A Selective Review Of Artificial Intelligence

Andrew Law

Commissioned by British Telecommunications
Research Laboratories
Martlesham Heath
Ipswich
IP5 7RE

Cognitive Science Research Paper

Serial No. CSRP 078.

Cognitive Studies Programme
The University of Sussex
Brighton
BN1 9QN

SSX
078

1. Overview

1.1. Introduction

This report provides a selective review of Artificial Intelligence (AI) research. Its purpose is to indicate some of the major areas of research, selected applications, issues to be resolved, and possible directions for future AI research.

The sources of information used in this document include the proceedings of recent AI conferences (ECAI '86, AAAI '86, Expert Systems '85), articles from relevant journals (e.g., Artificial Intelligence, Artificial Intelligence Review), and discussions with those involved in AI research at Sussex.

Whilst an attempt has been made to cover the major areas of AI research, it has not been possible to cover the whole field. Notable omissions from the report include reviews of research in machine learning and robotics.

The two major sections of the document are *Section 2: Applications* and *Section 3: Applications Infra-Structure*. There is no section on general AI theory or techniques (e.g., knowledge representation, planning, etc.). Instead, developments in these areas are discussed with respect to their possible applications. Section 2 contains a discussion of expert systems, natural language, vision, and speech processing. Section 3 contains discussions of the hardware, AI software development environments and knowledge elicitation techniques that are, or could be used in building the applications discussed in Section 2. It is expected that the reader has some general knowledge of AI. However detailed knowledge of the areas discussed is unnecessary.

1.2. Summary

This section provides a summary of the report. The section headings refer to the section headings of the rest of the report.

Section 2. Applications

2.1. Expert Systems

2.1.1. Applications

Expert systems are now being used in a variety of domains (e.g., engineering, medicine, education, mathematics, geology etc.,) and for a variety of purposes (e.g., diagnosis, monitoring, quality control, design etc.). The expert systems product and services in North America is presently worth $280 million and it has been suggested that this will rise to $1.9 billion by 1992.

2.1.2. Existing Problems and Future Developments

There are various problems with existing expert systems; they do not learn from experience; they often do not generate very illuminating explanations; they allow only very restricted forms of user interaction; and their performance does not degrade gracefully on the peripheries of the problem solving territory. Many existing expert systems reason with uncertain data or rules of inference. The current means of reasoning under uncertainty usually involves the use of some probability based inference technique, e.g., Bayes' theory. However, a major problem is that there are various problems with using this technique.

Future developments in expert system technology can be seen as, in part, attempts to reduce the need to represent uncertainty explicitly in expert systems. These developmentwill also contribute to the solution of some of the other problems mentioned above. Three of these developments are considered:

(a) *Blackboard architectures* offer the opportunity of representing various levels or types of knowledge independently. They also allow these different knowledge bases to make independent contributions to the reasoning process. These system will be used more widely in the future since they allow us to represent far more complex reasoning processes than traditional architectures allow.

(b) *Mixed-Initiative* expert systems allow the user and system to interact in a more flexible manner e.g., they are more flexible about who can take the initiative in the problem solving process.

It is expected that both of these developments in expert system architecture will have more commercial realisations in the next five to ten years.

(c) *Deep representation* is at present an ill-defined area of research. However, a common goal seems to be to produce models of the problem domain. It is hoped that these systems will improve the flexibility of expert systems, increase the complexity of the tasks expert systems can perform, improve their explanatory abilities and reduce the need to reason under uncertainty.

The major problems with deep representation systems concern the difficulty of producing adequate conceptualisations of the problem domain (e.g., spatial and temporal representations) and the problem of traversing large search spaces that are produced.

In this section there is also a discussion of the use of many sorted logics and the contribution they could make to the solution of the search space problem.

Where there are already adequate models/conceptualisations of a domain (e.g., electronic circuitry, simple mechanical devices and medicine) deep representation techniques are likely to have some impact. However the problem of conceptualisation will remain a major impasse in the application of deep representation systems in many other domains.

2.2. Natural Language Processing

2.2.1. Applications

Three main applications for natual language processing (NLP) systems are considered; machine translation, text summarisation and human-machine communication.

Reasonable, but not perfect performance can be achieved with existing machine-translation techniques. Pre- and post-processing can contribute to improving the performance of machine translation systems, to the extent that a four fold improvement (over unaided translation) in productivity can be achieved. A possible application for machine translation would be in translating electronic mail between countries.

Another application of NLP is in text summarisation. A prototype system is reported that summarises news and weather items. It can also translate between different languages. However, the author knows of no commercial text summarisation systems.

The final area of NLP application considered is at the human-computer interface. Several tools are now commercially available for building natural language interfaces to databases. However, in this section it is pointed out that NL communication will not be appropriate for all forms of human-computer interaction and that other forms of communication are available. The need and use of NLP techniques in interface design will depend as much on the success of the development of other forms of communication (e.g., pointing devices, high resolution graphics etc.), as on the solution to existing problems in NLP techniques.

2.2.2. Exisiting Problems and Future Developments

The major problem in NLP seems to involve the disambiguation of linguistic terms.

Semantic parsers will help in some disambiguation tasks. Tools for building semantic parsers are now becoming available. However the resulting interfaces are likely to be very domain-dependent, e.g., we might envisage a system that can deal with enquiries about train times, but not also about plane times and the weather.

It is expected that work on models of extended discourse will be needed to solve some problems of disambiguation and will contribute to making extended interaction with computers far more natural. Research on language acquisition may contribute important information about the nature of the knowledge involved in discourse and the pragmatics of language use. However it is unlikely that we will see many practical NLP systems based on models of extened discourse or the pragmatics of language use in the next 5-10 years.

Parallelism is unlikely to have a radical affect on practical NLP systems in the next ten years. It may have a limited effect in terms of the production of more efficient NLP systems.

23. Vision

*23.1** Applications

There are already commercial applications for image processing techniques, for example in optical character recognition, remote sensing **and** factory automation. The developments in hardware, particularly in VLSI design **and array** processors, mean that we are likely to see more efficient commercial low-level vision packages for these sorts of task.

23.2, Existing **Problems and** Future **Developments**

Low-level vision research seems to be an area of relative consensus and practical success. However, there is considerably more work to be done in higher-level processing, particularly in the areas of determination of surface properties, colour vision, motion perception, and the perception or planning of movements through cluttered space.

A promising area of research is the investigation of situation specific model based techniques.

Considering the problems in high-level processing, it is likely that most practical vision systems used in robotics, quality control etc. in the next five to ten years will rely heavily on additional sources of information (e.g.. lasar guidance) and "environmental support" such as limiting the number of possible objects in their field of operation or providing "clues" like deep shadows etc.

2A. Speech

2.4.1. Applications

Certain tasks will require speech processing (e.g., the processing of speech passing through telecommunications networks), and in situations where people can not use their hands to type (e.g.. in flying aircraft).

2.4.2. Existing Problems **and** Future Developments

The major problem with many existing speech processing systems is that they are limited in the form and quantity of input (they only deal with limited vocabularies, they often can not deal adequately with different voice types etc). It is likely that future speech processing system will utilise more sophisticated versions of the heterarchical processing method found in HEARSAY-H. It seems unlikely that we will produce connected speech understanding systems of human levels of competence in the next ten years.

Section **3.** Applications Infra-Structure

3.1. **Hardware**

3.1.2. Future **Developments**

Two main approaches are considered: the "evolutionary[11] approach and the "revolutionary[11] approach.

Evolutionary

It is likely that advances in VLSI design will make the most significant contributions to increases in efficiency of general purpose software in the next five years. It is possible that the work on "abstract instruction sets" for functional languages will also have some effect.

Revolutionary

In this section several "revolutionary" approaches are discussed. It seems that parallelism will affect only a limited number of tasks (e.g., low level speech and vision) in the next five years. It is likely that further developments in VLSI design and array processors will contribute in a quantitative way to these tasks. It is possible that limited parallelism will be used in general purpose programming.

While there seems to be considerable theoretical interest in connectionism, the possible consequences of the research are still unclear. Very little practical, applicable connectionist software is likely to appear in the next five years.

3.2. AI Languages and Environments

In this section there is a discussion of two issues: Developments that will aid in the increase of the production of AI based software and developments that will make software produced in AI development environments more relevant to the tasks at hand.

3.2.1. Increasing Productivity

It is possible that further developments of high level AI languages and interface devices will mean that more people can be involved in the production of AI software. Systems such as the Programmer's Apprentice are also likely to increase productivity significantly. A system is descibed, based on the Programmer's Apprentice, and used in telecommunications software development in Italy.

3.2.2. Increasing Relevance

Further work will be done on identifying the tasks to which the various high level AI programming languages are best suited.

Popular forms of these languages will be provided on specialised hardware, perhaps with limited parallelism.

It is also likely that we will see more multi-language environments, with cleaner, more efficient implementations. With further improvements in the memories of small target user machines, it may be possible to directly port programs developed in large software environments (running on large expensive machines) onto the smaller machines. Various ways of achieving this are discussed.

3.3. Knowledge Elicitation

As expert systems become more complex, and the demands for further sophistication increase, it is likely that existing knowledge elicitation techniques will be inadequate. There will be increasing pressure for the development of new knowledge elicitation techniques.

2. Applications

2.1. Expert Systems

An expert system embodies organised knowledge about some narrow domain. This knowledge enables it to perform some or all of the tasks in this domain, in a manner which we would consider requires "expertise" if performed by a human. The concept of the expert system arose in the '70s when AI abandoned, or postponed, attempts to develop generally intelligent machines, and turned instead to the development of systems that could solve narrowly focussed real world problems.

There are three fundamental components of a simple, classical expert system. The first component is the *knowledge base*. This contains symbolic representations of an expert's rules of judgement. These usually take the form of the "IF (some condition) THEN (some action or assertion)". The second component is the *inference engine*. This applies the rules from the knowledge base to the data to develop some conclusion(s). The final component is the *interface*. This is the channel through which information for and from the expert system passes. For example, it could be a natural language interface, allowing users to specify questions in natural language and, possibly, to receive natural language based explanations. In other expert systems it could simply process the outputs of some set of input devices (e.g., a temperature gauge) and drive some output devices or give advice to the user.

More sophisticated expert systems contain additional components. For example, schedulers or planners may organise the order in which rules are applied. Other components could include "justifiers" for providing justification of conclusions, and knowledge acquisition devices, which either automatically develop new rules when necessary, or which support the user in adding new rules or amending old ones.

2.1.1. Applications

Why do we need expert systems? Humans experts are scarce, and therefore expensive, often over-worked, mortal and fallible. However, computational systems can embody aspects of their expertise and can aid or replace them, or communicate their expertise to learners. Hence, expert system technology may, in principle, preserve expertise, make expertise cheaper, and make human experts more effective. While expert systems have mainly grown out of university research departments and R&D departments in large multinational corporations, they are now being used in a wide range of domains (e.g., education, medicine, engineering, science, mathematics, geology, knowledge engineering, etc.) and by many different institutions. Expert systems can be implemented to perform one or more of a range of functions, including fault diagnosis and interpretation, monitoring, prediction and control, design planning, and component compilation.

2.1.2. Existing Problems and Future Developments

Investment in expert systems technology in North America is estimated to be $400 million in 1986, with a product and services market worth $280 million in 1986 — this is expected to rise to $1.9 billion in the next six years (Hewett, 1986). Therefore, the expert systems market appears relatively healthy, and the application of expert systems technology seems to have been successful in many areas.

However, the extent to which we will see continued investment will depend, in part, on how we can improve the performance of expert systems by increasing their efficiency and accuracy, by extending their range of applications, and on the ease with which they can be created.

In this section there is a discussion of some of the limitations of existing expert system technology **and** ways of overcoming them. Four main areas of development are considered: techniques for reasoning under uncertainty, the development of blackboard architectures, mixed initiative expert systems **and** the development of deep representation systems. These developments **are** all primarily concerned with extending the **range** of applications of expert systems **and** in improving their problem solving abilities. However, it is shown that development in some of these areas also relates to improving explanatory facilities, and to introducing learning abilities to expert systems.

Other sections of this report discuss other important developments that may contribute to the survival or expansion of the expert systems market. In Section 3.2 there is a disussion of the tools that are available for expert system development, **and** in Section 3.3 there is a discussion of the problems of knowledge elicitation (the process of eliciting knowledge from experts in order to represent it in expert systems).

There are three main functions which, ideally, we should want an expert system to perform:

- To solve all problems in the domain.
- To present, and (when required) explain the result in an "acceptable[11] manner.
- To learn from experience, e.g., to restructure knowledge or amend its knowledge base.

These are all functions which human experts should be able to perform, however the performance of existing expert systems in each of these areas is often disappointing — when compared to the human.

At present most expert systems do not learn from "experience", and there is growing concern about the need for better explanatory capabilities.

The most important factor is that expert systems often cannot solve the problems with which they are presented. For example, a common criticism of expert systems is that, unlike human experts, they do not "degrade gracefully". As human experts approach the "boundaries" of their knowledge, they gradually become less proficient. That is, as human experts become uncertain about a domain, their performance degrades gracefully rather than precipitously, like expert systems (and many other computer programs). This means that expert systems are not very robust. A possible reason for this is that they cannot call on the sources of information or processes that a human might in these circumstances (common sense, first principles, analogies, etc.).

