**To appear in:**
Janice Glasgow, Hari Narayanan, Chandrasekaran, (eds), *Diagrammatic Reasoning: Computational and Cognitive Perspectives*, AAAI Press 1995?

# Musings on the roles of logical and non-logical representations in intelligence.

**Aaron Sloman,**
**School of Computer Science, The University of Birmingham,**
**Birmingham, B15 2TT, England,**
**A.Sloman@cs.bham.uk.ac**

### Abstract

This paper offers a short and biased overview of the history of discussion and controversy about the role of different forms of representation in intelligent agents. It repeats and extends some of the criticisms of the 'logicist' approach to AI that I first made in 1971, while also defending logic for its power and generality. It identifies some common confusions regarding the role of visual or diagrammatic reasoning including confusions based on the fact that different forms of representation may be used at different levels in an implementation hierarchy. This is contrasted with the way in the use of one form of representation (e.g. pictures) can be *controlled* using another (e.g. logic, or programs). Finally some questions are asked about the role of metrical information in biological visual systems.

## Introduction

Discussion of how to represent and manipulate knowledge intelligently has a very long history, going back to ancient times, as evidenced by the systematic codification of mathematical knowledge in Euclid, and Aristotle's invention of a formal approach to reasoning. An important later development was Kant's discussion in his *Critique of Pure Reason* regarding the non-analytic nature of some mathematical knowledge. Although his concepts lacked clear definition, and his understanding of logic was apparently restricted to what could be learnt from Aristotle, Kant attempted to show (roughly speaking) that some mathematical knowledge was 'synthetic', i.e. it went beyond what could be derived from definitions using only logic. Important ideas about how to represent and manipulate information were also contributed by Descartes, Boole, Frege, Peirce, Russell and many more. Scientists and mathematicians have often had to invent new specialised forms of representation, with associated manipulative techniques, in order to deal with new problems. The invention of the arabic numeral notation was a major landmark. Another was the invention of notations and techniques of differential calculus by Newton and Leibniz.

Perhaps the greatest logicist of all time was Gottlob Frege. A century ago he attempted to demonstrate that all arithmetical concepts could be defined in logical terms and all arithmetical knowledge could be derived entirely from those definitions and logical axioms. He thereby claimed to show that in a very precise sense arithmetical knowledge was analytic, not synthetic as claimed by Kant. In the process, Frege invented predicate calculus after generalising the concept of a function to allow non-arithmetical arguments and values and demonstrating the possibility of higher order functions, thereby providing a deep foundation for much of what followed in computer science (including the lambda calculus and Lisp).

But even Frege, while attempting to show that Kant was wrong about arithmetical knowledge, conceded that geometrical knowledge could not be derived from logic, though as far as I know he had nothing positive to say about how we obtain or use geometrical knowledge. Later, Bertrand Russell dealt Frege a body-blow by showing that the logical system on which he had tried to base arithmetic was inconsistent, as it led to the paradox of the set of all sets that do not contain themselves (or the predicate 'heterological' which describes all and only those predicates that do not describe themselves), sowing the seeds that later led to Gödel's incompleteness results.

The development of computers in the middle of this century added enormous momentum to studies in software engineering, cognitive modelling and AI, as shown by the 47 page size of Minsky's annotated bibliography produced as early as 1963 for the Feigenbaum and Feldman volume. Much of this work raised new issues about representations and their dynamics, as illustrated by some of the articles reprinted in that collection, for instance Gelerntner 1959, Lindsay 1963 and Minsky 1961. The latter includes a discussion of some complexity issues in problem solving, and claims that structured, computed descriptions will be needed: "To be useful, these should reflect some of the structure of the things they designate, abstracted in a manner relevant to the problem area." (p.413 in Feigenbaum and Feldman). There's nothing new under the sun.

