Graeme

9/2/88

Knowledge Representation and Inference Two

Lecture Five:

Three Levels for Describing Knowledge-Based Systems

and

Blackboard Systems

1 Introductory Comments

These notes are divided into two parts. The second part covers Blackboard Systems. The first part introduces a three level scheme for describing and talking about knowledge-based systems in general. It is presented, not because it only applies to Blackboard Systems, but because it applies to all types of knowledge-based systems and this is the first lecture in which a knowledge-based system architecture is discussed.

2 Three Levels for Describing Knowledge-Based Systems

2.1 Introduction

Knowledge-based systems, as with any type of system, AI-based or otherwise, can be viewed in different ways. These different views are useful in talking about different aspects of a system and enable important abstractions and distinctions to be maintained when describing them. Two views, or *levels* as they are called here, commonly used when talking about many types of computer-based systems are:

- The **Symbol Level** which focuses on the data structures and algorithms used by the system to perform in the way required. In knowledge-based systems, for example, Symbol Level issues might include access locking techniques, parallelism, inheritance hierarchies, forward and backward chaining inferenceing, storage mechanisms, and general system management and control problems. In other words, the Symbol Level deals with how a *machine* views and processes the information maintained in a knowledge-base.

- The **System Engineering Level** which focuses on a system from the point of view of a system designer and builder. It is concerned with what a designer needs to cope with, the structural complexity of large knowledge-based systems in our case. Techniques which might be used at this level include: successive refinement by specialisation, part or whole decomposition, version management, exception handling, and information abstraction mechanisms. In other words the System Engineering Level deals with the organisational aspects of designing and building systems and how *people* view and process the information in a system.

A third level which comes between the symbol level and system engineering level has been proposed by Brachman and Levesque for knowledge-based systems [BL86b].

- This third level is called the **Knowledge Level**. It is concerned with what knowledge is represented in a knowledge-base, or knowledge-based system, or what is representable in a system. At this level knowledge is treated as an abstract commodity, quite independent of the Symbol Level concerns of efficiency and integrity on the one hand, and the System Engineering Level concerns of understandability and maintainability on the other. In other words the Knowledge Level presents a view of the *information content* of a knowledge-based system without considering how that content might be dealt with either by machines or people.

The Knowledge Level view of a knowledge-based system is central to an understanding what the system is intended to do. The Knowledge Level account of system is thus *primary*, the Symbol Level and System Engineering Level accounts must play a supporting role. It is at the Knowledge Level that it must be made clear what information about the domain of a system is explicit and implicit in the knowledge-base. For different representation and inferencing schemes that might be used to construct a knowledge-based system more or less deductive activity will be necessary in providing an item of information. The amount of reasoning that has to be done in using particular schemes provides a useful metric for comparing schemes at the Knowledge Level since there is a tradeoff between how expressive a representation scheme is and how computationally tractable the associated reasoning is. For example, it is not a good idea to combine inferencing mechanisms, such as inheritance and productions rules, without first considering the Knowledge Level consequences.

2.2 The Use of Knowledge Level Descriptions

The basic idea of developing a Knowledge Level description of a knowledge-based system is an example of the use of the ideas presented by Alan Newell in his paper *The Knowledge Level* [New81]. The following subsections present some of the ways in which Newell's ideas can be used in developing a functional approach to the Knowledge Level description of knowledge-based systems.

2.2.1 Competence

An important characteristic of the Knowledge Level is that knowledge is considered to be a *competence* notion, "being a potential for generating actions", as Newell puts it. There are two ways of looking at this. First, we might say, if an agent believes p, and if p logically implies q, then the agent is likely to believe q — though not necessarily. Competence might thus be thought of as a plausible abstraction for reasoning about beliefs of an agent. Secondly, we might say, if an agent imagines the world to be in one in which p is true, and if p entails q, then it imagines the world to be one where q also

happens to be true. In other words, if the world the agent believes in satisfies p, then it must also satisfy q. The notion of competence need not be taken to mean that an agent appreciates the consequences of what it knows, but merely one where *we* examine those consequences.

