security benefits). Because knowledge, frequently expressed as a collection of rules, is crucial to the operation of such systems, they are also known as 'rule-based' or 'knowledge-based' systems.

Expert systems have proved strikingly successful in several fields. Already they can outperform skilled human practitioners in medical diagnosis, prospecting for minerals and in many other fields. Because of these dramatic successes they are emerging from artificial intelligence research laboratories and finding applications in general computing practice, with significant consequences for the way people build high-performance computer systems. The knowledge-based approach to software design replaces the tradition of:

DATA + ALGORITHM = PROGRAM

with a methodology based on:

KNOWLEDGE + INFERENCE = SYSTEM

which is an evolutionary change with very significant consequences.

So an expert system is based on a body of knowledge. There are in fact another three essential components of a fully fledged ES: the 'inference engine', which forms new rules for interpreting data on the basis of existing rules and data; the 'knowledge-acquisition module', through which the system acquires knowledge from its own experience, and from that of human experts; and the 'user interface', which allows non-experts to interrogate and use the system. Let us look at the two core modules — the knowledge base and the inference engine — in greater detail.

A knowledge base contains facts (or assertions) and rules. Facts can change rapidly — for example, during the course of a consultation. Rules are the longer-term information about how to generate new facts or hypotheses from what is presently known. How does this differ from a conventional database? The major difference is that a knowledge base is more creative. Facts in a database are normally passive: they are either there or they are not, to be filed or retrieved by the user as required. A knowledge base, on the other hand, actively tries to fill in missing information in the light of what it already 'knows', and independently of the immediate demands for data.

Rules in the IF...THEN format (known as 'production rules') are a favoured method of expressing 'rule-of-thumb' knowledge. For example:

> IF the home team lost its last home game, AND the away team won its last home game by two goals
> THEN the likelihood of a draw is multiplied by 1.088.

These rules are not embodied in program code: they are data for the inference engine, which is a high-level interpreter.

There are two major high-level strategies for

Weather Forecaster

HULK data files are created by the user as BASIC program files, thus making them easily accessible for inspection and editing. The file we used is typical.

We ran **HULK** using this file, and were asked for a hypothesis about the data; the hypothesis was:

TOMORROW=1

meaning that we are interested in those samples where the last variable has value 1 — in other words, days on which it rained on the following day. We want to use **HULK** to establish rules to enable us to predict from today's weather what tomorrow's will be.

Next we are able to enter rules for testing. Each rule is applied to the data file and its success in predicting rain is measured and reported, both as a single rule, and when used in combination with previous rules. **HULK** advises whether or not to add the rule to the rule set, but the decision is left to the user.

We quickly found three rules for predicting rainfall tomorrow:
1) If it rains more than 2mm today

AND

2) If the sun shines fewer than 3.5 hours today

AND

3) If maximum temperature is less than 6 degrees higher than today's minimum temperature

THEN

you can be 83% confident in predicting rain for tomorrow (given, of course, that this month's weather data is a fair sample of the whole year).

Although this is a BASIC program file, it is not intended to be RUN; it is simply a convenient way of storing data

```
1 WEATHER,30,5
```

Line 1 describes the file: containing its name (WEATHER), the number of data samples (30), and the number of data items (5) per sample. The file contains weather data for one month in London: each day being described in terms of minimum and maximum temperature (in centigrade), daily rainfall (in mm.), hours of sunshine, and a Boolean variable whose value is 1 if it rains the following day, and 0 if it doesn't.

```
100 MINIMUMT
200 MAXIMUMT
300 RAINFALL
400 SUNSHINE
500 TOMORROW
```

Lines 100-500 give variable names to the data in each sample

```
1001 D01,54,110,175,32,1
1002 D02,42,125,041,62,1
1003 D03,76,112,077,11,1
1004 D04,27,105,018,43,0
1005 D05,30,120,000,95,0
1006 D06,44,106,000,55,0
1007 D07,48,094,000,51,1
1008 D08,68,092,055,48,1
1009 D09,64,102,048,41,1
     --------"--------
1028 D28,58,154,000,20,1
1029 D29,67,088,064,42,0
1030 D30,45,096,000,68,1
```

Lines 1001-1030 contain the data, with one sample per program line, and each line starting with a label identifying that sample. Notice that HULK values must be integers, so all data has been multiplied by 10 to preserve precision:

Line 1001, for example, should really read
1001 D01,5.4,11.0,17.5,3.2,1 and so on for the rest of the file.

TOMORROW = 1

ASKING FOR RULE [1]

RULE IS: **RAINFALL > 20**

CONTINGENCY TABLE	SUCCESS	FAILURE
RULE TRUE	12	2
RULE FALSE	5	11

SUCCESS RATE = **76.3%**

BITS PER SAMPLE

BEFORE	**RAINFALL > 20**	**1.00**
AFTER	**RAINFALL > 20**	**0.85**

YOU ARE ADVISED TO [KEEP] THE RULE OK

RULE HAS BEEN [ADDED] TO THE RULE SET

Bits Per Sample
This is a measure of how many of the data samples are contributing to the success of the rule