Artificial Intellignce: Art or Science?

Artificial Intelligence: Art or Science?

Alan Bundy

DAI Research Paper No.

Paper delivered to the Royal Society of Arts on 2nd March 1988 and to appear in the Society's Journal.

Artificial Intelligence: Art or Science?

Alan Bundy

Department of Artificial Intelligence
University of Edinburgh

Abstract

Computer programs are new kinds of machines with great potential for improving the quality of life. In particular, expert systems could improve the ability of the small, weak and poor members of society to access the information they need to solve their problems. However, like most areas of computing, expert systems design is currently practiced as an art. In order to realise its potential it must also become an engineering science: providing the kinds of assurances of reliability that are normal in other branches of engineering. The way to do this is to put the techniques used to build expert systems and other artificial intelligence programs onto a sound theoretical foundation. The tools of mathematical logic appear to be a good basis for doing this, but we need to be imaginative in their use – not restricting ourselves to the kind of deductive reasoning usually thought of as 'logical', but investigating other aspects of reasoning, including uncertain reasoning, making conjectures and the guidance of inference.

Acknowledgements

Some of the research described in this paper was supported by SERC grant GR/E/44598 and earlier SERC and Alvey grants. I would like to thank Mike Uschold for comments on an earlier draft.

1 A New Kind of Machine

The advent of the computer has provided us with a new kind of machine: the computer program. We are used to physical machines, like steam engines, looms, pianos and cars. Computer programs are like these more familiar physical machines in that their behaviour is determined by laws, but they are unlike physical machines in that they are not made of physical stuff.

Note that I am carefully using the term *computer program*, and not *computer*. Computers are physical machines. They are made from silicon chips, wires and metal boxes, and this stuff can be seen, touched, smelt, tasted and heard. Computer programs are made from procedures, loops, conditionals, *etc.* These components cannot be sensed. *Representations* of computer programs can be sensed: a lineprinter listing can be seen; the holes in a paper tape can be felt. But these are no more the program than a particular gramophone record or musical score *is* Beethoven's 5th symphony.

To emphasise both the similarities and differences between computer programs and physical machines, we will call computer programs: *mental machines*. Other examples of mental machines are: cooking recipes, car repair instructions, knitting patterns and musical scores. Originally all these other mental machines required humans to interpret and execute the instructions constituting the mental machine. Then special purpose physical machines were invented to run them automatically, *eg* the Jacquard loom to execute knitting patterns, the player piano to execute musical scores. Once we can run such sets of instructions totally automatically then their

mechanical nature becomes more apparent: like physical machines their behaviour is determined by laws and rules. This justifies our calling them mental machines.

What distinguishes the computer program from these earlier mental machines is that they are not limited to some particular kind of task like knitting, cooking or making music, but can be used for any kind of task. Nor are they limited in complexity. They can be arbitrarily complex in a sense that can be made mathematically precise.

The differences between these new mental machines and the physical machines we are used to is much greater than the differences between any two physical machines. Despite the very different technologies underlying the steam engine, the internal combustion engine and the electric motor, they are all made from physical stuff, they can all be used to turn spindles and they can all be used for similar tasks, *eg* moving vehicles. Computer programs are not made from physical stuff, they do not have moving parts and they perform mental not physical tasks.

Computer programs are having and will have a profound influence on our society. This influence can be for good or ill, and will probably be both. In order to minimise the ill and maximise the good it is vital that people understand their nature and potential. Unfortunately, because computer programs *are* so very different from anything that has come before, very few people *do* understand their nature and potential. In these circumstances it is up to the engineers dealing with the new machines to explain them to the general public as well as they can.

A new branch of engineering is responsible for the theoretical advances and practical application of computer programs, but at the moment it is more an art than a science. This is normal in the early stages of a branch of engineering. We will argue below that computing must become more of an engineering *science*. In doing so it will be following in the footsteps of other engineering sciences from civil engineering to electronics. Despite the newness of the machines with which it deals, computing can learn many methodological lessons from its predecessors along this road from art to science.

2 What is Artificial Intelligence?

In this paper I want to address only one aspect of computing, the aspect with which I am myself involved: *artificial intelligence* (AI) and its application to *expert systems*. I will now try to explain what these terms mean.

