similar problem is encountered.

- **Inferencing:** This is a form of reasoning with facts and rules using heuristics or some search strategies.

- **Pattern matching** A process of describing objects, events or processes in terms of their qualitative features and logical and computational relationships.

- **Knowledge Processing** Knowledge consists of facts, concepts, theories, heuristics methods, procedures and relationships.

- **Knowledge bases.** Collection of knowledge related to a problem or an opportunity used in problem. Reasoning occurs based on this knowledge base.
Contrasting AI with Natural Intelligence

Important commercial advantages of AI are:-

1. AI is permanent as long as computer system and programs remain unchanged
2. AI offers ease of duplications and dissemination as compared to long apprenticeship for natural intelligence.
3. AI can be less expensive than natural intelligence.
4. AI being a computer system is consistent and thorough; natural intelligence may be erratic since people are erratic, they don’t perform consistently.
5. AI can execute certain tasks much faster than humans can.
6. AI can perform certain tasks better than many or even most people.

Natural Intelligence has the following advantages

1. Natural intelligence is creative, AI is uninspired- human always determine knowledge.
2. Natural intelligence enables people to benefit from use of sensory experience directly, while most AI systems must work with symbolic knowledge.

Modelling an AI System  A typical AI system consists of three subsystems, i.e.,

1. Perception Subsystem
2. Reasoning Subsystem
3. Action Subsystem(made of actuators/effectors)
Main Branches of AI  
AI research and application can be categorized into field’s along different dimensions.

- By the scope of cognitive/Intelligent capacities under research - the fields include: **Core cognitive facilities** e.g. searching, reasoning, learning, planning categorizing and recognizing. **Input/output facilities** e.g. sensors.

- By the type of major techniques - the fields include: Rule – Based system Case based system Neural networks Genetic programming Logical programming (e.g. Prolog) Functional programming(e.g. lisp)

- By the domains of application - the fields include: game playing theorem proving data mining etc

- By the design Symbolic AI – Designers explicitly program all of the AI “knowledge.” Connectionist AI – Designers “teach” an artificial neural network what the AI needs to “know.” Evolutionary AI – Designers give the AI the ability to refine itself

AI application areas

- Game Playing
  
  - Much of the early research in state space search was done using common board games such as checkers, chess, and the 15-puzzle
  
  - Games can generate extremely large search spaces. These are large and complex enough to require powerful techniques for determining what alternative to explore

- Automated reasoning and Theorem Proving
  
  - Theorem-proving is one of the most fruitful branches of the field
  
  - Theorem-proving research was responsible in formalizing search algorithms and developing formal representation languages such as predicate calculus and the logic programming language

- Expert System
One major insight gained from early work in problem solving was the importance of domain-specific knowledge. Expert knowledge is a combination of a theoretical understanding of the problem and a collection of heuristic problem-solving rules.

Current deficiencies:

- **Lack of flexibility**; if human cannot answer a question immediately, he can return to an examination of first principle and come up with something.
- **Inability to provide deep explanations**
- **Little learning from experience**

- **Natural Language Understanding and Semantics**

  - One of the long-standing goals of AI is the creation of programming that are capable of understanding and generating human language.

- **Modeling Human Performance**

  - Capture the human mind (knowledge representation).

- **Robotics**

  - A robot that blindly performs a sequence of actions without responding to changes or being able to detect and correct errors could hardly be considered intelligent.
  - It should have some degree of sensors and algorithms to guide it.

- **Machine Learning**

  - Learning has remained a challenging area in AI.
  - An expert system may perform extensive and costly computation to solve a problem; unlike human, it usually doesn’t remember the solution.
  - Examples include:
    - Decision tree learning
    - Genetic algorithms
    - Neural networks
Application Domains of AI

1. Application domain areas include:
   
2. Military
3. Medicine
4. Industry
5. Entertainment
6. Education
7. Business

Problem Solving Techniques in AI

Problems are tackled in AI using two main approaches namely:

- Search technique
- Modelling natural phenomena (e.g. evolution and neural networks).
Revision Questions

Example 📘 What is Artificial Intelligence?
- Describe the Turing Test.

Solution:

Artificial intelligence is a study in which computer systems are made that act like people. People in this group include Kurzweil, 1990 who particularly thought that AI is the art of creating computers that perform functions that require intelligence when performed by people. Rich & Knight were also in this group and they considered AI as the study of how to make computers do things which at the moment people are better at.

The turing test is whereby The human interrogator thinks he/she is communicating with a human.

To pass Turing Test the computer must:
- Process natural language;
- Represent knowledge;
- Reason;
- Learn and adapt to the new situations.

EXERCISE 1. 📘 What is intelligence? and Can intelligence be mechanized?

References and Additional Reading Materials


LESSON 2

Understanding kbs

2.1. Problem Solving Techniques in AI

Recall, typical problem solving (and hence many AI) tasks can be commonly reduced to:

- Modeling natural phenomena, and
- Search

Techniques that model natural phenomena generally concentrate on the concept of knowledge representation.

In a nutshell, some problems highlight search whilst others knowledge representation.

2.2. Knowledge Based Systems-Overview

Definition: a knowledge-based system is a software system capable of supporting explicit representation of knowledge and of exploiting it through appropriate reasoning mechanism in order to provide high-level problem solving performance.

- Knowledge based systems are systems that use knowledge representation as their central concept.
- Knowledge based systems are developed for a specific task domain.
- It is therefore a specific, dedicated, computer based problem solver, able to face complex problems, which if solved by man, would require advanced reasoning capabilities like:
  - deduction,
  - hypothetical reasoning,
  - model based reasoning,
  - analogical reasoning,
- learning etc. Knowledge based systems are developed for a specific task domain.
2.3. KBS-Tasks

Typical tasks for which knowledge based systems can be built include:

- Diagnosis
- Scheduling
- Design
- Planning
- Configuration

2.4. What is Knowledge?

- **Knowledge** is a theoretical or practical understanding of a subject or a domain. Knowledge is also the sum of what is currently known, and apparently knowledge is power.

- Those who possess knowledge are called **experts**.

- Anyone can be considered a domain expert if he or she has deep knowledge (of both facts and rules) and strong practical experience in a particular domain. The area of the domain may be limited.

- In general, an expert is a skilful person who can do things other people cannot.

2.4.1. Classification of Knowledge

- **Criteria 1**
  - **Priori Knowledge**
    - Universally true and cannot be denied without contradiction e.g. mathematical laws, logical statements.

  - **Posteriori Knowledge**
    - Represents knowledge that is verified using sensory experiences. Such knowledge can be denied based on new knowledge without the need for contradictions.
• Criteria 2
  • Procedural (Compiled) Knowledge
    – Considers the manner in which things work under different sets of circumstances. Includes step-by-step sequences and how-to types of instructions; it may also include explanations.

  • Declarative Knowledge
    – Descriptive representation of knowledge. It tells us facts: what things are e.g. “Elizabeth Kiiru is a young Kenyan woman”.

  • Tacit Knowledge
    – Unconsciously knowing how to do something. By definition, tacit knowledge is not easily shared. Tacit knowledge consists often of habits and culture that we do not recognize in ourselves. The concept of tacit knowledge refers to a knowledge which is only known to you and hard to share with someone else, which is the opposite from the concept of explicit knowledge.

  • Meta Knowledge
    – Knowledge about knowledge.

2.4.2. Conceptual structure of a kbs (conceptual schema)

A knowledge-based system is generally composed of two parts:

• Central Part (Kernel)
  – This implements the basic problem solving capabilities of the knowledge-based system.
  – The kernel is in turn composed of three main components:
    * Knowledge base (KB)
    * Reasoning mechanism-inference engine.
    * Working memory
Peripheral Part: This provides additional functions necessary for a practical and effective use of the knowledge based system e.g. user interface, explanation system, learner, knowledge acquisition facility, etc.

Diagram - Conceptual Schema

2.4.3. Components of A Knowledge Based System

Based on the conceptual scheme above, a typical knowledge based system has the following components.

- A knowledge base
- An inference engine
- A user interface
- Explanation facilities
- Learning facilities

The knowledge base contains the domain knowledge useful for problem solving. In a rule-based expert system, the knowledge is represented as a set of rules. Each rule specifies a relation, recommendation, directive, strategy or heuristic and has the IF (condition) THEN (action) structure. When the condition part of a rule is satisfied, the rule is said to fire and the action part is executed.

- The database/working memory includes a set of facts used to match against the IF (condition) parts of rules stored in the knowledge base.
The inference engine carries out the reasoning whereby the KBS reaches a solution. It links the rules given in the knowledge base with the facts provided in the database.

The explanation facilities enable the user to ask the KBS how a particular conclusion is reached and why a specific fact is needed. A KBS (specifically an expert system) must be able to explain its reasoning and justify its advice, analysis or conclusion.

2. The user interface is the means of communication between a user seeking a solution to the problem and a KBS.

3. Diagram Depicting components of KBS

2.4.4. Criteria for choosing to implement a KBS

4. Only certain types of applications are suited to knowledge based implementation. The following criterion is used in identifying suitable domains. The fields under study:

- Should be able to be reduced to a series of rules rather than mathematical formula or equations. – Knowledge based systems not applicable if domain has a large number of complex calculations.

- Is well understood so that well-defined knowledge can be formulated and represented in Computer

- Should not encompass problems which take too short (i.e. less than half an hour) or too long (larger than say one week) a time to solve.
ICS 2405 Knowledge Based Systems

- Should have general agreement among recognized experts in the domain to be computerized.
- The knowledge within the problem domain should be sufficiently large to warrant the development of a knowledge-based system.
- ‘Tame’ experts who are agreeable to involvement.

However, any application that requires access to specialist knowledge is a potential area for knowledge based systems technology.

2.4.5. Characteristics of a KBSs


*6. No matter how fast the system can solve a problem, the user will not be satisfied if the result is wrong!

*7. On the other hand, the speed of reaching a solution is very important. Even the most accurate decision or diagnosis may not be useful if it is too late to apply, for instance, in an emergency, when a patient dies or a nuclear power plant explodes.

*8. KBSs apply heuristics to guide the reasoning and thus reduce the search area for a solution.

*9. A unique feature of a KBS is its explanation capability. It enables the system to review its own reasoning and explain its decisions.

*10. KBSs employ symbolic reasoning when solving a problem. Symbols are used to represent different types of knowledge such as facts, concepts and rules.

*11. We should be aware that an expert is only a human and thus can make mistakes, and therefore, a KBS built to perform at a human expert level also should be "allowed" to make mistakes.
In KBSs, knowledge is separated from its processing (knowledge base and inference engine are split up). A conventional program is a mixture of knowledge and the control structure to process this knowledge.

When an expert system shell is used, a knowledge engineer or an expert simply enters rules in the knowledge base. Each new rule adds some new knowledge and makes the KBS smarter.

### 2.4.6. KBS Comparison

<table>
<thead>
<tr>
<th>Human Experts</th>
<th>KBS</th>
<th>Conventional Programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use knowledge in the form of rules of thumb or heuristics to solve problems in a narrow domain.</td>
<td>Process knowledge expressed in the form of rules and use symbolic reasoning to solve problems in a narrow domain.</td>
<td>Process data and use algorithms, a series of well-defined operations, to solve general numerical problems.</td>
</tr>
<tr>
<td>In a human brain, knowledge exists in a compiled form.</td>
<td>Provide a clear separation of knowledge from its processing.</td>
<td>Do not separate knowledge from the control structure to process this knowledge.</td>
</tr>
<tr>
<td>Capable of explaining a line of reasoning and providing the details.</td>
<td>Trace the rules fired during a problem-solving session and explain how a particular conclusion was reached and why specific data was needed.</td>
<td>Do not explain how a particular result was obtained and why input data was needed.</td>
</tr>
</tbody>
</table>

### 2.5. Knowledge engineering

This refers to the process of developing knowledge-based systems.

