INTELLIGENT SYSTEMS:
A BRIEF OVERVIEW

Aaron Sloman

1983

Cognitive Science Research Paper

Serial No: CSRP 027

The University of Sussex,
Cognitive Studies Programme,
School of Social Sciences,
Falmer,
Brighton BN1 9QN

INTELLIGENT SYSTEMS: A BRIEF OVERVIEW

Aaron Sloman
Cognitive Studies Programme
University of Sussex
Brighton England

NOTE:

This is a slightly modified version of a paper written for a study group set up by the Science and Engineering Research Council and the Department of Industry in October 1982.

This paper, like many others produced in Britain during 1982-3 uses the initials 'IKBS' to stand for 'Intelligent Knowledge Based Systems'. My personal view is that what people were really concerned with would have been better described as 'Applied Artificial Intelligence'. There were historical reasons why some supporters of AI were afraid to use the term 'Artificial Intelligence' in Britain. However, it has since become clear that there was no reason for the fear, and the barbaric mouthful 'IKBS' remains merely as an embarassment.

The original remit of the study group was to investigate architectures required for IKBS. The group soon decided that there was little of interest worth saying about architectures in general, and broadened its discussions, ultimately producing a set of recommendations on how to fund research in AI.

This paper was included in the annexes to Volume 1 of the report: Intelligent Knowledge Based Systems: A Programme for Action in The UK' published by the SERC and Dept. of Trade and Industry in 1983. The report is available from I.E.E.

This version of the paper has a number of minor changes, to enable it to be read in isolation.

Introduction

The study of intelligent knowledge based systems (IKBS) is now widely acknowledged to be of crucial importance for the future of Information Technology. As computing systems become increasingly powerful and complex they will need to be given knowledge and intelligence, both to make good use of the new power in performing complex tasks, and also to enable people to instruct, control, interrogate or modify them: significant improvements to the man-machine interface will require machines sharing human knowledge and reasoning powers.

This document is an attempt to characterise the field. It builds on the working papers presented by Alan Bundy, Yorick Wilks and myself to the SERC IKBS workshop on Jan 6-7 1983, and on submissions by other participants. It also builds on some aspects of the papers submitted by Karen Sparck Jones to the Alvey Committee. It is intended to summarise matters which are widely agreed by workers in the field, but it suffers from having been produced in too short a time, and inevitably describes the field from only one of several possible viewpoints: so omissions and errors are to be expected. The paper can therefore, at best, have an illustrative function rather than a definitive one.

Although the study was originally meant to characterise architectures for IKBS, several contributors to the study argued that it was not (at present) possible to attempt a general characterisation of such architectures, partly because there are already too Htany different architectures being explored with no common pattern emerging, and partly because we don't yet know enough to give a general characterisation. Nevertheless it is possible to make some illustrative remarks giving a tentative, provisional, approximate, characterisation of **the field.** And it is possible to identify some research problems concerning possible structures for intelligent systems.

Our strategy is top down. We start by placing the field of IKBS in a broader context, namely work in Artificial Intelligence. First we characterise long term aims of AI, then describe some ideas which have emerged so far, and go on to mention some unsolved problems, including some which may yield to investigations over the next decade, and which could play a role in applied work. It is hoped in this way to counter-act a rather narrow view of the scope of IKBS which might be generated by the considerable publicity recently given to the 'expert systems' sub-field.

The Context, of: work on IKBS: Scijence and Engineering

Previous documents have stressed the applied nature of the field of IKBS. However, in order to understand the prospects for the field it is important to see its relations to the wider scientific study of intelligent systems.

This includes both

(a) the empirical study and modelling of existing intelligent systems (mainly human beings) and

(b) the theoretical analysis and exploration of possible intelligent systems and possible mechanisms, representations, etc. used by such systems.

This scientific work informs and is informed by engineering activities. These fall into two main classes:

(c) attempting to deal with problems of existing intelligent systems (e.g. human learning or emotional difficulties) and

(d) designing new useful intelligent or semi-intelligent machines.

Theoretical and applied work in this area produces various by-products such as new tools for software engineering and new computing resources for education. The term 'Artificial Intelligence' is variously used to refer to the combination of all these activities, or to some subset. In the former sense it is equivalent to 'Cognitive Science'. However, it is most commonly thought of as encompassing (b) and (d), though sections devoted to the others can be found in conferences on AI. The recently introduced term 'IKBS' seems to be mainly focussed on (d). However, merely designing systems without ever studying the general princples they illustrate, e.g. without attempting to understand the relative merits of different representations, algorithms, architectures, is a recipe for chaotic and unprincipled proliferation of systems. So (d) should not be pursued in isolation from (b). Moreover, insofar as part of the aim of (d) is to define machines which can communicate **with** human

beings, and display some of the kinds of intelligence which human beings display, work on (d) should include some study of human beings, both so that the systems are adequately matched to the people who have to use them, and also because some of the methods used by the human mind may be better than what can be dreamed up by even the cleverest programmers. So (d) needs to be related to (a). The interaction is two-way, e.g. because the observed inadequacy of programs based on theories about how people work may indicate a need to revise the theories.

Both the scientific and the applied work are in their infancy, though the infants seem very robust. In particular, there has recently been an explosion of activity in an applied subfield known as "Expert Systems" or "Knowledge Engineering". Early applications have shown the commercial and industrial potential of expert systems, and attracted considerable funding, which is one of the reasons why the field of IKBS is receiving so much attention. The short term benefits are clear but existing systems are inherently limited and longer-term research and development are required to extend the range of applications. This will require major quantitative and qualitative extensions of the underlying capabilities, for instance to enable systems to combine several kinds of knowledge in order to solve new problems.

In order to understand the scope and limits of this work, and to get a feel for the medium term problems, it is useful to put them in the context of the long term goal of understanding and designing really intelligent systems, described in the next section.