The analysis of knowledge at the Knowledge Level is the study of what is *implicit* in what an agent holds as true in its knowledge-base, which is precisely the domain of *logic*. As Newell says: "Just as talking of *programmerless* programming violates truth in packaging, so does talking of *non-logical analysis* of knowledge". This does not imply that agents do, can, or should use some form of logical calculus as a representation scheme, but only that the analysis of what is being represented is best carried out in a logical framework. Again as Newell puts it, "knowledge is to be characterized *functionally*, in terms of what it does, not *structurally* in terms of physical objects with particular properties and relations".

2.2.2 Functionality

Newell's view of knowledge at the Knowledge Level is similar to the standard notion of an abstract data type: to specify what is required of a desired entity, or collection of related entities, and to specify the desired behaviour under a set of operations, but not the structures and algorithms that might be used to realise that behavior — in other words not the Symbol Level aspects. Levesque [LB85] has further developed these ideas in order to produce a formal foundation for a functional approach to knowledge representation.

2.2.3 Comparison of Knowledge-Based Systems

Most descriptions and surveys of knowledge-based systems to be found in the literature focus on the *mechanisms* and *techniques* used, such as inheritance hierarchies and cancellation, or frames and active slot values, or productions rules and forward and/or backward chaining. This makes it difficult to compare different systems. The Knowledge Level description of a knowledge-based system provides a way of comparing across different systems using the same, similar, or different Symbol Level techniques, and constructed using the same, similar, or different System Engineering Level methods.

Brachman and Levesque [BL86a] propose a scheme whereby knowledge-based systems, and database systems, are thought of as ways to express information about a domain, information that can be understood as declarative sentences. By expressing such sentences in a sufficiently general language, such as first-order logic, comparisions can be made across different knowledge-based systems by asking what *range* of sentences in the language can be expressed. Issues of organisation, structuring, and abstraction, are thus made irrelevant; all that counts is what sentences about the domain can or cannot formed.

2.2.4 Hybrid Systems

Another issue which occurs at the Knowledge Level involves the different *kinds* of knowledge that a system might need. For example, a simple fact such as *Skye is an island* (a declaratively stated fact) clearly interacts with a definition such as *An island is a body of land completely surrounded by water* (a definitional fact). This raises questions about how different kinds of knowledge used in a system affect one another. In general knowledge-based systems will have to use a *hybrid* approach to the kinds of knowledge used since no one single kind will typically suffice. It is at the Knowledge Level that the important interrelation of the different kinds of knowledge must be sorted out.

2.3 Some Comments

It is easy to forget that a representational scheme used in a knowledge-based system is first and foremost *representational*, and that consideration of how to tailor a scheme either to machines, at the Symbol Level, or to people, at the System Engineering Level, must be preceded by a good understanding at the Knowledge Level of what the system can or cannot represent.

A serious and continuing criticism that can be made of knowledge-based systems work, both their research and their engineering, is that little attempt has been made to adopt a disciplined approach to the description, comparison, and evaluation of the different techniques and methods currently available above the Symbol Level. As a result it is hard to say what progress has been made in the subject of Knowledge Representation and Inference. The ideas of Newell, Brachman, and Levesque, and those of others who have been developing similar formal approaches to Knowledge Level issues, represent an important development in the subject of knowledge representation and inference.

3 Blackboard Systems

The Blackboard Model [Nii86], which is an example of a knowledge-based systems architecture for a particular kind of problem solving, will now be presented.