AI is the building of computer programs which emulate human intelligence, *ie* the striving to create programs that equal or exceed human intelligence without necessarily achieving that intelligence in the same way as humans. In 1988 we are nowhere near achieving that goal, but we can build computer programs that rival human intelligence in narrow areas of expertise, for instance, medical diagnosis or mechanical trouble shooting. Such programs can be commercially useful in medicine, industry, education, defence, *etc*. They are called expert systems.

Many expert systems consist of a collection of if/then rules, of the form:

IF	$effect_1$ is present
AND	$effect_2$ is present
	$\vdots$
AND	$effect_n$ is present
THEN	*cause* is present with certainty factor c

together with observed symptoms of the form '$effect_i$ is present with certainty factor c'.

For instance, here are some toy expert system rules for crime detection. By using the word 'toy' I am implying that I am not suggesting crime detection as a serious task for expert systems nor am I suggesting that these rules should be taken seriously.

RULE-1	IF	*suspect* had motive
	AND	*suspect* had opportunity
	THEN	*suspect* committed *the crime* with certainty factor $0 \cdot 4$
RULE-2	IF	*witness* saw *suspect* commit *the crime*
	THEN	*suspect* committed *the crime* with certainty factor $0 \cdot 8$
RULE-3	IF	*suspect* was at *the scene of the crime*
	THEN	*suspect* had opportunity with certainty factor $0 \cdot 9$
RULE-4	IF	*suspect*'s fingerprints were found at *the scene of the crime*
	THEN	*suspect* was at *the scene of the crime* with certainty factor $0 \cdot 7$

where the terms in italics are variables which can take different values according to the case.

The observed symptoms in a particular case would be supplied as facts, *eg*

- FACT-1: John's fingerprints were found at Mary's house with certainty factor $0 \cdot 6$.
- FACT-2: Helen saw John commit Mary's murder with certainty factor $0 \cdot 7$.

where 'John' is the *suspect*, 'Mary's house' is *the scene of the crime*, 'Helen' is the *witness* and 'Mary's murder' is *the crime* in this particular case.

A collection of such rules and facts is called a *knowledge base*. The number, c, associated with each rule or fact is meant to associate some uncertainty with it, *eg* on a scale of 0 to 1, where 0 means false, 1 means true and intermediate values mean intermediate degrees of uncertainty.

A fault diagnosis system might use a knowledge base to reason from the observed symptoms of a fault to possible causes. Several rules might be chained together to go from observed symptoms via intermediate causes to final causes. The certainty factors on each rule in the chain are combined arithmetically to give an certainty factor to the final cause. Several such chains might suggest the same final cause, in which case the individual certainty factors are accumulated to give an overall certainty factor to it. Several different final causes might be suggested for the same set of observed symptoms. The certainty factors associated with the rival causes enables them to be rank ordered. The sequence of rules can be used as an explanation of the expert system's reasoning in coming to its conclusions, and gives a basis on which a human user might decide whether to accept this conclusion.

For instance, our crime detection expert system might chain together RULE-4 and RULE-3 with FACT-1 to infer that 'John had opportunity' with certainty factor $0 \cdot 378$. This is done by matching FACT-1 with the IF part of RULE-4 and then matching the THEN part of RULE-4 with the IF part of RULE-3. Many alternative schemes are used for combining certainty factors. I have multiplied them together ($0 \cdot 6 \times 0 \cdot 7 \times 0 \cdot 9 = 0 \cdot 378$), which is mathematically correct if they are interpreted as conditionally independent probabilities. Additionally, our expert system might use FACT-2 and RULE-2 to infer that 'John had opportunity' with certainty factor $0 \cdot 56$. If these two inferences are taken as independent evidence then the system can combine them to give an overall certainty factor of $0 \cdot 72632$, given by the probability calculation: $0 \cdot 378 + 0 \cdot 56 - 0 \cdot 378 \times 0 \cdot 56 = 0 \cdot 72632$.

The UK expert system community has been very successful in the development of small scale, commercial, rule-based, expert systems. A typical example is a fault diagnosis system for an electronic gadget, consisting of a knowledge base of

less than 1000 rules (and sometimes less than 100 rules), running on a personal computer, in one of the many commercial expert system shells. An expert system shell is a program which can chain rules together and combine certainty factors, but which comes without any rules. Customers provide their own knowledge base for the task they want solved. Part of the success consists of the unexpected (to me anyway) discovery of a large number of commercially interesting problems which yield to such a simple mechanism. Expert systems are especially flexible computer programs. They can chain together their rules in a variety of ways according to the problem they are called upon to solve. Each rule corresponds to a piece of expert knowledge which can be understood independently of the others. A chain of such rules can be used to explain the reasoning of the system to a user.