Long term aims of AI: understanding/designing intelligent systems

Intelligent systems designed during the next decade will at best exhibit only a subset of what may be required in the long term. Different subsets may be relevant for different applications. What follows is an attempt to describe, at a very general and abstract level, the union of the kinds of abilities which people in the field of AI have begun to try to understand and replicate. This gives a very rough and provisional characterisation of an intelligent system as one which has some combination of the features listed below. It is perhaps worth stressing that although the list may seem very ambitious, it actually reflects the spread of research which is already in progress, though not all aspects have been pursued to the same depth. Thus, the list represents, from an AI viewpoint, an answer to the question: what are the features of human beings (and some other animals) which make them different from inanimate mechanisms and unintelligent plants and animals, and which are currently hard to explain and replicate.

Having compiled a list of features of intelligent systems, we can then move on to ask what underlying mechanisms or capabilities seem to be required for the production of these features.

NB it must not be assumed that all AI workers would agree with this list, since any general characterisation of intelligence, if possible at all, must be the result of research yet to be done.

Characteristics of intelligent systems: a tentative overview
(The order is not significant.)

(1) Having a general range of abilities, including
(a) the ability to cope with varied objects in a domain
(b) the ability to cope with a variety of domains of objects
(c) the ability to perform a variety of tasks in relation to any object,
(d) the ability to recognise which sub-ability to use.

NOTE: 'object' here is a neutral term, covering such diverse things as physical objects, spoken or written sentences, stories, images, scenes, mathematical problems, social situations, programs, etc. 'Coping' includes such diverse things as perceiving, interpreting, producing, using, acting in relation to, predicting, etc.

(2) Various forms of discovery, learning, or self-improvement, including: qualitative extensions to new domains, new kinds of abilities, and quantitative improvements in speed of performance, complexity of tasks managed, etc. Important special cases include the discovery of new concepts, heuristics or generalisations within a domain, the creation of new domains, and the novel combination of information about several different domains to solve a new class of problems. The more complex examples overlap with what we ordinarily refer to as 'creativity'.

(3) Performing inferences, including not only logical deductions but also reasoning under conditions of uncertainty, non-monotonic reasoning (e.g. making use of implicit assumptions which may be cancelled by new information), reasoning with non-logical representations e.g. maps, diagrams, networks.

(4) Being able to communicate and co-operate with other intelligent systems, especially human beings.

(5) Being able to co-ordinate and control a variety of sensors and manipulators in achieving a task involving physical movement or manipulation.

(6) Coping flexibly with an environment which is not only complex and messy, but also partly unpredictable, partly friendly, partly unfriendly and often fast moving. This includes the ability to interrupt actions and abandon or modify plans when necessary, e.g. to grasp new opportunities or avoid new dangers.

(7) Self-awareness, including the ability to reflect on and communicate about at least some of one's own internal processes. This includes the ability to explain one's actions.

(8) Coping with a multiplicity of "motivators", i.e. goals, general principles, preferences, constraints, etc. which may not all be totally consistent in all possible circumstances. This need can arise either because a single high-level goal can generate a multiplicity of inter-related sub-goals, or because a system has a collection of independent sources of goals, requirements, etc.

(9) The ability to generate, or appreciate, aesthetic objects. This is often thought of as distinct from cognitive abilities, but there are reasons for thinking that aesthetic processes are involved in many cognitive processes, and vice-versa. E.g. elegant proofs not only give pleasure: they generally provide more insight than messy ones.

The notion of intelligence is bound up not only with what can be done, but also with how it is done (i.e. the style, or manner). For example:

(1) When confronted with messy, ill-defined problems and situations, and incomplete or uncertain information; an intelligent system should degrade gracefully as the degree of difficulty/complexity/noise/incompleteness etc. increases, rather than merely 'crashing', or rejecting the problem. Degrading gracefully may involve being slower, less reliable, less general, less accurate, or producing less precise or complete descriptions etc.

(2) Using insight and understanding rather than brute force or blind and mechanical execution of rules, to solve problems achieve goals, etc. E.g. instead of exhaustive trial and error searching there should be selection of alternatives based on some analysis and description of the current state of a problem-solving process. This is closely connected with a requirement for speed and generality.

(3) Plans should not be created simply by applying pre-defined rules for combining primitive actions to achieve some goal, but should rely on the ability to use inference to answer hypothetical questions about 'What would happen if..'. This should also play a role in the ability to make predictions, or test generalisations.

(4) Conflicting goals should not be dealt with simply by means of a pre-assigned set of priority measures, but for example by analysing the reasons for the conflict and making inferences about the consequences of alternative choices or compromises.

These lists are not proposed as a definition of 'intelligence'. The list merely summarises salient aspects of the most intelligent systems we already know, namely (adult?) human beings. It also summarises kinds of AI research already being pursued in a more or less fragmentary fashion.

No existing AI system fulfils even a subset of these criteria, except in very restricted domains, with rather generous interpretations of concepts like 'generality', 'graceful degradation', 'flexibly', etc. Nevertheless there are many examples of fragmentary progress concerned with such varied topics as:

vision, robotics, speech processing, natural language (text) understanding, abstracting and generation, (including the comprehension and generation of stories), machine translation, plan formation, inferring plans from actions, intelligent tutoring systems, problem-solving, non-monotonic reasoning, mathematical theorem proving, automatic programming, automatic debugging, concept learning, game playing, the analysis of music, and of course a growing collection of "expert system" activities such as interpreting mass-spectrographs, medical diagnosis, oil prospecting, fault diagnosis in machines, the planning of computer configurations, etc.

There is no sharp boundary between such work and other fields of computer-science and engineering, and perceived boundaries change as our understanding deepens. For instance compilers capable of accepting algebraic expressions were once thought of as intelligent because previously only human beings had been able to do such things.

Practical and theoretical work in these subfields has begun to indicate some of the underlying enabling features required for systems which have the sorts of intelligent capacities listed above. The design and analysis of such enabling abilities tends to distinguish work in AI from other fields, even though the boundaries are fuzzy and mobile!

What has to be built in to intelligent systems?

By analysing the requirements for intelligence listed above, and especially by looking in great detail at specific examples and trying to model them, we can begin to formulate and test hypotheses about underlying abilities and mechanisms required for intelligence. To illustrate this, I shall present some of the requirements which have emerged so far. The connections with features of intelligent systems listed previously should be evident in most cases.

What follows is at best the tip of an iceberg. It may prove to be an unimportant or misunderstood tip. Further, the list is still somewhat heterogeneous and disorganised.

