EXPERT SYSTEMS (ESs)

One of the largest areas of applications of artificial intelligence is in expert systems (ESs), or knowledge based systems as they are sometimes known.

ESs have been successful largely because they restrict the field of interest to a narrowly defined area that can be naturally described by explicit verbal rules.

ESs seek to embed the knowledge of a human expert (eg a highly skilled physician or lawyer) in a 'computerised consulting service' that -- because such systems do not get bored, or tired, or old -- *preserve* and *disseminate* the knowledge so that it can be useful to others.

An expert system provides advice derived from its knowledge base, using a reasoning process embedded in its inference engine, the 'thinking' part of the system.

ESs use backward chaining (deduction) as the basis of inference, because they start from a 'most likely' hypothesis (obviously the choice of this will play a large part in the system's success and efficiency) then look for evidence to support this hypothesis. If, after requesting relevant information from the user, this initial hypothesis cannot be supported, then the system will default to the 'next most likely hypothesis' and so on.

The process is analogous to that of medical diagnosis (so you will not be surprised that some of the earliest applications of ESs were to medicine) where a physician will start with the most likely diagnosis and perform tests to confirm it. If these tests are inconclusive, then further testing is necessary. If the results contradict the initial diagnosis then the physician must form another diagnosis, and this process continues until the physician is able to confirm a diagnosis.

OPERATION OF THE SYSTEM

There are three modes to this:

- **Knowledge acquisition**
- **Consultation**
- **Explanation**

KNOWLEDGE ACQUISITION

The designer of the system must liaise with people in order to gain knowledge and these people must be acknowledged experts in the appropriate area of activity, for example physicians, lawyers or investment analysts. The knowledge engineer acts as an intermediary between the human expert and the expert system. Typical of the information that must be gleaned is vocabulary or jargon, general concepts and facts, problems that commonly arise, the solutions to the problems that occur, and skills for solving particular problems. This process of picking the brain of an expert is a specialised form of data capture and makes use of interview techniques. Having acquired the information the knowledge engineer is also responsible for the self consistency of the data, and a number of specific tests have to be performed to ensure that the conclusions reached are sensible.

The knowledge engineer may use specialised software systems to help monitor the performance of an ES under development. When the human expert who is the source of the knowledge spots an error in the program's performance, in either the program's conclusions or its line of reasoning, such a system assists in finding the source of the error in the database by explaining the program's conclusions, retracing the reasoning steps until the faulty (or missing) rule is identified. It may then assist in knowledge acquisition by modifying faulty rules or adding new rules to the database

CONSULTATION

The system is in this mode when a user is interacting with it. The user interacts by entering data in English and the system responds using a backward chaining (deductive reasoning) process to derive an answer to the questions posed by the user. As explained earlier the user may during this time be asked for information that can be used to support the system's hypothesis, with appropriate backtracking if contradictory evidence to this hypothesis is found.

EXPLANATION

This mode allows the system to explain its conclusions and its reasoning process. This ability comes from the AND/OR trees created during the deduction process. As a result most expert systems can answer the following 'why' and 'how' questions

- Why was a given fact used?
- Why was a given fact not used?
- How was a given conclusion reached?
- How was it that another conclusion was not reached?

This ability to provide explanations is the big advantage of ESs over neural network based (NN) systems, and the reason why, despite the notable success of NN systems in many current application areas, ESs are likely to remain for a long time the AI technique of choice for safety-critical applications such as medical diagnosis, and ones where for legal reasons a verbal defense of a decision must be available if requested.

STRUCTURE

EXTERNALLY:

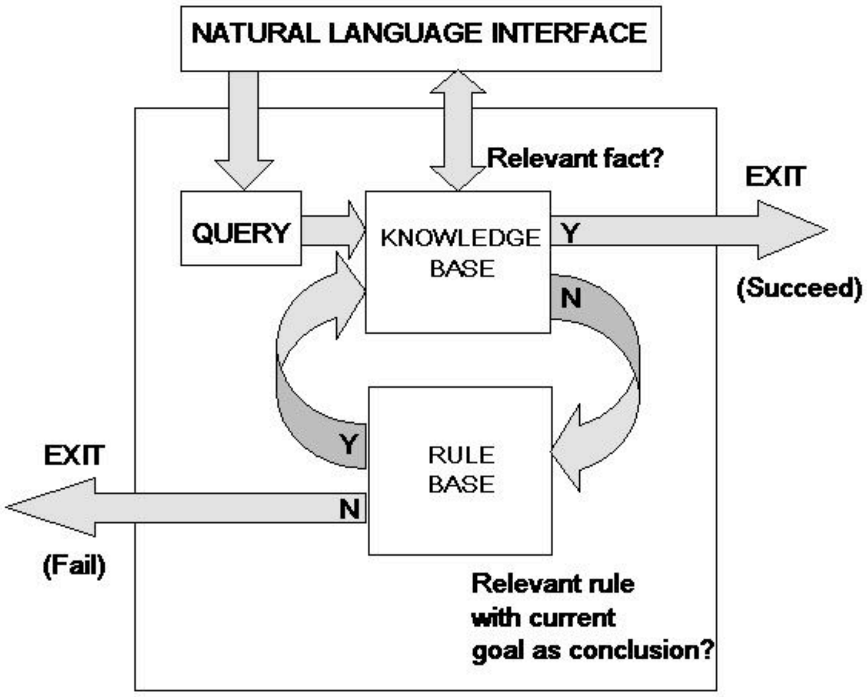
- Communication with the system is ideally provided by a natural language interface, so that it can be easily used by a person well-acquainted with the application area but not necessarily experienced with AI systems.