But these expert systems have, as yet, only been applied to a narrowly defined task and most are only in prototype form. Many companies have built fault diagnosis systems, but these are mostly for in-house use on a small piece of machinery. Many disease diagnosis systems have been built, but they are usually for a narrow ranges of diseases and are not yet in wide-spread use in hospitals or consulting rooms. Similar remarks could be made about other kinds of expert systems.

Some observers have been disappointed by the limited success and slow commercial exploitation of expert systems, which has not matched some of the early predictations about their potential and power. Such slow development is typical of new technologies, as is exaggeration during the early phases by researchers not versed in the difficulties of marketing and other commercial realities. However, some of the slowness has been caused by problems with the technology itself. These problems are addressed below.

3 The Potential of Artificial Intelligence

Most of the interest in expert systems is not because of their proven capability, but because of their potential. Many people recognise the tremendous benefits that could be brought to industry, government, medicine, education, voluntary groups and ordinary people in their homes, by the easy and cheap availability of information on a wide range of topics. Particularly, if a computer was available to help people find just the information they wanted and then to help them apply it to their problem. Expert systems can do this by reasoning from the user's problem to pinpoint the information required to solve it.

How many of our problems arise because we are ignorant, *eg* because we don't know: how to cure our sickness, or how to become richer, or whether our rights are being infringed, or where to go for help? The information we require is known, but it is hidden from us in legal or medical tomes, or in the heads of professionals.

A third world farmer may have plants decimated by disease, but not know what the disease is nor how to cure it. This knowledge may not be available in the farmer's village at all, or not at an affordable price. An expert system for diagnosing plant diseases could assist in identifying the disease and suggesting an affordable cure.

Some unemployed people do not have enough money to support themselves. Although they might be entitled to more benefits, they may not be claiming them because they lack access to independent advice. An expert system on DHSS regulations might advise them on their rights and on how to go about claiming them.

A group of objectors to a scheme to build a new road might feel lost in the maze of regulations that they have to cope with to phrase their objections. An expert system might help them locate the relevant parts of those regulations and then assist them in formulating their case.

A small company designing a new widget might not be able to afford the investment in time and effort required to make the widget as efficiently and hence

as competitively, as possible. A expert system might assist them in the design by interfacing to mathematical modelling packages, giving expert advice on the design choices, *etc*, thus enabling them to make a competitive product.

It is not a coincidence that my examples all involve the small, the weak and the poor: I believe that these are the groups with most to gain from the provision of cheap information and advice. Expert systems could speed the provision of true equality and democracy.

Of course, all new technological advances have the potential for both good and bad applications. Above I have suggested some good (I think) applications, but the same technology could also be used to take power away from ordinary people and give it to an elite, or even lodge it with the computer systems themselves. We need to be aware of both the potential benefits and dangers of expert systems, and try to encourage the benefits and avoid the dangers.

An example of a bad (I think) application would be an expert system for the launch on warning of nuclear missiles. Such a system would interpret the sensory data collected by military satellites, assess whether they constituted evidence of an enemy missile attack and decide whether to retaliate and if so how. Some military planners have speculated on the desirability of such a system as a way of coping with the speed of modern guided missiles and the way these may force defenders to take deployment decisions in a matter of minutes, or risk losing their capacity to retaliate. The flexibility of expert systems seems to make them the most appropriate kind of computer program. Unfortunately, the technology is not equal to the task. Computer programs are usually unreliable – current expert systems inherently so, because of their built-in certainty factors, *etc*. Unreliable computer programs are normally made acceptably reliable by a period of 'on the job' trial, error and correction. However, a launch on warning system could never be so corrected because we have no spare planet on which to test it. (Similar remarks apply to any other kind of computer program.)

4 Unreliability of Expert Systems

This inherent unreliability of expert systems is a fatal flaw in a launch on warning system, but it is also a serious limitation in any application. In a real time application, unreliability can cause chaos. In a life critical application, unreliability can cause injury or death. Even in more mundane applications, unreliability can cause expensive mistakes and lead to users rejecting the product. The unreliability of expert system technology makes it difficult to decide whether it will help solve a customer's problem, and so makes it difficult to predict whether an application will be successful. Thus reliability is a very desirable feature both for suppliers and users of expert systems. In the rest of this paper I want to examine the causes of unreliability in expert systems and point the way to a remedy.