(1) Rich stores of domain-specific knowledge.
Some early AI researchers hoped to base intelligence on a few algorithms or axioms or general principles producing intelligent or semi-intelligent behaviour. This approach is now discredited. A large amount of knowledge of different kinds seems to be required, for almost any intelligent ability. This includes both general-purpose know-how relevant to different domains, and domain-specific knowledge. The latter includes factual generalisations about the domain, particular facts, and procedural or heuristic knowledge relevant to the domain. It seems that if reasonably fast responses are to be achieved, the stores will often have to be redundant, in that information which could in principle be inferred from other information will be stored explicitly when first inferred, since re-discovering the appropriate derivation will often take too long compared with accessing the store.

(2) Powerful and varied descriptive resources.
Not all knowledge is best expressed in the same way, and in

particular, unlike many computing applications, AI systems cannot make do with collections of numerical measurements or strings of text either for long term storage or for the short term representation of problems, hypotheses, plans, etc. A variety of formalisms and representational structures is needed for storing assertions (including assertions with variables, for instance), condition-action rules, and a variety of special purpose structures, e.g. speech waves, images, maps, diagrams, etc. Procedural information often needs to be stored in such a fashion that the procedures can be both executed and examined, for instance for self-debugging. It is arguable that predicate logic should play a dominant role because of its power and generality, though some would object that it is not rich and powerful enough to capture all the kinds of subtleties which can be expressed in images or natural language. Thus some argue that really intelligent systems will have to make use of natural language for some of their internal representations.

Any claim that ONE formalism is suitable for all purposes would seem to be unfounded at present. In the history of the human intellect the development of a multiplicity of notations has had an enormous intelligence-amplifying role, including, besides natural language, logical notation, arabic numerals, chemical formulae, matrices, tensor calculus, cross-reference tables, musical notations, maps, circuit diagrams, transition nets, pie-charts, histograms, programming languages.... If people need a variety of formalisms, then intelligent machines may do so for the same reasons.

(3) Inference procedures: general and specific.
These are required for extracting information implicit in knowledge stores, such as logical databases, stored sentences, maps, or whatever. Besides general purpose inference mechanisms (e.g. logical deduction) various task-specific and domain-specific mechanisms are required, e.g. for geometrical reasoning, for propagating constraints, for propagating measures of probability or certainty, for solving equations, for inferring the possible consequences of executing strategies, etc. The need for a variety of inference methods follows in part from the need for a variety of formalisms. In addition there is the further possibility of using specialist knowledge about a domain to short-cut inferences in that domain: thus inference itself becomes a process that uses knowledge as opposed to merely being a process which extends knowledge in a knowledge-free manner. Some forms of creativity require procedures for mapping knowledge from one domain onto another, i.e. inferring analogies and new applications for old information. A current research issue is how to combine different types of knowledge, e.g. knowledge of the geometry of a circuit and knowledge of electrical properties of circuit elements, in solving problems.

While logical inference is extremely general and powerful, non-logical inferences are often required, such as those required for inferring a route from a map, or a construction plan from a picture of what is to be constructed.

(4) Self monitoring
Self awareness was described as one of the features characterising intelligent systems. However it is also one of the abilities

underlying some of the other features. E.g. the ability to learn may depend on awareness of features of one's strategies and performances, as does the ability to explain one's actions. Programs which synthesise and debug programs need to be able to relate the static structure of the programs and structure of the execution process. An intelligent system which has to relate to others may have to be able to observe some of its own internal processes in order to interpret some of the behaviour of others. So»e forms of self monitoring may be best implemented using parallel processors. (See below.)

(5) Meta-principles

Although notions like a single general purpose problem solver, or a uniform learning algorithm have been discredited, there does seem to be scope for a variety of types of domain-independent »eta-rules or meta-principles to play a role in intelligent systems. Examples would be rules for preferring plans with few steps to plans with several, rules for searching for bugs in strategies, strategies for generating modifications to partially successful plans, rules for assigning priorities when there are conflicts between goals (e.g. if Gl and G2 can't be pursued simultaneously, but experience shows that pursuing Gl often produces side-effects which subsequently facilitate G2, then do Gl first). It is not clear how large the set of generally useful meta-rules is, nor whether they need to be programmed in from the start or whether they can be learned or derived from still more general principles.

(6) Strategies for controlling search

Often, finding a solution to a problem or performing a task requires exploration of branching sets of possibilities. The combinatorial explosion needs to be constrained by a variety of techniques, including means-ends analysis, heuristic search, use of meta-rules, processing in parallel at different levels of abstraction or in different domains, constraint-propagation (Waltz-filtering, relaxation), changing to a new representation of the problem. Once again a large collection of problem-specific and domain specific techniques seem to be needed. Sometimes, a combinatorial explosion is de-fused by added information. Low level vision is the prime example, where it has been found that at least in good viewing conditions there is far more disambiguating information in natural images than was previously thought, reducing the need for search. Sometimes a total change of technique can, at least in special cases remove the explosion, for instance using relaxation to simulate breadth-first search, and more recent work on highly parallel machines reported in papers by Ballard and Hinton in IJCAI-1981.

Work on such diverse topics as language understanding, vision and plan formation has led to the discovery that it is often useful for a problem to be tackled at different levels of detail in parallel, where the lower-levels are relatively myopic and manipulate minutiae whereas higher-levels enable strategic decisions to be taken about the overall direction of processing, e.g. assessing the relative importance of some of the options available at lower levels, or producing sketchy global solutions to problems and leaving lower levels to sort out details subject to the need to fit the higher level plan. This can enormously reduce the amount of combinatorial searching which would have to be done if processing occurred only at

the lower level, or if the lower levels had to complete their task before higher level structures were examined.

(7) Matching and describing

Many kinds of structures, e.g. 'input' structures, general knowledge structures, and temporary internal structures generated during processing, all need to be matched with other structures as part of the process of analysis, recognition, inference, testing conditions for action. The simplest sort of match is an equality test. In addition it is often useful to be able to match different objects against the same general template, e.g. in applying a general logical rule to particular instances. More generally it is necessary to be able not merely to say that two things don't match, but rather to describe both what they have in common and how they differ. Intelligent systems seem to require a collection of general purpose and special purpose techniques (e.g. unification, parsing, network-matching), some of which may be built on others.