(a) Reasoning under uncertainty

Solutions to certain tasks involve sources of information that are "uncertain" for various different reasons. For example the data source may be noisy, ambiguous, imprecise or incomplete, the object world may be unpredictable or the rules of inference may be uncertain (the nature and the importance of the distinctions in forms of uncertainty **are** discussed below).

It has **been argued** that the use of classical logic alone is inadequate when reasoning under uncertainty, as the reasoning process operates with truth values which **are** different from TRUE **and** FALSE (e.g., Mamdani et al., 1985) This has led to the development of other reasoning and representational formalisms: e.g., modal, fuzzy or multivalued logics. Some of these techniques have been used in practical expert systems (e.g., MYCIN, Shortliffe, 1976; PROSPECTOR, Duda et al., 1978).

Mamdani et al., (1985) point out that all these systems use similar techniques and can be classed together as "variations on the Bayseian inference theme*[1]. In these systems the degree of uncertainty is usually represented by a numerical measure. The value is propagated through the inference chain. In the case of numerical representations, the numbers are often interpreted as a probability, and the inference step is based on Bayes' theory (although there are other techniques, see below).

However, there has been some dissatisfaction with the use of these techniques. Some objections concern theoretical issues, others are more pragmatic objections.

Theoretical Objections

(i) The justification of numerical computations by appeal to techniques such as Bayesian inference is theoreticaly invalid. This is because there are often dependencies between the rules — this is a violation of the assumption of conditional independence found in Bayes* theory.

(ii) Various sources of uncertainty exist, but they have been confused and represented as a single source in the systems developed.

These two reasons may not be considered all that important if our only concern is performance, and if we are satisfied with the performance of existing systems. However, there are also many pragmatic reasons for objecting to current techniques.

Pragmatic **Objections**

(i) *Development process* — The numerical information (e.g., certainty factors attached to rules or data) is often "elicited[11] through the process of iterative development. That is, there is a cycle of performance assessment and subsequent hand "tuning" of certainty factors. In the development of large systems this can be a laborious process.

(ii) *Computational demand* — It is often difficult to apply the Bayesian inference rule at each step of the inference chain. This is because not only the prior probabilities but also the joint probabilities are required to compute the probability of the hypothesis from the evidence. This requires an enormous amount of data and computation (often leading to ad hoc, piecemeal solutions bringing about the theoretical objection mentioned in (i)).

(iii) *Modularity* — Due to the dependencies between rules it is often difficult to add new rules without them affecting the certainty factors of other rules.

(iv) *Generality of applications* — The systems produced are often very "local*[1] in their potential for application. The result of the tuning process is often appropriate for only a limited domain. The resulting systems often have to be significantly altered in order for them to be applied in new domains.

(v) *Intelligibility* — All inference techniques require that the value of each proposition has a unique interpretation (TRUE **and** FALSE do have unique interpretations), unfortunately this is often not the case with numerical values. Also, and related to the previous point, the explanations produced by these systems have been considered "unnatural", or at least unhelpful. This means that the operations of the system are difficult to understand, check and debug.

Before we can identify ways of overcoming the problems of reasoning under uncertainty, it is necessary to examine the types of sources of uncertainty. Once this is done it is possible to identify what can be improved and how. A first attempt at identifying some important sources of uncertainty and techniques for overcoming the problems are discussed below (however this list should not be considered as exhaustive. For a more comprehensive review of sources of uncertainty see Cohen, 1986 and Reichgelt & van Harmelen, 1985). Almost all the techniques discussed here concern how to reduce uncertainty, not how to develop better techniques for reasoning under it.

Sources of Uncertainty

(i) Uncertainty in the object world

Uncertainty may arise because the object world is intrinsically unpredictable (e.g., in the areas of quantum mechanics or thermodynamics). There does not seem to be any way of avoiding inference under uncertainty in these circumstances.

(ii) Uncertainty in the fidelity of the input to expert systems

There **are** two main sources for input to expert systems: input devices (for example temperature guages used in monitoring systems), and users, who provide the data about which the system has to reason (for example a set of symptoms).

Uncertainty may be introduced into the reasoning process because the instruments used to guage the object world are unsatisfactory. They may be too "coarse", they may have limitations in the range of their sensing capabilities or they may be unreliable. This may mean that they can not disambiguate important features, or it may mean that certain information is missing.

Production of better input devices (e.g.. through making them more reliable, increasing their fidelity, increasing their range) may improve the performance of reasoning systems by decreasing the degree of uncertainty.

It may also be possible to improve performance by knowledge based post-processing of the outputs of the input devices. An example of this sort of technique is the partial use of top-down processing sometimes found in speech, NLP and vision processing systems (see Sections 2.2, 2.3, 2.4). This sort of technique allows the system to clarify or disambiguate the outputs of low level processing (even the outputs of the input devices) through the use of higher-level knowledge.

When dealing with a "noisy" object world (e.g., as in speech processing), it is also possible to limit or change the world in which the system will operate. For example, in speech processing the domains in which the systems work need to be restricted if we want to produce optimal performance, (e.g., talk slowly, in monotone, only use the words specified in the vocabulary X etc.)

Finally there is the case where there is missing information. This may be due to the inadequate nature of the input devices or because the user has not provided the appropriate data.

There seem to be four main approaches to dealing with the problem of missing data: First, it is possible to use alternative sources of information which provide supporting evidence (for example see the discussion below on blackboard architectures). Second, it may be that the user has not specified all the information which is required or has not specified it clearly enough. In this case it is necessary for the system to recognise that the information is inappropriate and to ask the user to provide the appropriate information, perhaps also giving advice about how to get this information. There is a more detailed discussion of these sorts of mixed initiative system below. The third approach involves the use of intelligent planning. As Cohen (1986) points out. uncertainty is often due to the timing of the provision of evidence, and therefore it can be minimized by ordering actions so that the timing of the presentation of evidence is most facilitative. He gives the example of buying a box which is the correct size for a birthday present. If the box is bought first, then certain information is missing in deciding the size of the box. It makes much more sense to buy the present before the box so that one can be certain that it will be the right size.

The fourth approach involves the development of systems that can reason under incomplete information, for example, by making assumptions and being able to refine their reasoning if those assumptions are confirmed or denied at a later point. These systems, while they reason under uncertainty, reason *about* it, rather than *with* it. Systems employing non-monotonic and default logics allow for this sort of inferential process. In these systems assumptions are made which may have to be revised in the light of new information. This means that they have to maintain information about the source of the uncertainty.

One of the first formulations of a non-monotonic logic is given by McDermott & Doyle (1982), and the best known practical system based in non-monotonic logic is Doyle's Truth Maintenance System (TMS) (Doyle, 1979). There is an extensive mathematical (e.g., Hughes, 1972) and philosophical (e.g., Bradley & Swartz 1979) literature on these logics and many alternative formulations of non-monotonic logics have been developed (e.g., see Gabbay, 1985). However few expert systems have been built which use TMS systems (an exception is the EXTASE system, see Worden et al., 1986).

It is expected that TMS systems will become more refined over the next five years and will be used in commercial expert systems. It is also possible that TMS packages will become available in expert system development environments in the next five years.

(iii) Uncertainty in the rules of inference

Uncertainty may arise in the rules of inference, not because they faithfully reflect the unpredictable character of the environment (as in case (i)) but for three other reasons: First, the uncertainty represented in the rules may reflect our uncertainty about the validity of the applicability of a rule generally or the applicability of a rule in a particular situation — that is, we are unable to produce an adequate theory. Second, we may need to produce systems that reason quickly. Although we may have an adequate theory, in order to produce practical systems, we may need to couch that theory in terms of more general (and less certain) rules of thumb in order to decrease the search space and so increase the speed of inference. Finally, it may be that we have a "good theory", but we do not have the appropriate expert system architecures or representational formalisms to implement it. These three reasons have often lead to expert systems operating with (uncertain) rules of thumb or heuristics.

If uncertainty does have to be introduced into the rules of inference, either because we can not produce an adequate theory, because it would be impractical to adequately implement the theory, or because the object world is intrinsically unpredictable, then there are alternatives to the use of the Bayesian method. Other methods include the use of "fuzzy" logic (Zadeh, 1983), Dempster-Shafer theory (Shafer, 1976) and certainty factors (Shortliffe & Buchanan, 1975). However, it is unclear whether these avoid the problems mentioned above. Also Fox (1984) has suggested that uncertainty should be represented (and presented) qualitatively with explicit semantics rather than as numbers representing probabilities. He suggests that this would make the systems operations more intelligible. He has produced a hierarchy of belief terms (e.g., "certain", "likely", "unlikely") based on a subset of the English vocabulary. In the inference process these can be combined according to a set of rules. He suggests that this will contribute to overcoming the problems of intelligibility, as there will be no need to interpret the meaning of a number. The meaning of the linguistic term is presumably regarded as transparent and more natural.

However it is unclear whether this system would address any of the other problems with Bayesian techniques mentioned above (e.g., computational demand, laborious development process, violation of the assumption of conditional independence). Indeed, it is questionable whether users, knowledge engineers or experts would produce common interpretations of the qualitative values of propositions such as "possible", "likely", "unlikely". Thus, it is questionable whether the problem of intelligibility would be solved by this technique.

The last three developments discussed here involve attempts at producing architectures, techniques and representational formalisms that should allow us to implement more complex and adequate theories.

(b) Blackboard Architectures

Traditional expert systems (including those that reason under uncertainty using numerical inferencing methods) usually have a common architecture: a single knowledge base and a single inference engine, although each of these may be very different in each implementation.

However, more recently, new architectures have been suggested. Probably the best known example is the blackboard architecture. In this sort of architecture more than one knowledge base is provided. The knowledge bases may differ in terms of the representational formalism used (e.g., logic or frames), or in the domain of information represented (knowledge about syntax or semantics) or possibly in the inference strategy that is used. Each knowledge base contributes to the problem solving process by presenting its hypotheses on a "blackboard", and by reading information off it.

This sort of architecture was developed in speech understanding where it was recognised that many sources of information were required to solve the problems (e.g., HEARSAY-II; see Erman & Lesser, 1980). In these sorts of system the knowledge bases differed mainly in terms of the "level" of information represented (e.g., at the "top" there was knowledge of semantics and syntax, at the bottom there was knowledge of phonetics). However knowledge bases could be distinguished by characteristics other than "levels". For example an expert system might contain three knowledge bases, one concerned with knowledge about the structure of the engine, another with the functions of its components and the third with knowledge about its electrical components.

With multiple sources of knowledge it may not be necessary to introduce uncertainty (and its associated problems) into the inference process. Instead it may be possible to solve the problem by providing several sets of simple but truth conditional rules. Each set on it's own may be unable to solve the problem, but, through co-operation with the other sets using the blackboard, the solution can be reached with certainty.

(i) Explanation and justification

These sorts of architecture may contribute to the production of more adequate explanations and justifications. Multiple lines or forms of reasoning about different aspects of the problem should provide more convincing justifications than a single line of reasoning. Also, when asking for explanations, users could specify which parts of the inference process they are interested in referring to the operations of the appropriate knowledge bases.

(ii) Learning and knowledge acquisition

Hayes-Roth (1984, 1985) has developed a domain independent blackboard system capable of "learning" when supported by human expert interactions. The human expert watches the system operate and overides any incorrect decisions. The system then questions the experts about the reasons for interventions and modifies the rules appropriately. This brief description of the system is based on a report by Boyle (1985) — unfortunately the author could not acquire the original reports and so cannot comment on the success of the system.

(c) Mixed Initiative Expert Systems

A mixed initiative expert system is a system which interacts with the human user for advice during the problem solving process and allows flexibility in terms of who can take the initiative (as opposed to only interacting before and after the problem solving process). There are three main reasons why mixed-intiative expert systems are required.

First, few experts work entirely alone in solving problems. They rely on colleagues to give them specialised advice or alternative perspectives on problems — especially when they become uncertain about their knowledge of a particular domain. To some extent the development of blackboard architectures reflects this fact. However, it is likely that for some time expert systems will not be able to work entirely autonomously (particulary in very complex domains), they are likely to need human help. Mixed-initiative systems should provide this facility.

Second, it is likely that if expert systems are to be professionally accepted, they will have to take greater account of the user. As Kidd (1985) points out, users approach tasks with their own intentions, expectations and constraints; these can significantly affect the choice or acceptance of an appropriate remedy. She also suggests that an important part of giving, and accepting, appropriate advice from an expert is the negotiation involved between the expert and consulant. It is likely that if we are to produce expert systems whose decisions are to be accepted by users in a wide variety of domains then we need to develop expert system architectures that allow for flexibility in terms of the ways users can interact with the systems.

Finally, as mentioned above, users sometimes do not provide appropriate information for the expert system to operate with. Therefore an expert system may need to be able to reconsult the user after an initial attempt at solving the problem.

A major problem, however, is that at present most expert systems are built using knowledge of how to do the job, not of how to be an assistant (Worden et al., 1986). It is expected that, over the next few years, greater emphasis will be placed on attempting to understand co-operative processes, and to build expert systems with architectures and knowledge that can support human-machine co-operation. Some work has already been done, for example, Worden et al., (1986) have been developing a "framework for building expert systems which can co-operate in something of the manner we would expect a human assistant to co-operate — accepting advice from the user when he knows best, not repeating mistakes unnecessarily, attempting to understand reasons behind user interventions and sometimes knowing when it is out of its depth" (p. 319).

The work of Kidd (1985) on what users want from an interaction with an expert system, and O'Malley, Draper & Riley (1985) on constructive co-operative interaction should be relevant to understanding some of the problems involved in developing mixed-initiative expert systems.