What do we mean by the term *unreliable* as applied to an expert system? It is a catch-all term, and can include any of the following overlapping phenomena.

- **Fragility** (non-robustness): The system may fail in unexpected ways.
- **Unpredictability**: The user either cannot specify the circumstances under which the system will produce an answer or cannot specify the type of answer that it will produce.
- **Brittleness** (non-flexibility): The system cannot deal with problems on which it has not been previously tested.
- **Discontinuity**: The system gives very different output in response to similar input.

To see why expert systems are inherently unreliable, consider the following examples.

Suppose our expert system for crime detection is investigating the murder of Mary. When it asks a witness whether 'John hated Mary', the witness can choose to associate a certainty factor with the assertion. What would an answer of $0 \cdot 5$ mean? The system's designer may have intended $0 \cdot 5$ to mean a mild form of hate. The rules of the system will embody this meaning and draw appropriate inferences from it. However, the witness might use $0 \cdot 5$ to mean that someone hated Mary intensely, but that it might not have been John. Consequently, the expert system may fail to suggest a culprit for Mary's murder or may suggest an innocent person.

Most expert system shells provide a facility for attaching numbers to rules and assertions to represent 'uncertainty', but few explain what these numbers mean. Many alternative interpretations are possible. This may cause the numbers to be used differently by different knowledge engineers and users, leading to a clash of expectations about the expert system and, hence, to unreliability.

Most expert systems work by chaining rules together in a form of inference. The actual chains formed will usually depend on the particular problem. Sometimes there are lots of very long chains formed, most of which do not amount to anything. Expert systems are particularly prone to this when they have lots of rules or when they have some very general rules, *ie* expert systems designed to tackle a wide range of problems are especially prone. The system becomes bogged down in the possibilities and may not return with an answer in a reasonable length of time. This phenomenon is called the *combinatorial explosion*. It is difficult to predict in advance when an expert system might explode combinatorially. This unpredicatability is a form of unreliability.

Some expert system builders try to avoid this problem by attempting to eliminate the chains of rules that are not going to come to anything. A standard way to do this is to add extra, so called *heuristic*, conditions to the IF parts of rules. A heuristic is a rule of thumb that is not guaranteed to work but often does. These heuristic conditions do not contribute to the truth of the rule, but rather refer to the situations in which the rules should be used. We say that they are *control knowledge* rather than *factual knowledge*. For example, to avoid RULE-3 being used in a situation where the suspect has an alibi and the rule is bound to fail, the system builder might add an appropriate heuristic condition, *ie*:

RULE-3a	IF	*suspect* does not have an alibi
	AND	*suspect* was at *the scene of the crime*
	THEN	*suspect* had opportunity with certainty factor $0 \cdot 9$

Note that this heuristic condition is not required to establish the truth of the rule, but only to control its use.

Such control knowledge tends to make assumptions about the state of the inference process: assumptions which are often violated, particularly when a new user adds additional rules to the knowledge base or someone decides to use the same rules for a slightly different task. This can cause unpredictable behaviour, *eg* a rule not being used when it is appropriate and the system failing to produce an answer when it is expected to. Thus this mixing of control and factual knowledge causes another form of unreliability.

In general, the unreliability arises because of a lack of theoretical understanding of the technology of expert systems. The design of expert system shells and expert system rules is an art form – an art that can lead to a useful and successful product in the right hands, but can, more often, lead to an unreliable mess in the wrong hands.

5 Comparison with Engineering Sciences

Contrast the situation of expert systems with more mature branches of engineering. Before building a bridge, a structural engineer makes a number of drawings and carries out elaborate calculations. These calculations enable the behaviour of the bridge to be predicted under a wide variety of stresses. As a result the engineer can say, with confidence, that for a specific range of loading and weather conditions the bridge will behave robustly, predictably, flexibly and continuously. If the engineer does not carry out these calculations or does so inaccurately, and the bridge falls down, then he/she is guilty of professional negligence (*cf* the Tay Bridge disaster). Of course, the bridge may still fall down even if the engineer makes all the right calculations. Freak weather conditions may exceed those assumed in the calculations, or a faulty component may not meet its assumed specifications. So the engineer's calculations do not provide a guarantee of reliability, but only some weaker kind of assurance.