(8) Communication between sub-systems

Where different sub-systems concerned with different tasks use overlapping sets of information, it is often convenient to have databases which are shared between several different sub-processes. These are sometimes referred to as 'blackboards', though normally the term is used when there is only one blackboard in a system, as in the Hearsay speech understanding system. Such databases may merely be passive structures which are accessed by various processes when they need information. Alternatively, some sub-systems may attach active 'demons' to a database. Such demons will monitor changes to (or possibly interrogations of) the database, and in certain circumstances will react by starting a new sub-process, or interrupting an existing process. (More on this below.) Such demons provide one way of implementing self-awareness.

(9) Very large very fast flexible memory stores

If intelligence requires very large amounts of knowledge, capable of being deployed very flexibly, e.g. using sophisticated matching techniques rather than simple comparisons, then that seems to require some sort of fast content-addressable file store, where the contents can be addressed by means of patterns or schemas as well as by simple keys such as are found in current database systems.

(10) Rapid re-organisation of part of memory

Besides long term storage of information of many kinds, an intelligent system will need to be able to represent current percepts, hypotheses, goals, plans, derivations, etc. These will change partly because the environment changes (e.g. perceptual hypotheses), partly because of the intrinsic nature of intelligent processes. This requires a memory mechanism which allows complex temporary structures to be built rapidly, and then discarded after use, making space for new ones. One strategy commonly used for this purpose is garbage collection, though others are possible in principle, e.g. constantly clearing the temporary work space and re-building structures from scratch, which may be better suited e.g. to visual systems.

(11) Parallelism: fine-grain and coarse-grain

Coarse-grain parallelism is required for systems which decompose

into a small number of modules which have to perform different tasks concurrently, e.g. controlling or monitoring motors or sensors. Certain sorts of self-monitoring may be achieved by allowing some processes to examine the data-structures and activation frames of others, which continue independently. Fine-grain parallelism may be useful on a large scale when a process naturally decomposes into a large number of relatively independent sub-processes, e.g. in low-level vision and perhaps some kinds of long-term memory mechanisms. In addition an essentially serial process may be speeded up if each of the steps is made up of a number of sub-steps which are independent, e.g. perhaps unification in a prolog like language. If each main step is composed of n sub-steps on average, then approximately an n-fold speed up can be achieved, in theory. In theory, many kinds of search can be speeded up by the use of parallel processing: e.g. using a breadth-first strategy is guaranteed to find the shortest route to a solution if there is one. However, in practice problem spaces often branch too fast and the combinatorial explosion can re-appear in space instead of time. Some problems may allow a combination of breadth-first and depth- or best-first search to benefit from massive parallelism, though it is still unclear what the range of problems is for which time to find a solution will be significantly improved by such techniques. Moreover, the management of ever growing sets of processes may itself take up time and communication bottlenecks can occur. So it remains an important open question how far intelligent processes can be helped by new architectures with massive parallelism.

Many intelligent processes seem to have a global structure which is inherently serial. E.g. in constructing a long proof, although it is possible to branch forwards from the axioms and backwards from the theorem in parallel, it is not possible to work on all the steps simultaneously, since which steps are required depends on the results of other steps. There is a need for considerable research on such topics. The Japanese fifth generation project seems to be based in part on the assumption that massive parallelism at a low level can always be relied on to produce a massive increase in speed.

(12) Interrupt mechanisms

These seem to be required for systems which act in complex and partly unpredictable environments, where rapid action may sometimes be called for. The use of interrupts can also aid modularity: a relatively simple general purpose strategy may be used in a variety of different circumstances, if a monitoring process is able to interrupt it from time to time and make adjustments. Using different monitoring processes with different tasks and circumstances makes it unnecessary to clutter up the general strategy with large numbers of conditional instructions. (Homeostatic systems can be seen as a limiting case of this??) When parallelism is used to explore alternative routes to solving a problem, it will be necessary to interrupt the still unsuccessful processes when a solution has been found. Plan execution systems operating in a complex and partly unpredictable environment may need to be interruptable by monitoring processes analysing sensory information. The alternative to allowing interrupts is constant polling, which is acceptable when not too many different kinds of monitoring are needed and reactions don't have to be very fast. However, the polling alternative can lead to non-modular, messy programs, whose only advantage is that it may be

easier to prove theorems about them!

If the processes which can generate interrupts can vary in importance, and the processes they interrupt can vary in importance, the interrupt mechanisms may have to use different 'priority' levels. In the simplest cases this may rely on pre-assigned numerical priorities, but in more complex cases the decision about which sorts of interrupts to accept and which to suppress »ay itself have to rely on knowledge based inferences.

EliiilUS £MJ22L§ £j2Ü§t!}!=L£ §2f§JBElSL§ 2LL architectures

Some fragmentary versions of the sorts of abilities characterised above have been assembled into working programs. A few principles and strategies for organising knowledge-based programs have begun to emerge, though all are still controversial, and at best relevant only to the design of relatively simple systems. What follows is a somewhat disorganised collection of examples. There may be some underlying general architectural principles waiting to be formulated, but so far they have eluded me.

* 2j*£ji!l§Il§ El!i§ i2£§L§Q£§ engine

One sort of architecture, which works well for relatively simple problems, is to have a database of facts and rules, together with an inference engine' which repeatedly selects items from the database and uses them to make changes to the database. In some cases the processing is 'data-driven' in that any item which can be a basis for an inference is capable of being selected to generate a change. In other cases the processing is 'goal-driven[1], i.e. besides the facts and rules there is a collection of unsatisfied goa3s and only items which relate to these goals can be selected to generate new actions (including creation of new goals).

Examples of these 'data-base plus inference-engine' architectures include production-systems and prolog, though each can be used as a virtual machine on which to construct more complex virtual machines.

More complex examples associate numerical probabilities or likelihood values with database items, and include in the inference engine algorithms for propagating such values.

Such systems are useful where domain-specific knowledge can be broken down into a collection of relatively isolated chunks which can be combined in different ways for different purposes.