(d) Deep Representations

What is a deep representation? It does not seem possible to give a definitive description of what deep representations are. Indeed a recent Alvey workshop on the subject failed to provide a unanimous definition. The characteristics identified below were found in three important overview articles (Cohn, 1985; Bobrow, 1984; Brasden, 1985). However, these characetristics should not be considered as a definitive set, but as indicating some general properties of deep representation systems. Bobrow (1984) should be consulted for a fuller review of the characteristics of deep representation techniques. (This edition of the journal is a special issue on deep representation techniques and systems, and reviews of some of the major problems and developments in deep representation.)

Cohn (1985) defines deep representation in contra-distinction to traditional expert systems representations, which he suggests have "shallow knowledge[11]. They have shallow knowledge because they contain rules which are heuristic rules of thumb. There is often no notion of causality represented in the rules, for example, they may simply associate a set of observed symptoms with a diagnosis. On the other hand, deep representation techniques usually have some model or models of the problem domain. These models are often based on causal and/or functional descriptions. It is important to point out that there is not one deep model of a domain but usually several. These models will differ in terms of relative "depth" and in terms of the types of deep descriptions (structural, functional and/or causal). For example, a deep representation of an engine might contain deep structural models (this might specify engine capacities, stresses, dimensions, etc.) Another model might be a functional one (specifying the function of each component). It may also require a dynamic model of the engine (for example, this could produce desciptions of the engine in various running states).

Why do we need deep representation? It is not suggested here that all expert systems should be based on deep representations. However, Cohn (1985) and Brasden (1985) both suggest that the some of the consequences of the shallow/associationist representations of knowledge are that they are less flexible (they are restricted to operating within a very narrow domain), they not provide useful explanations, and they often require the system to reason under uncertainty (along with its associated difficulties).

The reason why deep representation based systems are expected to be more flexible is that they should reason from general, "first principles". These should be applicable to other domains which have similar characteristics. For instance, Brasden (1985) suggests that "the knowledge held is general, rather than specific to one purpose. The more general, the deeper the knowledge" (p. 8). Cohn (1985) also suggests that the lack of understanding of shallow models "tends to limit the range and nature of the problems the system is able to solve" (p. 300).

One reason why these systems are expected to give more adequate explanations is that, rather than listing the set of symptoms and the resulting diagnosis, deep representation based systems can generate descriptions from the underlying causal or functional model. It is suggested that this will be more convincing and understandable.

Brasden (1985) also seems to suggest that the use of probabilistic inference techniques (and the problems associated with them) has arisen from our lack of deep representations, and that one crucial feature of deep representation systems is that they are considered to be less probabilistic "The deeper the knowledge, the more precisely we can define the relationship between items, and the less we need deal in probabilities" (Basden, 1985, p. 8). It seems that this statement might apply to many areas of expert system applications (those where we fully understand the mechanisms involved), but not all of them. For example, quantum mechanics could be considered as a deep model of all physical processes. However probabilities would need to be involved in representation of the behaviour of the components because they are intrinsically indeterminate.

There are several existing systems which use "deep" representation techniques. For example, ABEL (Patil, 1981): this is an expert consultant for acid-base imbalances; CADUCEUS (Pople, 1982): this diagnoses internal medical problems; CASNET (Weiss, 1978): this system diagnoses eye diseases. It should be pointed out that expert systems with deep representations or models, may also contain more shallow or traditional components. For example, Gallanti, et al., (1986) describe a diagnostic expert system that will have both deep and shallow components. It is expected that the shallow heuristic knowledge will make the system more efficient.

However there are many problems in attempting to build and use practical deep modelling systems. Some of these problems are reviewed here as well as developments that might overcome them.

(i) Problems with deep representation techniques

Cohn (1985) outlines four main problems involved in producing and using expert systems that use deep representations.

The first problem concerns conceptualisation. This is the process of analysing the problem domain and developing a means of describing it — for instance, what are the basic properties involved, the relationship between objects etc. For some domains, the appropriate desciptions already exist (e.g., descriptions of electrical ciruitry, simple mechnical devices and medicine), these are the areas that are most likely to be affected by developments in deep representation. However, in other areas it is far more difficult to identify the conceptual structure of the domain. A particularly difficult problem is the production of deep representations for time and space. Much effort is presently being placed on developing adequate representations of these notions. (For example, in 1986, the section on knowledge representation of the proceedings of AAAI (Vol. 1) contained 19 papers. Nearly half of them concerned problems with representing time.) Cohen suggests that conceptualisation in general is going to be the hardest problem in deep representation. "The only work that really begins to address this problem ... are the concept discovery systems of Lenat (Davis & Lenat, 1982; Lenat, 1982)" (p. 300). However, it is unlikely that these sorts of systems will be of any practical use in the next ten years, and at present it does not seem that there are any rigorous, non-computational methods of knowledge elicitation/acquisition that are well suited to the domain of deep knowledge. A review of current knowledge acquisition techniques for deep representation medical expert systems can be found in Gotts (1984).

The second problem involves trying to ensure "high fidelity axiomatisation". A high fidelity of axiomatisation might be measured by the closeness of its simplest model to the intended interpretation. The problem is that a given formalisation may have a much simpler model than that intended, and it is often difficult to identify that simpler model.

The third and related problem is that of ensuring a consistent knowledge base. Cohn suggests that it can be a non-trivial problem to determine whether a given set of axioms has a model at all, "The problem is especially hard since any system with the expressive power of first order predicate calculus (and surely we need at least this amount of expressiveness) is only semi-decidable; there is no decision procedure to compute whether an axiomatisation is satifiable (p. 301).

Cohn suggests that systems such as TEIRESIAS (Davis, 1982) are likely to be useful for this sort of problem. TEIRESIAS was built to help with the maintenance of large rule bases by using notions such as rule models and type checking to check rules for sensibleness. Also, systems such as Cunningham's model finding program (Cunningham, 1985) are likely to be very useful for these sorts of problems.

The final problem Cohn mentions concerns the use of systems based on deep representations. The use of such representations in complex domains is likely to produce massive search spaces because, if expert systems are going to reason from first principles, the inference chains will be very long. Searching these large search spaces will be very expensive.

In the next section there is a discussion of sorted logics as a representation language for deep reprsentation systems and how they could contribute to the solution of some of the problems mentioned above.

(ii) Many sorted logics

What are many sorted logics? In many sorted logic the items in the object world are divided into different sorts (e.g., objects, relationships or types of objects and relationships). The sorts of the arguments of all the non-logical symbols in t?rhe language and the sorts of the results of function symbols are specified. For example we could provide the sort "integer" and produce axioms involving numerical operators that only took integers as their arguments and only returned integers as results. To this extent, sorted logics are very similar to typed programming languages (such as PASCAL), and they share their problems too.

Several mechanised many sorted logics have been proposed or built (e.g., some of the more recent work Cohn mentions includes Reiter, 1981; Walther, 1983). There are several problems with these logics. Cohn suggests that Walther's logic is the only one with a sound theoretical foundation. Also, a major problem is that that many sorted logics have a restricted expressive power compared to unsorted logics. Cohn recommends LLAMA (Cohn, 1983a, 1983b) as being particulary expressive.

What are the advantages of many sorted logics as a representation language? The major advantage is that sortal logics can reduce the search space. Cohn argues that there are two major reasons for this; (a) no further inference will be done on ill-sorted formulae (e.g., where the sort of the argument of a predicate and the sort of an input are different), (b) axiomatisation will be far smaller than with non-sorted logics, either because fewer axioms will be required, or because the axioms themselves are smaller. There may be fewer axioms because the axioms that would normally be needed to explicitly express sortal information can be absorbed into the sortal notation and the inference machinery. Taking the example given above about numerical operators: in a non-sortal logic it would have been necessary to provide an additional axiom representing the rule that numerical operators only take integers as their arguments. Similarly, the axioms will be far smaller because sortal preconditions do not have to be explicitly stated — this also makes it a convenient notational tool.

Cohn also suggests that they will help with the fidelity problem discussed above. This is because the structure it imposes on the axiomatisation and restrictions imposed on the interpretations of the non-logical symbols, reduces the number of possible models.

References

Boyle, C.D.B. (1985). Acquisition of control and domain knowledge by watching in a blackboard environment. In M. Merry (Ed.) *Expert Systems '85: Proceedings of* **the** *Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems* C.U.P: Cambridge.

Bowbrow, D.G. (1984). Qualitative reasoning about physical systems: An introduction. *Artificial Intelligence, 24,* 1-5.

Bradley, R. & Swatz, S. (1979). *Possible Worlds: An Introduction to logic and its philosophy.* Basil Blackwell: Oxford.

Brasden. A. (1985). What is deep knowledge? In *Proceedings of the ALVEY Deep Knowledge Workshop, No.].* University of Sussex. 10-12 July 1985.

Cohen, P.R. (1986). Numeric and symbolic reasoning in expert systems. *Proceedings of ECAI '86, Vol. I,* 413-426.

Cohn, A. (1983a). Mechanising a particularly expressive many sorted logic. Ph.D. Thesis, University of Essex.

Cohn, A. (1983b). Improving the expressiveness of many sorted logic. *Proceedings of AAAI '83.*

Cohn, A. (1985). Deep knowledge representation techniques. In M. Merry (Ed.) *Expert Systems* '55: *Proceedings of the Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems* C.U.P: Cambridge.

Cunningham, J.C. (1985). Comprehension by model-building as a basis for an expert system. In M. Merry (Ed.) *Expert Systems '85: Proceedings of the Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems* C.U.P: Cambridge, sp

Davis, R. (1982). Expert Systems: where are we, and where do we go from here? AI Memo No. *665,* MIT: Mass.

Davis. R. & Lenat, D. (1982). *Knowledge Based Systems in Artificial Intelligence.* McGraw-Hill.

Doyle, J. (1979). A truth maintenance system. *Artificial Intelligence, 12,* 231-272.

Duda. R.O.. Gaschnig, J.. Hart, P.E., Konolige, K.. Reboh, R., Barrett, P. & Slocum, J. (1978). Development of the PROSPECTOR consultant system for mineral exploration. Final Report, SRI Projects 5821 and 6415, SRI International, Inc., Menlo Park, Calif.

Erman, L.D. & Lesser V.R. (1980). The HEARSAY-II speech understanding system: A tutorial. In W. **Lea** (Ed.) *Trends in Speech Recognition* 361-381. Prentice Hall. Englewood Cliffs: N.J.

Fox, J. (1984). Language, logic and uncertainty. Imperial Cancer Research Fund Laboratories Report.

Gabbay, D.M. (1985). Intuitionistic basis for non-monotonic logic. *Artificial Intelligence, 25,* 75-94.

Gallanti, M., Gilardoni, L. Guida, G. & Stefanini, A. (1986). Exploiting physical and design knowledge in the diagnosis of complex industrial systems. *Proceedings of ECAI '86, Vol. 1,* 335-349.

Gotts, N.M. (1984). Knowledge acquisition for medical expert systems — A review. AI in Medicine Group Report, AIMG-3 Sussex University.

Hayes-Roth, B. (1984). BBI: An architecture for blackboard systems that control, explain and learn about their own behaviour. HPP-84-16. Stanford.

Hayes-Roth, B. & Hewett, M. (1983). Learning control heuristics in BBI. HPP-85-2. Stanford.

Hewett, (1986). Commercial expert systems in North America. *Proceedings of ECAI '86, Vol.2.*

Hughes, G.E., Cresswell, M.J. (1972). *An Introduction to Modal Logic.* Methuen: London.

Kidd, A.L. (1985). What do Users Ask? — Some Thoughts on Diagnostic Advice. In M. Merry (Ed.) *Expert Systems '85: Proceedings of the Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems* C.U.P: Cambridge.

Lenat, D.B. (1982). Heuristics: Theoretical and experimental study of heuristic rules. *Proceedings of AAAI '82.*

Mamdani, A., Efstahiou, J. & Pang, D. (1985). Inference under uncertainty. In M. Merry (Ed.) *Expert Systems '85: Proceedings of the Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems* C.U.P: Cambridge.

McDermott, D. & Doyle, J. (1982). Non-monotonic logic I. *Artificial Intelligence, 13,* 41-72.

Merry, M. (1985). Expert systems — some problems and opportunities. In M. Merry (Ed.) *Expert Systems '85: Proceedings of the Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems* C.U.P: Cambridge.

O'Malley, C.E. Draper, S.W. & Riley, M.S. (1985). Constructive interaction: A method for studying human-computer-human interaction. In B. Skackel (Ed.) *Human-Computer Interaction — Interact '84.* North-Holland: Amsterdam.

Patil, R.S. (1981). Causal representation of patient illness for electrolyte and acid-base diagnosis. PhD. Thesis, Laboratory for Computer Science MIT: Mass.

Pople, Jr. H.E., (1982). Heuristics methods for imposing structure on ill-structured problems: The structuring of medical diagnostics. In P. Szolovits (Ed.) *Artificial Intelligence and Medicine. AAAS Selected Symposium Series 51.* Westview Press: Bolder, Colarado.

Reichelt, H. & van Harmelen, F. (1985). Relevant criteria for choosing an inference engine in expert system. In M. Merry (Ed.) *Expert Systems '85: Proceedings of* *the Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems* C.U.P: Cambridge.

Reiter, R. (1981). On the integrity of typed first order data bases. In H. Gallaire, J. Minker & J.M. Nicholas (Eds.) *Advances in Data Base Theory, Vol. L* Plenum Press.