There is not, yet, the same tradition in the field of artificial intelligence. In fact, there is a common assumption that analogous assurances of reliability could not be given; that algorithms do not exist on which analogous calculations could be based; that the heuristics on which expert systems are based lead to an inherently unreliable product.

I will argue that this is wrong. It is possible to base expert systems upon techniques which lend themselves to the giving of reliability assurances. To do this we need to make AI into an engineering *science*; we need scientific knowledge about AI so that we can design large, usable and reliable expert systems. But before we can gain this scientific knowledge we need first to understand what form it would take. Therefore, we need to understand what kind of discipline AI is.

6 The Nature of Artificial Intelligence

Some people pursue AI research with the intention of building useful applications for commercial, military, educational, *etc.*, use. I will call such research: *applied* AI. Other people pursue AI research with the intention of modelling human intelligence. Such research is usually called *cognitive science*. However, much AI research does not fall into either of these camps. Indeed, until the recent interest in applied AI, the vast majority of AI fell outside these camps. I shall call this third camp: *basic* AI.

The aim of basic AI is to develop computational techniques with the potential for emulating intelligence. We want to invent new such techniques, improve them and discover their properties and relationships to other techniques. We don't want to be concerned immediately with whether this is how humans do things. Nor do we want to have to justify our new techniques on commercial grounds.

AI techniques include:

- algorithms, for instance for chaining together rules to draw inferences;
- formalisms, for instance for representing changes caused by actions;
- organisational schemes, for instance for coordinating the interaction of several programs; and
- knowledge elicitation techniques, for discovering the reasoning that people use to solve a problem.

Many examples of these can be found in [1].

Basic AI is a little like applied mathematics, which develops mathematical techniques with the potential for modelling the physical world, but independent of any specific application. It is also a little like 'pure' engineering, which develops new engineering techniques, but without trying to sell the test rig as a product. The analogy is developed more fully in table 1. Computer science is like mathematics: they are the super-fields of which AI and applied mathematics are sub-fields. Psychology is like physics: they are the fields to which AI and applied mathematics are applied. Cognitive science is like theoretical physics: they are the sub-fields of psychology and physics in which the applications of AI and applied mathematics are studied.

Computer Science	Mathematics
Psychology	Physics
Basic AI	Applied Mathematics / Pure Engineering
Applied AI	Applied Engineering
Cognitive Science	Theoretical Physics

Table 1: An Analogy with Mathematics and Physics

Hence we can expect basic AI to continue to grow by the gradual discovery and development of new techniques, just as applied mathematics and pure engineering continue to grow. The application of these techniques to expert systems will produce a steady improvement in their power, range and reliability. There will not be a single identifiable event: 'the invention of artificial intelligence' to which we could assign a date. We cannot expect a brief period of accelerated funding to produce all the techniques that we require. We cannot expect the discovery of 'Laws of Nature' in basic AI, just as we would not expect them in applied mathematics or engineering. However, laws about human or animal minds might be discovered in cognitive science: the application of basic AI to psychology.

New AI techniques often first emerge as *ad hoc* tricks during a process of *exploratory programming*. Exploratory programming is a process of trial and error development of a program for a task and is part of the art of artificial intelligence. In the past some researchers have been content to stop there, declaring that the program *is* the theory. But in making AI into an engineering science we must not be content with this stage. We must go on to isolate the new technique, clean it up and gain a theoretical understanding of it. This theoretical understanding should provide the basis for the calculations required in giving reliability assurances.

7 The Role of Logic

In making the calculations required in bridge building, algebraic and differential equations and arithmetic play a crucial role. More generally, arithmetic, algebra and calculus have played a fundamental role in physics, applied mathematics and engineering. Mathematics is also important in AI. The branch of mathematics that has proved most fruitful is logic.

Logic has been used for representing knowledge and inference in all subfields of AI. It is used both directly as a representation language for knowledge, and indirectly as a theoretical tool for the analysis and rational reconstruction of *ad hoc* techniques and for the comparison of alternative formalisms. In particular, some researchers have used logic to represent expert system rules and have used the form of inference sanctioned by logic, *deduction*, to chain those rules together.

What makes logic such a powerful tool? Logics provide not just a language in which to write expert system rules, but a systematic way of relating this language to the real world being represented – called a *semantics*. This semantics gives the logic user a way of checking the correctness of the representation, and a way of checking the correctness of the programs for manipulating the representation. For instance, a deduction program should generate all and only the *logical consequences* of the knowledge base. The logical consequences of a knowledge base are just those further rules and facts that are true in any interpretation of the original knowledge base. Terms like 'logical consequence', 'semantics' and 'deduction' can be given a precise mathematical meaning and the correctness and completeness of the deduction program can be proved.