## Logicism in AI

Many people have been bemused by the logicist claim, put forward by some extremely intelligent AI folk, that systems built solely on logical representations and general logical forms of inference might exhibit human-like intelligence. The logic-based approach in AI was pioneered by John McCarthy during the 1950s and 1960s. For instance, in McCarthy 1968 he argued that not only the facts used by an intelligent agent but also the rules and heuristics for processing them could be expressed using logic, an idea which was subsequently embodied in the programming language Prolog and its successors, and to some extent in rule-based expert system formalisms. Some of the logicist ideas were developed further in the influential paper by McCarthy & Hayes 1969, which included a list of criteria for assessing forms of representations, including metaphysical adequacy, epistemological adequacy and heuristic adequacy.

At the second International Joint Conference on AI at Imperial College London, I challenged the assumption that logic could suffice for AI, in Sloman 1971. That paper led to an unscheduled inconclusive debate with McCarthy, Hayes and others, following the formal sessions at the conference. My 1971 paper was in turn criticised by Hayes 1974, to which there was a partial response in Sloman 1975 and in my later papers. (Some of the arguments on both sides are regularly re-invented.)

Disagreement persisted throughout the 1970s between those who sought totally general, logic-based, means of representation and inference (mainly people in the AI laboratory at Stanford) and those who argued for special purpose domain-oriented or problem-oriented representations and mechanisms (mainly people at MIT). There are relevant articles in Bobrow and Collins 1975, including the influential article by Woods, and further relevant discussion can be found in the collection of reprints edited by Brachman and Levesque 1985, especially Hayes 1974, Hayes 1985, Funt 1980, and my 1975 article.

Alongside the theoretical debates, designers and developers in AI and software engineering continued to use whatever forms of representation seemed best suited to their problem, some in ignorance of the theoretical debates and some explicitly taking sides, e.g. Bundy 1973 (though he later claimed that his diagrams could have been replaced by purely logical derivations). For instance, AI vision researchers used 2-D arrays, histograms, semantic nets, or logical databases for different stages in the interpretation of images and many of them would simply have laughed at any suggestion that they do it all using logic. (In fact such an interaction occurred around 1974 between an early Prolog proselyte in Edinburgh and a group of vision researchers at Sussex and Essex universities in the UK.)

I remember some of the debates being very muddled on account of confusion over definitions: for example there were some who treated the semantic nets used by Winston 1975 in his concept learning program as instances of analogical (pictorial, iconic) representations, because the topology of the net corresponded closely to the topology of the physical structures represented, whilst others regarded them as instances of symbolic or logical representations, because the use of explicit labels for relationships ensured translatability between the nets and logical formulae, a point developed further in connection with "frame" representations by Hayes 1979.

What is not always noticed is that two formalisms with similar expressive power and the ability to validate the same derivations, may differ in their heuristic power, for example in facilitating searches for solutions to problems: that was one of the main points in Sloman 1971, and before that in Lindsay 1963.

Parallel debates raged in psychology, e.g. see the volume edited by Nicholas in 1977 (especially Pylyshyn 1973, reprinted therein), and still continue. Although some of the experimental results (e.g. such as time taken to perform 'mental rotation' tasks being proportional to the amount of rotation required) are open to alternative interpretations, it does seem well established by neurophysiological research that a significant portion of the brain concerned with visual processing uses representations that preserve some aspects of retinal relationships (especially neighbourhood relationships), though not much is known about how those representations are used, nor how the higher-level tasks of vision are achieved, nor how tactile and motor representations concerned with spatial structure and motion integrate with visual representations. (I'll return to animal vision below.)

Since the 1970s the debate has continued in AI; for instance, an extreme logicist position was taken by Kowalski 1980 and elaborated in his later publications, to which my 1985 paper arguing for multiple knowledge representation formalisms was, in part, a response. However, during the 1980s the debate came to be dwarfed by other issues, for instance the clash between connectionist and symbolic AI, and a host of extremely difficult technical problems faced by the logicists, such as the problems of accounting for human abilities to reason with incomplete information which could later be extended or corrected, and the problems of efficiency in theorem provers.