Although such systems have the advantage of simplicity and modularity, they can be hard to understand and control when the databases grow large. Their very modularity implies that potentially anything can interact with anything else, and preventing unwanted interactions can involve the use of ad hoc tricks. In some cases it might be better for designers to think in terms of higher level organising principles. Research is needed on good design aids to facilitate this.

The status of the numerical representations of uncertainty and the algorithms for propagating them is often very unclear. It »ay often be better to represent alternative sets of hypotheses explicitly, and use

explicit domain-specific meta-rules for selecting between them, or at least determining partial orderings, rather than ill-defined quantitative concepts and dubious general-purpose algorithms.

Further, these systems are intrinsically suited only to types of processing which operate only at one level, whereas we have already seen that sometimes it is desirable that processing at different levels of abstraction should go on in parallel.

Despite these critical comments, useful systems have been built using the simpler models, some of which have proved very profitable investments. In some cases the designers have subsequently abstracted 'shells': i.e. collections of domain independent tools for building expert systems, often including a formalism, algorithms for propagating uncertainty measures, mechanisms for generating explanations, a rule editor, and a collection of debugging aids.

* §:~LH§£§1 RI2Lff i>ff§.: blackboards and actors

Another popular strategy, mentioned previously, is to have collections of active modules, possibly with different internal architectures, communicating through a global 'blackboard', i.e. a globally accessible database. This is one way of implementing systems in which processing at different levels proceeds in parallel, as in Hearsay.

An objection to this is that often the blackboard is effectively partitioned by the formats of the assertions stored there, and therefore, for some systems, it may be both simpler and also far more efficient instead to use a mechanism for broadcasting messages to a named class of modules (where the members of the class need not be known individually to the sending process). Instead of one central bottleneck, there is then one bottleneck per class of recipients, which can enhance opportunities for parallelism.

An extreme version of the latter is Hewitt's 'actor' architecture, also found in SMALLTALK, where each module essentially has a fixed set of acquaintances to which it can send messages. However, such architectures are so general that it is hard to say anything useful about them.

* iJl§L§L£ky §M heterarchy

During the late 1960[f]s and early 1970[f]s it became fashionable to criticise programs which had the following organisation:

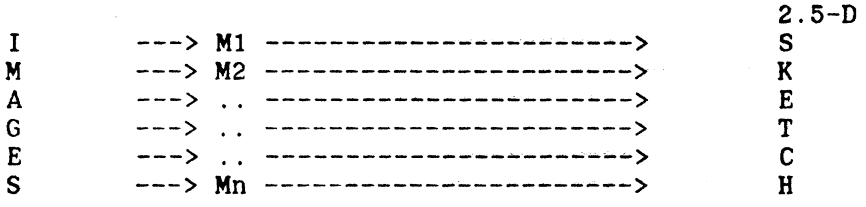
input -> Ml -> M2 -> M3 ->. . . . .-> Mn -> output

where each module waited until the previous module had completed its transformation of the data. For instance, Ml might be an edge-detector, M2 a line-finder, M3 an image segmenter, M4 a 2-D to 3-D interpretation program. Or Ml might find phonemes in speech, M2 might find words, M3 might do syntactic analysis, M4 semantic interpretation, etc. Instead of this hierarchic organisation programmers attempted to allow control to flow between modules in a much less rigid fashion, for instance allowing higher level knowledge to be used to disambiguate some of the data at a lower level, and thereby reducing the total search space. This alternative program organisation was referred to as 'heterarchy[1]. Unfortunately, such programs proved very hard to control. Moreover, when they worked it was difficult to understand why, or to analyse the conditions under which they would or would not work. One reaction (see below) was to claim that ambiguity at low levels could be eliminated, or

at least reduced, by doing more elaborate low-level processing, based on a good theory as to the nature and origins of the input data. Much recent work on vision has taken this line. Another approach has been to try to achieve better designs by thinking of sub-processes as running in parallel on separate machines, rather than as competing for control on a single serial processor. Where each process has its own processor, and a well-defined interface to the rest of the system, and no one module can gain control and hog the whole machine, it is in principle easier to analyse the behaviour of the system. For instance, termination under all circumstances can be proved if one module has the power to send an abort signal to all the others if they have not solved the problem within a certain time.

* Architectures for vision: low levels

Recent work in vision reported in the vision tutorial suggests that at the low levels the required architecture may take the form:



In other words, a collection of modules of different sorts, with their own internal architectures and their own processors, take information from the retinal array(s) and store new information in a common database of 'visible surface descriptions', sometimes called a 2.5 D sketch. Higher-level processes would operate on this database to produce a collection of different databases based on a variety of reference frames. It is an open question whether something like the above organisation is useful for lower levels of speech. In fact it may turn out to be a form of architecture useful for a variety of problems. Where there are classes of problems to which many forms of knowledge are potentially relevant, it may be useful to allow a collection of different modules all to study a problem in parallel, and store whatever cues or clues they find in a shared database (a 'blackboard'). Another module could then be examining the database for patterns suggesting a good strategy to solve the problem.

In the case of vision, and perhaps speech understanding, it is worth noting that the modules which find and interpret low level features, e.g. finding edges or inferring shape from shading or depth from binocular disparity, may have a great deal of implicit knowledge about physics and geometry compiled into them in a procedural form. In principle it would be possible to express all the knowledge explicitly in general rules and assertions. However, the resulting system would run far too slowly, at least using current implementation techniques. It has been suggested that some of the algorithms are compiled down into the 'wiring' in animal visual systems, and that the same will be required for artificial visual systems with a speed constraint. However this does not in principle rule out the specification of such systems in (e.g.) a logical form. The logical form would not be used at run-time, however.

* Architecture of an intelligent perceiver?
So far there has been little study of the way in which a visual system should relate to the rest of an intelligent agent. It is arguable that there should be a variety of different routes between different sub-databases of the visual system and non-visual modules (e.g. hearing, touch, planning, problem-solving, control of movement, etc.), but this is a wide-open research area. Similar questions arise about speech: besides the recognition of words and syntactic structures it seems clear that many other features of speech can play an important role in communication, or in making inferences about the speaker. For instance, pitch, intonation contours, stress patterns, speed, etc. may all be relevant. So again there may be a need for many routes from low level as well as high level modules of a speech understanding system to other modules in an intelligent understander.