3.1 Some Historical Remarks

Historically, the Blackboard model arose from abstracting features of the HEARSAY-II speech-understanding system developed between 1971 and 1976. HEARSAY-II understood a spoken speech query about computer science abstracts stored in a database. It *understood* in the sense that it could respond to spoken commands and queries about the contents of the database. The second example of a system based upon the Blackboard model was the HASP system. The domain of the HASP system was ocean surveillance, and its task was the interpretation of continuous passive sonar data. As the second example it not only added credibility to the claim that the Balckboard model provided a general approach to problem-solving, but it also demonstrated that it could be abstracted into a robust model of problem-solving. Since HEARSAY-II and HASP many other systems have been built using the Blackboard model of problem-solving.

The first use to the term Blackboard in AI seems to have been made by Alan Newell in the following quote:

> Metaphorically we can think of a set of workers, all looking at the same blackboard: each is able to read everything that is on it, and to judge when he has something worthwhile to add to it, [New62].

Newell was concerned with the organisational problems of the types of problem-solving programs which existed at the time, such as, checker-playing programs, chess-playing programs, and theorem-proving programs, most of which were based upon the generate-and-test search model. These programs suffered from *rigidity* as Newell called it. The scheme that he went on to propose for building more flexible systems led to the development of what we now call production systems, and the OPS language systems.

In a paper published in 1966 (and later published in [Sim77]), Simon mentions the term blackboard in a slightly different context from that of the Newell quote. Simon was writing about an information-processing theory for the discovery and development of ideas:

> In the typical organisation of a problem-solving program, the solution efforts are guided and controlled by a hierarchy or *tree* of goals and subgoals. Thus, the subject starts out with the goal of solving the original problem. In trying to reach this goal, he generates a subgoal ... If the subgoal is achieved, he may then turn to the now modified original goal. If difficulties arise in achieving the subgoal, sub-subgoals may be created to deal with them ... we would specify that the goal tree be held in some kind of temporary memory, since it is a dynamic structure, whose function is to guide search, and it is not needed when the problem solution has been found ... In addition, the problem solver is noticing various features of the problem environment and is storing some of these in memory ... What use is made of [a feature] at the time it is noted depends on what subgoal is directing attention at the moment ... over the longer run, this information influences the growth of the subgoal tree ... I will call the information about the task environment that is noticed in the course of problem solution and fixated in permanent (or relatively long-term) memory the *blackboard*.

Although Newell's and Simon's concerns derive from different contexts, the problem-solving method they were using was the goal-directed generate-and-test search method. They encountered two common problems in using this method: the need for previously generated information during problem solving and lack of flexible control. It was Simon who proposed the Blackboard idea to Raj Reddy and Lee Erman for use in the HEARSAY-II project.

3.2 Uncertainty Again

A *problem-solving model* is a scheme for organising the reasoning steps and domain knowledge required to construct a solution to a particular kind of problem. In lecture three on uncertainty a number of schemes were introduced for explicitly representing and reasoning about the uncertainty of the domain knowledge. However, it is not necessary to explicitly represent the uncertainty about the state of knowledge in a system; the uncertainty about its beliefs; much of a systems uncertainty about its domain, or world,

can be resolved by adopting an appropriate control strategy, or problem-solving scheme. By exploiting properties of an uncertain domain in the design of a problem-solving scheme, the effects of uncertainty can be reduced. For example, consider the possible strategies for solving a jigsaw puzzle: you can work on *anything next*, or you can attempt to build the frame of the puzzle first — exploiting the fact that frame pieces have at least one straight edge. Having completed the frame you can again work on *anything next*, or you can search for pieces which extend the frame inwards — exploiting the property that pieces fit together, and knowing one of two fitting pieces says someting about the others shape. Often the strategy of completing the frame first and working inwards will complete the task faster than adopting the *do anything* strategy. In both cases, there is uncertainty about where any given piece of the puzzle goes, but with a suitable control strategy this uncertainty can be *graded*: Corner pieces (assuming the puzzle has a rectangular perimeter) are least uncertain, edge pieces are next, and so should be placed next, and then pieces which look as if they might extend the frame should be dealt with since their placement is less uncertain than those whose placement is completely unconstrained.