In the last few years (since about 1990) there appears to have been a revival of interest in the debate, as shown by symposia on diagrammatic or spatial reasoning at AI conferences, a special issue of *Computational Intelligence* (Narayanan 1993), this volume and a volume being edited in parallel with it by Peterson, forthcoming. Work on multi-media interfaces has also led some (e.g. Wahlster 1994) to study the relationship between verbal and diagrammatic forms of communication and good ways of integrating them: an area that is bound to attract increasing attention as the use of computers becomes more widespread and dissatisfaction with their communicative limitations grows.

## What's wrong with logicism?

There are several different bases for attacking logicist AI. One that is popular with some philosophers uses Gödel's incompleteness theorem, or similar metamathematical 'limit' theorems, in an attempt to demonstrate that human beings (or at least human mathematicians!) have capabilities that cannot be replicated in a logic-based formal system or any mathematically equivalent system (including all computing systems that are in principle implementable as Turing machines). There is a considerable philosophical literature on whether Turing-equivalent machines could ever implement mental states and processes, much of which hinges on Gödel's theorem and related results. The best known recent attack of this form is Penrose 1989, discussed at length in the commentaries following Penrose 1990 and criticised at length in Sloman 1992. I shall say no more about this line of argument, which essentially assumes that if logicist AI fails then AI fails. There is a broader conception of AI as the study of principles for designing or explaining (actual or possible, natural or artificial) intelligent systems, which is not committed to any particular restriction of formalisms or mechanisms (Sloman 1994a).

The main basis of my criticism of logicists in 1971 was the oft-noted fact that human beings fruitfully use many different forms of representation (Hayes 1974 calls them 'schemes'), including natural languages, gestures, maps, musical notation, dance notation, Venn diagrams, Euler diagrams (often confused with Venn diagrams!), dress-making patterns, programming languages, blueprints, flow-charts, 3-D models of molecules, many types of data-structures used in computing systems, and a host of special-purpose mathematical notations (including, for instance, the number notation in which concatenation of digits stands for the combination of multiplication by 10 and addition).

More recently, connectionist representations have also added to the variety. For instance, a neural net may include information about many instances of some category superimposed and distributed over a collection of synaptic weights.

My claim was and is that this proliferation of types of representation is not simply a quirk of human nature but a profound and universal requirement for intelligence. In part my views had been shaped by reading Kant's (somewhat confused and unclear) argument that some forms of mathematical discovery extend our knowledge in ways that seem distinct from logical inference: this matched my own experience of doing mathematics.

Whether they are aware of these debates or not, many computer programmers, both inside and outside AI, naturally use different sorts of representations carefully crafted to meet the requirements of particular problems. Often this is done without reflection on the variety of forms of representation available and the reasons for selecting some rather than others. A good, experienced programmer, like good, experienced mathematician, intuitively adopts a fruitful approach, without necessarily knowing how or why.

The papers in this volume discuss the issue explicitly. The main point is not what can be expressed in various formalisms, but how they compare in *heuristic* power, e.g. how the syntax of specialised representations facilitates control of search for a solution to a problem. Sometimes the improvement results from the way the syntax restricts what can be represented, thereby reducing branching in search spaces. Sometimes it results from the 'indexing' power of analogical or pictorial representations, which allows good candidates to be found first, because the syntax encodes a heuristic evaluation function to guide the search. (Both points are frequently re-discovered, so they must be important!)

An example of the first point is the difficulty of drawing an object that is both round and square (except when viewed end on!), compared with the ease of construction of the assertion **round(obj)&square(obj)**. An example of the second point is the use of direction in tracing a good route from A to B on a road map: roads going off in completely the wrong direction can usually (but not always) be ignored.

## Confusions about efficiency

One thing that emerges from close analysis of the papers in this volume and other publications, including the papers in the special issue of the journal *Computational Intelligence* on computational imagery (Vol 9, no 4, 1993), is that there is still no clear consensus as to what the options are, how they differ, and what criteria there are for choosing between them in building working systems, either for engineering purposes or for the purpose of modelling human or animal capabilities.