Practitioners of Knowledge Engineering are called Knowledge Engineers.

Thus the development of a Knowledge based system should follow a professional approach with respect to:

- qualifications,
• personality, and
• attributes.

2.5.1. Process of Knowledge Engineering

Consists of five activities:

• Knowledge acquisition

• Knowledge Validation

• Knowledge Representation

• Inferencing and

• Explanation.

2.5.2. Tasks of knowledge engineers

• Extracting knowledge from people. (KA)

• Representing knowledge in some form. (KR)

• Including knowledge in a computer program which makes use of the knowledge.

• Validating the software system produced.

2.6. Types of knowledge based systems

•.16. In general, knowledge based systems are classified according to the human behavior they attempt to mimic. On the basis of this, we have:

•.17. Expert Systems: They model the higher order cognitive functions of the human mind They are used to mimic the decision making process of the human mind.

•.18. Neural Networks: They model the brain at the biological level They are adept at pattern recognition and introduce the concept of learning into computing.
**19. Case Based Reasoning:** Models the human ability to learn from past experience. They borrow from the legal system where past cases are used as a basis for making decisions in the present cases.

**20. Fuzzy Logic systems:** Provide mechanisms for handling uncertain knowledge.

2.7. **Knowledge acquisition**

**21.** Knowledge acquisition is the process of extracting knowledge (facts, procedures, rules) from human experts, books, documents, sensors or computer files and converting it into a form that can be stored and manipulated by the computer for purposes of problem solving.

**22.** It occurs throughout the entire development process for a KBS.

2.7.1. **Acquiring Knowledge**

**23.** The important characteristics of knowledge are that it is experiential, descriptive, qualitative, largely undocumented and constantly changing.

**24.** There are certain domains where all these properties are found and some where there are only a few.

**25.** The lack of documentation and the fact that experts carry a lot of information in their heads, makes it difficult to gain access to their knowledge for developing information systems in general and expert systems in particular.

**26.** Therefore, knowledge engineers have devised specialized techniques to extract and document this information in an efficient and expedient manner.
2.7.2. Knowledge Acquisition Process

- **Identification**: Identify the problem including data, criteria for solutions to meet, available resources, etc.

- **Conceptualization**: Determine the key concepts and relationships by characterizing the data, flow of information, the domain structure, etc.

- **Formalization**: Understand the underlying search space, uncertainty issues, etc.

- **Implementation**: Translate acquired knowledge into the program.

- **Testing**: Validate and verify.

2.7.3. Knowledge elicitation

- **It implies that knowledge acquisition is accomplished from a human expert.**

- **Interaction between the expert and a program or knowledge engineer where the purpose of the program is to:**
  - Elicit knowledge in a systematic way,
• Store such knowledge in some representation, and
• Compile the represented knowledge into a runnable program.

• Sources of Knowledge
• The Knowledge Engineer acquires knowledge from a number of sources:

  • Written sources, such as books, manuals, standard procedures- An important source of knowledge at the start, when the knowledge engineer learns the vocabularies and general topics of the domain.

  • Experts

  • Observations of the actual process

  • Expertise of a procedural nature, in which a sequence of tasks is performed, direct observation of the operation also can provide input to the knowledge base.

  • Experts make decisions based on an abstract and intellectual process that even experts themselves have difficulty understanding and verbalizing. The knowledge engineer should elicit knowledge from the expert by helping the expert to discover and communicate this process, using a host of knowledge acquisition methods.

• Methods of Knowledge Elicitation

  • Face to face interview with experts – knowledge engineers interview the experts.

  • Protocol analysis – This is a documentation of how the expert behaves and processes information during problem solving. Usually the experts think aloud. Observation – The experts are observed at work.

  • Questionnaires –This is where questions are sent to experts for responses.

  • Analysis of documented knowledge – It involves extraction of knowledge from sources such as books, journals, articles, magazines, mass media materials.
• Rule induction (computer aided knowledge acquisition) – rule induction can be viewed as a system that accepts examples and develops classification rules.

2.7.4. Issues with Knowledge Acquisition (Problems)

• Machine representation is lower in form than human usage of knowledge;

• Many participants are involved and they have varied backgrounds causing communication challenges

• Experts may not express their knowledge.

• Mismatch between the way experts hold their knowledge and the way computers represent knowledge.

2.7.5. Characteristics of Knowledge Acquisition

• Knowledge acquisition is a labour and time intensive process.

• Currently knowledge bases for knowledge based systems are crafted by hand, this is a severe limitation on the rapid deployment of such systems.

• Biggest ‘bottleneck’ in system development.

• Most expensive part (money, time & labour).

• Automating KA the ultimate goal.

2.8. Players in the Development Team

There are five members of the KBS development team:

• the domain expert

• the knowledge engineer

• the programmer

• the project manager

• the end-user.

The success of their KBS entirely depends on how well the members work together.
2.8.1. Domain Expert

- The domain expert is a knowledgeable and skilled person capable of solving problems in a specific area or domain.
- This person has the greatest expertise in a given domain.
- This expertise is to be captured in the KBS.
- Therefore, the expert must:
  - be able to communicate his or her knowledge
  - be willing to participate in the expert system development
  - commit a substantial amount of time to the project.
- The domain expert is the most important player in the KBS development team.

2.8.2. Knowledge Engineer

- The knowledge engineer is someone who is capable of designing, building and testing a KBS.
- The knowledge engineer’s main tasks are:
  - interviews the domain expert to find out how a particular problem is solved.
  - establishes what reasoning methods the expert uses to handle facts and rules and decides how to represent them in the expert system.
  - chooses some development software or an expert system shell, or looks at programming languages for encoding the knowledge.
  - responsible for testing, revising and integrating the expert system into the workplace.

2.8.3. Programmer

- The programmer is the person responsible for the actual programming, describing the domain knowledge in terms that a computer can understand.
•4. The programmer needs to have skills in symbolic programming in such AI languages as LISP, Prolog and OPS5 and also some experience in the application of different types of expert system shells.

•5. In addition, the programmer should know conventional programming languages like Java, C, Pascal, FORTRAN and Basic.

2.8.4. Project Manager

•6. The project manager is the leader of the KBS development team, responsible for keeping the project on track.

•7. The project manager makes sure that all deliverable and milestones are met, interacts with the expert, knowledge engineer, programmer and end-user.

2.8.5. End-User

•8. The end-user, often called just the user, is a person who uses the KBS when it is developed.

•9. The user must not only be confident in the KBS performance but also feel comfortable using it.

•10. Therefore, the design of the user interface of the KBS is also vital for the project’s success; the end-user’s contribution here can be crucial.
Example 🎨. What is knowledge acquisition?

Solution:
Knowledge acquisition is the process of extracting knowledge (facts, procedures, rules) from human experts, books, documents, sensors or computer files and converting it into a form that can be stored and manipulated by the computer for purposes of problem solving.

Exercise 2. ✒️ Describe the issues in knowledge acquisition.

Learning Activities/assignments

• Explain how you may acquire knowledge to build a system

• Discuss ways of acquiring knowledge.

• Discuss problems associated with knowledge acquisition.
LESSON 3

Knowledge representation

- A framework for storing knowledge and manipulating knowledge
- A set of syntactic and semantic conventions that make it possible to describe things
- There is always a relationship between the form in which knowledge is represented and the way in which the knowledge is used. You can use domain-specific or general-purpose representation.

3.0.6. What to represent

Let us first consider what kinds of knowledge might need to be represented in AI systems:

- Objects - Facts about objects in our world domain. e.g. cars have wheels, cows are herbivores.
- Events - Actions that occur in our world. e.g. The power-sharing deal was sealed on 28th February 2008
- Performance - A behavior like playing volleyball involves knowledge about how to do things like serve, boost, spike and block.
- Meta-knowledge - knowledge about what we know.
- Thus in solving problems in AI we must represent knowledge
- There are two entities to deal with, i.e. facts and representation of these.
- Facts - truths about the real world and what we represent. This can be regarded as the knowledge level

Representation of the facts which we manipulate. This can be regarded as the symbol level since we usually define the representation in terms of symbols that can be manipulated by programs.
3.1. Uses of knowledge

Knowledge may be used in the following ways:

3.1.1. Learning

acquiring knowledge. This is more than simply adding new facts to a knowledge base. New data may have to be classified prior to storage for easy retrieval, etc. Interaction and inference with existing facts to avoid redundancy and replication in the knowledge and and also so that facts can be updated.

3.1.2. Retrieval

The representation scheme used can have a critical effect on the efficiency of the method. Humans are very good at it. Many AI methods have tried to model humans.

3.1.3. Reasoning

Infer facts from existing data.

- If a system only knows: *Daisy is a world class model. All world class models are beautiful.*

- If things like *Is Daisy a world class model?* or *Are world class models beautiful?* are asked then the answer is readily obtained from the data structures and procedures. However a question like *Is Daisy beautiful?* requires reasoning.

Note: The above are all related. For example, it is fairly obvious that learning and reasoning involve retrieval etc.

3.2. Properties of Knowledge Representation Systems

The following properties should be possessed by a knowledge representation system.

- Representational Adequacy

the ability to represent the required knowledge;

- Inferential Adequacy
the ability to manipulate the knowledge represented to produce new knowledge corresponding to that inferred from the original;

- Inferential Efficiency the ability to direct the inferential mechanisms into the most productive directions by storing appropriate guides;

- Acquisitional Efficiency the ability to acquire new knowledge using automatic methods wherever possible rather than reliance on human intervention.

Note:

- To date no single system optimizes all of the above.

- Currently, logic based representation is still the theoretical foundation of various kinds of knowledge representation.

3.2.1. Knowledge Representation Schemes

A number of knowledge representation schemes (or formalisms) have been used to represent the knowledge of humans in a systematic manner. This knowledge is represented in a knowledge base such that it can be retrieved for solving problems. Amongst the well-established knowledge representation schemes are:

- Natural language
- Logic
- Predicate and Propositional Logic
- Conceptual or Terminological Logics
- Production Rules
- Semantic Networks
- Frames
- Conceptual Dependency Grammar
- Conceptual Graphs
- Ontology
- XML / RDF
3.3. Logic Representation

3.3.1. Representation, Reasoning, and Logic

The objective of knowledge representation is to express knowledge in a computer-tractable form, so that agents can perform well.

A knowledge representation language is defined by:

- Its syntax which defines all possible sequences of symbols that constitute sentences of the language (grammar to form sentences)
- Its semantics determines the facts in the world to which the sentences refer (meaning of sentences) Each sentence makes a claim about the world.
- Its proof theory (inference rules and proof procedures)

A representation that has clear syntax and semantics is a logic representation.

- Logics include:
  - Propositional logic
  - Predicate logic (FOPC) HOPC
  - Fuzzy Logic
  - Temporal logic
  - Description Logics
  - e.t.c.

- Propositional Logic: Syntax
Syntax-3

Example ....

Solution: ....

EXERCISE 3.
LESSON 4

Semantic networks

- A semantic network is a structure for representing knowledge as a pattern of interconnected nodes and arcs. Nodes in the net represent concepts of entities, attributes, events, values. Arcs in the network represent relationships that hold between the concepts.

  - A semantic network is a graph theoretic data structure whose nodes represent word senses and whose arcs express semantic relationships between these word senses.

