RULE-BASED EXPERT SYSTEMS AND BEYOND: AN OVERVIEW

John Kingston

AIAI-TR-25

Abstract

There's a good deal more to expert systems than rules, and there's a lot more to Artificial Intelligence than expert systems. This paper will start with a brief overview of expert systems, particularly rule-based expert systems, and the types of task for which they are suitable. This will be followed by a description of other ways of representing knowledge in expert systems, including a quick look at the state of the art in software and hardware. The final section will describe some of the other tasks that Artificial Intelligence performs, outside the area of expert systems.

This paper was presented at the one-day Expert Systems 'plenary session', which formed part of the British Association of Accountants' Conference 1987, held at the Kelvin Conference Centre, Glasgow, from 13-15 April 1987. It was the first of six papers presented.

Artificial Intelligence Applications Institute
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
United Kingdom

1. Some definitions

1.1. What is an expert system?

An expert system can be defined as a computer system which performs at an expert level a task usually performed by a human expert in that field. There are also some subsidiary requirements:

- It must be knowledge-based; that is, knowledge must be explicitly coded into the system
- There must be some aspect of 'intelligence', or controlled search, in the method of inference used

Expert systems can make good use of either statistical evidence, or the knowledge of a human expert, concerning a problem; but without one of these two sources of information, they cannot function.

1.2. What is knowledge?

An expert system must have knowledge explicitly coded into the system to be be classified as an expert system. But what is knowledge? Leaving aside the centuries of philosophical debate on this topic, a working definition is given below.

Knowledge is not simply facts about the world which are stored in memory. The knowledge that a human uses to tackle a problem goes well beyond this. Such knowledge can be categorised under three headings:

- facts about the world
- strategies for solving problems in a particular area
- strategies for solving problems in general

2. Rule-based expert systems

2.1. What is a rule-based expert system?

A rule-based expert system is an expert system that encodes its knowledge in the form of IF-THEN rules. A typical rule looks like this:

IF condition-1 is true
AND condition-2 is true

THEN conclude goal-1 is true
AND execute action-1
AND execute action-2

An example might be:

IF there is a SHARE which is being traded
AND there is a DEALER who has executed MORE THAN 20 deals in the SHARE today
AND the price of the SHARE has changed by LESS THAN 3 pence today

THEN conclude that the DEALER is possibly breaching the Financial Services Act
AND investigate how much COMMISSION the DEALER gets for each deal in the SHARE

Rules are generally used to represent knowledge about strategies for solving problems in a particular area. Rule-based systems also maintain a small database of facts about the world, so that they can perform reasoning; if a fact about the world matches a condition of a rule, that condition is judged to be fulfilled. Many rule-based expert systems are developed using expert system *shells*. A shell provides facilities for writing rules easily, often in a format which resembles English syntax, and also provides a strategy for solving problems in general - that is, it has built-in algorithms for deciding which rule is to be used when. This strategy is known as the shell's *inference mechanism.* A shell can be thought of as a rule-based expert system without any knowledge, or as a framework around which an expert system can be developed.

2.2. Inference mechanisms: strategies for solving problems in general

Rules are only used when they are required by the system (as opposed to a system written in COBOL or a conventional language, when the programmer has to explicitly specify the flow of control between different parts of the program). This makes each rule comparatively independent of other rules in the system; this *modularity,* as it is known, is a typical feature of expert systems. Note also that many expert systems gather their initial data by asking questions of the user; these questions are linked to rules so that, if a particular rule is never considered relevant by the system, questions associated with that rule will never be asked.

Different rule-based systems use various different methods for deciding when each rule is required, but nearly all of them use either a *forward chaining* technique, or a *backward chaining* technique. Forward chaining systems work by making deductions from initial data. The user is allowed to supply data in any order he/she likes, and rules will operate when they have enough information - that is, when all their conditions are matched by incoming data, or by the conclusions of other rules. Hopefully, a rule whose conclusions include a solution to the problem being investigated will eventually be used. Backward chaining systems work by focussing on a particular part of the problem (that is, a solution to the problem, or a solution to a sub-problem); rules operate only if one of their conclusions is the solution or partial solution under consideration. If a rule is used, its conditions become the new solutions that the system focusses on.

2.3. What are rule-based systems used for?

Commercial applications of rule-based expert systems include:

- Devising a work plan for an audit that makes efficient use of staff time. This system asks for details about the client organisation, such as whether it deals in foreign currency, whether it has a stock control system, and so on, and for details of the audit manager's views, based on his knowledge of the firm, and the level of scrutiny required for various parts of the audit. It has recently been tested in the field by Coopers & Lybrand, who developed it. It runs on a portable PC.

- Advising on whether to give credit, using the expert system shell SAVOIR.

- Advising on the details of Statutory Sick Pay. This system can be purchased from Expertech, Slough.