The control or problem-solving approach to uncertainty is often used when the task of the system is to interpret noisy, but converging, data. For example, outline-finding algorithms used in image analysis systems must contend with noisy image data, but sometimes use the partially-interpreted image as a source of knowledge from which to infer expectations that can be used to guide further processing of the image data. The HEARSAY-II speech understanding system works in a similar way on the signals received from a microphone. In such systems the decision to pursue a particular hypothesis or ignore it is related to how certain the hypothesis is, ie how much supporting evidence and how much contradictory evidence has so far been collected for it. Such systems are said to establish and extend *islands of certainty*; those parts, or islands, of an image interpretation that are relatively certain are used to guide the interpretation of the rest of the image, for example.

Uncertainty influences the behaviour of problem-solving approaches, whereas it has little, if any, effect on the way a degree-of-belief-based approach works. For example, MYCIN did not pursue hypotheses whose CF (Confidence Factor) was less than 0.2, and thus made a simplistic discrimination on which hypotheses to develop or not. But, in general, uncertainty has a passive role in MYCIN-like systems, and a much more active role in the control strategy, approach to uncertainty in which the problem-solving scheme used is design to deal with it in some appropriate way.

3.3 The Blackboard Model

The Blackboard model of problem solving is a highly structured special case of opportunistic problem solving. In addition to opportunistic reasoning as a knowledge-application strategy, the Blackboard model prescribes the organisation of the domain knowledge and all the input and intermediate and partial solutions required to solve a problem. The set of possible partial and full solutions to a problem is called the *space of possible solutions*, or just the *space of solutions*.

In the Blackboard model, the solution space is organised into one or more application-dependent hierarchies. Information at each level in a hierarchy represents partial

solutions and is associated with a unique vocabulary, or set of representations, which describes the information. The domain knowledge is partitioned into independent modules of knowledge which transform information on one level, possibly using information at other levels of the hierarchy into information on the same or other levels. The knowledge modules perform the transformation using algorithmic procedures or heuristic rules that generate actual or hypothetical transformations. Opportunistic reasoning is applied within this overall organisation of the solution space and task-specific knowledge; that is, which module of knowledge is applied is determined dynamically, one step at a time, resulting in the incremental generation of partial solutions. The choice of a knowledge module is based upon the solution state (particularly the latest additions and modifications made to it) and on the existence of knowledge modules capable of improving the current state of the solution. At each step of knowledge application, either forward or backward inferencing techniques can be applied, or any other of the many kinds of reasoning available, such as event driven, goal driven, model driven, expectation driven, etc.

3.4 What's In A Blackboard System

The Blackboard model is a relatively complex problem-solving scheme which prescribes the organisation of knowledge and system control within an overall organisation. It is usually described as having three major components:

- **Knowledge Sources (KSs).** The knowledge required to solve the problem is partitioned into *knowledge sources* which are kept separate and independent.
  - The objective of each KS is to contribute information that will lead to a solution to the problem. A KS takes a set of current information on the Blackboard and updates it according to its encoded specialised knowledge.
  - The KSs are represented as procedures, sets of rules, or logic assertions. To date most of the KSs have been represented as either procedures or a sest of production rules. However, systems that deal with signal processing either make liberal use of procedures in their rules or use both rule sets and procedurally encoded KSs.
  - The KSs modify only the Blackboard or control data structures, which are also often kept on the Blackboard, and only the KSs modify the Blackboard. All modifications to the solution state are explicit and visible.
  - Each KS is responsible for knowing the conditions under which it can contribute to a solution. To do this each KS has *preconditions* that indicate the condition on the Balckboard which must exist before the KS can be *activated*.
- **A Blackboard data structure.** The problem solving state data kept in a global database, the *Blackboard*. KSs produce changes to the Blackboard which lead incrementally to a solution to the problem. Communication and interaction among the KSs takes place only via the Blackboard.
  - The purpose of the Blackboard is to hold computational and solution-state data required by and produced by the KSs. The KSs use the Blackboard data to interact with each other indirectly.