  - Concepts can be represented as hierarchies of interconnected concept nodes (e.g. animal, bird, canary)

  - Any concept has a number of associated attributes at a given level (e.g. animal \( \rightarrow \) has skin; eats etc.)

- Some concept nodes are super class of other nodes (e.g. animal \( > \) bird) and some are sub classes (canary \( < \) bird)

  - Often, sub classes inherit all the attributes of their super class concepts.

  - Some instances of a concept are exempted from the having certain attributes held by their concept classes (e.g. ostrich is excepted from flying)
4.0.2. Advantages of Semantic Networks

- **Advantages**
  - Easy to visualise and understand.
  - The knowledge engineer can arbitrarily define the relationships.
  - Related knowledge is easily categorised.
  - Efficient in space requirements.
  - Node objects represented only once.
  - Standard definitions of semantic networks have been developed.

- **Disadvantages of Semantic Networks**

**Limitations**

- The limitations of conventional semantic networks were studied extensively by a number of workers in AI.
- Many believe that the basic notion is a powerful one and has to be complemented by, for example, logic to improve the notion’s expressive power and robustness.
- Others believe that the notion of semantic networks can be improved by incorporating reasoning used to describe events.
- Limited in handling quantifiers e.g. “Every dog has bitten a postman”
- Binary relations are usually easy to represent, but some times is difficult e.g. try to represent the sentence: “Nengo caused trouble to the party”.

4.1. Frames

These are knowledge representation formalisms in which stereotyped information on objects are represented.

**Features:**

- capture object attributes and their values;
• search done by matching:

**Structure:**

• Node and collection of attributes(slots).

4.1.1. **Advantages:**

• can cope with missing values- close matches are presented.

4.1.2. **Disadvantages:**

• has been hard to implement, especially inheritance.

**Example** 🎨. Try to represent the following two sentences into the appropriate semantic network diagram:

- isa(person, mammal)
- instance(Duggan Kariuki, person) all in one graph
- team(Duggan Kariuki, Cardiff)
- score(Gor Mahia, AFC, 1-3)

Misiko gave Ngicu the book
4.1.3. Discrimination function

Discrimination function is a function that maps input features into a classification space.

•.1. They are used in many pattern recognition techniques.

4.1.4. Classification techniques

There are two broad classes of classification techniques. These are

• numeric and
• non-numeric methods.

•.2. The numeric methods are classification techniques that use measurements of geometric patterns.

•.3. They can be deterministic or statistical.

•.4. The non-numeric methods are classification techniques that depend on symbolic processing such as those dealing with fuzzy sets.

•.5. Our scope of treatment will only cover numeric-methods of classification include deterministic and statistical techniques

• k-NEAREST NEIGHBOUR CLASSIFICATION: The nearest neighbour classification is one in which a new pattern is placed in the class to which it is closest.

• Hamming distance
• Euclidean distance
• City-block distance
• Square distance
Revision questions

Example. what is a semantic network? give its a advantages and disadvantages

Solution:
Advantages.
Easy to translate to predicate calculus.
Disadvantages.
Cannot handle quantifiers; nodes may have confusing roles or meanings; searching may lead to combinatorial explosion; cannot express standard logical connectives; can represent only binary or unary predicates.

Exercise 4. Describe blind search.

Exercise 5. Describe guided search.

Exercise 6. Describe the knowledge representation formalisms giving their advantages and disadvantages.

Exercise 7. Learning Activities
What is pattern recognition?
What is classification?
Explain the meanings of the terms feature vector and feature space.
What is discrimination function?
Describe the nearest-neighbour classification technique.
LESSON 5
Logic programming and prolog

Contents

• Basic introduction to logic
• Structure of a Prolog program
• Basic Prolog programming
• Data objects and structures
• Summary

5.1. Introduction

• Logic programming refers to a family of languages and an associated programming style based on writing programs as a set of assertions (facts and inference rules).

• The execution of a logic program is a kind of deduction on the specified facts and inference rules.

• Prolog is such a logic programming language.

5.2. Prolog

• Prolog is the most widely used language to have been inspired by logic programming research. Some features:

• Prolog uses logical variables. These are not the same as variables in other languages. Programmers can use them as ‘holes’ in data structures that are gradually filled in as computation proceeds.

• Unification is a built-in term-manipulation method that passes parameters, returns results, selects and constructs data structures.

• Basic control flow model is backtracking.
•.10. Program clauses and data have the same form.

•.11. The relational form of procedures makes it possible to define ‘reversible’ procedures.

•.12. Clauses provide a convenient way to express case analysis and nondeterminism.

•.13. Sometimes it is necessary to use control features that are not part of ‘logic’.

•.14. A Prolog program can also be seen as a relational database containing rules as well as facts.

5.2.1. Prolog as a ‘declarative’ language

•.15. Clauses are statements about what is true about a problem, instead of instructions how to accomplish the solution.

•.16. The Prolog system uses the clauses to work out how to accomplish the solution by searching through the space of possible solutions.

•.17. Not all problems have pure declarative specifications. Sometimes extralogical statements are needed.

•.18. Example: Concatenate lists a and b
5.2.2. Structure of a Prolog program

- Programs consist of procedures.
- Procedures consist of clauses.
- Each clause is a fact or a rule.
  - facts for describing object features or relationships, and
  - rules for inferring new properties and relationships from other properties and relationships.
- Programs are executed by posing queries.

**.19. example**

![Diagram of Prolog program structure]

- **A Fact**
  - A fact has the format:
    - `predicatename(arguments)`
      * Where arguments-ANY TERM
      * (CONSTANT, VARIABLE, FUNCTION EXPRESSION)
  - For instance:
female(monica).

* expresses the fact that monica is a female

parent(monica, bob).

* expresses the fact that monica is a parent of bob

● A Rule
● A rule has the following form:

- X :- Y1, Y2, ... ,Yn.

  - (Simply HEAD:-BODY. Read as HEAD if BODY)
  - which may be read as "X is true if Y1 is true and Y2 is true and ... and Yn is true"
  - X represents the head of the rule and Y1, Y2, ... ,Yn represents the body.

  - ":-" is read "if"

● The BODY may be a CONJUNCTION or a DISJUNCTION of predicates

● For instance:

  - mother(X, Y) :- parent(X, Y), female(X).

  - means X is mother of Y if X is parent of Y and X is a female.

5.2.3. Querying a Prolog Database

● Once a prolog program is written, the user may ask different questions.

  ● A question to Prolog is always a sequence of one or more predicates (called goals) as, for instance:

  ● ?- parent(X, rachael).

  - % Who is Rachael’s parent ?
• \(?-\) predecessor(monica, X).

> % Who are Monica’s successors ?

Example 3

\begin{verbatim}
parent(monica, bob).
parent(tom, bob).
parent(tom, rachael).
parent(bob, ann).
parent(bob, pat).
parent(pat, kenny).
female(monica).
male(tom).
male(bob).
female(rachael).
female(ann).
female(pat).
male(kenny).
\end{verbatim}

% Monica is a parent of
% Notice that in Prolog the
% are written in lower case
% Monica is a female
% Tom is a male
5.2.4. Prolog Program-Continued

mother(X, Y) :- % X is the mother of Y if
    parent(X, Y), % X is a parent of Y and
    female(X). % X is female

% Note that a variable starts with an upper
predecessor(X, Y) :- % Rule pr1: X is a predeces
    parent(X, Y). % X is a parent

predecessor(X, Y) :- % Rule pr2: X is a predece
    parent(X, Z), % X is a parent
    predecessor(Z, Y).% Z is a prede

5.2.5. How Prolog answers questions

6. If the question consists of a predicate with no variables like, for instance

7. ?- parent(bob, pat).% Is Bob a parent of Pat ?

    • Prolog tries to demonstrate that this fact logically follows from the facts and
      the rules in the program.

    – It returns Yes if the proof is successful and
    – No otherwise.
8. Prolog uses the **closed world assumption** which states that all relevant, true assertions are explicitly represented or are derivable from those explicitly represented.

- Based on CWA, any assertion that is neither explicitly represented nor derivable, is considered to be false.

9. If the question consists of one or more predicates with variables such as:

- `?- parent(Y, kenny), parent(X, Y).`

  - % Find X and Y such that Y is a parent of Kenny
  - % and X is a grandparent of Kenny

10. Prolog will look for all the instances of the variables from the question (X and Y in the above example) such that the predicates in the question logically follow from the facts and the rules in the program.

11. One says that Prolog tries to satisfy the goals "parent(Y, kenny)" and "parent(X, Y)".

12. Prolog returns the found pairs of the values of X and Y, or No if no such pair is found.

- `?- parent(Y, kenny), parent(X, Y).`

  - X = bob
  - Y = pat

13. In all the queries the goals are satisfied by matching the questions with the facts.

14. If a goal cannot be satisfied in such a way, Prolog will try to use the rules from the program.

15. Prolog backtracks and tries any applicable rules if one goal fails.
5.3. Unification

- Two terms unify if substitutions can be made for any variables in the terms so that the terms are made identical. If no such substitution exists, the terms do not unify.

- The Unification Algorithm proceeds by recursive descent of the two terms.
  - Constants unify if they are identical
  - Variables unify with any term, including other variables
  - Compound terms unify if their functors and components unify.

5.3.1. Order of Clauses

When writing Prolog programs one should use the following problem solving heuristic:

- try the simplest idea first.

5.3.2. Data objects

- Syntactically, all data objects in Prolog are terms.

- As known from logic, a term could be a constant, a variable, or a function the arguments of which are terms.

Complete Syntax of Terms
5.3.3. Variables

- Variables are strings of letters, digits and underscore characters.
  - They start with an upper-case letter or an underscore character.
  - e.g. haschild(X) :- parent(X, Y). % X has a child if %there is Y such that X is the parent of Y

- Each time a single underscore character occurs in a clause it represents a new anonymous variable.
  - haschild(X) :- parent(X, _). % X has a child if % X is the parent of someone

- If the anonymous variable appears in a question clause then its value is not output when Prolog answers the question:
  - ?- parent(X, _). % Who is a parent ? Or
    - X = monica; % Find X such that X is a %parent of someone.
    - X = tom;
    - X = bob;
    - X = pat

- The lexical scope of variable names is one clause.

5.3.4. Compound Terms

- Some atoms have built-in operator declarations so they may be written in a syntactically convenient form. The meaning is not affected. This example looks like an arithmetic expression, but might not be. It is just a term.

- Constants are simply compound terms of arity 0.

- owino means the same as owino()
5.3.5. Structures

• 19. Structured objects (structures) are objects that have several components.
  
  • The components themselves can, in turn, be structures.

• 20. A structured object corresponds to a function (or functional term) in logic.
  
  • E.g.: date(december, 3, 1983)

• 21. All structured objects in Prolog are trees, represented in the program by terms.

• 22. It should be noted that the functions of Prolog are seen as compound data structures similar to the records of Pascal: they can be constructed and analyzed.
  
  • They are not functions applied to arguments to denote the corresponding result.

5.3.6. Lists

• 23. A list is a data structure that is either empty or consists of two parts: a head and a tail. The tail itself has to be a list.

• 24. The empty list is represented as[].

• 25. A list with Head and Tail is represented as .(Head, Tail)

• 26. Lists are handled in Prolog as a special case of binary trees.

