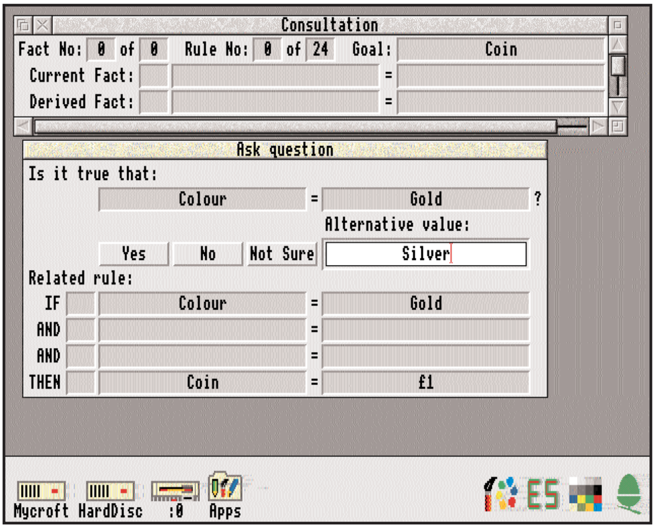
Expert Witness

Robin Watts interrogates and reviews the Observess expert system shell from Cherisha Software

For many years, one of the fields of computer science that has attracted the most attention is that of artificial intelligence - the idea of producing machines that can think. This has been toyed with for years, but so far relatively little progress seems to have been made.



Consultation

Fact No: 0 of 0 Rule No: 0 of 24 Goal: Coin

Current Fact: =

Derived Fact: =

Ask question

Is it true that:

Colour = Gold ?

Alternative value:

Yes No Not Sure Silver

Related rule:

IF Colour = Gold

AND =

AND =

THEN Coin = £1

Consulting the Observess coinage knowledge base

A major reason for this is simply the vast quantities of data involved; in order for a machine to act like a human it needs to have access to the huge amounts of background information on which we base our decisions. Every time we choose a course of action (be it whether to go out with or without a coat, whether to cross the road, or to turn left or right at a junction) we subconsciously consider many different factors.

The effort involved in moving all this information onto a computer is a huge one - every piece of information you give the computer seems to require several others to put it in its context. To try and counteract this problem a sub-field of artificial intelligence has sprung up - that of expert systems. Here the computer is only fed information on very limited subjects - hence it becomes an expert in one particular field, and remains totally ignorant about everything else. In common with many human experts, the computer is not much good at conversation, but if you just answer its questions you will probably get a reasonable response.

An expert system can be neatly split into two sections. Firstly, there are the rules it applies - known as the knowledge base. These can be given in various forms, but the most common is known as the IF - THEN method (more about this later).

The second part of the expert system is known as the inference engine. This is the part of the program that actually tries to apply the rules from the knowledge base to build up a dictionary of working facts and to draw conclusions from these.

Traditionally, expert systems have been written from scratch as they are required, but more recently people have begun to take advantage of this split. Expert systems have appeared which contain an inference engine, but no knowledge base. Users can add their own knowledge base, and effectively build expert systems for themselves. These systems are known as expert system shells.

The first such shell to appear for the Arc is called Observess from newcomer Cherisha Software. Observess is a Desktop application that allows you to enter and edit rules to build your knowledge base, and then consult with the expert thus formed. The knowledge base can then be saved out as a datafile and distributed to other users.

Observess installs an icon on the icon bar which when clicked produces a window that shows information about the currently loaded knowledge base - its title, purpose, size etc. To create a knowledge base we simply fill in these fields and then add all the rules required using the Add Rule menu option.

Observess works by keeping a list of variables and their values. The program will first try to apply its rules to work out values for each of the variables, but if this fails it will ask the user a question designed to elicit enough information to work out more about a variable s value.

Ultimately Observess tries to obtain the value of a Goal variable, and will make recommendations as to what value it thinks this variable has. On the way you can give it some Sub-goal variables, and it will tell you as it discovers their values.