  - The Blackboard consists of objects from the solution space. These objects can be input data, partial solutions, alternative hypotheses, and final solutions, and possibly control information.
  - The objects on the Blackboard are hierarchically organised into levels of analysis. Information associated with objects, their properties, on one level serves as input to sets of KSs, which in turn, place new information on the same, or other levels.
  - The objects and their properties define the *vocabulary* of the solution space. The properties are represented as attribute-value pairs. Each level uses a distinct subset of the vocabulary.
  - The relationship between the objects are denoted by named links. These relationships can be between objects on different levels, such as *part-of* or *in-support-of*, or between objects on the same level, such as *next-to* or *follows*.
  - The Blackboard can have multiple Blackboard panels. In other words, a solution space can be partitioned into multiple hierarchies.
- **A Control System.** The Knowledge Sources respond opportunistically to changes in the state of the Blackboard under the overall control of a Control subsystem..
  - There is a set of control modules that monitor the changes on the Blackboard and decide what actions to take next.
  - Various kinds of information are made globally available to the control modules. This information can be on the Blackboard or kept separately. It is used by the control modules to determine the *focus of attention* of the system.
  - The focus of attention indicates the next thing to be processed. It can either be the KSs to be activated next, or the Blackboard objects to be worked on next, or a combination of both, ie which KSs to apply to which objects. Typically systems employ one of these strategies, but not more than one.
  - The solution is built one step at a time. Any type of reasoning step, data driven, goal driven, model driven, etc, can be applied at each stage of the solution formation. As a result, the sequence of KS invocation is dynamic and opportunistic, rather than fixed and preprogrammed.
  - Pieces of problem-solving activity occur in the following iterative sequence:
    1. A KS makes changes (or just one change) to the Blackboard objects (or one object). As these changes are made, a record is kept in a global data structure that holds the control information.
    2. Each KS indicates the contribution it can make to the new solution state. This can be either defined *a priori* for an application, or dynamically determined. Each possible KS contribution is put into an *Agenda*, and is called a *Knowledge Source Agenda activation Record (KSAR)*, as part of the control information.
    3. Using information from 1 and 2, a control module selects a focus of attention.
    4. Depending upon the information contained in the focus of attention, an appropriate control module prepares it for execution as follows:

a) If the focus of attention is a KS, then a Blackboard object, or set of objects, is chosen to serve as the context of its invocation. This is called the *knowledge scheduling approach.*

b) If the focus of attention is a Blackboard object, then a KS is chosen which will process the object. This is called the *event scheduling approach.*

c) If the focus of attention is a KS and an object, then the KS is ready for execution. The KS is executed together with the context.

- Criteria are provided to determine when to terminate the overall process. Usually, one of the KSs indicates when the problem-solving process is terminated, either because an acceptable solution has been found, or because the system cannot continue for lack of knowledge or data.

3.5 Problem-Solving Behaviour and Knowledge Application

The problem-solving behaviour of a system is determined by the knowledge-application strategy encoded in the control modules. The choice of the most appropriate strategy depends upon the characteristics of the application task and on the quality and quantity of the domain knowledge relevant to the task. Basically, choosing a particular Blackboard region and choosing a particular KS to operate on that region determines the problem-solving behaviour. Generally a KS uses information on one level as input and produces as output information on another level. Thus, if the input level of a KS is on a lower level (closer to data) than its output level, then it is an example of a *bottom-up* or *forward reasoning*, KS application.

Conversely, a commitment to a particular type of reasoning step is a commitment to a particular knowledge application method. For example, if we are interested in applying a data-directed forward reasoning step, then we would select a KS whose input level is lower than its output level. If we are interested in goal-directed behaviour, we would select a KS whose output information satisfies a goal at a lower level. Using the constructs in the control component it is possible to make any type of reasoning step happen at each step of knowledge application.