Even above the phonetic level the same considerations may apply: there are many features of style or communicative strategy besides those which contribute to literal meaning. Detection of such features may have to proceed in parallel with such things as word-recognition and parsing or semantic analysis, and may feed into a variety of higher level processes.

Little is known about the possible architectures for the higher levels of vision or any other perceptual system, though there is much exploratory work. There is a lot of work on strategies for using general, or domain-specific, visual knowledge to guide higher-level interpretation processes.

* Architectures of language understanders
There is a wide variety of approaches to the design of language understanding systems. Language understanding, like vision, often involves the problem of interpreting ambiguous fragments, subject to global consistency constraints and there are many different approaches to this including simple depth-first search with backtracking, and parallel breadth-first search.

The architectures vary so much as to defy any brief overview. For instance some do syntactic parsing as a conceptually distinct process from inferring meaning, whereas others make no such distinction. Those which do include parsing may either have a complete initial parsing process before semantic interpretation begins, or else may (like Winograd) interleave the two in order to constrain the combinatorics of syntactic ambiguity. Another distinction is between parsers which separate an explicit 'declarative' representation of a grammar from the procedures which operate on the grammar and the input sentence, and those which use only a procedural representation of the grammar, which operates directly on the input sentence. A different dimension again involves the distinction between parsers which work top down, those which are bottom up, and those which combine top-down and bottom up processing. Further distinctions can be made according to the degree of parallelism employed.

The distinction between parsers with an explicit grammar and those without becomes blurred again when we consider that some of the former don't have a procedure to interpret the grammar, but instead compile it into a collection of procedures which are then run.

Another distinction concerns the extent to which non-linguistic knowledge, e.g. knowledge about the physical environment, or other particular features of the context, enters directly into the process of understanding a sentence, as opposed to being invoked only after a representation of the meaning has been derived solely on the basis of syntactic and general semantic knowledge.

* 'Local' constraint-propagation
Constraint-propagation mechanisms have attracted a lot of attention as devices for controlling the attempt to resolve ambiguous data, subject to the requirement of global consistency, or global optimality where some consistency has to be sacrificed. Waltz filtering is the very simplest case, where each ambiguous fragment looks at possible interpretations of its neighbours and eliminates those of its own interpretations which are not compatible with any neighbouring one. But this does not cope with propagation of constraints involving inequalities, or relations between several nodes of a network. It can also lead to 'gangrene' in the case of noise, occlusion, etc., as Hinton pointed out in 1976. I.e. if a possible interpretation is eliminated wrongly as a result of noise, etc, the effect can be eventually to eliminate all interpretations.

A slight generalisation of this is to use relaxation, in which alternative hypotheses, instead of being completely eliminated acquire a lower 'credibility' value if inconsistent with their neighbours. In some cases the process of repeatedly allowing nodes to raise or lower the credibilities of other nodes linked to them by constraints, will converge to a single stable solution.

* Connection machines
Related forms of computation are being investigated in which instead of using definite chunks of memory to store items of information, the system represents information in the form of a pattern of activation of a collection of units, and different items of information may be stored by superimposing different activation patterns. Inference then occurs by allowing patterns of activation to spread using excitatory and inhibitory links. These sorts of architectures may not be well suited to all sorts of problems. E.g. they may be very good for perceiving or interpreting structures or patterns embedded in large collections of information, as in speech or vision and some forms of problem solving. But they may be less well suited to deep chains of reasoning such as those required for proving mathematical theorems.

* Complete intelligent systems?
Probably the nearest things to a complete intelligent system are mobile robots. However current robots are architecturally very crude. Simple robot plan-formation systems exist which essentially have a library of plans, a collection of plan-constructing techniques, a factual knowledge base, and an inference engine which can be used by the plan-constructor. When presented with a 'goal' the plan-constructor makes a plan, possibly invoking a variety of sub-modules in the process. Plan formation and evaluation may require the ability to simulate the execution of the plan in a hypothetical world. (This influenced the development of back-tracking languages, context mechanisms, etc.) For executing the plans in the real world, a plan executing module is required which takes a (possibly incomplete) plan description and arranges for appropriate signals to be sent to motor sub-system(s). Some kind of visual or

tactile monitoring subsystem operates in parallel with all this, feeding information to the factual knowledge base. Then depending on whether speed is crucial, either the plan executor can repeatedly poll the knowledge base to ensure that the plan is still appropriate, or else a 'demon mechanism' can be used to monitor it in parallel with execution and interrupt when necessary.

At that stage a new module or collection of modules may have to be invoked to analyse the situation and decide whether to continue or revise the plan, and in the latter case call the plan constructor (or perhaps a special plan-modifier) to produce the new plan.

When there is not just a single top-level goal but a more complex system of 'motivators' including several short and long term goals, general principles of ethics, prudence, etc., then things can become far more complicated, especially if the environment permits both dangerous and fast-moving situations. However, as far as I know, nobody has begun to build an intelligent system of this sort, though unintelligent versions are to be found in a variety of automatic control systems.

This brief and sketchy survey of architectures indicates the variety of forms which have so far been investigated. At the most abstract level it is easy to give a general characterisation of such architectures in terms of composition of modules using standard concepts from computer science. E.g. there are procedure-calling hierarchies, pipe-lines, concurrent processes. There are different kinds of control hierarchies, schedulers, interrupt mechanisms. There is a variety of types of communication between modules, e.g. procedure arguments and results, a global database, dedicated message channels, etc. These general-purpose architectural concepts may be applied in describing either the fine grain, or the more global features of intelligent systems.

But the only general characterisations which seem to be applicable are not specific to intelligent systems: rather the same architectural concepts can be illustrated in other computing applications. This is not to say that there is nothing general to be said about intelligent systems, only that no clear pattern has emerged so far. (Perhaps someone should attempt a comparative survey of different sorts of complex systems constructed in different branches of computing. This may help to highlight distinctive features of intelligent systems, if there are any.)