Shafer. G. (1976). *Mathematical Theory of Evidence.* Princeton University Press. Princeton.

Shortliffe, E. H. (1976). *Computer-Based Medical Consultations: MYCIN.* American-Elsevier: New York.

Shortliffe. E.H. & Buchanan, B.G. (1978). A model of inexact reasoning in medicine. *Mathematical Biosciences, 23,* 351-379.

Walther, C. (1983). A many sorted calculus based on resolution and paramodulation. *Proceedings of IJCAI '83, VoL 8.*

Weiss, S.M., Kulikowski, C.A., Amaral. S. & Safir, A. (1978). A model-based method for computer aided medical decision making. *Artificial Intelligence, 11,* 145-178.

Worden, R.P., Foote. M.F., Knight, J. & Anderson, S.K. (1986). Co-operative expert systems. *Proceedings of ECAI '86, Vol. 1,* 357-368.

Zadeh, L.A. (1983). The role of fuzzy logic in the management of uncertainty in expert systems. *Fuzzy Sets and Systems, 11,* 199-227.

2.2. Natural Language Processing

Natural language processing (NLP) involves computer generation and understanding of textual, natural language inputs.

2-2*1. Applications

Three main **areas** for NLP applications **are** considered here: in machine translation, text summarisation and as a medium for human-computer communication.

(a) Machine Translation

It is not yet possible to provide general, perfect machine translation packages. However, in some domains high, but not perfect, translation performance may be quite acceptable. For instance, in machine translation there is a demand for bulk translation or translation, between "odd" combinations of languages, where there is a skill shortage (e.g., between Dutch and Greek). Commercially viable machine translation systems exist (e.g., LITRAS [German-English, English-German] based on METAL; Slocum. 1986). While they do not produce perfect translations, they can produce massive increases in productivity especially if pre- and/or post-processing is made available. For example, provisional post-editors (i.e., people who were not involved in the development of the system) working on the output of METAL were able to revise 29 pages a day. This contrasts with the average daily output of a translator, which is around 5 to 8 pages a day. See King (1986) for a review of some of the current work (including a report of progress in the EUROTRAN project, the largest machine translation project at present), and for a general discussion of the prospects of machine translation. His main conclusions are that machine translation will not entirely replace human translation, and that "there is no prospect of them getting drastically better". However, they do offer massive improvements in productivity, especially when dealing with "every-day", non-specialised material and where pre- and post-processing is introduced. A possible application of machine translation techniques is the provision of an automatic translation service for messages passed by electronic mail between different countries.

(b) **Text Summarisation**

Another area for the application of NLP techniques is in text summarisation. Schank (1985) reports on a program called FRUMP which summarised unedited news items and could translate them into other languages (given that it used Schank's "language-independent internal meaning representations"). Target languages included Chinese and French. However, this system was never commercially available.

(c) **Human-Computer Communication**

The most common form of explicit human-human communication is via natural language. At present the dominant form of communication with computers is via programming languages. Most of the general public who do not program might find it a laborious means of communication or difficult to learn. It is hoped that research into NLP will provide interfaces that will change this, so that much of human-computer interaction can proceed via natural language.

However, it must be recognised that there **are** other ways of making computers more generally accessible, and that there are circumstances where natural language communication is inappropriate. Ways of making computers more generally accessible include providing languages and devices that are more natural to use and easier to learn. The development of high level programming languages, direct manipulation inerfaces , high resolution graphics, bit map screens and pointing devices have, and will, make the user's job much easier. An important question will be whether these sorts of techniques will provide a means of communication with computers which are convenient and easy enough to use, so that further progress in NLP will become redundant.

There **are** many tasks where natural language communication with the computer is inappropriate. In choosing a high level programming language for a task, it is recognised that different languages **are** suited to different tasks, they can differ considerably in terms of efficiency, expressive?n power, or problem orientation; natural language is no exception to this. Natural language tends to be ambiguous and imprecise. Therefore, in tasks where precision is important, such as the specification of an algorithm, other languages or forms of communication (e.g., high level programming languages or pointing devices) may be more appropriate.

Two areas of human-computer interaction are considered here: The first concerns explanation systems; these explain the operations and the output of computational processes. The second concern interfaces to large informations sources, such as databases.

(i) Explanation

In the field of expert systems it is likely that if we are to produce generally acceptable systems then we will have to improve their explanatory capabilities. At present expert systems rely on "direct production" explanation which often means all that is provided is an execution trace; these can become incomprehensible if it is too long. There is now a growing interest in the application of NLP techniques to language generation in explanations. For example there is an ALVEY project at Sussex managed by Dr. Mellish, investigating "Natural Language Generation From Plans'* — project number 010; the uncle is M. Cooper of BT Research.

(ii) **Data-base** front ends

The research on NLP of the last two decades is now beginning to show fruition in the development of commercially available tools for the development of NLP interfaces to knowledge bases. It is expected that this trend will continue and that we will see increasing numbers of domain-independent tools for constructing front-ends (e.g., the SRI TEAM system (Martin et. al. 1983) and the INTELLECT system), although, once constructed, the front ends are likely to be domain dependent (see below). Most of these do, and will, make use of technology that has been available for several years, and which can be found in Woods' LUNAR program (Woods, 1973). However ATNs, although flexible, are considered to be cumbersome, since they are a low-level notation requiring inelegant detail, and are not considered as robust as certain other techniques e.g., Kaplan's chart parser (Kaplan, 1973). (See [1] for a definition of an ATN.)

2.2.2. **Exisiting Problems and Future Developments**

A major problem with all commercially available techniques, including ATNs and chart parsers, is that there are still many cases of ambiguity which can not be resolved.

(a) Semantic Parsers

One way of overcoming some of the problems of ambiguity is to build systems that make use of semantic processing, e.g., systems that use rules which specify simple semantic restrictions about what kinds of object can participate in what kinds of relationships. Semantic parsers (e.g., Schank et al. 1980, Dahlgren & McDowell, 1986) are considered to be more robust than pure syntactic parsers. However, they tend to be suited to processing simple declarative sentences. There have been some successful attempts to build tools and techniques to improve their performance (e.g., Binot & Ribbens, 1986; Lytinen, 1986) but, at present, these techniques only work adequately in narrow domains, and hence NLP front ends to databases that are built using these techniques are likely to remain effective only for narrowly restricted domains.

(b) Models of Discourse

Another related problem is that current NLP systems are restricted by their models of context. This is because they are all based on sentence-by-sentence processing. This also makes some aspects of disambiguation difficult, for example, the disambiguation of ellipsis (e.g., incomplete text or statements) and pronouns (e.g., the referent of "his" in the sentence "He took his life" — Did he take his own life or someone else's?). Disambiguation of these features is generally associated with processes operating during extended discourse, which means that it is not possible to adequately process these features through sentence-by-sentence processing.

It is expected that work on the structure of discourse and the pragmatics of communication will make a contribution to overcoming these problems. The work done (e.g., Grosz, 1977) on the structure of discourse and (Cohen, 1978; Allen, 1983; Litman, 1986) on language as goal directed behaviour suggests that the deployment of relatively modest computational resources could overcome some of the problems mentioned above. However, it would seem that we are unlikely to see practical NLP systems employing these techniques for the next 5-10 years.

(c) Language Acquisition

There seems to be a growth in interest in language acquisition systems, i.e. systems that can substantially improve their linguistic abilities by learning new words and grammatical rules. As in other areas of AI applications, one reason for this interest is the recognition that adequate NLP systems will be knowledge rich and that an alternative to "hand coding" these systems is to provide them with facilities to augment their own knowledge bases. There has always remained some interest in the development of algorithms for learning syntax since Chomsky's theory of LAD (Language Acquisition Device). For example, see Wolff, (1980) Berwick, (1980), and Anderson, (1977), for description of systems that learn or augment grammars. However, there is now a growing interest in the production of a broader framework for understanding language acquisition in humans. This includes analyses of the development of general purposeful and communicative skills (e.g., speech acts). As mentioned above, current NLP systems are restricted by the lack of these sorts of "skills". Shatz (1983) provides an overview of the work being done in this field (mainly by developmental psychologists and linguists). Although it seems unlikely that we will be able to produce complex language learning devices in the foreseeable future, this research may provide insight into the constituent knowledge structures and processes required in order to be a practical language user (and understander) — in particular it may tell us more about the domain of discourse and goal structures.

(d) Parallelism

Any progress in introducing limited parallelism to NLP systems is likely to contribute in a quantitative manner (as in most other areas of computation) by producing faster and more efficient parsers over the next ten years. However the "connectionist" work is already influencing NLP (though less apparently than in vision or learning). For instance there is now a new connectionist technique called "simulated annealing" which is concerned with NLP. The author knows of no publications, but there is a project funded by the Joint Speech Research Unit at the University of Leeds — supervised by Prof. Geoffrey Sampson of the Linguistics and Phonetics department, and Eric Atwell of the Computer Science department. It is unclear what the properties of such a parser would be (beyond those of the connectionist machine — see Section 3.1), or the likelihood of producing working models. However, given the state of the art of connectionism. it is unlikely that we will see any radical changes brought about by macro-parallelism in practical NLP systems in the next ten years.

Notes

[1] **ATNs** (augmented transition networks) are grammar representations for natural language. They are an extension of finite state transition diagrams. Finite state transition diagrams are transition networks representing the possible rewrites of linguistic items (e.g., NP -> NP + VP). Augmented transition networks are similar but they include additional conditions and side effects operating on the arcs. These additions resulted in the power needed for handling features of language, like embedding and agreement, that could not be conveniently captured by context free grammars. ATNs can be viewed as either a grammar formalism or a parser (definition from Barr & Feigenbaum, 1983).

References

Allen, J. (1983). Recognising intentions from natural language utterances. In M. Brady & R. Berwick (Eds.) *Computational Models of Discourse,* MIT: Mass.

Barr, A. & Feigenbaum, E.A. (1983). *The Handbook Of Artificial Intelligence. Vol. 1.* Pitman: London.

Binot, J.L. & Ribbens, D. (1986). Dual frames: A new tool for semantic parsing. *Proceedings of AAAI '86,* 579-583.

Cohen, P. (1978). On knowing what to say: Planning speech acts. Technical Report 118, Dept. Comp. Sci. Univ. Toronto.

Grosz. B. (1977). The representation and use of focus in dialogue understanding. Technical Note 151, SRI International.

Kaplan, R.M. (1973). A general syntactic processor. In R. Rustin (Ed.) *Natural Language Processing.* Algorithmic Press: New York.

King. M. (1986). The prospect of machine translation. *Proceedings ECAI '86, Vol. 1.*

Litman, D. J. (1986). Understanding plan ellipsis. *Proceedings of AAAI '86,* 619-624.

Lytinen, S.L. (1986). Dynamically combining syntax and semantics in natural language processing. *Proceedings of AAAI '86,* 575-578.

Martin, P., Appelt, D. & Periera, F. (1983). Transportability and generality in a natural language interface system. *Proceedings of IJCAI '83.*

Schank, R. (1985). Intelligent advisory systems. In P. Winston. & K. Prendergrast (Eds.) *The AI Business: Commercial Uses of Artifial Intelligence.* MIT: Mass.

Schank. R. Lebowitz, M. & Birnbaum. L. (1980). An integrated understander. *AJCL, 6,(1).*

Shatz, M. (1983). Communication. In J.H. Flavel, & E.M. Markman (Eds.) *Handbook of Child Psychology, Vol. 3: Cognitive Development.* Wiley: New York.

Slocum. J. (1986). METAL: The LRC machine translation system. In M. King (Ed.) *Machine Translation Today — the State of the Art.* Edinburgh University Press: Edinburgh.

Woods. W.A. (1973). Progress in natural language understanding: An application to lunar geometry. *Proceedings of AFIPS Conference.*

2.3. Vision

Barr & Feigenbaum (1983) define vision processing as the task of understanding a scene from its projected images. However, in the field of vision research, a distinction is often made between signal processing and classification on the one hand and image understanding on the other. Signal processors transform an input image into another image that has desirable properties. For example, a signal processor may produce a better signal-to-noise ratio to facilitate human or machine inspection. Classification techniques classify images into predefined categories often by statistically based pattern matching techniques. These techniques are primarily concerned with the processing of two dimensional images. They will be refered to as "image processing techniques" in this section. The other form of vision processing, image understanding, involves a process that builds a description not only of the image itself but also of the scene it depicts. Image understanding, as Barr & Feigenbaum (1983) point out, is usually considered to require, in addition to image processing techniques, highly structured knowledge about the task world.

2.3.1. Applications

Image processing research has already had some commercial impact. For example, in the field of optical character recognition, remote sensing and medical image analysis. However there are many areas appropriate for image understanding applications once some of the existing theoretical and technical problems have been resolved.

(a) Human-Human Communication

At present telecommunications mostly proceed via textual and oral means. As with local human-machine interaction, human-human telecommunication might be enhanced by the provision of visual information. However, a major problem will be parsimoniously coding and sending this information. Image understanding techniques such as model based techniques (see below for more on model based techniques) might make the coding process more parsimonious. For instance, only visual information which is changing over time need be coded, as this is the source of any new information. Principles from AI image understanding vision research (e.g., on the perception of motion) are likely to be useful here.

(b) Data Translation and Manipulation

The trend towards digital storage of information throughout many areas of industry may necessitate the production of complex vision systems (and other knowledge based techniques) in order to transfer the vast stores of existing diagrammatic information into machine readable form. For example, the UK department of Ordnance Survey is now producing digital representations of maps for computer storage (and later computer processing). At present, this needs to be done "by hand"; in the future, image processing vision systems could be developed to perform the task much more cheaply, and possibly more accurately.