If you use deduction in an expert system then you will have a guarantee of what it will and won't do. You can safely build a large knowledge base and know that the system will behave reliably – that it will generate all and only the logical consequences of what it has been told, and that what it was told is what you intended it to be told. We should not be over confident about this state of affairs. What you told it may not be what you should have told it. Some of the logical consequences of what you told it may surprise you; mathematics is full of such surprises. And it may not be logical consequences that you really wanted to have generated. But deduction programs are sometimes useful and their authors do have some control over what they are able to do.

Contrast this with a program made from *ad hoc* techniques. Such a program may work well on a few toy examples, but this offers no assurance about how it will behave on other examples, particularly on big complex ones. It may not work at all, or it may produce results diametrically opposed to those you wanted. It is a bit like trying to build the Forth Road Bridge by scaling up a model footbridge.

In order to accelerate the use of logic within AI (and elsewhere in Computer Science), the Science and Engineering Research Council have recently launched an Initiative in Logic for Information Technology, [2]. This will fund logicians to work closely with computer scientists in the application of logic to AI. We hope that this will lead to the deeper understanding of existing AI techniques, an increase of their power and range, and the development of new techniques, all based on a solid theoretical foundation.

8 Examples of Logic-Based Techniques

My research group at Edinburgh University uses logic to build computer programs which emulate mathematical reasoning. We call our research programme the **DReaM Project**, for **D**iscovery and **Rea**soning in **M**athematics, [3].

Logic has long been used for representing mathematical formulae and deductions with such formulae. We have also used it to represent the process of controlling the search for a proof, the learning of new knowledge and the formation of mathematical representations of informally stated problems.

In particular, our use of logic to control the search for a proof offers a solution to the problem of unreliability caused by the combinatorial explosion. Instead of inserting heuristic control conditions into our mathematical formulae or into the rules of an expert system, we provide a separate logical language for expressing control information. We call this this control language a *meta-logic* and we use it to build *proof plans*, [4].

A proof plan is a sequence of problem solving methods, each of which produces a particular kind of effect on the mathematical formulae it manipulates. For instance, one method might bring two symbols in the formula closer together. Another method might eliminate certain kinds of symbol. The meta-logic is used to

give a precise description of the conditions under which each method is applicable and what effect it has if it succeeds. Given a problem and a definition of what it means to solve it, the descriptions of these methods can be used to form a proof plan to guide the search for a solution.

Proof plans are also potentially useable to guide the rule chaining of an expert system. This logic based technique appears to have a number of advantages including solving problems of reliability, namely:

- **Efficiency**: The proof plans will guide the inference process and avoid the combinatorial explosion associated with deep search in large knowledge bases.
- **Predictability**: The meta-logical proof plan gives a formal account of the success/failure of the control strategy. With this we can better predict the behaviour of the expert system. We are able to say, not only what kind of conclusion can, in principle, be inferred, but how likely it is that any particular conclusion will be inferred within a reasonable period.
- **Flexibility**: Sometimes a method may fail even though it is applicable, *eg* because a rule of the right form is not in the knowledge base. However, since the preconditions and effects of a failing method are known, it is possible to replan and patch the gap in the plan with a new subplan.
- **Maintainability**: By separating the factual from the control knowledge we improve the modularity of the expert system, making it easier to modify either type of knowledge. The unreliability caused by mixing factual and control knowledge is avoided. This is particularly important for large scale knowledge bases where the factual knowledge may be accumulated and improved over a number of years and put to different uses by changing the control strategy.
- **Generality**: The use of a meta-logic to express the plans makes it possible to develop a few general purpose plans capable of guiding many different inferential chains.
- **Explanatory Power**: By expressing the control decisions in a meta-logic we have provided an alternative language for explaining the reasoning of the expert system, *ie* instead of answering "why" questions by printing out a chain of rules, we can describe the current method and its place in the overall plan. Such meta-logical explanations are often more intelligible to the human user. This argument applies especially when the user is being asked to interact with the expert system in order to help guide the search.
- **Co-operability**: We can specify not only the methods available to the expert system, but also the problem solving abilities of the user. Proof plans can consist of a combination of system and user methods. This enables the system and the user to co-operate in the solution of a problem. Furthermore, the improved explanatory power described above enables users to understand better what the system requires of them and what the system is capable of doing for them.