INTERNALLY there are three major parts to the system:

- The **knowledge base**:
  This database gives the context of the problem domain and what are generally considered to be a set of useful facts. These are the facts that could be used to satisfy the premise part of the IF-THEN rules (the 'A' parts of possible assertions of the form '**IF** A **THEN** B').

- The **rule base**:
  This holds the set of rules of inference that are used in reasoning. Most of these systems use IF-THEN rules to represent knowledge. Typically systems can have from a few hundred to a few thousand rules.

- The **inference engine** or **rule interpreter**:
  This is the 'brain' of the system, and controls how the IF-THEN rules are applied to the facts. In realistic systems this should allow for the acquisition of further information from the system's user -- who will be prompted for further input via the natural language interface -- which can be used to refine a hypothesis or resolve conflict between currently competing hypotheses.


NATURAL LANGUAGE INTERFACE
Relevant fact?
EXIT
QUERY
KNOWLEDGE BASE
Y
N
(Succeed)
Y
RULE BASE
EXIT
N
(Fail)
Relevant rule with current goal as conclusion?

The inference engine is not explicitly shown; it however controls the flow of information around the above modules, retrieving appropriate facts and rules during the reasoning process

The inference engine therefore does not contain domain-dependent knowledge and is -- together with support software that more easily enables the new user to customise the system for their intended application area -- the major part of what one gets when buying an expert system shell. There are many of these software products on the market (such as *KnowledgePro*) and their wide availability has contributed to the growth in use of expert systems, reported in 2003 by Siegel and Shim to be worth close to $1 billion in the US alone.

(A well-written expert system shell is probable easier for a novice user to configure correctly than a neural network (which are available as general purpose software simulators) is to train; as was discussed in the last section there are many potential pitfalls with neural network training such as sensitivity to initial weights and size of training rate, possibility of 'overtraining,' etc.)

A simple example of deductive reasoning

The knowledge base contains, amongst other facts:

green(Fritz).

The rule base contains, amongst other rules:

IF green(x) THEN frog(x).
IF frog(x) THEN hops(x).

Query: Does Fritz hop?

Step 1
Knowledge base is examined to see if 'hops(Fritz)' is a recorded fact. It’s not.

Step 2
Rule base is examined to see if there’s a rule of the form **IF A THEN hops(x)**; x=Fritz.
There is, with **A=frog(x)**; x=Fritz. But is the premise 'frog(Fritz)' actually true?

Step 3
Knowledge base is examined to see if 'frog(Fritz)' is a recorded fact. As with 'hops(Fritz),' it’s not, so it’s again necessary to look instead for an appropriate rule.

Step 4
Rule base is examined to see if there’s a rule of the form **IF A THEN frog(x)**; x=Fritz.
Again, there is a suitable rule, this time with with **A=green(x)**; x=Fritz. But now is 'green(Fritz)' true?

Step 5
Knowledge base is yet again examined, this time to see if 'green(Fritz)' is a recorded fact, and *yes* -- this time the premise is directly known to be true.

One can therefore finally conclude that the original assertion 'hops(Fritz)' was also true.

Failure of a query

If the required fact 'green(Fritz)' had *not* been found in the knowledge base, the rule base would have yet again been examined, this time looking for a rule of the form
IF A THEN green(x); x=Fritz.
However there is no rule in this database of this form, specifying a condition 'A' under which things are green; this would cause the system to exit in fail-mode, effectively concluding that 'No, Fritz doesn't hop.'

NOTE:

- The interpretation of *failure as negation* -- even though in practice failure to discover evidence something was true could also be just because the database was incomplete or inadequately maintained.