Principles or representation techniques from AI image understanding research might be used in areas such as computer aided design (CAD). In CAD it is often necessary to generate and manipulate representations of three dimensional objects. Representational formalisms which have been developed in AI might contribute to these sorts of systems. For example, generalised cylinders are a form of representation developed in vision research. It provides a useful means of representing complex objects by part/whole segmentation.

Other areas for the application of AI image understanding research include the use of vision in providing robots (e.g., autonomously guided vehicles) with vision sensing, this is expected to increase their versatility and flexibility. It is likely that AI vision research will be necessary in this domain in order to provide three-dimensional information. However, there are many problems yet to be resolved before general purpose AI vision systems or practical vision packages can be produced, although there are many potential applications.

2.3.2. Existing problems and Future Developments

In order to understand the direction vision in which research is going, and what are the possibilities and problems, it will be helpful to examine the history of vision research.

Vision research in the 60's and 70's can be described under two main headings which reflect two alternative strategies for vision processing: the high level "blocks world" and "low level" vision processing research.

In low level research there was an attempt to identify methods for extracting the "important" features or changes in an image. For example, there was an attempt to build line and region finding algorithms. It was hoped that this low level information could be combined to provide higher level information. However, as Brady (1981) points out, by the end of the seventies the consensus was that low level vision alone was incapable of providing useful, rich descriptions. As in speech (e.g., HEARSAY II; Erman et. al., 1976) and natural language processing (e.g., SHRDLU; Winograd, 1972), it was realised that there was a need for top down information flow to provide the necessary additional constraints.

The high level vision processing research consisted of attempts to overcome the combinatorial explosion and the problems of low level vision by identifying high level constraints. A good example of this is the work on line labelling (e.g., Clowes, 1971; Waltz, 1975). This involved methodologically working out all the ways planes could meet in space, all possible appearances of these junctions and developing a notation for representing these configurations. Recognising an object was then simply a matter of finding a consistent labeling for a line drawing of the object.

More recently AI vision research has split, not into groups reflecting competing strategies (top down vs bottom up), but into groups exploring and attempting to define the computational properties of different co-operating modules. Some of these modules parallel those identifiable in the human visual system (identifiable through psychological experiments, studies of visual illusions and brain damaged patients etc.) e.g., stereopsis, interpretation of surface contour, determination of surface orientation from texture, motion and depth. Most of this work has been inspired by the work of Marr and is concerned with elaborating his theory of vision processing (see Marr, 1978 for a description of some of the theory).

Brady (1981) points out that the major areas of success and consensus involve those modules that are at the "bottom" of the hierarchy in vision processing — those that deal with the image as opposed to subsequent representations of it. Brady (personal communication — seminar at Sussex) has recently suggested that that the bottom-up data driven approach is far from exhausted, and that he didn't forsee the need for knowledge-based systems in basic research for quite some time, until the bottom-up approach had been taken much further. It is expected that considerable progress will continue to be made in this area. This will be supported by advances in VLSI design and progress in the commercial availability of array processors. This may mean that in the medium term, say within five years, low level vision packages will become commercially available. These should embody the necessary balance between efficiency and flexibility.

Brady points out that as we move up the vision processing hierarchy there is less consensus and many more problems to be solved — these are reviewed in Brady, (1981). The problems include those of identifying the details of the "surface orientation map", determination of surface properties, the question of colour vision, motion perception, and the perception or planning of movements through cluttered space. He also points to the possible influence of micro and macro-parallelism on vision research and suggests "It is likely that our conception of compututation will change as a result of such developments. Vision will be one of the first areas to benefit from such advances" (Brady, 1981, p. 11).

The resolution of some or all of these problems may be a necessary prerequisite for providing general vision processing packages as these sorts of information (depth, texture, colour, shading and motion) will aid greatly in the disambiguation of the scene. The rest of the volume describes some of the more recent attempts in dealing with these problems.

However there are two other (not mutually exclusive) approaches to developing better vision systems. The first involves the use of more high level, situation specific information or models in disambiguation. The second approach involves the use of multiple sources of information and/or the simplification of the environment to reduce the need for disambiguation.

(a) High Level Model Based Techniques

There is a growing interest in the potential role of situation specific, model based representation of objects that working vision systems might encounter. Detailed information (e.g., abstract geometric models) about classes of objects the system is likley to encounter can help improve the performance of vision systems. Brookes (1981) reports on a system which uses "high" level object representations in order to identify aircraft in an airport.

(b) Multiple Sources of Information and Changing the Environment

It may be possible to avoid the problem of producing very complex, general vision systems by providing simpler systems with additional sources of information. For instance, information from vision processing could be used in conjunction with other sources of non-visual information e.g., acoustic, temperature etc. The combination of various sources of information is likely to require the development of more sophisticated architectures, representations and control strategies such as the those being used in blackboard architectures in expert systems (see section 2.1.). Another way of overcoming the problems of producing complex vision systems is to alter or restrict the range of visual information which the system needs to process. For instance by guaranteeing that only a certain set of objects enter into the scene or by providing additional clues in the scene such a sharp contrast or shadows. It seem that practical vision systems requiring more than image processing will have to be supplemented by these sorts of information.

This section draws heavily on the work of Brady, (1981; 1983). The first reference is an introduction to a special issue of the AI Journal on vision. Although the article is now five years old, most of the issues raised are still important today. The second reference includes an annotated bibliography on more recent vision research, concentrating in particular on image processing hardware, parallel image understanding algorithms (including connectionist theories), and robotic vision.

References

Brady, M. (1983). Parallelism in vision. *Artificial Intelligence, 21*, 271-283.

Brady, M. (1981). Preface — the changing shape of computer vision. *Artificial Intelligence, 17*, 1-15.

Brookes, R.A. (1981). Symbolic reasoning among 3-D models and 2-D images. *Artificial Intelligence, 17*, 274-282.

Clowes, M.B. (1971). On seeing things. *Artificial Intelligence, 2*, 79-116.

Cohen, P.R. & Feigenbaum, E.A. (1983). XIII Vision. In *The Handbook Of Artificial Intelligence, Vol. 3*. Pitman: London.

Erman, L.D., Hayes-Roth, F., Lesser, V.R. and Reddy, D.R. (1976). The HEARSAY-II speech understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys, 12, (2)*, 213-253.

Marr, D. (1978). Representing visual information. In A.R. Hanson & E.M. Riseman (Eds.) *Computer Vision Systems*. Academic Press: New York.

Waltz, D. (1975). Generating semantic descriptions from drawings of scenes with shadows. In P. Winston (Ed.) *The Psychology of Computer Vision*. McGraw-Hill: New York.

Winograd, T. (1972). *Understanding Natural Language*. Academic Press: New York.

2A. **Speech** Processing

Speech processing involves computer production and understanding of speech (either single words or connected speech). The input for such a system would be human speech (perhaps recorded and perhaps with additional background noise), the output could be some representation speech, for example a piece of text. However, the output could include more than simply a list of words; it may contain information about accent* stress, intonation etc.

2A.I. **Applications**

Why do we need speech processing systems? The reasons are similar to those for NLP (see Section 2.2), and it seems that further progress in each field will be related for two reasons. First, sophisticated speech processing systems are of little use unless we can process their output (the input to NLP systems). The second reason is that knowledge from higher levels (e.g., grammatical or pragmatic knowledge) may need to be used to solve ambiguities at lower levels and vice versa (e.g., intonation can contribute to clarifying grammatical ambiguity).

Therefore, the results of speech processing could be used to improve the performance of NLP systems. However, it is unlikely that we shall want to take on the additional tasks of producing speech processing systems simply to solve the remaining problems of NLP. Also, it will be inappropriate to give priority to speech in HCI in all circumstances, the circumstances under which would be an inaapropriate form of human-machine communication are likley to be the same as those for NLP (see Section 2.2.).

However, there are circumstances where HCI should proceed, at least in part. via speech. We can talk much quicker than we can type; also, if programmers/operators can communicate with computers through speech then their hands are left free to perform other functions. In some circumstances it will be inconvenient for the operator to use their hands for communication, e.g., in flying an aircraft. Speech contains additional information that could be important in some systems (e.g.. intonation indicating reference or urgency, both in input and in output). Finally, there are obviously applications in the area of telecommunications, where users can communicate with remote information sources (such as databases) via the telephone. This could be done without the need for extra equipment at the user's end.

2*4.2. Existing Problems **and** Future Developments

The ARPA-SUR (Advanced Research Projects Agency — Speech Understanding Research) initiative aimed to achieve systems for connected-speech understanding capabilites (as opposed to systems which recognised isolated words). This initiative produced five major speech understanding systems in the 70*s; HEARSAY-I, n, (see Erman et al..l980), HARPY (see Lowerre and Reddy. 1980), WHIM (Wolf and Woods, 1980) and SRI/SDC (Walker, 1978). This initiative and the projects are reviewed in Barr and Feigenbaum (1983).

The major problem with these systems is that they suffer from an aspect of the "micro-worlds[11] problem (Dreyfus. 1982). in that most of them work adequately only in "perfect" environments. When an extended vocabulary, background noise or voice variation (e.g accent, intonation) is introduced, the performance of these systems decreases significantly. These speech systems could operate within less than perfect environments but they all seemed to reflect certain performance constraints, for example even in word recognition "Sizable vocabularies (more than a hundred words) can be realistically utilized with speaker-dependent templates. Smaller vocabularies (on the order of one or two dozen words) can be reliably utilized in talker-independent systems" (Flanagan et al., 1980, p.442, cited in Barr & Fiegenbaum, 1983).

It is expected that progress in the development of VLSI and array processors will have a quantitative effect on low level speech processing (see Section 3.1). However the major problems remaining are not ones which can be solved by developments in the speed of hardware alone; the major problem will concern knowledge representation, organisation and acquisition.

Generally, speech processing has has contributed to, general AI techniques — for example, island driving searching and blackboard architectures were developed in speech processing. Because of the nature of speech processing, success in this area is likely to contribute to the development of techniques for other systems that depend on co-operative decision making using multiple expert systems. For example, tasks requiring the coordination of several co-operating robots. This is because it is now recognised that significant progress will probably not be made until a wider range of knowledge is incorporated [1] Other forms of knowledge can be used to restrict the search task by using expectancies. For example, in HEARSAY-I, semantic knowledge (the rules of chess) and syntactic knowledge (natural language grammar) were used to disambiguate speech. Therefore, speech processing systems will probably consist of richly interconnected collections of knowledge based systems, utilising many types and levels of knowledge. Some of the ideas found in Sloman (1985) and Sloman et al. (1978) are relevant to this sort of task. Current work on developing more adequate blackboard systems (e.g., Boyle, 1985) and other architectures for dealing with multiple sources of representation and reasoning (e.g., Kauffman & Grumbach, 1986) will contribute to this task.

It is not expected that levels of human speech recognition competence will be reached within the next ten years. The major problems for producing practical systems will involve overcoming the "micro-worlds" problems of restrictions on vocabularies, speech types and speed. However production of speech is likely to come close to human performance over this period.

Notes

[1] The assertion that future success in speech processing will depend our ability to identify, represent and integrate multiple sources of knowledge has recently been challenged, and should not be regarded as uncontentious. For example Durham (1986) suggests "Current thinking is that much information resides in the speech signal itself, if only one could see how to use it" (p. 26).

References

Barr, A. & Feigenbaum, E.A. (1983). *The Handbook of Artificial Intelligence Vol* 2. Pitman: London.

Boyle. C.D.B. (1985). Acquisition of control and domain knowledge by watching in a blackboard environment. In M. Merry (Ed.) *Expert Systems '85: Proceedings of the Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems.* C.U.P: Cambridge.

Dreyfus, H. (1982). From micro-worlds to knowledge representation. In J. Haugland (Ed.) *Mind Design.* MIT: Massac.

Durham, T. (1986). A quantum leap towards the hearing computer?. *Computing, 18th September*, 26-27.

Erman, L.D. & Lesser V.R. (1980). The HEARSAY-II speech understanding system: A tutorial. In W. Lea (Ed.) *Trends in Speech Recognition*. 361-381. Prentice Hall, Englewood Cliffs: N.J.

Flanagan, ?nJ. Levinson, S. Rabiner, L. and Rosenberg, A. (1980). Techniques for expanding the capabilities of practical speech recognisers. In W. Lea (Ed.) *Trends in Speech Recognition*. 425-444. Prentice Hall, Englewood Cliffs: N.J.

Kauffmann, H. & Grumbach, A. (1986). MULTIple worlds in LOGic Programming. *Proceeding of ECAI '86, Vol. 1*, 291-305.

Izard, S. (1986). Levels of representation in computer speech synthesis and recognition. In M. Yazdani (Ed.) *Artificial Intelligence: Principles and Applications*. Chapman and Hall Computing: NY.

Lowerre, B. & Reddy, R. (1980). The HARPY speech understanding system. In W. Lea (Ed.) *Trends in Speech Recognition*. 340-360. Prentice Hall, Englewood Cliffs: N.J.

Sloman, A. (1986). Real time multiple-motive expert systems. In M. Merry (Ed.) *Expert Systems '85: Proceedings of the Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems*. C.U.P: Cambridge.

Walker, D.E. (Ed.) (1978). *Understanding Spoken Language*. North-Holland: New York.