In particular it is too easy when arguing about the relative efficiency of different representations to forget that *actual* efficiency depends not only on the abstract properties of representing structures and processes manipulating them (usually implemented in software in virtual machines), but also on the properties of the underlying engines that implement those processes. For example, visual or spatial reasoning processes that introspectively seem very simple and efficient for the human brain may actually make use of enormously 'expensive' computations performed by vast numbers of neurons working in parallel. Attempting to replicate those processes on computers might be far less efficient than using different representations whose manipulations fit more directly onto computers.

For example, in some cases it may be cheaper and easier to add indexing and constraint mechanisms to a logical representation in a computer than to give the computer human-like map-reading capabilities.

Confusion on these points will probably reign for some time, and often leads to debates at cross-purposes. I'll mention a few of the common confusions before moving on to discuss some ways in which different forms of representations interact.

## Misconceptions about logical and analogical representations

My 1971 paper defined analogical representations as those that used properties and relationships in the representing medium, rather than explicit symbols, to represent properties and relationships in the situation depicted. This was misinterpreted by some readers as the claim that analogical representations are always isomorphic with what they represent, even though the paper presented a counter-example in the form of a 2-D picture of a 3-D scene.

The paper argued that in *some* cases analogical representations were more efficient, e.g. in controlling search. This was often misinterpreted as saying what some people actually believe, namely that logical representations are always inferior to pictures or diagrams, or that there are only two sorts of representations, logical (Fregean, applicative, symbolic, verbal) and analogical (pictorial, iconic, 'direct').

All of these are mistakes. I have never denied that logic remains the most powerful and general form of representation available, which, because of its applicability across domains and across problems, supports powerful forms of analogy-making and learning. Analogical representations (e.g. pictures and diagrams) can be very powerful in some cases, for reasons discussed in several papers in this volume, but in other cases logic wins, especially where disjunctive, negative, conditional, or quantified information is involved. In yet other cases more specialised representations, tailored to the needs of the domain, may be best, e.g. the use of arabic numerals in long multiplication and division. Most programming languages fit neither the logical nor the analogical models, though they often contain elements of both. (Even Prolog uses analogical representations of lists, and to some extent the order of processing is depicted by order of symbols, though backtracking complicates this mapping.)

However, a logical system can be self-sufficient in the sense that in principle it is possible to have a completely logic-based (virtual) machine, whereas pictorial representations need to be embodied in a machine that also uses non-pictorial representations to control the use of the pictures. For an example see Funt 1980. I've explained some of these points in more detail in Sloman 1985. (These comments about logic being more self-sufficient might be refuted by producing a purely pictorial general purpose programming language.)

It is a mistake to suppose that there are just *two* forms of representation: besides logical and analogical representation there are all sorts of other types, including those mentioned above. In Sloman 1993a, 1993b, 1994b, 1994c, I have begun to develop the notion of a mind as a self-modifying, self-monitoring, control system, and to generalise the concept of 'representation' to cover a host of different types of information-bearing, causally effective, control states. These control states can exist in virtual machines at different levels of abstraction, like function definitions in a Lisp virtual machine or rules in a Prolog virtual machine or OPS-5 virtual machine. From this general viewpoint, all sorts of different types of representations, internal and external, can be seen as having syntax, semantics, pragmatics and inference methods. Syntax is a matter of the available structures and forms of variation. Semantics is concerned with one thing representing (depicting or denoting) another. Pragmatics is concerned with the functional roles of various sub-systems in a larger whole. Inference methods are simply the pragmatically useful syntactic transformations.

I suspect that there are still many new forms of representation waiting to be discovered, all with their own syntax and forms of transformation suited to particular subject domains and types of problems. In particular I believe that attempts to understand and replicate human-like *visual* capabilities will not succeed without some radically new forms of representation that integrate information about spatial structure and motion with information about possible changes, causal relations, and functional roles in a deep way (Sloman 1989). The fact that we don't yet know how to give machines visual capabilities that even begin to match the sophistication of human vision, or even squirrel vision, is one reason why it has been hard for research in AI to make use of the obvious fact that visual representations play a powerful role in human (and animal) intelligence.