- In certain types of expert system, under some circumstances, there is a possibility of *infinite looping* when attempting to retrieve missing information; in these cases it is necessary to apply termination criteria that explicitly detect this and allow the system to always fail gracefully.

IMPLEMENTATION OF EXPERT SYSTEMS

Procedural vs. Declarative Languages

A procedural program consists of a sequence of commands. It's necessary for the programmer to think carefully, for each new problem, about the steps that must be carried out in order to solve it and the order in which they must be done. Neural networks, as an example of a 'number crunching' application, are typically implemented in procedural languages such as Java, C/C++, etc.

A declarative program in contrast is a sequence of facts and rules, a set of conditions that describe a solution space. There is some dependence on the order in which these are written, but not nearly as much as in procedural programs.

Declarative programming languages such as Lisp, Prolog and Miranda are usually the first choice for implementing rule-based AI systems such as ESs. This is because the syntactical structure of the rules as written in these languages can be closely matched to a chosen form of knowledge representation, and the means by which a query is resolved to a related model of reasoning (for example in Prolog, logical inference).

However sometimes languages like Lisp are used only for prototyping when the ES is intended to be marketed as a commercial product. The reason usually given for this is that programs written in languages like C/C++ execute more quickly, but nowadays with fast computer systems widely and cheaply available this isn't such a problem; other considerations may be the desire to supply compiled (object) code rather than source code in order to protect intellectual property, and the fact that a product may be more saleable if it is written in a well-known programming language like C rather than one like Lisp or Prolog with which a potential customer might be unfamilar.

Some areas of financial application of ESs are:

- **Detection of possible frauds:**

  Authorizer's Assistant is an American Express ES used for credit authorisation and for weeding out bad credit risks amongst new card applicants.
  Escape is used by the Ford Motor Company to assess insurance claims.

- **Appraising loan applications**

  Countryside Home Loans, Inc., a US-based mortgage provider with close to $100 billion in assets, has since 2003 used an expert system to automate the process of approving a home loan. In this case the major advantage of the ES is *time saved* -- a decision can be obtained from it in 30 seconds that would have taken a human underwriter up to a week.

Note that these two areas are also ones to which subsymbolic systems, neural networks in particular, have been applied. It would be very interesting to see a benchmark comparison of a neural network and ES solution, but to my knowledge no such studies have been published. Benchmarking studies are unfortunately rare in AI applications, even ones of a more limited scope which compare the ability of say, neural networks training using different methodologies.

- **Preparation and analysis of reports**

  CoverStory extracts marketing information from a database and automatically writes marketing reports.

- **Audit planning**:

  ESs can be used to help prepare a consistent and regulation-compliant presentation of accounts. From the point of view of the auditor they can also be used to select audit programs and/or test samples, and to formulate a judgement based on the findings.

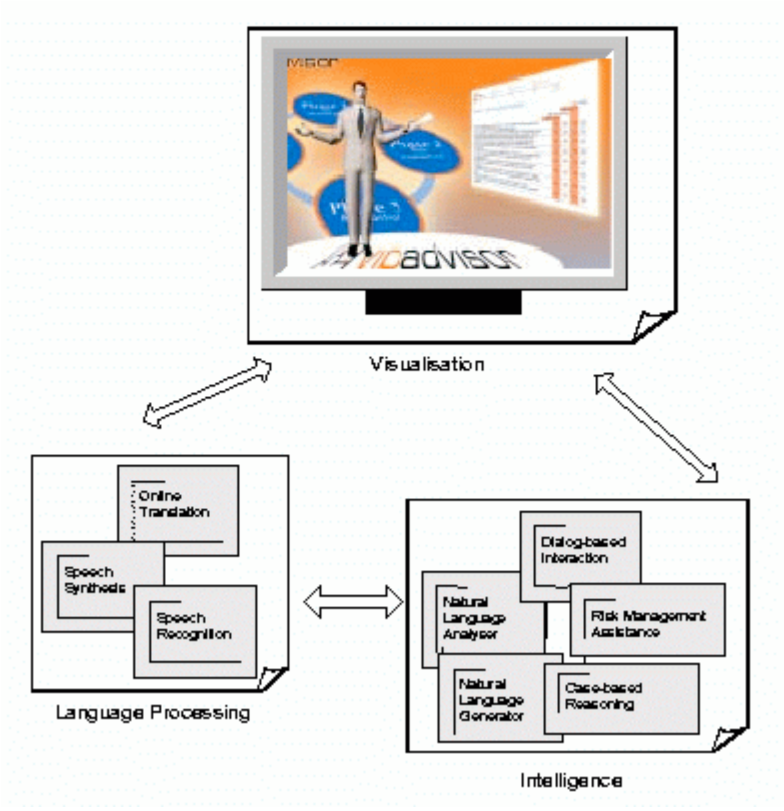
- **Tax planning**:

  Taxation has a complex set of rules and procedures that make it an especially suitable area of application for ESs. Tax expert systems may be used for estate planning, tax research, and for determining the tax consequences of stock transactions.