Wolf, J. and Woods, W. (1980). The HWIM speech understanding system. In W. Lea (Ed.) *Trends in Speech Recognition*. 316-339. Prentice Hall, Englewood Cliffs: N.J.

3. Applications Infra-Structure

3.1. Hardware

This section looks at the possibilites for progress in hardware development. One of the most contentious issues raised here concerns parallelism, which can also be considered as an issue of knowledge representation, programming or AI technique. Also some developments in parallelism do not involve a radical departure from traditional underlying hardware (e.g., communicating sequential processors). For the sake of clarity, all major questions concerning parallelism are discussed under this section.

3.1.1. Exisiting Problems

Why should we be concerned about hardware development? There are two main reasons. The first concerns the need for greater efficiency from hardware and the programs that run on it. Enkintrap (1986) reports that computer usage in large installations has been increasing by between 40-60% a year. However traditional processing power has only been increasing by 15-20% a year.

The second reason concerns what parallelism might offer us in terms of the development of new AI techniques and concepts, and perhaps even a new "paradigm" of computation. Bobrow and Hayes (1985), in their review of progress in AI, show that several eminent researchers believe that little significant progress has been made in the last decade. Their suggestion is that we have learned to refine and commercially apply several ideas from the research of the sixties and seventies, but no significant theoretical advances have been made. However, over the last few years there has been some optimism displayed about the potential effects of developments in "connectionism". For example, Boden suggests "one area where it seems that there may be a breakthrough is that of connectionist ... machines ... exploratory research should become possible within the next decade" (Bobrow & Hayes, 1985), and Feldman suggests "many of the core problems of AI, such as pattern matching, context sensitivity, representation of real world knowledge and plausible inference, are better approached on a non-symbolic computational basis" (ibid). He goes on to argue that connectionist machines may offer an alternative.

3.1.2. Future Developments

There are currently two main approaches to the development of hardware and software: The serial or "evolutionary" approach concerns making existing hardware and software more efficient. The non-serial or "revolutionary" approach involves advances in micro- and macro-parallelism (connectionism), which is concerned with radical (and not so radical) changes in the underlying hardware of the machines we use, and possibly radical changes in our conception of computation as a whole.

(a) The Evolutionary Approaches or 'do it faster'

Two important developments are considered briefly here. Changes in VLSI design and research on "abstract instruction sets", aimed at accelerating the speed of programming languages running on conventional and, potentially, parallel hardware. Neither of these approaches directly offer computing any radical departure in terms of new concepts/techniques etc. However, the increases in speed of processing that they do offer may bring about new possibilities in applications.

(i) VLSI Design

Current research into VLSI design shows that some major advances have been, and will be, made in increases in processing speed in the short and medium term. Improvements in VLSI design involve better, more efficient chip architectures, for example, closer spacing of components. VLSI design should not be seen as an alternative to the advances made in parallelism. For example, the development and commercial realisation of array processors (see below) have been enabled by advances in VLSI design. However, it is suggested below that it is unlikely that we will see the commercial production of full scale dedicated hardware based on cheap "customised" VLSIs.

It should be noted that, while developments in VLSI chip design will aid in producing more efficient AI and traditional software applications, AI techniques will also aid in VLSI design. It is expected that A.I will make significant contributions to this research by providing VLSI designers with computer aided design (CAD) techniques and other support tools. Wolf, Kowalski &McFarland, (1986) discuss some of the ways AI based tools can aid VLSI designers. Finegold, (1985), Kahn, (1985), and Lathrop & Kirk (1986) provide descriptions of existing tools. It is expected that VLSI design will be a useful test bed for complex CAD techniques that will later be used in other areas.

(ii) Warren's Abstract Machine (WAM)

There have been attempts to design computer architectures and languages where the basic machine language is a form of symbolic logic and the basic machine operation is a form of logical inference. This has been stimulated by the Japanese Fifth Generation Project. Prolog is the best known (but not the only) logic based language. Attempts are also being made in building similar macchines for other functional languages such as LISP.

What are the prospects of building such machines and what sort of increases in speed can we expect? In this section I briefly examine Warren's work on the design of an abstract instruction set for Prolog (called WAM) and Tick's work on the design of a hardware sequential processor, especially adapted for the WAM. Their belief is that large scale parallelism is unlikely to be realised and/or will not run practical Prolog programs. However, they believe that it will be possible to build WAM-based Prolog systems with small-scale parallelism, that these systems will run very fast and that there will be no need for special Prolog dialects.

Warren (1983, 1980) has designed a Prolog abstract instruction set and Tick has designed the processor, (reported in Tick & Warren, 1984). Experimental prototypes have been built and tested. The "High-Speed Prolog-Machine" (Nakazaki et. al 1985) uses WAM working on "exotic", non-standard serial hardware. This machine produces 280,000 logical inferences per second (280 Klips). This can be compared to more traditional Prologs. For example, the following are benchmarks for POPLOG Prolog, including garbage collection:

SUN-3	12.0 Klips (estimated by extrapolation)
GEC-63	7.6 Klips
VAX-780	6.4 Klips
SUN-2	3.7 Klips
VAX-750	3.5 Klips

These figures correspond to the state of the system in January 1986, there have been significant improvements since then.

It should be noted that there is no need to build special hardware for the WAM. An experimental prototype of a modified version of the WAM has been implemented on traditional hardware. This system also produces impressive performance. Knodler et. al. (1986) report the WAM running on traditional hardware (a Motorola 68000 sequential processor) can provide 10 Klips (expected to rise to 40 Klips in **later** implementations), and that this can be improved with PIAG to 400 Klips. PIAG is **a** "programmable adaptive instruction generator", it is reported in Bursky (1985) **and** it does not seem to require the development of new hardware.

It should be pointed out that most Prolog Benchmarks quote only the 'naive reverse* test. A number of more comprehensive comparisons have shown that this can be quite misleading, since the naive reverse method tests only a small subset of Prolog operations. Different programs can give very different comparative results for different Prolog systems. For example, on naive reverse tests, compiled Quintus is four or five times as fast as POPLOG Prolog, yet Thorn-Emi found tha on tests involving a large Prolog program the differences between Quintus Prolog and POPLOG Prolog were mostly only a few percent, so they chose POPLOG. as they considered it had a better environment. This also indicates that the surrounding environment is very important (see Section 3.2 for a discussion of AI environments).

All the figures for the WAM serial implementations seem to compare favourably with existing parallel implementations of Prolog, Furthermore, the programmer's task is no more complicated than with traditional implementations, and there do not seem to be the "book-keeping" or control problems that exist with parallel implementations (e.g., Shapiro's Concurrent Prolog runs at 5 Klips on a SUN-3, this is expected to rise to 65 Klips; PARLOG runs at 2.5 Klips on a SUN-3 — these figures were reported at ECAI 1986 by Shapiro).

In any case, Warren's research should not be seen as an alternative to the work on parallelism as he argues that the WAM is parallel compatible. Warren suggests that once (or, if) the problems with parallel implementations are overcome, he expects a parallel implementation of WAM-based Prolog to come close to producing giga lips (i.e. 1,000 Klips). In the next section, I consider some of the approaches to parallelism and their prospects for success.

(b) Revolutionary Approaches: *Do it at the Same Time^9

In the AI "public eye" there seems to be the impression that there is a single approach to parallelism, with unified goals and techniques. This often leads to disappointment when anouncements are made about developments in parallelism that are subsequently identified as "not REAL parallelism". This section shows that there are various forms of parallelism. In the discussion, the different techniques and underlying goals are reviewed, as well as the possibilities of achieving these goals. This section draws heavily on Ramsay (1986).

(0 General purpose computing engines

One goal of parallelism is to produce very fast general purpose computing machines. That is, the aim is to produce machines that perform the same functions as present computers, but which perform the functions much faster. There are three main techniques for achieving this goal.

Communicating sequential processors

The first is to utilise existing Von-Neumann, serial hardware but to connect machines together, with different machines doing different parts of the task in parallel. As the programs run, they read in data from various devices and output the results to other devices. The other devices consist of other computers running different parts of the program.

The advantage of this technique is that there is no need to invest heavily in the development of new hardware. However there are many problems with this approach. The programmer's task is much harder, he will have to specify what parts of the programs will run on different processors. In order for any improvements to be made in efficiency, programmers need to split up the tasks into the right number of processes (compatible with the number of available processors). If there is a disparity between demand and supply for the number of processors, then the technique can become very inefficient. While there is no need to invest in radically new hardware, these techniques do demand the use of several machines.

For further details see Hoare. (1978) and Brich-Hansen. (1978).

Dataflow programs and machines

This technique is characterised by two features: First, the development of a new formalism which makes explicit the operations and the connections between them (specifying where to get input, where the output goes to). Once these languages are learned, this will make the programmer's job much easier, since the task of "controlling" the parallelism is left to the machine. Second, this technique involves the development of hardware on which the parallelism in these formalisms can be run.

Two types of formalism exist for dataflow machines: diagrammatic or graphical languages and textual languages. In graphical languages, the connections between the various operations are indicated by drawing lines; a program written in a graphical language would look like a network or tree, with nodes and connections. Textual languages would look very similar to present day languages (which are also mostly textual). Sharp. (1985) provides a review of dataflow languages.

Ramsay believes that textual languages will be very difficult to work with. However, he suggests that it would not be difficult to produce a new generation of editors and compilers which could handle the diagrammatic languages.

A number of designs for such machines exist. Srini (1986) surveys the performance of several of the most widely reported machines, some of which produce "impressive[11] performances. However the number of processors in each machine is very low. and to attain better performance will require an increase in the number of processors. Ramsay suggests that the problem is that the manufacture of each processor is a complex and expensive task, each with many subcomponents. Dataflow machines will provide massive increases in performance, but are likely to have millions of lower level components.

Functional and single assignment languages

The final strategy for improving the performance of general purpose programs is to return to the mathematical foundations of programming languages. Many languages are based on mathematical formalisms, e.g., lambda calculus (LISP, POP-11) and predicate calculus (Prolog). In pure form these seem well suited to parallelism, as each program is made up of nested expressions, each of which (in the pure form) should be capable of evaluation independently (and hence in parallel). As Ramsay suggests, the crucial point is that the evaluation of an expression in these languages can have no side effects which linger on after its value has been used in the value of some enclosing expression. As there are no side-effects, there are no hidden dependencies between expressions. It is therefore easy to see which expressions can be evaluated in parallel: they are just those expressions which appear as a subexpression of something at a higher level.

There are now several designs for architectures for these machines (Darlington & Reeve, 1981; Kennaway & Sleep, 1984). The major problems with this work are the same as for dataflow architectures. That is, it will be difficult to provide enough processors, and much of the processing power will need to be dedicated to book-keeping.

The work of Shapiro on Concurrent Prolog, and of Gregory on PARLOG can be seen as part of this movement (reported by Shapiro at ECAI 1986). However neither of these languages run on "real" parallel hardware, they run on traditional hardware which simulates parallelism. Also, neither of them are pure forms of Prolog, they involve the programmer in the additional tasks of specifying the control of the parallelism. Therefore, they involve the programmer in additional tasks, such as learning a new notation. Also almost all of the current schemes for the parallel execution of logic programs use parallelism in restricted ways. This is mainly due to the problem of efficiently implementing a general AND/OR parallel system (Pollard, 1982). Also Prolog programs are well known for their non-deterministic nature. This means it is very likely that they will spawn a large number of tasks which are of no use. Ramsay concludes that the book-keeping of any parallel executions of Prolog may well outweigh the advantages which may be obtained — and may not compete with other work on faster serial processing (e.g., Warren and Tick's work, see above).

(i) Essentially distributed algorithms: Dedicated machines and array processors

Dedicated machines

Another goal of parallelism is to produce machines that perform a certain task or sets of tasks much faster. Many tasks are of the form where the same computation has to be performed for each of a large number of inputs, and can be done independently. They are typical of the tasks found in low-level vision and speech.

Ramsay points out that the work on VLSI design means that if we have some algorithm to perform such a task then it can, in principle, be placed on one of these chips. If each chip needs connections only to its immediate neighbours (as should be the case in the task specification), then we should be able to link processors together to do the tasks in parallel.

Ramsay suggests that the major problem is that, despite the advances made in VLSI design, it is still unlikely that dedicated machines (those with algorithms "built into" the chips) will become cheap or commercially available for some time. If chip manufacture is to be profitable, then we must reach volume production. The question is, how sure can we be that we have produced the "right" algorithm and VLSI design for some task.

Array processors

Slightly more general machines are becoming cheaply available. Fountain (1983) describes CLIP4, which is a commercially available machine with 32*32 and 128*128 processors for about the price of a good workstation. These machines have similar uses as dedicated machines, however, they have much simpler and more general processors. Hillis' book (Hillis, 1985) gives an account of how existing programs (including those written in symbol manipulating languages like LISP) can be adapted to run on these machines. The major problem with array processors is the limitation on memory size of the processors. This limitation means that a good deal of the machine's time can be spent loading data onto the input lines and getting the results off the output line. Ramsay concludes that, acknowledging the caveat above, "it is clear that these machines are going to have a major impact in the immediate and short term future wherever suitable problems are recognised" (Ramsay, 1986 p. 20).