## Confusions regarding the syntax of analogical representations

Besides the confusions already mentioned, there are many common errors that are made about analogical representations, such as that they need to be continuous (not so, for a discrete ordered list can be an analogical representation of a sequence of events) and that they obey the same principles of compositionality as sentences or logical formulas (some diagrams do, but not all pictures, for there are no unique ways of decomposing typical photographs or paintings into parts, or replacing parts with other items without ruining another part of the picture or leaving empty spaces).

Related to the mistake that pictures always have a unique decomposition on which a formal semantics can be based is the erroneous but common assumption (e.g. Hayes 1974) that analogical or pictorial representations are necessarily isomorphic with what they represent, in that parts, properties and relations in the representation correspond unambiguously and systematically with parts properties and relations in what is depicted. This is a common misinterpretation of Sloman 1971, which merely stated that analogical representations use properties and relations in the representing medium to represent properties and relations in the represented domain. This does not imply isomorphism. Although we agreed on many points, Hayes was partly at cross-purposes with me as he apparently thought that I meant by 'analogical' what he meant by 'direct', whereas I regarded his direct (and isomorphic) representations as merely a special case of analogical representations.

Anyone who works on 3-D machine vision knows that pictures are typically not isomorphic with what they depict. The scene depicted by a picture will necessarily include things, like the far side of a cube, that do not correspond to any component of the picture. Likewise a line drawing of a wire-frame cube will typically contain several junctions that do not correspond to junctions in the cube, namely where projections of edges cross in the picture. Deciding which image junctions do and which do not depict scene junctions may include a search for a globally optimal interpretation. Insofar as I can make sense of what Hayes meant by 'direct', many images are not direct representations, which is one of the reasons why vision is so hard to explain and replicate.

The lack of isomorphism, the lack of a unique context-free segmentation and the presence of much local ambiguity are the sorts of things that add up to make computer vision extremely hard, or even *impossible* without a sophisticated mixture of top-down and bottom-up, or knowledge-based and data-driven, processing. Very often working out the interpretation is not a matter of applying a semantic interpretation function but of solving a problem. Sometimes, in the presence of noise or occlusion, there is no unique solution, merely a collection of candidates requiring a selection based on extraneous considerations (e.g. interpreting on the basis of what you were expecting to see, or optimising some function).

Most AI work on the use of pictorial or diagrammatic representations in reasoning or planning by-passes the problems of visual interpretation, by using special-purpose analogical representations that simplify the problem, e.g. Funt 1980, Glasgow 1983, and papers in this volume. It is possible that the brain does the same: why create an ambiguous representation to manipulate when you *start* by knowing what it is a representation of? However, we sometimes use what we *see* as a basis for reasoning or problem solving, e.g. looking at the shape of an armchair and its relationship to a door, in order to work out how to get it through the door by rotating it. In such cases we do not choose the representation: the environment presents us with it, and all the hard problems of vision are there.

## Symbiosis between logical and analogical representations

In human thinking there often seems to be a rich interplay between different forms of representation. For example, one can reason about numbers by manipulating algebraic expressions, expressed in a notation that is equivalent to a logical formalism. This sort of process now pervades science and engineering. However we often find it useful to think of numbers as forming a spatially ordered series, along which something can move in either direction. Thus it is sometimes helpful, especially for a child learning to understand numbers, to think of subtraction as moving backwards a fixed number of steps. Multiplication of integers is definable logically using a recursive definition, but it can also be thought of in terms of replication of spatial groups, usually producing a rectangular array of objects. Replacing arrays of objects with areas gives an interpretation for multiplication of non-integers (fractions and reals). It is more common to teach such arithmetical operations in terms of their spatial analogues than in terms of direct logical definitions. (How many primary school children, or their teachers, would be able to cope with the latter?)