  ExpeTAX, used by Coopers and Lybrand, is an ES with around 3000 rules that interacts with the user via a question-and-answer format before suggesting the best tax options.

- **Investment and portfolio management**

  VIP Advisor is a 'virtual assistant for personal financial advice' developed 2002-04 as part of an EC-funded expert systems project. It also used 3D avatar technologies and 'chatterbot' interfaces that had up to that time been used mainly in the entertainment industries:



FUZZY LOGIC

One major criticism of traditional expert systems has been that the rules they use are in many cases too precise; they don’t capture the 'shades of grey' of everyday life.

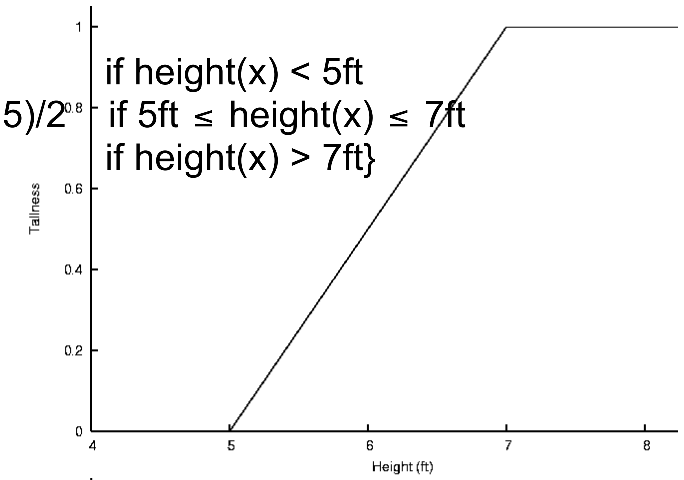
For example in the rule

IF *credit rating is GOOD and cost of purchase LESS THAN £5000* ***THEN*** *accept sale*

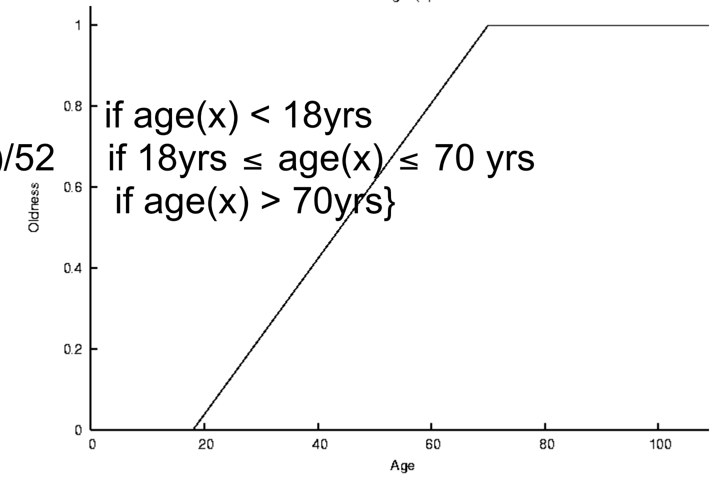
a sale would not be made if the intended purchase were for exactly £5000, no matter how good the customer’s credit rating might be. And what, in any case, does 'GOOD' mean? Is it reasonable to divide the whole population into just two classes, those with GOOD and BAD credit ratings?

Example: fuzzy set membership functions for TALL and OLD

$$\text{TALL}(x) = \begin{matrix} 0 & \text{if height}(x) < 5\text{ft} \\ (\text{height}(x) - 5)/2 & \text{if } 5\text{ft} \le \text{height}(x) \le 7\text{ft} \\ 1 & \text{if height}(x) > 7\text{ft}\} \end{matrix}$$



$$\text{OLD}(x) = \begin{matrix} 0 & \text{if age}(x) < 18\text{yrs} \\ (\text{age}(x) - 18)/52 & \text{if } 18\text{yrs} \le \text{age}(x) \le 70\text{ yrs} \\ 1 & \text{if age}(x) > 70\text{yrs}\} \end{matrix}$$



Combining fuzzy assertions via 'fuzzy logic operators'