• 27. For improved readability Prolog provides a special notation for lists:
  
  • [ Item1, Item2, ...] or
  
  • [ Head | Tail] or
  
  • [ Item1, Item2, ... | Others]
5.3.7. Some operations on lists-Membership

- We define the relation member( X, L), where X is an object and L is a list.
- X is a member of L if either
  - X is the head of L, or X is a member of the tail of L.
    - member( X, [X | Tail] ).
    - member( X, [Head | Tail] ) :- member( X, Tail ).

5.3.8. Concatenation

- 28. We define the relation conc( L1, L2, L3) where L1 and L2 are two lists and L3 is their concatenation.
- conc corresponds to append in LISP.
- 29. conc( [], L, L). % the concatenation of an empty list [] %with a list L is L
- 30. conc([X | L1], L2, [X | L3] ) :- % a non-empty list has the form [X | L1]
- 31. conc( L1, L2, L3). % the concatenation of [X | L1] with a list L2 is % the list [X | L3], where L3 is the concatenation of L1 and L2
- 32. This program can be used for concatenating given lists:
  - ?- conc([a,b,c], [1,2,3], L).
    - L = [a,b,c,1,2,3]
- 33. conc could also be used in the inverse direction for decomposing a given list into two lists:
  - ?- conc(U, V, [a,b,c]).
    - U = []
    - V = [a,b,c];
    - U = [a]
Add
Adding an item X to the beginning of a list L can be programmed as a relation

- add(X, L, [X | L]). % X is added in the front of the list

Delete

- Deleting an item X from a list L can be programmed as a relation

  - del(X, L, L1) where L1 is equal to the list L with the item X removed.
  - del(X, [X | Tail], Tail). % if X is the head of the list then the result of deletion of X is the tail of the list.
  - del(X, [Y | Tail], [Y | Tail1]) :- % if X is in the tail, then it should be deleted from there. del(X, Tail, Tail1)

Sublist

sublist(S, L) :- % S is a sublist of L if conc(L1, L2, L), % L is composed of two lists, L1 and L2, and conc(S, L3, L2). % L2 is composed of S and some L3.

5.3.9. Operators

Any atom may be designated an operator. The only purpose is for convenience; the only effect is how the term containing the atom is parsed. Operators are ‘syntactic sugar’. Operators have three properties: position, precedence and associativity.
Examples of operator properties
Revision questions

Example. Give an example of a Prolog program consisting of procedures, clauses, facts, rules, and queries.

Solution:

Exercise 8. ....
LESSON 6
Inference and Knowledge Processing

Outline

• Process of reasoning and reasoning types

• Reasoning methods
  
  – Deductive Reasoning
    * Single inference
    * Multiple inference
  
  – Inductive reasoning

6.1. The Process of Reasoning

Reasoning: The process of drawing inferences or conclusions; moving from what is known (fact) to what is unknown (inference).

6.1.1. Types of Reasoning:

1. Reasoning from signs:

2. Cause and effect:

3. Reasoning by analogy:

4. Reasoning by example:

• Reasoning from signs:
Assuming that a sign or symbol represents or indicates something specific.

• Example: making a phone call and receiving a busy signal. The assumption is that someone is already on the line.

The sign, busy signal is used to infer that someone is on the phone. There should be enough information to relate the sign to the inference

• Beware of stereotypes like: “Luos are proud and loudmouthed”. E.g. given Wafula is a Luo, then you infer Wafula is proud and loudmouthed.
• **Cause and effect:**
Inferring that one event causes another

  • “I was late because my alarm didn’t go off.”

or reasoning with the result in mind

  • “If I don’t study, I will fail the test.”

Important that there is enough information to link the events.
Beware of assuming that things that happen sequentially are causally related

  • “I got HIV/AIDS because I had unprotected sex”

• **Reasoning by analogy:**
Drawing a parallel between two similar events, people, or traits.
Based on comparisons.

  • Example: If all mothers are women, and Ann is a mother, then Ann must be a woman.

Care should be taken to always check analogies for validity, because these don’t always work backwards.

  • Example: Because Ann is a woman doesn’t mean she’s a mother.

• **Reasoning by example:**
Basing conclusions on facts, illustrations, or cases.

  • Example: Conclusions drawn from the results of polls or surveys.

  • Example: Mende has an approval rating of 40%, which is 15% more than his closest challenger Mbu; so he will be elected in the next election.”

When drawing conclusions, always ask, “Do I have enough facts to support this?”
6.2. Reasoning Methods

Knowledge Representation allows encoding or formalization of quantitative facts and knowledge. Reasoning and inference allows generation of new and useful knowledge, conclusions and recommendation. Inference methods fall into two general categories.

- Deductive Reasoning.
- Inductive Reasoning.

6.2.1. Deductive Reasoning

- The method used in propositional and predicate calculus.
- The inference engine combines the rule (or predicates) to arrive at the final answer.
- Starts with a set of axioms, or commonly accepted premises that one cannot derive from a system itself.
- Using the axioms and proven formulae, deductive reasoning can deduce new conclusions.

- **Deductive Reasoning: Example Axiomatic System**

1. Akidiva is a man.
2. Akidiva is a Luhya
3. All Luhyas are Kenyans.
4. Kibaki is a leader.
5. All Kenyans are either loyal to Kibaki or hate him.
6. Everyone is loyal to someone.
7. People only try to cheat leaders they are not loyal to.
8. Akidiva tried to cheat Kibaki.
9. All men are persons
**Deductive Reasoning: Axiomatic Systems**

1. man (akidiva)
2. luhya (akidiva)
3. \( \forall X (luhya(X) \rightarrow kenyan(X)) \)
4. leader (kibaki)
5. \( \forall X (kenyan(X) \rightarrow loyal_to(X, kibaki)) \lor hate(X, kibaki) \)
6. \( \forall X \exists Y ((person(X) \rightarrow person(Y) \land loyal_to(X,Y)) \)
7. \( \forall X \forall Y (person(X) \land person(Y) \land try_to_cheat(X,Y) \rightarrow \neg loyal_to(X,Y)) \)
8. try_to_cheat (akidiva, kibaki)
9. \( \forall X (man(X) \rightarrow person(X)) \)

**Process of Deductive Reasoning**

The Inference process consists of two parts:

- **Single Inference.**
  - Process of applying inference rules to combine two pieces of knowledge to derive a new premise.
  - May be an intermediate step or the final recommendation.

- **Multiple Inference**
  - The sequence or order of applying the single inference process to the entire knowledge base in order to derive final conclusions.

*Note: The knowledge-processing capability of an intelligent system is composed by the two parts above.*

**Inference in Propositional Logic and Calculus**

There are a number of rules of inference in propositional logic that allow derivation of new statements from combining two previously accepted ones.

- Modus ponens
- Hypothetical syllogism
• Modus tollens

Modus Ponens has to be a tautology for it to be true.
Tautology: The truth value of a statement form is always T no matter what the truth values of its components are.
Contradiction: Opposite of tautology. Always F.

• Inference in Predicate Calculus

Predicate calculus uses rules of inference in propositional calculus. Modus ponens and hypothetical syllogism are used.
Predicate calculus has predicates and quantifiers (Existential and Universal) that do not exist in propositional calculus.

• Quantifiers: Provide more power for the system because the content of propositions can be broken into predicates - unification and resolution.

6.2.2. Rules of Inference

<table>
<thead>
<tr>
<th>Rules of Inference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Law of Detachment: p → q</td>
</tr>
<tr>
<td>p</td>
</tr>
<tr>
<td>Therefore, q</td>
</tr>
<tr>
<td>Law of the Contrapositive: p → q</td>
</tr>
<tr>
<td>Therefore, ¬q → ¬p</td>
</tr>
<tr>
<td>Chain rule (Law of the Syllogism): p → q</td>
</tr>
<tr>
<td>q → r</td>
</tr>
<tr>
<td>Therefore, p → r</td>
</tr>
<tr>
<td>Law of Disjunctive Inference: p v q</td>
</tr>
<tr>
<td>p v q</td>
</tr>
<tr>
<td>Therefore, p</td>
</tr>
<tr>
<td>Law of Double Negation: ¬¬p</td>
</tr>
<tr>
<td>Therefore, p</td>
</tr>
</tbody>
</table>

ICS 2405 Knowledge Based Systems
• Deductive Reasoning: Axiomatic Systems

A system using the below approach is called a natural deductive system

1) man (akidiva)
2) luhya (akidiva)
3) $\forall X (luhya(X) \rightarrow kenyan(X))$
4) leader (kibaki)
5) $\forall X (kenyan(X) \rightarrow loyal_to(X, kibaki) \lor hate(X, kibaki))$
6) $\forall X \exists Y (person(X) \rightarrow person(Y) \land loyal_to(X,Y))$
7) $\forall X \forall Y (person(X) \land person(Y) \land try_to_cheat(X,Y) \rightarrow \neg loyal_to(X,Y))$
8) try_to_cheat (akidiva, kibaki)
9) $\forall X (man(X) \rightarrow person(X))$

• Resolution and Clausal Form

6.2.3. Resolution in FOL

An algorithms for proving facts true or false by virtue of contradiction e.g. to prove X is true, we show that the negation of X is not true e.g.

Fact:

• Not feathers (X) or bird (X)
Sentence 1 states “either X does not have feathers or else X is a bird“
Sentence 2: X has feathers.
To prove “X is a bird”, we first add an assumption that is the negative of that predicate. The facts then become:

- Same as above.
- Same as above
- Not bird (X)

In sentences 1 & 2, not feathers (X) and feathers (X) cancel each other out. Resolving 1 & 2 produces the resolved, sentence and which is added.

- Same as above.
- Same as above
- Same as above
- bird(X)

It is clear 3 & 4 cannot both be true: either X is a bird or not-thus a contradiction. Thus we have just proved our first assumption -3 is false, 4 must be true.

6.2.4. Unification

Takes two sentences in FOL and finds a substitution that makes them look the same required to prove theorems using resolution.
If two predicates are identical, then they match by definition.
If one or both contain variables, then appropriate substitutions must be found using unification as follows:

- A variable can be replaced by a constant.
- A variable can be replaced by another variable.
- A variable can be replaced by a predicate, as long as the predicate does not contain that variable.
• **Unification Example**

Consider hates (X, Y)

- hates (Onyango, Kales)
- hates (Okeyo, Cabbage)

Unify 2 with 1 by binding Onyango to X and Kales Y, also unify 3 with 1 by binding Okeyo to X and Cabbage to Y.

- Hates (X, Vegetable (Y))
- Hates (Onyango, Vegetable (Kales))
- Hates (Z, Kales)

Unify 6 with 1- X with Z, Y with Kales.
Unify 4 and 5, bind Onyango to X, Kales to Y.

*Note: Prolog uses a generalized version of unification algorithm called match.*

**6.2.5. Multiple inference in deductive reasoning**

Single inference has a solid theoretical foundation.

- Applying rules of inference to the entire knowledge base requires tools for analysis and heuristics that could improve the speed of system response - MI

Multiple Inference (MI)- Involves testing rules or predicates to find the one which must fire next.

The commonly used methods of multiple inference are: Graphs, Trees, and the And/Or

- Graph.
- Backward Chaining.
- Forward Chaining.

**6.2.6. Graphs**

- Uses graph theory.
- Easily understandable.
- Consists of nodes connected by arcs
• **Directed Graph:**
The direction of an arc connecting two nodes is important.

  • Path: The sequence of nodes and arcs connecting the beginning node of the path to the end node of the path
  • Cycle: The ending node of the path is the same as the beginning node.