Further work might perhaps attempt to relate architectures to types of computational functions, for instance distinguishing functions of the following sorts and their architectural requirements:

(a) systems which merely take a decision (e.g. recognition nets)

(b) systems which analyse complex structures (e.g. sentences)

(c) systems which synthesise complex structures (e.g. plans)

(d) systems which do both (e.g. generating parse trees or proofs)

(e) systems which continuously monitor a stream of incoming information

(f) systems which continuously control changing objects

lHl£lii£§Sl H£2wLedge Based Systems

So far I have not attempted to separate IKBS from AI. IKBS **have** been defined as:

> 'semi-intelligent systems for carrying out a single complex task. This implies working with a large, incomplete, uncertain and rapidly changing knowledge store, use of inferential procedures for applying this knowledge in reacting to variegated and unreliable inputs in a changing environment, and the use of sophisticated and flexible control mechanisms'.

Depending on which words in this definition are stressed, this can sound more or less ambitious. I don't believe there are any major boundaries between the simplest rule-based systems and the most complex intelligent systems. Rather there are very many different boundaries: the space of possibilities has many discontinuities. So perhaps the best way to interpret this definition, in the context in which it was produced, is to regard IKBS as those systems which are difficult but not impossible to design within the next five to ten years, and which may potentially have some practical use. The field of IKBS design could therefore be described as 'Applied AI'.

All of the points I have listed under long term aims are potentially capable of playing a role in such designs.

WHITHER NOW?

The l5LE2Xt§ILce of domain sgec.ifj>c knowledge

The most difficult type of task in designing an intelligent system is usually finding out what the domain specific knowledge is that needs to be deployed in the system. By contrast designing new languages, shells, virtual machines, is comparatively easy. Curiously, the hardest kinds of knowledge to articulate in a useable form are those which are shared by all ordinary people, not the esoteric knowledge of experts in science, medicine, chess, or whatever. How the meanings of sentences in ordinary language are inferred and how they are represented, how ordinary vision works, what it is that we know about spatial and physical properties of objects that enables us to manipulate them in every day life – these familiar and widely shared, yet strangely inaccessible forms of knowledge need to be actively explored if we are to make good use of the potential of computers in the next decade or two. But we can't assume that such research will produce commercial payoffs in the near future. Thus provision should be made to support such research independently of applications-oriented projects.

ZH£iJ}ff developments

All the systems and architectures mentioned so far have known weaknesses. For instance the simple system of rules plus inference engine used in 'expert systems' seems to work (so far) only for very narrow domains, where it is possible to get by with **very** shallow knowledge in the form of rules which more or less **exhaustively** cover possible situations. Coping with more complex tasks, **for** instance diagnosing a whole variety of computer faults, requires the **ability** to creatively combine different types of knowledge of the **function of** the system, the structure of the system, the functions and structures of components, and the relationships between such things as electrical, mechanical, and geometrical properties of parts.

We do not yet know how to represent much of this knowledge (e.g. spatial relationships are in general hard to represent in a useful way, except for mathematically simple structures). We don't know what the knowledge is that needs to be represented. And we don't know how to arrange for intelligent control of the deployment of such knowledge, e.g. deciding when to combine different sorts of information about a machine, or deciding at what level of detail to think about it. Projects which address the issue of how to combine different kinds of knowledge are therefore worth supporting.

There are many other issues on which research is desirable. Some of them have to do with rather general problems like how to reason with uncertain information, or how to represent complex networks of causal relations. Some have to do with the need for far more domain-specific knowledge (e.g. knowJedge of the syntax and semantics of English, knowledge about the structures and behaviour of physical objects, knowledge of anatomy physiology and pharmacology, etc.) Some have to do with the design of good representations for a task. Some have to do with how to make inferences using particular sorts of representation. Some have to do with how to control inferences. Some have to do with ways of putting the various sorts of components together into complete working systems. All these issues require considerable theoretical and practical study.

£h8oi g EI2Jl§ct_s_ to support

There are two very different strategies the SERC or Alvey directorate might adopt in planning a directed national research programme. One is to select particular application areas and invite bids for funds to work on these areas. Another is to identify a collection of research issues and invite people to propose projects which address those issues. The two strategies might both be adopted. I propose a concentration on the latter, for the following reasons.

There are many different kinds of 'demonstrator' projects which would provide a good framework for addressing important research issues. Among candidates which have already been proposed are projects concerned with intelligent database managers, projects concerned with trying to combine different visual modules in a working system to guide a mobile vehicle, projects concerned with developing intelligent software tools, projects concerned with developing one or more expert consultants, projects concerned with computer-based education, projects concerned with various medical problems, etc.

Since so many different kinds of projects are all capable of addressing

important research issues, and leading to useful applications, there will be no easy way of deciding in advance which are to be specially encouraged, unless a considerable amount of research is done to assess the potential usefulness of the alternatives. This might involve, for example, attempting to weigh up the usefulness of intelligent teaching aids in schools and the home, against the usefulness of intelligent advisers for doctors. Any such evaluation if done properly would have to be an elaborate exercise, consulting potential users, costing the different kinds of benefits, assessing the chances of success, etc. Moreover, the assessment would necessarily involve making complex ethical or political judgements about the best way to spend tax-payers money. For these reasons, it would not be appropriate for a small study such as this to select application areas to concentrate on.

I would propose therefore that instead of selecting a particular set of application areas, the SERC should spell out the kinds of general problems for IKBS design which are worth addressing, and require people who submit proposals to indicate which such problems they are addressing. In addition, people may be asked to give evidence of the usefulness or even commercial exploitability of the tasks they are undertaking.

This is not an attempt to restrict the programme to fundamental research. On the contrary, there are many issues which are of great theoretical interest yet are both suitable for relatively short term research, and relevant to potential applications. Some are domain-specific, such as issues concerning the kinds of knowledge of programming required by an intelligent debugging aid. Others are more general, such as issues concerning how to represent and make inferences about uncertainty.

I therefore propose that a central task for stage three of the study should be to create the list of issues concerning the design of IKBS which are generally thought to be important, and to use them as the basis for formulating a partial set of criteria for assessing demonstrator projects.