How a piece of knowledge is stated often presupposes how it is to be used. Given a piece of knowledge about a relationship between information on two levels, that knowledge can be expressed in top-down or bottom-up application forms. These can further be refined. The top-down form can be written as a goal, an expectation, or as an abstract model of the lower-level information. For example, a piece of knowledge can be expressed as a conjunction of information on a lower level needed to generate a hypothesis at a higher level (a goal), or it can be expressed as information on a lower level needed to confirm a hypothesis at a higher level (an expectation), and so on. The Blackboard framework does not presuppose nor prescribe the knowledge-application, or reasoning methods used. It provides constructs within which any reasoning method can be used. As a result many interesting problem-solving behaviours can be and have been implemented using the Blackboard model.

3.6 Some Comments

3.7 Summary and Warning

To summarise, the basic approach to problem-solving of the Blackboard model is to divide the problem into loosely coupled subtasks. These subtasks roughly correspond to areas of specialisation within the task. For a particular application, the designer defines a solution space and knowledge required to find the solution. The solution space is divided into analysis levels of partial or intermediate solutions, and knowledge is divided into specialised knowledge sources that perform subtass. The information on the analysis levels is globally accessible on the Blackboard, making it a medium of interaction between the knowledge sources. Generally, a knowledge source uses information on one level of analysis as input and produces information on another level as output. The decision to employ a particular KS is made dynamically using the latest information on the Blackboard. This approach to problem decomposition and knowledge application is very flexible and has been made to work well in diverse applications. There is, however, one caveat: how the problem is partitioned into subproblems makes a great deal of difference to the clarity of the approach, the time taken to find solutions, the resources required, and the ability of the system to solve problems at all.

3.8 Some Comparative Comments

The Blackboard model, when applied to the building of knowledge-based systems, has certain advantages over those provided by production rules, frames and objects, or deduction systems, whether used separately or in combination in the context of a hybrid knowledge-based system, at both the Symbol Level and Knowledge Level.

1. Knowledge sources provide a mechanism for structuring production rules (Symbol Level), and a means for differentiating different kinds of domain knowledge (Knowledge Level).

2. Control modules remove much of the inflexibility associated with conflict resolution strategies in production systems (Symbol Level), and provide a means for differentiating different kinds of control knowledge (Knowledge Level).

3. The use of a global data structure means that KSs can communicate indirectly without having to send each other messages, as in the object-oriented paradigm. This means that incremental programming, with all the debugging and refactoring of representational objects that this implies, can proceed without the system builder having to work out the details of explicit inter-object communication, as new objects are created, or as old objects are destroyed, renamed, split into multiple objects, etc (Symbol Level). It also means that certain data - events, states, etc - which do not lend themselves to reification (being made explicit, or concrete) can be represented without introducing unnecessary objects or meaningless messages (Knowledge Level).

4. Having control modules and control data structures (which may be part of the Blackboard) represents a more structured approach to meta-level inference than

that typically found in logic programming approaches (Symbol Level), and it is more convincing as an approximate model of human reasoning (Knowledge Level).

5. It provides a rationale for combining procedures, rules and structured objects in a more or less principled way (Symbol Level). It is *more or less* because at the lowest level of analysis it is still possible to do unprincipled things with powerful primitives, as it is in any system. However, there is a consistent story concerning many aspects of the internal structure of declarative representations of knowledge and their procedural interaction in problem-solving (Knowledge Level).

3.9 Example Systems

These notes are intended to be supplemented by Chapter 11 of the Knowledge Representation and Inference One notes by Peter Ross, which contains descriptions of the HEARSAY-II, HASP/SIAP, and OPM Blackboard systems.