• **Tree**
A directed graph with no cycles which starts with a beginning node (root) and ends with ending nodes (leaves).
Paths start from the root and end with leaves in a tree
There is only one path from the root to each leaf.
Used extensively in A.I. And Expert Systems e.g. and/or graph and decision tree.

```
  Root
    A
    /   \
   /     \ 
  B       C
   |     /   |
  E     F   G
        /   / \
       /     H
```

Directed graph, with branches,
A - B - E etc.

• **And/Or Graph**
Rules and predicates are shown with nodes and arcs.
The two arcs in the AND diagram are connected by another arc signifying that both A and B must be true to reach conclusion C from this rule.
6.2.7. Backward and Forward Chaining.

Two well known and commonly used methods of multiple inference in expert systems are

- Backward Chaining.
- Forward Chaining.

6.2.8. Inferencing

The process of accessing the knowledge stored in the knowledge base in order to make conclusions.

The program for inferencing is typically called the inference engine or control program.

In rule based systems it is also called a rule interpreter.

Purpose: Direct the search through the knowledge base. The engine determines

- Which rules to investigate,
- Which to eliminate, and
- Which will attribute to a match.

6.2.9. Criteria for choosing inferences

- The logical reasoning process.
- Design features of the system.
• The inputs and where they come from.

• The outputs and where they go.

• How the inputs and outputs map to forward or backward chaining.

6.2.10. Forward Chaining

The system requires the user to provide facts pertaining to the problem. The inference engine tries to match each fact with the antecedent or if part of a rule. If the match succeeds, the rule fires and the consequent of that rule is established and is added to the known facts of the case. The process continues until the inference engine has drawn all possible conclusions by matching facts to antecedents of rules in the knowledge base. Among these could be goals of the system.

• The Forward Chaining algorithm
  1. Enter new data
  2. Fire forward chaining rules
  3. Rule actions infer new data values
  4. Go to step 2
  5. Repeat until no new data can be inferred
  6. If no solution, rule base is insufficient
• **Reasons for forward chaining**
  When everything that can possibly be concluded about a set of data needs to be known.
  Many conclusions are possible from a single data item.
  It is important to communicate new conclusions to the user immediately.

• **Characteristics of forward chaining:**
  • Good for monitoring, planning, and control
  • Looks from present to future.
  • Works from antecedent to consequent.
  • Is data-driven, bottom-up reasoning.
  • Works forward to find what solutions follow from the facts.
  • It facilitates a breadth-first search.
  • The antecedents determine the search.
  • It does not facilitate explanation.

6.2.11. **Backward Chaining**

The multiple inference starts with a question or goal.
The inference engine starts from the consequent of a rule and goes backward to the antecedent or IF part of the rule.

• **The Backward Chaining algorithm**
  1. State a specific goal (question)
  2. Find rules which resolve the goal
  3. At runtime, answer questions to satisfy the antecedents of the rules as required
  4. Obtain a result (goal resolved or not)
• **Backward Chaining example** (see diagram in the exercise)

The goal is to establish whether G is true.
The inference engine checks to see which rule has G as its consequent.

  • Rule 4 fires

To conclude from this that G is true, E or F must be shown to be true.

E becomes the current goal of the system
The inference engine checks to see which rule has E as its consequent

  • Rule 2 fires

The current goal becomes D
The inference engine checks to see which rule has D as its consequent
There is none.
It asks the user to provide information about D
D is True
The inference engine concludes that E is true and G is also true.

D is False The system fails to establish the truth of E
In this case, it backtracks and picks up F to get at the goal G.
B and C become the current goal. B is not the consequent of any rule and the system asks the user if it is true.
B is False

  • No matter what C is, the conditional part of rule 3 is false and the inference engine fails to establish whether G is true.
B is True

- The inference engine pursues C etc

**Reasons for Backward Chaining**
- There is a clear set of statements which must be confirmed or denied.
- A large number of questions could be asked of the user, but typically only a few are necessary to resolve a situation.
- It is desirable to have interactive dialogue with the user.
- Rule execution depends on data gathering which may be expensive or difficult.

**Characteristics of backward chaining:**
- Good for Diagnosis.
- Looks from present to past.
- Works from consequent to antecedent.
- Is goal-driven, top-down reasoning.
- Works backward to find facts that support the hypothesis.
- It facilitates a depth-first search. The consequents determine the search.
- Facilitates explanation.

**Backward vs. Forward chaining**

**Backward Chaining**
- Useful when the number of goals is small and the and/or graph does not have numerous levels.
- In some systems, the user has the choice of telling the inference engine which goal it must pursue - Goal selection by the user.

**Forward Chaining**  Performs well when the number of goals is large, the user has a given set of facts at the start and wants to find the implications of these facts.
6.2.12. Inductive Reasoning

Most inference rules in A.I. have their theoretical basis in deductive reasoning. Most learning is through experience which is an inductive process.

- Exposure to a number of similar circumstances that lead to a certain consequence leads to generalization and development of rules for dealing with that group of circumstances.

Inductive Reasoning: The process of going from specific cases to general rules. Example: Qualitative methods e.g. statistics.

6.2.13. Methods for inductive reasoning in expert systems include

- Decision trees.
- ID3
- Case-based approach: reasoning by analogy

• Decision trees

Must have a set of representative cases for which the goal or decision value is known. E.g. the following data for the possible cases in the Mortgage Loan Case.

<table>
<thead>
<tr>
<th>Steady job</th>
<th>Adequate Assets</th>
<th>Adequate income</th>
<th>Approve Loan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

• ID3

The tree in its raw form is inefficient. Try to find a tree with the minimum number of nodes. Method used is called ID3 developed by Quinlan. Example is as shown.
• **Case-based approach: reasoning by analogy**
  - Intuition – You don’t pursue theory – the answer just appears
  - Heuristics – Based on rules of thumb based upon experience
  - Generate and test – trial and error.
  - Abduction – reasoning back from a true conclusion to the premises that may have caused the conclusion.
  - Default – assume general or common knowledge by default.
  - Auto epistemic – Self-knowledge, reasoning about your own knowledge.
  - Non-monotonic – Previous knowledge may be revised when new evidence is obtained.
  - Formal reasoning – Synthetic manipulation of data structures to deduce new facts by following prescribed rules of inferences.
  - Metalevel reasoning – manipulating knowledge about knowledge.
Revision questions

Example. ForwardChainingexample

Solution:

- Forward Chaining using the same knowledge base consisting of the following rules
  1. $A \rightarrow C$
  2. $D \rightarrow E$
  3. $B \land C \rightarrow F$
  4. $E \lor F \rightarrow G$

- A and B are the known facts
  - Using rule 1 inference engine concludes that $C$.
  - Using rule 3 it concludes that $F$ is true.
  - Using rule 4 it then concludes that $G$ is true.
EXERCISE 9. Learning Activities

What is inference?
Is inference problem solving?
What are AI inference strategies?
Describe forward and backward chaining inference controls.
What does an inference engine do?
LESSON 7

Uncertainty

Overview Reasoning and Uncertainty
Motivation
Objectives
Sources of Uncertainty and Inexactness in Reasoning
  • Incorrect and Incomplete
  • Knowledge Ambiguities
  • Belief and Ignorance
Probability Theory
  • Bayesian Networks
Certainty Factors
  • Belief and Disbelief
Dempster-Shafer Theory
  • Evidential Reasoning

Important Concepts and Terms

Summary

7.1. Motivation

Reasoning for real-world problems involves missing knowledge, inexact knowledge, inconsistent facts or rules, and other sources of uncertainty

• while traditional logic in principle is capable of capturing and expressing these aspects, it is not very intuitive or practical
  • explicit introduction of predicates or functions
many expert systems have mechanisms to deal with uncertainty
  • sometimes introduced as ad-hoc measures, lacking a sound foundation
7.2. Objectives

- be familiar with various sources of uncertainty and imprecision in knowledge representation and reasoning

- understand the main approaches to dealing with uncertainty
  - probability theory
    - Bayesian networks
    - Dempster-Shafer theory
  - important characteristics of the approaches
    - differences between methods, advantages, disadvantages, performance, typical scenarios

- evaluate the suitability of those approaches
  - application of methods to scenarios or tasks

- apply selected approaches to simple problems

7.3. Sources of Uncertainty and Inexactness in Reasoning

reasoning under uncertainty and with inexact knowledge frequently necessary for real-world problems heuristics ways to mimic heuristic knowledge processing methods used by experts empirical associations experiential reasoning based on limited observations probabilities objective (frequency counting) subjective (human experience) reproducibility will observations deliver the same results when repeated

7.4. Dealing with Uncertainty

expressiveness can concepts used by humans be represented adequately? can the confidence of experts in their decisions be expressed? comprehensibility representation of uncertainty utilization in reasoning methods correctness probabilities adherence to the formal aspects of probability theory relevance ranking probabilities don’t add up to 1, but the “most likely” result is sufficient long inference chains tend to result in extreme (0,1) or not very useful (0.5) results computational complexity feasibility of calculations for practical purposes
7.5. Sources of Uncertainty

data data missing, unreliable, ambiguous, representation imprecise, inconsistent, subjective, derived from defaults, ... expert knowledge inconsistency between different experts plausibility “best guess” of experts quality causal knowledge deep understanding statistical associations observations scope only current domain, or more general knowledge representation restricted model of the real system limited expressiveness of the representation mechanism inference process deductive the derived result is formally correct, but inappropriate derivation of the result may take very long inductive new conclusions are not well-founded not enough samples samples are not representative unsound reasoning methods induction, non-monotonic, default reasoning

7.6. Uncertainty in Individual Rules

errors domain errors representation errors inappropriate application of the rule likelihood of evidence for each premise for the conclusion combination of evidence from multiple premises

7.7. Uncertainty and Multiple Rules

collision resolution if multiple rules are applicable, which one is selected explicit priorities, provided by domain experts implicit priorities derived from rule properties specificity of patterns, ordering of patterns creation time of rules, most recent usage, ... compatibility contradictions between rules subsumption one rule is a more general version of another one redundancy missing rules data fusion integration of data from multiple sources

7.8. Probability Theory

7.8.1. Basics of Probability Theory

mathematical approach for processing uncertain information sample space setX = {x1, x2, ..., xn} collection of all possible events can be discrete or continuous probability number P(xi) reflects the likelihood of an event xi to occur
• non-negative value in [0,1]

• total probability of the sample space (sum of probabilities) is 1

• for mutually exclusive events, the probability for at least one of them is the sum of their individual probabilities

• experimental probability
  
  – based on the frequency of events

• subjective probability
  
  – based on expert assessment

7.8.2. Compound Probabilities

describes independent events do not affect each other in any way

joint probability of two independent events A and B

union probability of two independent events A and B

7.8.3. Conditional Probabilities

describes dependent events affect each other in some way

an example of a conditional probability of event A given that event B has already occurred

7.8.4. Advantages and Problems: Probabilities

advantages

formal foundation

reflection of reality (a posteriori)

problems

• may be inappropriate the future is not always similar to the past

• inexact or incorrect especially for subjective probabilities

• ignorance
probabilities must be assigned even if no information is available assigns an equal amount of probability to all such items

- non-local reasoning
  - requires the consideration of all available evidence, not only from the rules currently under consideration

- no compositionality
  - complex statements with conditional dependencies can not be decomposed into independent parts

7.8.5. Bayesian Approaches

- derive the probability of a cause given a symptom

- has gained importance recently due to advances in efficiency
  - more computational power available
  - better methods

- especially useful in diagnostic systems
  - medicine, computer help systems

- inverse or a posteriori probability
  - inverse to conditional probability of an earlier event given that a later one occurred