(iii) Parallel representations of knowledge: The connectionist hypothesis

What is connectionism and why is there so much optimism? The "connectionist hypothesis" suggests that the computation performed by a network of processors need not be defined by the relation between the signals on some selection of input lines and the signals, when they arrive, on a selection of output lines. It is defined instead by the relation between the states of some selection of processors at the start of the computation and the states of some, possibly other, selection of processors at the point when an entire network is stable (Ramsay's, 1986 definition). These systems would run on hardware with tens of thousands of simple processors. A recent collection of papers (Hinton & Anderson, 1981) present research which has explored some of the properties of connectionist models. It is believed that they can be made to have the properties modelled by Boltzmans' laws of statistical thermodynamics. It is suggested that these systems will converge to steady states, will have learning abilites and that it may be possible to introduce minor damage to the networks without severe consequences. There seems to be an intuition amongst researchers that many of the properties of the human brain are mirrored in these systems — robustness, ability to cope with degraded inputs, ability to learn. "The trouble is that, although the reports on individual systems do seem to indicate that they do some of what we want, there is no impression of an overall theory which could be used either to explain human abilities or to construct practically useful machines" (Ramsay, 1986 p. 23).

3.1.3. Conclusions

Clearly, then, there is more than one approach to the improvement of efficiency in computing, and there is more than one approach to parallelism. Whether parallelism will be successful depends on what form of parallelism we are trying to produce, and this in turn depends on the uses to which it is put. However, it is unlikely that general purpose computing will be radically effected by changes in underlying hardware in the next ten years. We are more likely to see increased speed in sequential processing brought about by the work on WAM, and it is possible that further developments of WAM based systems will include limited parallelism.

One area where parallelism will have an effect within the next few years is in array processing. With the continued work on VLSI design and the present commercial availability of array processors, it might be expected that we will see improvement in these machines that can already produce impressive results compared to traditional machines. The only problem will be finding appropriate tasks, it is expected that vision and speech processing will be the main areas affected.

So far as macro-parallelism is concerned, we are still experimenting. It will probably be another five years before we are certain about the utilities of these (as yet non-existent) machines and possibly another decade before we begin to see any practical applications (if any). It is still unclear exactly what properties these machines will have, but it is clear that there are many tasks which will demand some form of macro-parallelism.

References

Bobrow, D.G. & Hayes, P.J. (1985). Artificial intelligence — where are we? *Artificial Intelligence, 25*, 375-415.

Brich-Hansen, P. (1978). Distributed processes: A concurrent programming concept. *Communications of the ACM, 21*, 934-941.

Bursky, D. (1985). Instruction generation technique speeds program execution. *Electronic Design, March 7*, 40-44.

Darlington, J. & Reeve, M. (1981). ALICE: A multiple reduction machine for the parallel evaluation of applicative languages. *Proceedings of the ACM Conference on Functional Programming Languages and Computer Architecture.*

Enkintrap, N. (1986). Quiet life falls under revolutionary attack. *Computer Weekly, July 3*, 22.

Finegold, A. (1985). The engineers apprentice. In P. Winston & K. Prendergrast (Eds.) *The AI Business: Commercial Uses of Artificial Intelligence.* MIT: Mass.

Fountain, T. (1983). Image processing by parallel computer. *Automation, September*, 8-15.

Hillis, W. D. (1985). *The Connection Machine.* MIT Press: Mass.

Hoare, C.A.R. (1978). Communicating sequential processes *Communications of the Association of Computing Machinary, 21*, 666-677.

Kahn, H.J. & Filer, N.P. (1985). An application of knowledge based techniques to VLSI design. In M. Merry (Ed.) *Expert Systems '85: Proceedings of the Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems.* C.U.P: Cambridge.

Kennaway, J.R. & Sleep, M.R. (1984). Towards a successor to von Neuman. In F.B. Chambers, D.A. Duce & G.P. Jones (Eds.) *Distributed Computing.* Academic Press: London.

Knodler, B. Neidecker, B. & Rosentiel, W. (1986). A Prolog machine for Warren's abstract instruction set. *Proceedings ECAI '86. Vol. 2.*

Nakazaki, R. & Konogaya, A. (1985). Design of a high-speed Prolog machine ICOT Technical Memorandum TM-0105, April.

Sharpe, J.A. (1985). *Data Flow Computing.* Ellis Horwood: Chichester.

Sloman, A. (1985). Real time multiple-motive expert systems. In M. Merry (Ed.) *Expert Systems '85: Proceedings of the Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems.* C.U.P: Cambridge.

Srini, V.S. (1985). An architectural comparison of dataflow systems. *Computer, 19*, 68-87.

Ramsay, A. (1986). Distributed versus parallel computing. *Artificial Intelligence Review, 1*, 11-25.

Tick, E. & Warren, D.H.D (1984). Towards a pipelined Prolog processor. *New Generation Computing,* 2, ***(4).***

Warren, D.H.D (1983). An abstract Prolog instruction set. SRI International Technical Note 309.

Warren, D.H.D (1980). An improved instruction set optimises tail recursion. Dept. AI Univ. Edingburgh, Research Paper 156.

Wolf, W.H., Kowalski, TJ. & McFarland, S.J. (1986). Knowledge engineering issues in VLSI synthesis. *Proceedings of AAAI '86,* 866-871.

3.2. AI Languages and Environments

This section discusses languages and AI environments. Two main issues are considered: First, research in these areas which, it is hoped, will increase human productivity. Second, approaches to making AI languages and software more relevant to the tasks at hand.

3.2,1. Increasing Productivity

An important problem in producing software is not just generating software and hardware that runs fast (see Section 3.1), but producing programmers who work fast as well. As hardware costs drop, human efficiency issues will become increasingly important. The problem of human efficiency is particularly important in AI software development. Sleeman has provided a definition of AI, characterised as the study of inherently ill-defined problems, where the task is to try to transform ill-defined problems into well-defined ones (Sleeman. 1985). Partridge (1986) gives a clearer definition, by comparing the nature of AI problems to traditional software engineering problems. Some of the characteristic features include:

AI Problems:

- Answers tend to be adequate or inadequate
- Context-sensitive problems
- Not completely specifiable

Software Engineering Problems:

- Answers are correct or incorrect
- Context-free problems
- Completely specifiable

The process of solving AI problems usually involves understanding the nature of the problem by iteratively refining its definition through the activity of programming — this is known as the RUDE cycle: Run — Understand — Debug — Edit. This is in contrast to traditional software design, where a large percentage of the work is done at the program specification stage, away from the machine. If we accept that this definition characterises the AI programmer's task, then a potential source of inefficiency is the time taken to pass through each iteration and the number of iterations involved.

There are two (not mutually exclusive) ways of increasing the production of software. The first is to involve more people. This in turn will (in part) involve making the systems more accessible, that is, making the use of the systems less "skills-dependent". A second way of improving productivity is to help those that have the skills to be more efficient, either through the use of intelligent interfaces or by the use of "unintelligent intelligence amplifiers". This in turn could lead to a third approach, which is to automise the task completely through automatic program synthesis.

(a) Making the Systems More Accessible

In a recent series of articles, Enkintrap (1986) suggested that parallelism could potentially solve many of the problems of computing.

He suggests that there is an "applications backlog". That is. we have now developed the necessary techniques for many commercial applications but one important factor holding us back is that we do not have enough skilled people to produce the applications based on these techniques. One way to overcome this problem is to make machines more accessible to those without skills. Implicitly, he suggests that advances made in parallelism will contribute to the solution of this problem: "Software development productivity will always be limited while computers require system designers to concentrate on the way the machine works, and that in turn will always be the case while von Neumann architectures are used" (Enkintrap 1986, p.21).

However, in Section 3.1 this conclusion was questioned. One reason is that parallelism is not going to have a radical effect in the next few years (it will mostly affect a small number of tasks, or will produce limited improvements in the speeds of general purpose high level languages). Another reason is that present developments in parallelism suggest that some parallel languages will be just as hard (if not considerably harder) to learn and use. It is likely that with languages like PARLOG, Concurrent Prolog or in programming communicating sequential processors, the programmer will actually need to know more about the underlying machine than they would if they had used conventional techniques.

However, Enkintrap is right in suggesting that machines will become more friendly — but not for some time yet, and not simply because we will have parallel architectures. The advances will be due in part to making programs much more efficient: programs operating in parallel should be faster than serial programs. This will allow interface designers to produce more complex tools or models of users, etc. This could be called "enabling technology", but the major advances will come from the theoretical advances made in HCI and AI These will also be "enabled" by parallelism, i.e., it may help us to produce more complex/appropriate cognitive models. However, as Sloman (1985) points out, although many features of human mental functioning are embodied in a massively parallel system, and much of our mental functioning is likely to have parallel characteristics, we may not need massively parallel systems to model it or to produce systems (e.g., interfaces) which use models of it. He suggests that some human mental functioning can be explained and modelled in much "cruder" forms of parallelism, perhaps in existing parallel systems or even in simulated parallel systems. Probably the most important factor will be the fundamental research that needs to be done in HCI in order to understand the cognitive processes of human users. If this is not done then we will have yet another backlog — we will have the enabling technology but no theories to "plug" into it.

Other ways of making AI systems less "skills-dependent" include the development of high level languages (see below), the development of intelligent editors (e.g., Waters, 1982) and the continuing work on high resolution graphics, bit map screens and pointing devices. Finally, the work in intelligent tutoring systems may contribute to increased productivity in an indirect manner. One result of this work (apart from introducing AI into the educational system) will be that we can produce "skills transfer" much more quickly, efficiently and sensitively than at present. There are likely to be "AI tutoring systems", i.e., not just systems that introduce AI techniques to tutoring systems in traditional areas (Maths, English, Physics, Electronics), but systems which teach AI languages. Most of these systems are still very experimental, but it is expected that considerable progress will be made in the next ten years. Also, some exisisting AI development environments contain "passive", unintelligent tutors. For example the POPLOG environment contains many tutorial files introducing novices to the system.

(b) Programmer's Support

A whole spectrum of approaches are, and will be, used in making those who have AI skills more efficient (some of those mentioned above fall into this category). At one end of the spectrum is the work being done on providing intelligent tools for programmers, for example, programmer's apprentices (e.g., CEDAR, Teitelman, 1985; Rich, 1985; Rich & Shrobe, 1978). Other examples include tools for supporting or automating the task of debugging (e.g. Elsom-Cook & du Boulay, 1986), commenting (e.g., Rich, 1985) and browsing (e.g. ECO, Robertson et. al., 1985). This work can be seen as a progression towards automatically synthesising the job of programmers. The other end of the spectrum includes less intelligent versions of some of the tools mentioned here — keyword searchers etc. Although the goal of achieving automatic program synthesis in the next ten years seems very remote, it is expected that these developments will contribute significantly to programmer's productivity. Indeed, practical applications of the Programmer's Apprentice (Rich, 1985) have been found in telecommunications. The SCAT system transforms descriptions in SDL (Specification Description Langauge) to CHILL (CCITT High Level programming Language). This partly automates the most crucial phase in the software development process, i.e., the transition from the project's detailed specification to the software implementation. This system has already been applied in the development of message handling in the Italian public packet switching network (see Barra et al. 1986a, for a review of it's performance, and Barra et al. 1986b, for a description of the system).

3.2.2. Increasing Relevance

(a) High Level Programming Languages

It is commonly believed that is easier to write programs in high level languages as they allow a more direct translation between the structure of a task and the structure of the program. They are also considered to be very easy to debug and maintain. These features have allowed us to develop applications (e.g., expert systems) that were not possible in lower level languages. They also provide us with tools for thinking about complex problems. However, all programming languages specify objects and procedures to solve a certain set of problems. The characteristics of high level languages derive from their purpose i.e. to solve AI problems; they were designed to produce systems that could display intelligent behaviour.

(b) Types of Language

Originally procedural languages like LISP were considered to be general purpose. However over the last 5-10 years it has been recognised that they are appropriate for only a limited subset of problems. Some problems are not best solved in terms of the procedural paradigm; they may be expressed more adequately in terms of rules (e.g., logic), or in terms of objects passing messages, or in terms of objects and procedures inheriting properties. These needs have paralleled and motivated the development of alternative languages such as Prolog and Smalltalk.

Over the next decade, it is expected that work will be done on clarifying the characteristics of the tasks for which these languages are best suited. It is also expected that the popular forms of these languages will be provided in fast single language environments running on specialised hardware (e.g., the Japanese PSI machine, Taki et. al., 1984), probably making use of limited parallelism. (See Section 3.1 for a discussion of the speeds of various machines.)

However, many of these large machines will be used for software development work. This means that the question of the facilites provided in the environment will be as important as the speed of the languages they run. Some relevant technical issues that should be considered in choosing any AI language system include:

- availability of tracing and debugging tools, including use of saved images;
- links to non-AI languages (e.g., C, Fortran, Pascal) and utilities (e.g. NAG libraries);
- presence of an integrated editor including "compile marked range" and "output to editor" facilities;
- good online help and teaching facilities;
- good window/mouse/graphics facilities — for more sophisticated applications;
- provision of library facilities for collaborative development work;
- provision of a library of utility programs;
- access to operating system facilities, including all I/O facilities; pipes, spawning sub-processes, mail etc.;
- the size of the image — if it is too big, memory costs or paging may be prohibitive;
- the range of machines and operating systems for which the environment is available.

(c) Multi-language Environments

Assuming that development machines will need to be used for many different tasks, it is likely that multi-language environments will be needed. It is expected that there will be an increase in the use of multi-language environments like POPLOG and LOOPS. It is possible that the languages in these environments will also become faster (perhaps through limited parallelism with hundreds of processors).