4 Required Reading

The required reading for this lecture is taken from the *Big Red KR Book*:

1. Chapter 4, page 41 — Levesque, H.J, and Brachman, R.J., *A Fundamental Tradeoff in Knowledge Representation and Reasoning.*
2. Chapter 24, page 411 — Brachman, R.J., Fikes, R.E., and Levesque, H.J., *KRYPTON: A Functional Approach to Knowledge Representation.*

5 Recommended Reading

The paper by Alan Newell called the *Knowledge Level* and the paper by Hector Levesque called *Foundations of a Functional Approach to Knowledge Representation* are well worth reading, particularly if you intend doing any kind of knowledge-based system project.

The double article by Penny Nii on Blackboard Systems in volume 7 of the AI Magazine provides a good introduction and description of some example systems. The first part of this article has been used in compiling some of these notes.

A Blackboard Shell built in Prolog by Peter Ross et al, as part of a project concerned with user modelling of command driven systems, is described in DAI Research Paper No 277, 1986.

References

[BFL85] R. J. Brachman, R. E Fikes, and H. J. Levesque. KRYPTON: a functional approach to knowledge representation. In R. J. Brachman and H. J. Levesque, editors, *Readings in Knowledge Representation*, chapter 24, pages 411–429, Morgan Kaufmann, 1985.

[BL86a] R. J. Brachman and H. J. Levesque. Knowledge level interfaces to information systems. In M. L. Brodie and J. Myopoulos, editors, *On Knowledge Base Management Systems — Integrating Artificial Intelligence and Database Technologies*, Springer-Verlag, 1986.

[BL86b] R. J. Brachman and H. J. Levesque. The Knowledge Level of a KBMS. In M. L. Brodie and J. Myopoulos, editors, *On Knowledge Base Management Systems — Integrating Artificial Intelligence and Database Technologies*, Springer-Verlag, 1986.

[LB85] H. J. Levesque and R. J. Brachman. A fundamental tradeoff in knowledge representation and reasoning. In R. J. Brachman and H. J. Levesque, editors, *Readings in Knowledge Representation*, chapter 4, pages 41–70, Morgan Kaufmann, 1985.

[Lev84] H. J. Levesque. Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23, 1984.

[New62] A. Newell. Some problems of basic organisation in problem-solving programs. In M. C. Yovits, G. T. Jacobi, and G. D. Goldstein, editors, *Conference on Self-Organizing Systems*, Spartan Books, Washington, D.C., 1962.

[New81] A. Newell. The Knowledge Level. *AI Magazine*, 2, Summer 1981.

[Nii86] P. H. Nii. Blackboard systems. *AI Magazine*, 7(3), Summer 1986.

[Jon86] J. Jones, M. Millington, and P. Ross. A Blackboard Shell In Prolog. *DAI Research Paper No 277*, 1986.

[Sim77] H. A. Simon. Scientific discovery and the psychology of problem solving. In *Models of Discovery*, D. Reidel Publishing Co., 1977.

6 Small Group Tutorial Preparation

At the end of the lecture a Simple Autonomous Mobile Device (SAMD), otherwise known as the Andy Russel Mouse after its creator, will be demonstrated so that you can observe its behaviour. For the small group tutorials to be held on Thursday 18 February (MSc) and Friday 19 February (AI/CS-3) you should prepare a brief (about one A4 page) written explanation of what you think the SAMD *knows* expressed both informally and in some formal way, such as first-order predicate logic (Knowledge Level description), together with an explanation of how you think its observed behaviour is achieved by its implementation (Symbol Level Description - if you think it can be described in terms of symbol manipulations, if not you should say how its implementation can be described). You should also indicate whether you think the device is *intelligent* or not, and be prepared to defend your opinion with a sound well constructed argument.

7 Advertisement

As those of you who have got as far as the Required Reading sections of the lecture notes for this course will have noticed, considerable emphasis is placed upon the reading of numbers of often difficult papers. If enough interest was shown by members of the class I would be interested in organising occasional extra sessions to discuss any of the papers set. My favoured time would be Wednesday afternoons at 5:00pm for about an hour. If you are interested in such an offer please indicate as much by mailing me at tim@ed.edai, so that I might gauge the demand. Thank you.

Tim Smithers
February 1988