Similarly, although many functions from numbers to numbers can be defined and discussed in a purely logical (or algebraic) way, people often find it very useful to think of them in terms of a 2-D graph which can be drawn on paper. Thus the exponentiation function can be pictured in terms of a curve whose steepness continually increases at an ever increasing rate.

There are many well known proofs of properties of numbers and operations on numbers that use diagrams, such as the proof that the sum of the first N odd numbers is always a perfect square, which can be seen by carving up an N by N square of dots appropriately. Start in the top left corner, where there is one dot. Consider all the immediate neighbours of that dot as the next number, namely 3, then all the immediate neighbours to the right and below as the next number, namely 5, and so on. A little reflection shows that each new set of neighbours (to the right of and below the previous set) contains exactly two more dots than the previous set, so only the odd numbers are involved, and they add up to form the square. Notice that I have used verbal descriptions of a class of pictures to help the reader grasp the pictorial proof of a theorem of arithmetic, which could be expressed logically.

words *describe* a picture *which proves* an arithmetical theorem.

This sort of interplay between different types of representation is typical of the use of pictures and diagrams in reasoning.

Here's another example: given a pair of numbers we can describe their parity as 1 or 0 depending on whether their sum is odd or even. Consider a set of transformations of such pairs all of the form of increasing or decreasing each number by 1. Thus the four possible transformations are (+1, +1), (+1, -1), (-1, +1), (-1, -1). Is it possible to start with a pair of numbers of odd parity and change it to a pair of even parity by repeatedly applying transformations from that set? Some readers will find that easy to answer whereas others will have to think quite hard. However, everyone finds it obvious that a bishop in chess, allowed only diagonal moves, can never change the colour of the square it is on. Seeing the isomorphism between these two problems makes the answer to the first one obvious (though seeing the isomorphism is not trivial).

Good mathematicians, scientists and engineers seem to switch rapidly between different ways of thinking and reasoning about numbers and numerical functions and relationships. They also use spatial and diagrammatic reasoning in many other contexts. For instance category theorists use diagrams to represent morphisms and operations on morphisms. It is commonplace to use diagrams to represent set-theoretic relations (e.g. Euler circles, Venn diagrams). What is not always noticed is that finite spatial structures can play a powerful role in thinking about infinite sets. For example many students first learn that infinite sets permit one-to-one mappings (bijections) from the whole set to a proper subset by being presented with a picture something like the figure below, demonstrating the bijection between the set of positive integers and its even subset.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **1** | **2** | **3** | **4** | **5** | **6** | **.** | **.** | **.** |
| $\vert$ | $\vert$ | $\vert$ | $\vert$ | $\vert$ | $\vert$ | $\vert$ | $\vert$ | $\vert$ |
| **2** | **4** | **6** | **8** | **10** | **12** | **.** | **.** | **.** |

How do we see that the picture can be continued indefinitely?

More interesting is the theorem that if **S1** and **S2** are two sets and there exists a bijection between **S1** and a subset of **S2** and a bijection between **S2** and a subset of **S1** then there exists a bijection between **S1** and **S2**. This is very easy to express logically and a proof can be formulated logically, but most people I have talked to find it almost impossible to think about the problem without using some sort of spatial image of the two sets and the mappings between them.

Another example, first drawn to my attention by Alan Bundy, is the difficulty of using logic to solve the equation:

**sin(x) = x**

for which people can quite easily find at least one solution by thinking of the definition of **sin** in terms of ratio of opposite to hypotenuse in a right-angled triangle, and then doing 'qualitative' reasoning by visualising the graph for **y = sin(x)** superimposed on that for **y = x**, and noticing that there is an intersection point at the origin. (There may be more than one intersection point, depending on the unit of measurement for angles.)

This ability to combine different forms of representation, including static and dynamic spatial representations, is a characteristic feature of human intelligence. Perhaps it is an essential feature of intelligence in general. We don't yet know much about how to implement it or explain how it works.