Each rule is given in a Basic-like IF - THEN format. In the following syntax definition, angle brackets indicate parts that should be replaced with appropriate names and square brackets enclose optional parts:

```
IF     [NOT] <variable1> = <value1>
  [AND [NOT] <variable2> = <value2>]
  [AND [NOT] <variable3> = <value3>]
THEN   [NOT] <variable4> = <value4>
```

This form for rules has the benefit of being extremely simple, but it has various disadvantages. Firstly, there is no simple way to do O Rs. For instance to do:

```
IF coin = 1p OR coin = 2p THEN metal = copper
```

requires two separate rules:

```
IF coin = 1p THEN metal = copper
IF coin = 2p THEN metal = copper
```

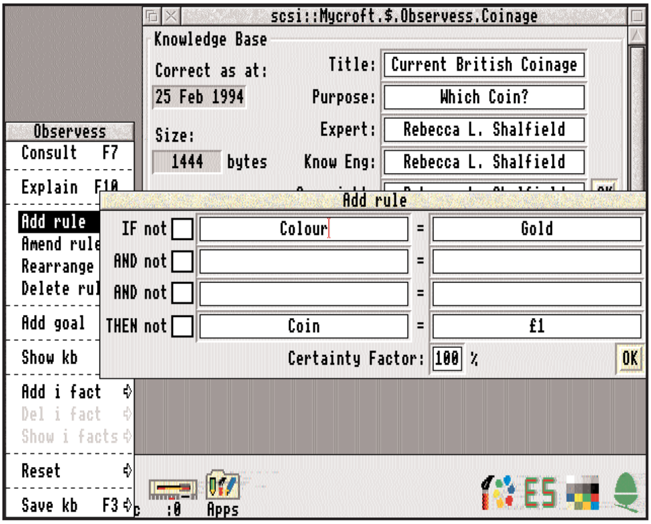
Secondly, to do a rule with more than 3 conditions requires the use of a link variable :

```
IF publisher = Cherisha AND author = Shalfield
AND subject = ExpertSystems THEN !link = true

IF !link = true AND computer = Archimedes
THEN program = Observess
```

Such cases have been considered by the author though and are mentioned in the manual. While consulting the expert, you obviously do not want the user to be asked what value link should have, as he will probably have very little idea what link means. To this end, prefixing a variable with ! (as above) indicates that the computer may never ask questions about this variable.

There are some other drawbacks to the simple form of rules used here - frequently an expert is expected to make decisions on incomplete data. For instance if a torch doesn t work and the batteries aren t flat, then you can be fairly sure that the bulb has gone, rather than one of the wires inside the torch having broken. But how do you express this probability in an expert system?



scsi::Mycroft.$.Observess.Coinage

Knowledge Base

Correct as at: 25 Feb 1994

Title: Current British Coinage

Purpose: Which Coin?

Expert: Rebecca L. Shalfield

Size: 1444 bytes

Know Eng: Rebecca L. Shalfield

Add rule

IF not	□	Colour	=	Gold
AND not	□		=	
AND not	□		=	
THEN not	□	Coin	=	£1

Certainty Factor: 100 % OK

Building up a knowledge base is a simple matter of adding rules

Observess solves this by the cunning expedient of adding Certainty factors to each of its rules. Every rule is given a percentage which represents how likely it is to be true. So the situation with our broken torch would become:

```
IF NOT batteries = flat THEN reason =
brokenbulb
 (90%)
```

Using these percentages Observess can be made to come up with suggestions in much more complex cases than it could otherwise. It will even give you several recommendations and the percentage chance of each one.

Once you have the knowledge base in place, you can then Consult with your new expert. To do this you simply choose the Consult option on a menu, and answer the computer s questions. For instance in the supplied Coinage knowledge base we are asked:

Is it true that: Colour = Gold ?