Advantages and Problems of Bayesian Reasoning

Advantages

1. sound theoretical foundation

2. well-defined semantics for decision making

Problems
1. requires large amounts of probability data
   (a) sufficient sample sizes
2. subjective evidence may not be reliable
3. independence of evidences assumption often not valid
4. relationship between hypothesis and evidence is reduced to a number
5. explanations for the user difficult
6. high computational overhead

7.9. Certainty Factors Belief and Disbelief

7.9.1. Certainty Factors
denotes the belief in a hypothesis H given that some pieces of evidence E are observed no statements about the belief means that no evidence is present in contrast to probabilities, Bayes’ method works reasonably well with partial evidence separation of belief, disbelief, ignorance share some foundations with Dempster-Shafer theory, but are more practical introduced in an ad-hoc way in MYCIN later mapped to DS theory

7.9.2. Advantages and Problems of Certainty Factors
Advantages

• simple implementation
• reasonable modeling of human experts’ belief
  – expression of belief and disbelief
• successful applications for certain problem classes
• evidence relatively easy to gather
  – no statistical base required

Problems
• partially ad hoc approach
  – theoretical foundation through Dempster-Shafer theory was developed later

• combination of non-independent evidence unsatisfactory

• new knowledge may require changes in the certainty factors of existing knowledge

• certainty factors can become the opposite of conditional probabilities for certain cases

• not suitable for long inference chains

7.10. Dempster-Shafer Theory Evidential Reasoning

• mathematical theory of evidence

• uncertainty is modeled through a range of probabilities instead of a single number indicating a probability

• sound theoretical foundation allows distinction between belief, disbelief, ignorance (non-belief)

• certainty factors are a special case of DS theory

7.10.1. Combination of Mass Probabilities
combining two masses in such a way that the new mass represents a consensus of the contributing pieces of evidence set intersection puts the emphasis on common elements of evidence, rather than conflicting evidence

7.10.2. Evidential Reasoning

• extension of DS theory that deals with uncertain, imprecise, and possibly inaccurate knowledge

• also uses evidential intervals to express the confidence in a statement
7.10.3. Advantages and Problems of Dempster-Shafer

- **advantages**
  - clear, rigorous foundation
  - ability to express confidence through intervals certainty about certainty
  - proper treatment of ignorance

- **problems**
  - non-intuitive determination of mass probability
  - very high computational overhead
  - may produce counterintuitive results due to normalization
  - usability somewhat unclear

7.11. Important Concepts and Terms

- Bayesian networks
- belief
- certainty factor c
- compound probability
- conditional probability
- Dempster-Shafer theory
- disbelief
- evidential reasoning
- inference inference
- mechanism ignorance

Bayesian classifiers can be optimized to perform very well. It is a theorem that Bayesian classifiers reduce to linear classifiers.

- knowledge knowledge representation mass function probability reasoning rule sample set uncertainty
7.12. Summary

- many practical tasks require reasoning under uncertainty
  - missing, inexact, inconsistent knowledge
- variations of probability theory are often combined with rule-based approaches
  - works reasonably well for many practical problems
- Bayesian networks have gained some prominence
  - improved methods, sufficient computational power
Revision questions

Example 🌼. Advantages of Probabilities

Solution:

advantages

formal foundation

reflection of reality (a posteriori)

EXERCISE 10. 🌼 What is statistical classification?

EXERCISE 11. 🌼 Discuss Bayesian classification technique, Bayes’ Rule for Single Event and Bayes’ Rule for Multiple Events

EXERCISE 12. 🌼 Discuss an example showing how Baye’s theorem may be used in a classification a problem.
LESSON 8
Approximate Reasoning and intelligent agents

8.0.1. Approximate Reasoning

•.1. inference of a possibly imprecise conclusion from possibly imprecise premises

•.2. useful in many real-world situations
  • one of the strategies used for “common sense” reasoning
  • frequently utilizes heuristics e
  • specially successful in some control applications

•.3. often used synonymously with fuzzy reasoning

•.4. although formal foundations have been developed, some problems remain

• Approaches to Approximate Reasoning
  • fuzzy logic
    – reasoning based on possibly imprecise sentences
  • default reasoning
    – in the absence of doubt, general rules (“defaults) are applied
    – default logic, nonmonotonic logic, circumscription
  • analogical reasoning
    – conclusions are derived according to analogies to similar situations
Advantages of Approximate Reasoning

- allows the emulation of some reasoning strategies used by humans
- can cover many aspects of a problem without explicit representation of the details
- can sometimes avoid lengthy inference chains

Problems of Approximate Reasoning

- nonmonotonicity
  - inconsistencies in the knowledge base may arise as new sentences are added
  - sometimes remedied by truth maintenance systems
- semantic status of rules
  - default rules often are false technically
- efficiency
  - although some decisions are quick, in general such systems are very slow
    - especially when truth maintenance is used

8.0.2. Fuzzy Logic

- approach to a formal treatment of uncertainty
- relies on quantifying and reasoning through natural language
  - uses linguistic variables to describe concepts with vague values
    - tall, large, small, heavy, ...
Advantages and Problems of Fuzzy Logic

Advantages foundation for a general theory of commonsense reasoning many practical applications natural use of vague and imprecise concepts hardware implementations for simpler tasks problems formulation of the task can be very tedious membership functions can be difficult to find multiple ways for combining evidence problems with long inference chains efficiency for complex tasks

8.0.3. Important Concepts and Terms

approximate reasoning common-sense reasoning crisp set default reasoning defuzzification extension principle fuzzification fuzzy inference fuzzy rule fuzzy set fuzzy value fuzzy variable imprecision inconsistency inexact knowledge inference inference mechanism knowledge linguistic variable membership function non-monotonic reasoning possibility probability reasoning rule uncertainty

Summary

attempts to formalize some aspects of common-sense reasoning fuzzy logic utilizes linguistic variables in combination with fuzzy rules and fuzzy inference in a formal approach to approximate reasoning allows a more natural formulation of some types of problems successfully applied to many real-world problems some fundamental and practical limitations semantics, usage, efficiency

8.1. Intelligent agents (IA)

Intelligent agents must have been coined from the term agent. In conventional interpretation agents are associated with personal representatives that interact with others to accomplish a task. In computational sense, agents can be seen as programs that help the users with routine tasks. However, several meanings to the term agent have been proposed as will be discussed below.

8.1.1. Definition of intelligent agent

Working definition of "agent"
An agent is a reusable software component that provides controlled access to (shared) services and resources. Example: a printer agent that provides printing services schedules requests to a shared printer.
Agents are the basic building blocks for applications, and applications are organized as networks of collaborating agents. Example: a desktop agent "recruits" the services of a screen and a connection agent to physically connect a call.

8.1.2. Agent and conventional programs

Common properties that make agents different from conventional programs:

- Agents are autonomous, that is they act on behalf of the user.
- Agents contain some level of intelligence, from fixed rules to learning engines that allow them to adapt to changes in the environment.
- Agents don’t only act reactively, but sometimes also proactively (initiate actions).
- Agents have social ability, that is they communicate with the user, the system, and other agents as required.
- Agents may also cooperate with other agents to carry out more complex tasks than they themselves can handle.
- Agents may move from one system to another to access remote resources or even to meet other agents.

8.1.3. Components of agents

The following may be specified for a typical agent: Owner. This is the user name, parent name or master agent name.

- **Author.** This is the person or the process that created the agent.
- **Account.** This is the owner’s account that may be used for billing or pointer to the account.
- **Goal.** This is the statement of successful task.
- **Subject description.** These are attributes of the goal.
- **Creation and duration.** This is the request or response time expected.
- **Background.** This is supporting information.
• *Intelligent system.* This is the engine that runs the agent’s intelligence, it may consist of rule-based expert system or neural network.

### 8.1.4. Characteristics of intelligent agents

Intelligent agents have several characteristics that are discussed below.

• *Autonomy.* Agents act or decide on their own. Sometimes they may do this to circumvent obstacles, or as they handle high level requests and seek more clarifications if necessary.

• Background *operation.* Agents work in the background, usually out of sight, perhaps somewhere in the cyberspace.

• Singularity of task. Agents work on a single task.

• *Communication.* Agents interact with other agents or humans.

• *Automation* of repetitive tasks. Agents work on special repetitive tasks.

• Support *conditional* processing. Agents may be rule-based systems thereby showing flexibility.

• *Learning.* Agents can learn; this goes beyond rule-based systems.

• *Reactivity.* Agents can perceive the environment and then respond.

• *Proactiveness.* Agents can take initiatives such as inhibiting behavior instead of just acting in response to environmental inputs.

• *Temporal continuity.* Agents are continuously running processes.

• *Personality.* Agents have a personality; they can interact with humans.

• *Mobility.* Agents can move across different architectures.

### 8.1.5. The need for agents

The following issues underline the need for agents:

• *Information overload.* The managers have so much information and they need some kind of help to cope.
• Massive bank of information over the years. A lot of information has been accumulated over the years and there is need to analyze it and discover any other knowledge held.

• Internet. The Internet requires search tools.

• Service support. Service support is necessary in many areas including network security, electronic commerce or employee support.

• Simplification of distributed computing. Agents can act as intelligent resource managers. Overcome user interface problems. Agents act as personal assistants adapting to the users. Handle information service management problem. Agents can provide services, service customization, monitor interacting features where systems are combined, enable using varied terminals, enable resource sharing and selection, diagnose problems, bill electronic users and provide security (firewalls).

8.1.6. TYPES OF AGENTS

Nwana’s classification of Agents

•.1. 1. Mobility: static or mobile;

•.2. 2. Reasoning model: deliberative or reactive;

•.3. 3. Ideal attributes: autonomy, learning and cooperation;

•.4. 4. Role: information, management;

•.5. 5. Hybrid: combination of the above.

• Classifying agents from organizational and personal view
  • Organizational agents. These are agents that work on behalf of businesses or computer applications.
  • Personal agents. These are agents that help users surf the Internet. They work for the users that create them.
ICS 2405 Knowledge Based Systems

- **Classifying agents from characteristics point of view**
  - Nature of agency. Agents can collaborate or negotiate.
  - Intelligence. Agents have varying degrees of reasoning and learning.
  - Mobility. Agents have degrees of mobility across platforms.
Revision questions

Example 🏷️. Classifying agents by intelligence level and power

Solution:

Level 0 agents. These are agents that follow orders to retrieve documents such as Netscape Navigator.

Level 1 agents. These are agents that are initiated by users to get relevant Internet pages such as Yahoo, Alta Vista, Inforseek.

Level 2 agents. These are agents that monitor information and notify of relevant information such as WebWatcher.

Level 3 agents. These are agents that learn and examine user profiles.

Exercise 13. 🏷️ Classifying agents by application area

Exercise 14. 🏷️ read on the following subtopics

Fuzzy Sets and Natural Language
Membership Functions
Linguistic Variables
Learning Activities

Variation of Reasoning with Uncertainty
Commonsense Reasoning
What is an intelligent agent?
What is a distributed intelligent agent?
What is a multi-agent system?
Discuss why agents are necessary.
Discuss the components of an agent.
Discuss the classification of agents.
LESSON 9

Expert Systems

9.1. Definitions: Expert Systems

• “a computer program that represents and reasons with knowledge of some specialist subject with a view to solving problems or giving advice.” – Jackson

• “an intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solutions.” – Edward Feigenbaum

• “a system that uses human knowledge captured in a computer to solve problems that ordinarily require human expertise.” – Efraim Turban and Jay Aronson

9.1.1. Expert Systems VS. Conventional Programs

• Expert systems simulate human reasoning about a problem domain, while conventional programs simulate the domain itself.