The fuzzy versions of the basic **NOT**, **AND**, **OR** classical logic operators are

truth(**NOT** x) = 1.0 – truth(x)
truth(x **AND** y) = minimum(truth(x), truth(y))
truth(x **OR** y) = maximum(truth(x), truth(y))

Example:

A = x is TALL **AND** x is OLD;
B = x is TALL **OR** x is OLD;
C = **NOT**(x is OLD) (or equivalently, 'x is YOUNG')

height	**age**	**x is TALL**	**x is OLD**	**A**	**B**	**C**
5' 5"	**30**	0.21	0.23	0.21	0.29	0.77
5' 9"	**19**	0.38	0.02	0.02	0.38	0.98
5' 10"	**54**	0.42	0.69	0.42	0.69	0.31
6' 1"	**47**	0.54	0.56	0.54	0.56	0.44

FUZZY EXPERT SYSTEMS

To date, expert systems are the major application area for fuzzy logic based technologies, and have been applied successfully in a wide range of fields including linear and nonlinear control; pattern recognition; financial systems; operations research; data analysis.

Fuzzy expert systems use IF-THEN rules like traditional expert systems, but allow the premises within them (for example 'credit rating is GOOD') to be true to a degree calculated by the extent to which they belong to a fuzzy membership function (like those for 'TALL' and 'OLD' above). The degree to which they apply is thus in general in the interval [0,1] whereas classical logic would allow only the limiting values of 0 (=FALSE) or 1 (=TRUE).

In a fuzzy expert system, *all* relevant rules are 'fired.' So if, for example, there are separate rules pertaining to OLD and YOUNG people, then someone of age 47 would have *both rules* applied to them as they are according to fuzzy logic both OLD (to the degree 0.56) *and* YOUNG (to the degree 0.44).

Because of this feature, a final decision from a fuzzy expert system requires defuzzification, a process that can take a number of forms. The choice of method depends most strongly on whether it makes sense to blend the conclusions of several jointly fired rules in some way (for example if the system was being used as a controller), or whether a 'crisp' output is needed which selects just one conclusion from the range of candidates (as for example in a legal expert system).

HYBRID SYSTEMS

Fuzzy logic gives some flexibility to rule-based AI, but it does not give it the ability to create its own rules. For this, a subsymbolic system such as a neural network or genetic algorithm is still required.

There are currently a number of systems that in some way combine rule-based (including fuzzy) AI with a subsymbolic element. The simplest way to do this is to have the subsymbolic system do preprocessing of the data (for example extracting statististically significant clustering using a self-organising neural network) which is then passed on to the rule-based part of the system for analysis and decision making. However there are also some more sophisticated types of hybrid system that utilise a subsymbolic element to create new rules or refine pre-existing ones; Searchspace, a London-based company whose clients include the London Stock Exchange and the Bank of New York, has used a combination of fuzzy logic and genetic algorithms to evolve very successful systems able to detect insider dealing rings and evidence of money laundering.

However even systems such as this are inflexible when compared with the way biological nervous systems are able to learn and adapt to a changing environment. The Searchspace system for example requires 'seed' rules or rule fragments (in a verbal form) for the genetic algorithm to work on, so it is in this sense still limited by, and dependent on, the insights of human experts.

Symbolic vs. subsymbolic AI: final thoughts...

General Electric (GE) had a problem. Their top locomotive field service engineer, David I. Smith, who travelled all over the US troubleshooting diesel locomotive engines and advising younger engineers what to do, was nearing retirement. Yes, he could train a small number of apprentices (which had been GE's traditional solution in a situation like this), but the company wanted his special skills to be more widely available than this would allow. So, over a period of 3 years, an expert system was built that embodied all that could be extracted of Smith's troubleshooting skills. The system worked well and is currently installed at every railway repair shop serviced by GE.

In a similar but much narrower area of expertise, German railways trained a neural network to emulate the skills of their top engineer in detecting, from the sounds it made when tapped, whether a train axle had a fault that would make it liable to fracture or otherwise fail. Again the idea was to preserve a rare skill and disseminate it more widely than the traditional apprenticeship scheme would allow. But in this case it was found that the AI system's skills not only emulated but *exceeded* those of the human expert who had provided the training data.

GE's expert system by its nature could never be better than the human expert from which its knowledge and rule bases were derived. The experience with the 'axle-tapper' neural net however demonstrates that the skills developed during a training process by such machines can exceed our own. What will happen when neural networks, based on an improved knowledge of the brain, are built that not only have pattern recognition but *reasoning* skills? Will such machines be able to think -- in a true, broad sense -- better than we can? Could they be more creative than us? And what might we (and, possibly, they) feel about that?