- Checking for errors in the seven most important forms of letters of credit. This particular system is being used by the Bank of America, although since it takes longer to ask all the necessary questions than it takes a human to do the checking manually, it is mostly used for training.

There are a variety of advantages to using rule-based systems. These include the following:

- Experts often find it easy to express their knowledge in a rule-like format

- Programmers typically find rule-based programming easy to use and to understand

- For small systems, at least, the modularity of rules means that new rules can be added to the system whenever they are needed, without having to worry too much about affecting the behaviour of other rules, or radically changing the behaviour of the rest of the system. Modularity also makes programs easier to understand, and therefore easier to debug.

2.5. What are the limitations of rule-based systems?

The limitations of rule-based systems often become obvious after a few weeks or months to those who use them to write programs. Some of them are listed below:

- In larger rule-based systems, modularity breaks down. In forward chaining systems in particular, introducing a new rule can have unexpected side-effects throughout the program. The programmer is usually forced to add more restrictions on the flow of control within the system than the system's built-in inference strategy provides.

- Most expert systems use a few procedures, for complex mathematical functions, or other tasks which are difficult to encode within the framework provided by the expert system building tool, or shell. It is not uncommon for a program to use a particular procedure more than once. Very few rule-based systems include facilities for storing procedures with a particular rule, or for passing procedures from one rule to another; instead, the procedure must be stored separately from the rules, and each rule must call it explicitly. This is especially annoying if the knowledge is structured in a taxonomic hierarchy, where the same procedure may be used by every member of a particular class.

- Representing knowledge as an unordered set of rules fails to take advantage of any explicit structure the knowledge may already have, such as taxonomies, or cause-effect relations. This not only limits the system's reasoning, but also makes it difficult to provide sensible explanations of cause and effect to the user.

- It might be thought that it would be possible to program a rule-based system to detect wrong choices, and to back up to a previous state in the computation when it did. However, rule-based systems usually destructively modify their database of facts about the world, for the sake of efficiency; it is therefore difficult to make a rule-based system return to a previous state. This feature also makes it difficult to provide explanation facilities.

- For some problems, neither forward chaining nor backward chaining is an ideal inference strategy; the programmer may have to use constructs which are complex or unnatural (that is, not used by a human expert) to make the system behave just as he wants.

It has been found that rule-based expert systems are best suited to tasks which involve diagnosis or classification. For more details, see Jackson (1986)

Researchers in Artificial Intelligence have put a lot of effort into overcoming these limitations, resulting in new ways of representing knowledge, new programming techniques, and new expert system building tools. This section describes techniques which have been developed to overcome each of the five limitations described above.

3.1. Lack of modularity and inability to pass procedures between rules

The techniques used to overcome these two limitations are **frame-based programming** and **object-oriented programming**. Because the two are closely linked, they are discussed together.

Many expert systems represent knowledge using *frames*, also known as *schemata*. Frames represent knowledge using *slots* and *values*. A slot can be thought of as an attribute of the object which is being represented by a frame.

A typical frame is of the form :

(**A-543-TGT**
  IS-A: Metro
  MILEAGE: 65 000
  PRICE-WHEN-NEW: 5 000
  CURRENT-WORTH: 1 900
  REGISTRATION: A)

In this case, the object *A-543-TGT* has five slots, one of which is REGISTRATION; the value of this slot is A. Most frame-based systems include functions for accessing and manipulating frames, slots and values; these functions are broadly analagous to the database query language which is provided for a database.

Frames are a very efficient way of representing knowledge which is structured in a tree-like form, because of *inheritance* of attributes. Inheritance is a process by which all facts which are true of a particular class are also true of each member of that class, unless explicitly stated otherwise. For instance, given the frame:

(**NISSAN-CAR**
  NEW-PRICE: (< £20,000)
  DESIGNED-IN: Japan
  MANUFACTURED-IN: Japan)

and a taxonomic hierarchy which showed that **Bluebird** and **Micra** were types of Nissan car, then a frame-based system would assume that all Bluebirds and Micras were priced under £20,000, were designed in Japan, and were manufactured in Japan, unless the system was explicitly told that some of them are manufactured near Sunderland.

3.2. Object-oriented programming.

Object-oriented programming is designed around a frame-based representation of knowledge. An *object* is simply anything that can be represented using a frame. Object-oriented programming is a technique which uses slots whose values are pointers to procedures written in a programming language. Procedures can only be called by sending a message to the slot to which it is attached. Sending a message is simply a special way of accessing that slot, and is typically done using one or more of the frame accessing functions. The two main advantages to object-oriented programming are that all procedures which are most relevant to a particular frame are linked to slots within that frame - this makes object-oriented programs genuinely modular; and that procedures can be inherited by one object from another.