• ES perform reasoning over representations of human knowledge, numerical calculations and data retrieval while conventional programs perform numerical calculations and data retrieval.

• ES solve problems using heuristic or approximate methods that do not guarantee success, while conventional programs provide algorithmic solutions.

9.1.2. Expert Systems VS. Other AI Programs

• Expert Systems deal with problems that require human expertise, while most AI programs focus on abstract mathematical problems or simplified versions of real problems.

• Expert Systems must be fast and reliable in order to be useful, most AI programs do not run very fast and are programs rather than supported software.
• Expert systems must be able to explain and justify their solutions, other AI programs do not have to meet this requirement because their users are the developers.

9.1.3. Expert Systems VS. KBS

• Expert system, knowledge-based system, and knowledge-based expert system. Are they all the same?

• **Knowledge-based systems** typically perform a task by applying rules of thumb to a symbolic representation of knowledge, instead of employing mostly algorithmic or statistical methods.
  
  – They tend to be more general than ES.
  – They may not contain any expertise.
  – They use domain-independent methods.
  – They effectively and efficiently perform tasks that do not require an expert.
  
  – Knowledge-based systems are also known as advisory systems, knowledge systems, intelligent job aid systems, or operational systems.
  – Knowledge-based systems are easier and cheaper to build than expert systems.

Basic Concepts

• Expertise

• Experts

• Knowledge Acquisition

• Inferencing

• Knowledge Representation

• Explanation
9.1.4. Expertise

Expertise is the extensive, task-specific knowledge acquired from training, reading, and experience.

- Knowledge that allows experts to make better and faster decisions than non-experts when solving complex problems.

- **Examples of expertise:**
  
  1. Theories about a problem area.
  2. Rules and procedures regarding a general problem area.
  5. Meta-knowledge (knowledge about knowledge)

- **Facts about expertise:**
  
  - It is usually associated with a high degree of intelligence, but not always the smartest person.
  - It is usually associated with a vast quantity of knowledge.
  - It is gathered via past successes and mistakes.
  - It is well-stored, organized, and quickly retrievable from the expert.
  - It allows experts to quickly recall patterns for solutions.

9.1.5. Experts

- An Expert is an individual or group of individuals that possess various levels of domain expertise.

  - Nonexperts outnumber experts in many fields Huge advantages of making the expertise available to non-experts.
Experts know more than mere facts or principles about a domain. They also know the extent of their knowledge and qualify their advice as the problem reaches their limits of ignorance.

Experts should be able to explain the results, learn new things about the domain, restructure their knowledge when necessary, break the rules when necessary, and determine when their expertise is relevant.

Experts are also able to “degrade gracefully” meaning that they gradually become less proficient at solving problems, but are still able to develop reasonable solutions.

An expert must:

• Be able to perform a task;
• Know how to perform the task;
• Be able to explain how they perform the task;
• Have the time to explain how they perform the task;
• Be motivated to cooperate in the enterprise.

9.1.6. Knowledge Acquisition

• Knowledge acquisition is the process of eliciting the knowledge from the expert in a manner that permits the translation of the knowledge into a representation for the expert system.

  Buchanan defines Knowledge acquisition as: “the transfer and transformation of potential problem-solving expertise from some knowledge source to a program.”

• Knowledge acquisition typically occurs by interviewing the expert.

  Knowledge may also be gained from books, the web, etc.
  The major “bottleneck” in developing an ES is knowledge acquisition.
  One day of interviewing will produce two to five units of knowledge.
• The difficulties in knowledge acquisition are:
  – Many facts and principles cannot be easily represented as mathematical theory or as a deterministic model.
  – Human expertise is usually set in the context of commonsense knowledge.

9.1.7. Inferencing
Inferencing is the ability to reason about the data provided in the knowledge base and the relevant information.
  • Inferencing in Expert Systems is performed by the inference engine.

9.1.8. Knowledge Representation
Knowledge representation is concerned with storing large bodies of useful information in a symbolic format.
  • Most commercial ES are rule-based systems where the information is stored as rules.
  • Frames may also be used to complement rule-based systems.

9.2. Types of Knowledge
Many expert systems do not represent causal knowledge, meaning they do not understand the underlying cause and effects in the system.

**Shallow knowledge:** Surface-level information that can be used to deal with very specific situations.
  • It is based upon empirical and heuristic knowledge.
  • Heuristic knowledge represents rules of thumb or empirical knowledge gained from experience that may help in a particular situation but is not guaranteed to solve the problem.
  • Heuristics can provide valuable shortcuts that reduce costs and time.

**Deep knowledge** can be applied to different tasks and different situations.
It is based upon a completely integrated, cohesive body of human consciousness that includes emotions, common sense, intuition, etc.

It is difficult to collect and validate. Explanation

- The explanation is the expert systems ability to explain its advice or recommendations.
  - Explanations must be easily understood by the non-expert user.

- Explanations are important because:
  - System users need to know that the program’s conclusions are basically correct.
  - Knowledge engineers need verification that the knowledge is applied properly.
  - Domain experts need to be able to trace the way that their knowledge is applied to determine if the knowledge elicitation is successful.
  - The programmers need a higher level view of the system.
  - Managers need to verify that the system’s mode of reasoning is applicable to their domain.

The Expert System Structure

The Expert System Shell

- User Interface
- Inference Engine
- Knowledge Base
- Domain Expert
- Working Memory

The Expert System shell:
A specialised software tool that is used to develop an expert system. It is made up of the following three components.
ICS 2405 Knowledge Based Systems

- The User Interface: It provides means for the non-expert to interact with the knowledge contained in the knowledge base.

- The Inference Engine: This is the core of the expert system. It determines how the rules in the knowledge base are processed. The inference engine can be backward chaining, forward chaining or both.

- The Working Memory: An area of memory containing
  - Observed facts.
  - New facts deduced from observed facts.

The Knowledge Base: Contains both factual and heuristic knowledge.

- Factual knowledge: Knowledge of the task domain that is widely shared, typically found in text books or journals and commonly agreed upon by those knowledgeable in the particular field.

- Heuristic knowledge: Less rigorous, more experimental, more judgemental knowledge of performance.

In contrast to factual knowledge, heuristic knowledge is rarely discussed, and is largely individualistic.

When in operation, the user interacts with the system through the user interface, which may use

- Menus,

- Natural language or

- Any other style of interaction.

The inference engine is used to reason with both the expert knowledge (extracted from the expert) and data specific to the problem being solved.

- The expert knowledge is represented in a scheme known as knowledge representation. This normally formalizes and organizes the knowledge.
ICS 2405 Knowledge Based Systems

Expert System Structure diagram

Consultation Environment
User Interface
User

Development Environment

Knowledge base
Knowledge engineer

Facts about
the specific
incident.

Interpretation
– Inferring situation descriptions from observations.

Prediction
– Inferring likely consequences of given situations.

Diagnosis
– Inferring system malfunctions from observations.

Participants in Building ES

Tool builder
Build

Documented knowledge

Expert

Acquire knowledge

Test

Support Staff

Expert System

Build

End-user

System Engineer

Use

Cooperate

Vendor

Provide

Tools, languages

Use

Acquire knowledge

Knowledge Engineer

Build

Use

Expert

End-user

Knowledge Acquisition

Plan: Agenda
Solution: Problem Desc

Inference Engine

Draws conclusions

Knowledge Refinement

Documented Knowledge

Knowledge Acquisition

Expert knowledge

95
• Design
  – Configuring objects under constraints.

• Planning
  – Developing plans.

• Monitoring
  – Comparing objects under construction.

• Debugging
  – Prescribing remedies for malfunctions.

• Repair
  – Executing a plan to administer a prescribed remedy.

• Instruction
  – Diagnosing, debugging, and correcting student performance.

• Control
  – Interpreting, predicting, repairing, and monitoring system behaviors.

9.2.1. Advantages of ES

• Increased Availability - Mass produce expertise
• Reduced Cost Cheaper than having more experts
• Reduced Danger Reduce exposure to hazardous environments
• Permanence Expertise does not leave with personnel.
• Multiple Experts Knowledge from multiple experts can be combined.
• Increased reliability Increase confidence that a decision is correct.
• Explanation Explain the reasoning behind the conclusion.

• Fast response May respond faster than a human, and is always available.

• Steady, unemotional, and complete response times. Stress and fatigue are known to affect human performance.

• Intelligent Tutor A user may run examples and sample programs.

• Intelligent Database

• Increased output and productivity ES work faster than humans on complex cognitive tasks. Can mean fewer workers and reduced costs.

• Increased Process and Product Quality Provide consistent advice, therefore reducing the size and rate of errors.

• Reduced Downtime When used for malfunction and repair prediction, downtime can be avoided.

• Easier Equipment Operation Complex equipment and older control programs can be simplified.

• Can work with incomplete or uncertain data Will be able to produce an answer, but it may not be certain.

• Vigilance - ES do not become bored, tired, sick. They don’t go on strike, have bad attitudes, or talk back.

• Knowledge transfer to remote locations - Easily transfer information across international boarders.

9.2.2. Disadvantages of ES

• Knowledge is not always readily available.

• Difficult to extract expertise from humans.

• Experts may approach the same situation differently.
• Difficulty in eliciting good situational assessments when the expert is under
time pressure.

• Users still have cognitive limitations.

• ES only work well with narrow, well defined domains.

• Difficulty of independently verifying experts conclusions.

• Experts tend to use a domain specific vocabulary that is difficult for others to understand.

• Knowledge engineers are scarce and expensive.

• End-users may not trust the technology. This is a known phenomenon.

• The ES may not be able to arrive at conclusions.

• The ES may sometimes produce incorrect recommendations.

9.2.3. Elements of Successful ES

Factors the affect the successful development of expert systems are:

• Having a champion in management - Management must support the project.

• Good user involvement and training - The users must feel ownership.

• A high level of knowledge.

• Having a cooperative expert to provide expertise.

• The problem to be solved should be qualitative (fuzzy) - Not require a numerical
  approach, quantitative.

• The problem must have a narrow scope.

• The ES shell must be high quality, while naturally storing and manipulating the
  knowledge.
• User friendly interface for novice users.

• The problem must warrant development of an ES. Important and difficult

• Knowledge engineers and developers must have good people skills.

• Must consider end-users’ job improvement.

• Management support is required.

• Business ES can usually be justified by showing a gain in competitive advantage

• rather than their cost-effectiveness.

9.2.4. Software for building Expert Systems

**Expert System tools:** Software that is used for constructing expert systems

The tools range from programs used for building expert systems to programs that can aid the knowledge acquisition process.

The main software tools for developing expert system fall into the following categories.

• Programming languages.

• Expert system shells.

• A.I. Tool kits.
Example: Discuss the advantages and limitations of expert systems.

Solution:

Advantages:
- Increased productivity and output since expert systems work faster than humans.
- Decreased decision-making time as expert systems can make decisions faster.
- Increased process and product quality as errors can be significantly reduced.
- Reduced downtime as expert systems (ES) are used for diagnosis of malfunctions and prescribing repairs.
- Capture of scarce resource as ES can store the expertise held by humans who may be taking long to train.
- Flexibility as ES can sense changing needs and advice accordingly, such as a product out of production.

Limitations:
- Knowledge is not always readily available;
- Difficulty in extracting expertise from humans;
- Variations in problem assessment by different experts;
- Human experts cannot abstract when under pressure;
- Cognitive limitations of users;
- Only work well in a narrow domain;
- Experts may not always validate their conclusions;
- Experts may not always use understood vocabulary;
- Knowledge engineers are few and expensive;

Exercise 15. Describe the qualities and qualifications of a human expert.