We are about to address the application of proof plans to expert systems to try and realise the potential advantages listed above.

How does this work assist applied AI? A team at Hewlett Packard in Bristol are investigating the use of explicit, meta-logical control knowledge in the building of large, robust and long-lived expert systems. The factual knowledge contained in such an expert system might change over its lifetime, as the user's circumstances change. The control knowledge may also change, as the user thinks of new things

to do with the factual knowledge. In both cases we want the system to continue to work to specification. Meta-logical control promises this flexibility. More *ad hoc* techniques do not. This is just one of the way in which better understood AI techniques can lead to bigger and more robust expert systems.

Other groups are applying logic to other aspects of the unreliability of expert systems. For instance, Paris and Vencovská are investigating the use of logic to put uncertain reasoning on a firmer theoretical basis, [5]. This will enable us to reduce the uncertainty about what certainty factors mean, with a consequent improvement in reliability.

Cox and Pietrzykowski are using logic to develop a technique for conjecturing causes to explain effects, [6]. Note that the rule structure quoted in section 2 is back to front logically: causes imply effects not effects causes. Cox and Pietrzykowski put the rules the right way round, but then need to invent an inverted form of deduction, called *abduction*, for forming conjectures. Some of the uncertainty associated with conventional expert system is to cope with the fact that the rules are back to front. Using the rules the right way round enables us to dispense with this aspect of uncertainty with a consequent improvement in reliability.

9 Conclusion

Expert systems are computer programs which use the techniques of artificial intelligence to reason about some narrow area of human expertise. Computer programs in general, and expert systems in particular, are new kinds of machines whose nature and potential are not yet fully understood. Expert systems have great potential to improve the quality of life, but this potential has not, so far, been achieved. To achieve it fully, we need to be able to build bigger and more reliable systems. Artificial intelligence, currently practiced as an art, needs also to become an engineering science: providing the kinds of assurances of reliability that are normal in other branches of engineering. We must put the existing *ad hoc* AI techniques, developed from exploratory programming, onto a firm theoretical foundation, so that we can predict the effects of using them.

In this task we are aided by the tools of mathematics, particularly mathematical logic. My own research has been particularly concerned with the formal analysis of such *ad hoc* techniques and their rational reconstruction with the aid of mathematical logic. In particular, I have used logic to develop techniques for controlling inference. Other AI researchers are using this methodology to analyse and rational reconstruct other *ad hoc* techniques.

This theoretical work is normally seen as being the province of academic researchers, but it need not be exclusively so. In any case, the success of the theoretical work is of crucial importance to applied researchers, in industry and elsewhere; without it they will not be able to build reliable expert systems. But the interaction is not all in one direction. Applied researchers have important feedback to provide on the utility of existing knowledge engineering techniques 'in the field'. Together we can provide reliable and powerful expert systems which can help improve the quality of life, especially for the small, the weak and the poor.

References

[1] A. Bundy, ed., *Catalogue of Artificial Intelligence Techniques*, Springer-Verlag, 1990. Third Edition.

[2] S. Abramsky, A. Bundy, and B. Richards, "Initiative in logic for information technology." SERC, 1987. Available from the secretary to the Computing Science Sub-Committee, SERC, Polaris House, Swindon.

[3] A. Bundy, "Discovery and reasoning in mathematics," in *Proceedings of IJCAI-85* (A. Joshi, ed.), pp. 1221–1230, International Joint Conference on Artificial Intelligence, 1985. Also available from Edinburgh as DAI Research Paper No. 266.

[4] A. Bundy, "The use of explicit plans to guide inductive proofs," Research Paper 349, Dept. of Artificial Intelligence, University of Edinburgh, 1988. Short version published in the proceedings of CADE-9.

[5] J. B. Paris and A. Vencovská, "The maximum entropy approach to inductive reasoning," in *Proceedings of the AI-workshop on inductive reasoning*, Riso National Laboratory, 1987.

[6] P. T. Cox and T. Pietrzykowski, "Causes for events: Their computation and applications," in *Lecture Notes in Computer Science: Proceedings of the 8th International Conference on Automated Deduction* (J. Siekmann, ed.), pp. 608–621, Springer-Verlag, 1986.