To give a simple example, the frame *A-543-TGT* might have a slot called *ANNUAL-DEPRECIATION*, whose value was a pointer to a procedure. This procedure would perform some arithmetic, using values of other slots in the frame (or other frames); in this example, it would subtract the value of the *CURRENT-WORTH* slot from the value of the *PRICE-WHEN-*

NEW slot, and divided the result by the age of the car, which can be calculated from the registration. The procedure would probably not be stored in *A-543-TGT* itself; it would be more sensible to store the procedure with the frame *VEHICLE*, and allow *A-543-TGT* to inherit it.

3.3. What systems have been built using frame and object-based systems?

Systems that are heavily based on object-oriented programming include:

- A system which generates financial plans. This system was launched in 1986, aimed at banks and other institutions.

- An investment advisor - most notable for displaying changes to the knowledge base graphically, helping the expert to see what's going on.

- A credit checker for a German bank, developed in Germany.

3.4. What are the advantages of a frame-based and object-oriented programming?

- Frames are clearly good at exploiting taxonomic structure in knowledge.

- The advantages of modularity have been mentioned above. However, modularity in object-oriented programming does not break down when the system gets large, provided the programmer defines some protocols, and sticks to them.

- Graphics can be implemented using frames. The updating of gauges, histograms, and other graphics that need to be altered during the run of a program is relatively simple to implement in an object-oriented system.

3.5. What are the limitations of frame-based systems?

- Allowing unrestricted overriding of inherited properties makes it impossible to define anything. For instance, a Leyland car MUST be designed and manufactured by British Leyland, or a partner, sub-contractor or subsidiary of British Leyland; so a frame-based system should generate an error if this fact is overwritten in frames such as *A-543-TGT* further down an inheritance hierarchy. Multiple inheritance (inheriting from more than one frame) also creates the possibility of inconsistency; for instance, if the inheritance hierarchy contained a frame representing a minibus, it might seem sensible to make it inherit characteristics from both *CAR* and *BUS*; but unless it was instructed otherwise, the system would inherit all the information from both *CAR* and *BUS*, providing the minibus with two categories for insurance purposes, and so on.

- Frames, like rules, destructively modify data structures, with the same disadvantages.

- Finally, it seems that some people find the behaviour of frame-based systems difficult to predict or understand

Frame-based systems are most useful for tackling problems which need a lot of reasoning using exceptions and defaults; problems involving looking for the best match between data and some conclusion, rather than a complete match; and problems for which the appropriate knowledge is structured in a taxonomic hierarchy

4. Inability to take advantage of cause-effect relationships

The technique which has been developed for overcoming this limitation is **model-based reasoning**, also called **simulation-based reasoning**. As the name suggests, this technique is based on explicitly representing a model of a process, such as the machining of parts on a factory floor (where parts have to go through several machines), or the flow of cash between various businesses and their creditors and debtors. This greatly facilitates the representation of causes and effects, and consequently the explanations provided to the user are also improved.

Model-based systems are typically implemented using object-oriented programming

4.1. What are such systems used for?

- Alerting the operator of a nuclear power plant to the underlying causes of alarms going off. This is necessary, since a lot of alarms may sound at once (500 in the first minute of the Three-Mile Island accident), causing the operator to panic. This system is still at the prototype stage.

4.2. Advantages of model-based systems

- The explicit representation makes it easy to understand the computer's reasoning
- The reasoning is usually accurate
- Very clear explanations can be provided

4.3. Limitations of model-based systems

- Explicitly representing a model in a system is computationally expensive
- For some problems, no information is available concerning the causes of various states of the world; for these problems, model-based reasoning is of little use
- For other problems, a model-based technique might require far more work than programming the system in a rule- or frame-based system, with comparatively little gain in performance.

Model-based systems are useful for tasks involving simulation, scheduling problems, and tasks that require causal explanations of the state of the world.

5. Destructive modification of the database

The major disadvantage of destructive modification of the database is that, once a choice has been made (that is, a rule has been used and the database has been updated), it is very difficult to undo that choice and back up to a previous state in the computation. Two techniques have been developed to overcome this limitation - *truth maintenance* and *hypothetical reasoning*.

5.1. Truth maintenance

The essence of a truth maintenance system is that reasoning can be performed based on assumptions, and can be undone if that assumption is found to be wrong. Once an assumption has been made, it is marked as an assumption, and all subsequent deductions are marked as being based on that assumption; if the assumption is ever found to be false, the system simply retracts all the but it is still computationally expensive if used all the time

5.2. Hypothetical reasoning

Hypothetical reasoning is not too difficult to implement in frame-based systems. If a system reaches a point where it has more than one possible deduction, the system creates new hypothetical 'contexts' to handle this. These contexts are structured in a hierarchy. In each new context, all the facts about the world are the same as in the context from which they were generated (their parent in the hierarchy), except the facts based on the most recent deduction. The system generates one new context for each possible deduction it can make. Thus, all possible states of the world exist at the same time. If any deduction is found to lead to an inconsistent state of the world, the context in which it was made is usually deleted; if it is not deleted, no more reasoning is performed within it.