Exercise 16. Discuss expertise.

Exercise 17. Discuss the definition of expert systems.

Exercise 18. Describe the structure of a human expert.
LESSON 10

Software Lifecycles

Describes the processes of:

- Design
- Development
- Testing
- Implementation
- Maintenance

All software has a ‘life cycle’

Characteristics of KBS Projects

- All KBS projects have unique requirements, but there are significant commonalities.
- There are also some significant differences from traditional software projects
- KBS projects require a different approach to be successful
- There are many different, evolving lifecycle models

Commonalities in KBS

Feasibility Study

- To determine whether the project is possible

Prototyping

- Develop early versions quickly
- Iterative process

Validation

- Need to demonstrate that knowledge is accurate
Maintenance
Feasibility Study
Need to determine whether a KBS solution to the problem is:

- Possible.
- Practical.
- Affordable.

A number of key pre-requisites should be examined.

Pre-requisites

- No known algorithmic solution exists Problem tends to be solved using “heuristic” knowledge
- Human experts can solve the problem satisfactorily There may be problems in using human experts, such as availability or cost
- Non-experts tend not to solve the problem satisfactorily.
- Not solving the problem has significant impact in terms of:
  - Cost.
  - Risk.
  - Delays.
  - Resources.

If the above do not apply, KBS is not the right approach to the problem

Prototyping

Has the advantages of:

- Speed - first version(s) can be produced quickly
- Can be used in “what if?” mode - try new ideas, generate enthusiasm for project
- Can be used to sell the project to management and/or users
- Incremental development - new features can be added as versions evolve
• Debugging can be done stage-by-stage.

Validation
Normal software systems are tested for procedural errors
KBS need to be tested AND validated
Validation is to ensure that the knowledge in the system is:
  • Accurate
  • Complete
  • Effective in solving problem.

Maintenance
All software systems require maintenance
  • software support for new versions, new platforms etc
KBS need additional maintenance
  • Additions to knowledge base
  • Modifications of inference techniques

Can be a very significant issue in dynamic knowledge domains

The Waterfall Model
  • Phased development
  • Sequential execution
  • Criteria used to determine completion of each phase (document- or code-driven)
  • Feedback loops confined within phases
  • Complete and correct requirements and specification needed before design

Eight stages:
  • Feasibility
ICS 2405 Knowledge Based Systems

- Requirements analysis
- Initial design
- Detailed design
- Code
- Integration
- Implementation
- Operation and maintenance

**Benefits of the staged approach to KBS development**

- Ability to plant the project and estimate the resources required to complete it.
- Ability to estimate the size and complexity of the software project.
- Availability of documents for monitoring and control.
- Ability to update estimates on the basis of the monitoring.
- Easy to fit project into a quality-management system.
Limitations of the staged approach to KBS development

In general, Knowledge Based Systems:
Have incomplete system/user requirements at the start of the project.

- Users do not understand what the system can do for them.
- Expertise which the system is aiming to capture may be difficult to categorise and represent.

Are complex and poorly structured.
Need to cope with uncertainty and/or incomplete knowledge.
The task of the KE is to adapt existing software-development life cycles to problems of KBS development.

Prototyping

Prototyping moves the developer and customer toward a "quick" implementation. It begins with requirements gathering.

- Meetings between developer and customer are conducted to determine overall system objectives and functional and performance requirements.

The developer then applies a set of tools to develop a quick design and build a working model (the "prototype") of some element(s) of the system. The customer or user "test drives" the prototype, evaluating its function and recommending changes to better meet customer needs.

Iteration occurs as this process is repeated, and an acceptable model is derived. The developer then moves to "productize" the prototype by applying many of the steps described for the classic life cycle.

In object oriented programming using a library of reusable objects (data structures and associated procedures) the software engineer can rapidly create prototypes and production programs.
The Prototyping Model

Types of Prototyping

- **Throw-it-away:**

  Involves the rapid development of an early working design.
  Used to develop the user’s requirements to expand the original specification.
  Identifies the required system and eliminates errors.
  The prototype is subsequently discarded.

  - advantages

    Applied in stages as the design of the system is carried out with suggestions from both users and developers contributing to the development. A flexible approach which reflects the rapidly changing requirements that are often needed in software development.

    - Incremental:

    Used after the normal design of the system
    Developed in stages with each stage evaluated before continuing to the next.
    The user can suggest changes at any stage.
    Once the design of any stage is completed, it is frozen.

**Benefits of prototyping**

- A working model is provided to the customer/user early in the process, enabling early assessment and bolstering confidence,

- The developer gains experience and insight by building the model, thereby resulting in a more solid implementation of "the real thing"
The prototype serves to clarify otherwise vague requirements, reducing ambiguity and improving communication between developer and user.

**Limitations of prototyping**

The user sees what appears to be a fully working system (in actuality, it is a partially working model) and believes that the prototype (a model) can be easily transformed into a production system.

The developer often makes technical compromises to build a "quick and dirty" model. These may be propagated into the production system.

Prototyping is applicable only to a limited class of problems.

- It is valuable when heavy human-machine interaction occurs, when complex output is to be produced or when new or untested algorithms are to be applied.

**Knowledge Acquisition and Design Process (KADS) Methodology**

- Came from a European-funded research project
- Based on the waterfall model
- Sequential development steps, each with substantial documentation required
- Used in the initial phases of KBS development (KADS II for later stages)
- Computer-based documentation system
- Other tools to help developers

**KADS Methodology**

In KADS, development of a KBS is viewed as a modeling activity.

- “A KBS is not a container filled with knowledge from an expert, but an operational model that exhibits some desired behavior which can be observed in terms of real-world phenomena.”

**Principles underlying the KADS approach**

- Introduction of multiple models as a means of coping with the process of knowledge engineering.
- Use of knowledge-level descriptions as an intermediate model between expertise data and system design.
**Knowledge Acquisition**  
Knowledge acquisition involves

- Eliciting knowledge in an informal form.
- Interpreting the elicited data using some conceptual framework.
- Formalizing the conceptualizations in such a way that the program can use the knowledge.

Knowledge acquisition can be viewed in two ways

- **Transfer view:** Extracting knowledge from a human expert and transferring to the KBS. Many problems!
- **Modeling view:** Use any data about the behavior of the expert in which the ultimate modelling decisions have to be made by the Knowledge Engineer.

**Principle 1: Multiple Models**

KBS construction can be viewed as a search through a large space of KE methods, techniques and tools.

In KADS, the KE space of choices should be controlled by introduction of a number of models.

The models include;

1. The organization model:
2. The application model:
3. The task model:
4. The model of cooperation:
5. The model of expertise:
6. The conceptual model:
7. The design model:

The organization model
The application model

- Function and problem: Defines what problems the system should solve in the organization and what function the system will be in this organization.

- External constraints: Specifies the external constraints that are relevant to the development of the application.

The task model
A description of the tasks carried out in realizing some function.
It is a decomposition of some high level function.
Tasks are described by

- A goal

- Inputs and outputs

- Relations to other tasks

- Agents performing the tasks.

Agent: A structure that as a whole performs the task.

- E.g. the university educating 20,000 students (the university is the agent)

The communication/cooperation model
Models communication between co-operating agents
ICS 2405 Knowledge Based Systems

- Communication tasks
- Transfer tasks
- User models

Describes transactions that cross boundaries between agents

- Information ingredients passed between agents
- Transaction (dialogue) plan (sequencing, deferral of questions)
- Assigns initiative for initiating transactions to co-operating agents.

The conceptual model
Abstract descriptions of the objects and operations that a system should know about, formulated in such a way that they capture the intuitions humans have of this behavior.

The language in which conceptual models are expressed is not the formal language of computational constructs and techniques, but the language that relates real world phenomena to the cognitive framework of the observer.

- Conceptual model = Model of expertise + Model of cooperation

The expertise model

- Associated with the agent.
- Describes the capabilities of an agent, especially knowledge intensive problem solving behavior.
- KADS identifies 4 knowledge types

Domain

- Concepts, relations and structures

Inference

- Knowledge roles
- Primitive inference steps
ICS 2405 Knowledge Based Systems

- Inference structure

Task

- Goals and (sub-)tasks organized in a control structure

Strategic

- Knowledge for combining and performing tasks (planning).
- Deadlock and conflict resolution.

Known as KADS library of expertise model

Relationship between knowledge layers

The design model

Looks at the following

- Application design: Specifies application specific components to be built
- Architectural design: Specifies the computational structures used to realize the application.
- Platform Design: Specifies the hardware and software to be used.
10.0.5. KADS Methodology

- Limitation - only applies to the capture and structuring of knowledge obtained verbally (e.g. interview)
- Not yet a standard method
- Influential in European research centres
- Gaining more ground for commercial applications
- KADS II

- Why Different Models?

There are MAJOR differences between traditional software and KBS software projects
Traditional life cycle models do not work well for KBS systems
KBS life cycle models try to encapsulate the nature of KBS projects and the differences

**Major Differences**

*Traditional approach:*
Primarily concerned with what the user wants. Assume complete and correct specification before design and implementation commences. Minimal back-tracking and iteration. Debugging is largely done on the complete system.

- **KBS approach:**
  Primarily concerned with acquisition and representation of knowledge. Cannot define a complete and correct specification before design and implementation. Use explorative programming, iterative design, and prototyping. Developed, and debugged, incrementally.

What Makes a Good KBS Life Cycle Model?

- Should be complete, covering all aspects of KBS development process.
- Should have a clear structure, based on hierarchical decomposition of the KBS development process.
- Should be simple, with a few key concepts.
Example. What Makes a Good KBS Life Cycle Model?

Solution:
Should be complete, covering all aspects of KBS development process
Should have a clear structure, based on hierarchical decomposition of the KBS development process
Should be simple, with a few key concepts

EXERCISE 19. Can normal life cycle models be applied to KBS?

EXERCISE 20. Define the terms communication and language.

EXERCISE 21. Describe the importance of speech acts.

EXERCISE 22. Discuss the meaning of the terms: intention, generation, synthesis, perception, analysis, disambiguation, incorporation.

EXERCISE 23. Describe the component steps of communication.

EXERCISE 24. Describe a model for two agents communicating in a language.

EXERCISE 25. Discuss the important items to have in natural language processing.

EXERCISE 26. What is a lexicon? What is the role of grammar in natural language processing?

EXERCISE 27. How can a statement in natural language be parsed?

EXERCISE 28. Discuss practical applications of natural language processing.

EXERCISE 29. Discuss aspects of machine vision.
Solutions to Exercises

Exercise 1.

issues in knowledge acquisition
Machine representation is lower in form than human usage of knowledge;
Many participants are involved and they have varied backgrounds causing commu-
nication challenges(Keng., Dom. Expert, System designers, Users, etc.).
Experts may not express their knowledge);
Mismach between the way experts hold their knowledge and the way computers
represent knowledge.

Exercise 2.

Exercise 13.

There are various types of agents under this category including:
Workflow and administrative agents that assist in organizing work schedules or
tasks.
Collaborative agents that work with other agents or people.
Desktop application support agents that help users of the desktop PCs.
Electronic commerce support agents that help users engage in electronic business.
Information support and management agents that help users retrieve, store, search
or interpret information.
Mail message support agents that help users dealing with electronic mails.
Network control and management agents that help users in controlling and managing
the network.
User interface creation agents are the agents that help users create interfaces or
present them with useable interfaces.
Operating system agents are those agents that help users of various operating sys-
tems get services.

Exercise 13