Multi-language environments, such as POPLOG allow the software designer to have an "AI Meccano Set". These are, at present, mostly made up of many tiny pieces, though in the future it is expected that there will be an increase in the size and efficiency of these pieces (for example, through the introduction of truth maintenance packages, production systems etc.).

The major problems with multi-language environments concern the interfaces between the languages and the problem of porting multi-language programs from the development environment, such as POPLOG (running on VAXs, GECs, SUNs workstations, APOLLOs, HP BOBCAT workstations and BLEASDALEs), to target user machines, which are usually much smaller and cheaper. One way to overcome the first problem (and, potentially in the future, the second) is to produce a common underlying virtual machine that supports the basic operations of all the languages used in the environment (as in the developments in POPLOG). This movement may also help with the portability problem, since the problems of providing a syntax transformation "front end" to the environment will mean considerably less work than hand coding the language from scratch.

Also, over the next few years, progress in hardware development means that small, cheap machines are becoming very powerful, some being able to run AI programs. This means that AI software development environments (or some subset of them) will be available, not only on the large development machines, but also on smaller cheaper machines such as Ataris and Macintoshs.

There are a number of options for making software produced in a development environment available on small cheap machines (without porting the whole environment):

- Deliver the program in a saved image without all the (unnecessary) environment libraries and documentation.
- It may be possible to link together only a subset of the core environemnt, leaving out those parts not required.
- If the program is written in some simple subset of a language that is already (or soon will be) available on small machines (e.g., Prolog, LISP. POP-11). then it amy be possible to simply port the program.
- Develop a full batch compiler for users (for the languages that exist in the environment). For example, at present, in POPLOG there are tools that translate POP-11 source code into assembler files, possibly for a different machine. Those files are used in re-building the POPLOG system, or building a new version on a new machine. There are plans to package these tools so that they can be used by software developers to package thier product into a run-time system. It is hoped that these tools will be available in the next few years, although it does depend on the availablility of funds.

All these approaches could mean that programs developed on large machines could be ported (along with the necessary parts of the development environment) to these smaller cheaper machines. This will allow us to use small, cheap target machines for programs developed on much larger hardware. There are already versions of POP-11, LISP and Prolog on the Macintosh, and there are plans to put some version of POP-11 on the Atari.

Some tasks (e.g., knowledge based systems for low level speech and vision processing — see Section 3.1) seem suited to the use of parallel architectures. At present there are very few software development environments specially designed to aid the designer in producing software for these sorts of tasks. However, it is likely that we will see the development of sophisticated environments (including appropriate software support tools) that would be suitable for the development of software that would run on (for example) array processors. For example, see Bisiani (1986) for a discussion of the AGORA environment.

(d) Shells Ts Multi-language Environments

High level language environments are obviously not the only tools for building expert systems. There are two types of tools for constructing expert systems: high level programming language environments and expert system shells. They both have advantages and disadvantages (this section draws heavily on Reichelt et. al. 1985).

Shells are designed specifically for expert systems development. They are usually abstracted from existing systems by taking out the knowledge base contents which are specific to their original application. It has been assumed that these shells would be appropriate to a number of tasks. Often this is not true, in particular, the inference engines are very domain specific. They are now generally considered as rather inflexible, and they often provide only meagre debugging and explanation facilities. Progress in the development of other support tools are mentioned in Section 3.3. on knowledge elicitation.

On the other hand, high level programming environments provide a much lower level set of tools (and not just for expert systems development). The engineer is left to construct the basic architecture of expert systems. This gives him the flexibility of designing the types of representation and inference engine he requires. However, the major problem is that they may present the engineer with too much choice and too little guidance in deciding what techniques are appropriate; he is also left with the problem of having to implement these techniques. In other words, he has a complex Mecanno set but, without wide experience in building models, it is difficult to conceptualise from the pieces exactly what it is possible to build. There is often also the problem of having to start from "nuts and bolts" every time he builds a model. Some environments provide more support than others. For instance, in POPLOG there exist many documented libraries giving examples of expert systems and general production interpreters. As mentioned, it is expected that more tools like these will be added and refined. While these are not always efficient implementations they give an indication of how to build them and the documentation gives an indication of the tasks for which they are appropriate.

It is expected that there will be work on identifying the appropriateness of different types of languages, control strategies and shells for different tasks. Some of this work has been started, for instance a review of criteria for choosing an inference engine for expert systems is given by Reichelt et. al., (1985). There are applications for resources to add more and better higher level components to POPLOG. There is also interest in developing general software engineering tools based on knowledge based system techniques. These would be used to aid the programmer in specifying the problem more clearly.

References

Barra, S. Ghisio, O. & Manucci, F. (1986a). Experience and problems of applications of automatic translation from SDL specifications into CHILL implementations. In *6th International Conference on Software Engineering For Telecommunication Switching Systems*. Eindhoven.

Barra, S. Ghisio, O. & Manucci, F. (1986b). SCAT, an automatic-programming tool for telecommunications software. *Proceedings of AAAI '86*, 831-835.

Bisiani, R. (1986). A software and hardware environment for developing AI applications on parallel processors. *Proceedings of AAAI '86*, 742-747.

Elsom-Cook, M. & du Boulay, B. (1986). A pascal program checker. *Proceedings ECAI '86. Vol 2.*

Enkintrap (1986). Quiet life falls under revolutionary attack. *Computer Weekly, July 3, 1986*, 22.

Partridge, D. (1986). Engineering artificial intelligence software. *Artificial Intelligence Review, 1*, 27-42.

Reichelt, H. & van Harmelen, F. (1985). Relevant criteria for choosing an inference engine in expert system. In M. Merry (Ed.) *Expert Systems '85: Proceedings of the Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems*. C.U.P: Cambridge.

Rich, C. (1985). The programmer's apprentice. In P. Winston, & K. Prendergrast (Eds.) *The AI Business: Commercial Uses of Artificial Intelligence*. MIT: Mass.

Rich, C & Shrobe, H.E. (1978). Initial report on a LISP programmer's Apprentice. *IEEE Transactions on Software Engineering, Vol. SE-4, (1).*

Robertson, D., Muetzelfeldt, R., Plummer, D. & Bundy, A. (1985). The ECO browser. In M. Merry (Ed.) *Expert Systems '85: Proceedings of the Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems*. C.U.P: Cambridge.

Sloman, A. (1986). Real time multiple-motive expert systems. In M. Merry (Ed.) *Expert Systems '85: Proceedings of the Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems*. C.U.P: Cambridge.

Sleeman, D. (1985). The low road, the middle road, and the high road. In P. Winston & K. Prendergrast (Eds.) *The AI Business: Commercial Uses of Artificial Intelligence*. MIT: Mass.

Taki, K. Yokota, M. Yamamoto, A., Nishikawa, H. Uchida, S. Nakashima, H & Mitsuishi, A. (1983). *Hardware Design and Implementation of Personal Sequentail Machines (PSI)*. ICOT: Japan.

Teitlman, W. (1985). A tour through CEDAR. *IEEE Transactions on Software Engineering, Vol. SE-11 (3)*.

Waters, R.C. (1982). The programmer's apprentice: Knowledge based editing. *IEEE Transactions of Software Engineering. Vol SE-8 (1)*.

3.3. Knowledge Elicitation

Knowledge elicitation has now been recognised as a non-trivial and important task in AI In order to provide intelligent systems we need to provide them with appropriate and well structured knowledge. Therefore, a major challenge will be producing methods (perhaps computationally supported) of transferring human knowledge into machines. There are, at present, fairly successful techniques for eliciting simple empirical knowledge, such as the associations between a series of symptoms and a diagnosis, for example, interviewing, lectures, protocol analysis, personal contruct theory and conceptual analysis. Wielinga & Breuker, (1984) and Brouwer-Janse & Pitt, (1986), provide a review of current techniques. There has also been some progress towards building computational systems to support the knowledge engineer, for example, TEIRESIAS (Davis, 1983) helps experts transfer their knowledge into expert systems, and then aids them in revising individual rules to guarantee that no important knowledge is missing.

Other systems circumvent any interaction with the expert. This is done through automatic induction of the rule-set e.g., EXPERTEASE (Freyenfeld, 1984). The major problem with these systems is that, because they learn from example, care must be taken in presenting the right rules and possibly in the right order — in some circumstances this will be an expert's job! However, any progress made in the domain of machine learning is likely to contribute to this area (see Mitchell et. al., 1986 for a recent review of the field, or the standard texts by Michalski et al., 1983, 1986). It is likely that we will produce more automatic induction systems and understand more about their limitations. It is possible that in specialised areas they could be of considerable use over the next ten years. However it is likely that the task of knowledge elicitation will become much harder, and different techniques (perhaps computational) will need to be developed.

This challenge is accentuated by two related factors. The first concerns the progress made in AI which is providing the enabling technology for more sophisticated expert systems. As progress is made in the development of AI techniques (e.g., blackboard architectures allowing multiple forms of knowledge representation and inference, advances in deep representation techniques, reason maintenance and mixed initiative expert systems etc.), the technical possibilities of building more sophisticated expert systems may lead to a demand for the elicitation of more knowledge types. That is, there is clearly a demand for more complex expert systems, however as the opportunity for building more complex expert systems arise, the task of eliciting that knowledge may become much harder. Therefore a major problem will be in developing appropriate knowledge techniques for these new types of knowledge. For instance, it does not seem that Kelly's "Personal Construct Theory" is an appropriate technique for elicitating knowledge involved in deep modelling (see Gotts, 1984). De Mantaras et al. (1986) give an example of the use of Personal Contruct theory in a knowledge based, knowledge elicitation tool.

The second factor concerns the parallel demands for more sophisticated performance in expert systems — particularly where widespread professional acceptability is needed. For example, with medical diagnosis systems, a decision's acceptance by the doctor, or even the patient will be very important. Part of this acceptance may depend on the ability of the system to explain and justify its conclusion.

For instance, as Kidd (1985) points out, users approach tasks with their own intentions, expectations and constraints; these can significantly affect the choice or acceptance of an appropriate remedy. She also suggests that an important part of the giving, and accepting, of appropriate advice from an expert is the negotiation involved between the expert and consultant. It is likely that if we are to produce expert systems whose decisions are to be accepted by users in a wide variety of domains, then we may need to model the users' intentions, expectations, dialogue and negotiation strategies, etc., in order to provide an acceptable form of interaction and so that appropriate answers can be given.

More generally, there is a realisation that expert systems should have good pedagogical and explanatory abilities. Present expert systems have extremely simple means of explanation (usually an execution trace). In the future, we will have to produce systems that can reason about their own reasoning, not only so that they are more flexible, but also so that they can provide more convincing and helpful explanations.

Therefore, while the results of AI research might in principle be able to satisfy the demands for increasing sophistication in expert systems, this will only be possible if we can "fill" this technology with the appropriate knowledge; this in turn will depend on whether we can develop adequate knowledge elicitation techniques.

The author knows of little work directly involved in developing knowledge elicitation techniques for these other forms of knowledge, however, there is research going on that is relevant to this problem. Gotts (1984) provides a review of deep knowledge elicitation techniques. Also, the work being done in the application of AI in education may tell us a lot about how to elicit the expert's explanatory power (see Yazdani, 1986 for a review of current knowledge based tutoring systems). There is also a SERC funded research project in the Cognitive Studies Programme at Sussex on explanation and constructive interaction (contact is Dr. Claire O'Malley). Research in NLP, particularly recent work on the structure of discourse and language generation may also contribute to our understanding of the process of negotiation (see Section 2.2). Kidd (1985) provides some references on other work being done on negotiation.

References

Brouwer-Janse, M.D. & Pitt, R.B. (1986). Knowledge acquisition: Methodological issues and problem-solving profiles. *Proceedings of ECAI '86*, 120-127.

Davis, R. (1983). TEIRESIAS: Experiments in communicatibg with a knowledge based system. In M.E. Sime & M.J. Coombs (Eds.) *Designing for Human-Computer Communication*. Academic Press: London.

Freyenfeld, F.A. (1984). Decision support systems. National Computing Centre.

Gotts, N.M. (1984). Knowledge acquisition for medical expert systems — a review. AI in Medicine Group Report, AIMG-3 Sussex University.

Kidd, A.L. (1985). What do users ask? — some thoughts on diagnostic advice. In M. Merry (Ed.) *Expert Systems '85: Proceedings of the Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems*. C.U.P: Cambridge.

de Mantaras, R.L., Cortes, U., Manero, J., Plaza, E., Salra, X. & Agusti, J. (1986). Knowledge elicitation using personal contructs application to document classification. *Proceedings of ECAI '86*, 128-134.

Michalski. R.S.. Carbonell. J.G. & Mitchell. T.M. (Eds.) (1986). *Machine Learning: An Artificial Intelligence Approach. Vol. II.* Morgan Kaufmann Inc.: Cal.

Michalski. R.S.. Carbonell. J.G. & Mitchell. T.M. (Eds.) (1983). *Machine Learning: An Artificial Intelligence Approach. Vol. I.* Morgan Kaufmann Inc.: Cal.

Mitchell. T.M.. Carbonell. J.G. & Michalski. R.S. (Eds.) (1986). *Machine Learning, A Guide to Current Research.* Kluwer Academic Publishers: Mass.

Yazdani. M. (1986). Intelligent tutoring systems overview. *Artificial Intelligence Review, 1,* 43-52.

Wielenga. R.J. & Breuker. J.A. (1984). Interpretation of verbal data for knowledge acquisition. T. O'Shea (Ed.) *ECAI '84: Advances in Artificial Intelligence.* Elsevier: North Holland.