6. Inability to mix forward and backward chaining

The technique used to overcome this has been to develop expert system building tools that incorporate both forward chaining rules and backward chaining rules, and to allow the programmer to mix and match them as he/she chooses. These 'hybrid' toolkits will be discussed in the next section.

7. The state of the art in software

Current interest is centred on hybrid knowledge representation - that is, providing forward chaining rules, backward chaining rules, and object-oriented programming in one expert system development tool, and allowing the programmer to mix and match them as he/she wishes. There are some very sophisticated expert system development tools available, such as ART, KEE, or Knowledge Craft. All these systems incorporate rule-based and object-oriented programming techniques, as well as built-in facilities for hypothetical reasoning, and a wide variety of user interfaces; some also provide facilities for truth maintenance, and programming of simulations. These toolkits are suitable for implementing expert systems to tackle most problems.

7.1. Systems built using hybrid toolkits that are commercially available

There are quite a number of such systems, including:

- An advisor to foreign currency option traders. This system is not cheap at US$110,000, but it incorporates an extensive knowledge base of the economic, marketing, and technical details of foreign exchange

- Risk assessment for marine underwriting - another Coopers & Lybrand development

- An equity investment advisor - the Alvey insurance club, known as ARIES, is developing such a system, which is nearing completion.

8. The state of the art in hardware

Most rule-based shells run on IBM PCs. Most of the large expert system toolkits, however, currently run only on single-user workstations. Various techniques are being used to integrate these tools into a commercial environment:

- Some commercial products include a link from a workstation to a database on a mainframe. The workstation may be used either as a front-end terminal, or a back-end

- At least one toolkit can now use IBM PCs as remote terminals

- The manufacturers of one of these toolkits are currently developing software which will translate queries made by that toolkit into a database query language

- Some expert systems are prototyped in the large toolkits, and then rewritten in an expert system shell

9. What else is going on in Artificial Intelligence?

A variety of tasks have been mentioned as being suitable problem areas for various types of expert system. However, the range of problems being tackled by Artificial Intelligence include:

- Natural language - enabling computers to interpret the meaning of utterances in ‘every-day’ English correctly

- Speech - enabling computers to correctly deduce what a human is saying

- Vision - correctly assigning light reflected by surfaces to objects

- Diagnosis - inferring system malfunctions from observables

- Interpretation - inferring situation descriptions from sensor data

- Prediction - inferring likely consequences of given situations

- Design - configuring objects under constraints

- Planning - designing actions

- Monitoring - comparing observations to plan vulnerabilities

- Repairing - executing a plan to administer a prescribed remedy

- Mathematical Reasoning

- Robotics

For more details, see Tate (1986).

9.1. Is this technology commercially useful?

Most of the above list of topics are still research areas, although partial solutions have been developed in some areas, and some of these are in use commercially. Given that the beginnings of expert systems were in the mid-60s, and they did not reach the market-place until the mid-80s, AI techniques for other areas which only began to emerge in the mid-1970s are unlikely to produce widespread commercial applications until the 1990s.

Domains for which AI seems to be able to extend or improve the range of available techniques include:

- Financial planning - this is a popular application area for commercial expert system tools. It is to be hoped that techniques and ideas from AI work on planning will be incorporated into future financial planning systems.

- Databases - topics under research include ‘intelligent’ database searches (using expertise to infer missing data), and using an ‘inheritance’ inference mechanism in databases.

- Operations research - ‘incremental’ versions of various algorithms have been developed, which are sensitive to changes to some given set of constraints; the long term goal of such techniques is operations research techniques which can be run interactively, rather than in batch mode.

- Real-time applications; the model-based and simulation programs are intended to run in real time, and the applications of such systems could be widespread, from foreign exchange dealing to computer vision.

- Electronic documents - Arthur Andersen use an expert Financial Statement Analyser, which reads (electronic) balance sheet submissions from companies, and puts them into a standardised form; to do this, it has to perform some natural language analysis.

10. Conclusion

Some useful work is being done with rule-based expert systems, but they have a number of limitations. Today’s expert system development tools overcome these limitations, although the range of hardware on which they are available restricts their commercial application. It can be hoped that tomorrow’s expert systems will overcome these hardware limitations; it is also quite possible that other areas of research in Artificial Intelligence may produce commercially viable techniques in a few years’ time.

References

P.Jackson, *Introduction to Expert Systems*, Addison-Wesley **1986**

A. Tate *AI: Coming out of its* "*Shell*" AIAI Technical report AIAI-TR-19 **1986**