NOTE:

This proposal was accepted by the workshop, and I was asked to produce a paper listing research issues, in consultation with other members of the study group. This was done (though with undue haste) and the paper, entitled 'IKBS Research Issues' included in the annexe to Volume 1 of the final report'. A slightly modified version is available as a Sussex University Cognitive Science Research Paper.

Acknowledgements

Early drafts of this paper were circulated to members of the IKBS architecture study and some of my graduate students. The following made helpful suggestions for improvements: Alan Bundy, Bob Kowalski, Guy Scott, Bob Searle, Ronan Sleep, Karen Sparck Jones.

APPENDIX
The role of logic

Despite comments about logic programming above it should be noted that predicate logic provides the most generally applicable formalism and inference system known so far. Although there is plenty of evidence that human beings, despite their great intelligence, are not very good at doing logical inference unless specially trained, it may still turn out that logic will play an important role in the design of artificial intelligent systems.

However, it is important to distinguish a number of different uses of logic.

(a) Logic can be used by theorists analysing the problems of designing an intelligent system. E.g. they may find it useful to formulate axioms characterising the nature of the task, and prove theorems about the properties of certain strategies, or certain representations. There is no doubt that this is an important role for logic, though it must not be forgotten that other notations and inference methods will always have a role. For instance, nobody would tolerate having to use logical notation for doing long multiplication: instead we prefer the analogical properties of the arabic numeral system, where spatial relations represent numerical relations.

(b) Logic may be used as a specification language for designing intelligent systems. This may prove to be a very important addition to the current collection of programming formalisms, especially when intelligent programming systems have been constructed which can directly interpret such specifications. Even before then, the use of logic may be a considerable aid to clarity of thought, and may facilitate co-operation between teams of designers. However, this is the use of logic discussed previously. Moreover, it seems clear that for some applications, alternative representations will be more convenient and less prone to errors. To take a simple example, someone designing an interactive system using graphical communication may find it very easy to characterise a hexagonal spiral pattern he wishes to be drawn, if he can write something like

```
repeat 120 times                    draw(length); turn(60);
length := length + increment        endrepeat;
```

Better still if he can use graphical input to characterise the pattern. There is no doubt that such things can be expressed in logic. But it seems pointless to try to force people always to use the logical notation when others are simple, clear and reliable. This is also an example of a type of program which is not amenable

R. Davis fExpert systems: where are we? and where do we go from here?' MIT AI Memo No 665 June 1982 (expanded version of IJCAI-1981 paper)

K.S. Fu, Syntactic methods of fiattern recognition (????) **(1982)**

K. Fuchi 'Aiming for knowledge information processing **systems**[1]

S. Hardy, 'Toward more natural programming languages' **1982** (in preparation)

ICOT 'Outline of research and development plans for fifth generation computer systems'

G.Hinton 'Using relaxation to find a puppet' In AISB Conference Proceedings, Edinburgh 1976.

G. Hinton, papers in Proceedings 7th IJCAI, Vancouver 1981 S. Isard, 'Tutorial paper on speech for SERC IKBS study' 1982

J. Mayhew 'Tutorial paper on vision for SERC IKBS study' 1982

C.S. Mellish and S. Hardy 'Integrating Prolog in the POPLOG Environment' (in preparation) 1983.

D. Michie Consultative documents

D. Michie (ed) Expert Sj^tjems rn the Micro-Electronic Age Edinburgh Univ Press 1979.

A. SIoman The Computer Revorutj^on rn PhUosojDhj;: Philosophy Science and Moi ei s 2l M M ' Harvester Press 1978.

A. Sloman 'The way ahead in image processing?' in Proceedings conference **9H Eilllica], and biological AffijiS £I2££S§lQ£ e(*« °- Braddick and A.** Sleigh, 1982. [Forthcoming]

K. Sparck Jones 'Intelligent Knowledge Based System: Papers for the Alvey Committee', 1982

G. Sussman A cjoniEiltational model of skHl, acquisition American Elsevier/North Holland 1975

Consultative papers circulated to group leaders

to parallel execution, especially if its output is made less predictable, e.g. by using commands like: draw(length + random(delta)) turn(60 + random(delta))

(c) The ability to construct and manipulate logical formulae may be built in to an intelligent system, for storing information and making inferences. Because of the power and generality of logic, it is clear that this is likely to be very useful. (E.g. it played a central role in the STRIPS problem solver.) However, it also seems clear that for many applications where speed is important and complex search spaces have to be cut down using domain-specific heuristics, it will be useful to make use of specialised inference strategies tailored to the domain, e.g. building specialist arithmetic procedures into the system rather than just axioms for arithmetic plus general logical rules. Of course, the use of specialised inference strategies may be specified by the designer using logic, but that use of logic is the one discussed previously.

(d) New machines may be constructed whose machine language is logic, in the sense that the most primitive symbols available to the programmer use logical formulas and the most primitive processes are logical inference procedures. An example would be a machine whose basic language was something like Prolog. There is no doubt that for many applications such a machine would be very useful. However, as with existing Prolog systems it is already clear that such a machine will have to have various additional features built in, at the bottom level e.g. arithmetic abilities. And of course extra-logical primitives would have to be built in for inter-process communication, file handling, etc. As a component of a larger system such machines could play a central role in the development of IKBS. However it is far from obvious that all sub-systems are best programmed in this fashion. In particular some of the operations required for low-level vision and speech might be very slow if implemented on such a machine (though there is no doubt that they could be so implemented).

Further some recent work on parallel neural-net like systems, cited above and in previously circulated papers may prove very important in the long run, and not only for low-level vision. These systems have little in common with logic machines. However, there is nothing to stop us describing them using logic, and trying to analyse their properties or prove theorems about them. This will again be an example of the first use of logic.

References (INCOMPLETE)

Hideo Aiso 'Fifth Generation Computer Architecture'

D. Bobrow and D. Norman 'Some principles of memory schemata' in Bobrow and Collins Representation and Understanding

D. Ballard, papers in Proceedings 7th IJCAI, Vancouver 1981

B.G. Buchanan and R.O. Duda 'Principles of Rule-Based Expert Systems', Report STAN.CS.82.926, Stanford Dept of Computer Science, 1982.