## Efficiency of representations may depend on underlying machines

One of the important points made in Hayes 1974 is that a working system may have different levels of description. At one level a software system may use 2-D arrays to represent spatial information. At a lower level the array might be implemented as a set of logical assertions about the contents of the locations of the array. E.g. the following might represent the fact that at the location (3,7) the colour is blue.

**value(3, 7, "blue")**

It might turn out that on a machine designed for very fast logical operations this would be more efficient than a more conventional array representation where slices in the array are mapped into contiguous portions of memory, usually at the cost of symmetry with respect to rows and columns. Thus, it is wrong to state without qualification that analogical representations are more efficient for a given problem than logical representations: it all depends on what runs fastest on a given machine. From this viewpoint, if brains don't use a logical engine that is merely a quirk of evolution.

I believe that this tempting relativistic compromise view misses an important point, namely that whatever the properties of the low level machine being used, for certain problems it may still be useful to ensure that there is a higher level virtual machine that uses representations suited to the problem. Software engineers do that all the time.

For example suppose there are many thousands of events recorded in a temporal sequence. It would be possible to store an unordered list of assertions of the following form in a logical database:

**event(label, features, time)**

If one had information about two events, with labels A, and B, and wished to know whether any other event occurred between them, then if the information were unordered it would be necessary to search the whole list to ensure that no other event had a time between those of A and B. If the items were ordered in the list according to time, then having found event A we would need to look only at its immediate predecessor and successor in order to see if any other event came between it and B. The time this saves on average compared with the unordered list will be roughly a linear function of the size of the list. For some problems a change from one representation to another can make a very dramatic difference to the type of function mapping database size to time required. For example, a switch from an exponential complexity function to a polynomial function may turn a totally useless system into one that delivers results in a reasonable time.

For some problems the speed-up gained by using the right representation in a high level virtual machine can, as the problem size grows, dwarf the potential speed-up gained by re-implementing on a different sort of physical computer (which usually gains at most a constant factor). For example, if a switch of representation replaces an exponential function with a linear function, this will, for large enough databases, dwarf any gain obtained by engineering advances or even highly parallel implementation on a fixed number of processors, which produces only a constant speed-up compared with an optimal serial implementation.

Thus discussion of efficiency of representations must include the representations used in virtual machines, and not only the physical or electronic structures. I believe Newell and Simon (Newell 1980) caused some confusion by referring to 'physical' symbol systems when they were actually talking about symbols that could occur in non-physical, abstract, *virtual* machines. Instead of talking about 'physical symbol systems' they should have talked about 'physically implementable symbol systems'.

## Simultaneous multiple representations

The example of an ordered list of events described logically shows that a representation can simultaneously be of more than one form. The individual information items use only a logical notation in which predicates are applied to arguments. At the same time, the *ordering* of those logical formulae in a list allows a new semantic relation to be used, in which the collection of items functions as an 'analogical' representation, in the terminology of Sloman 1971. (Note that from a logical point of view the information in the ordering is strictly redundant since it can be derived from the time values of the events.)

In other cases logical assertions can be organised in the form of a 2-D array (e.g. Glasgow 1993), or in the nodes of a network. Instead of simply using a data-structure whose locations have useful relationships, the items in the database may make use of a sophisticated indexing scheme which enables the neighbours of an event to be found quickly without scanning the whole database. Clever indexing schemes may implicitly implement orderings or other useful relationships just as well as storing assertions in a list, though perhaps with small (constant) differences in efficiency. An example would be an efficient index that mapped each event label onto an integer giving its location in the ordering (not the time: the listed events might have intervals of varying length). Then checking whether A and B were neighbours would be a matter of checking whether they mapped onto integers that differed by 1. Here one would be locating events not in contiguous bits of memory in the computer but in locations in an abstract space: the space of positive integers. The inverse mapping, from integers to stored assertions, would be needed for some applications (e.g. answering "What comes after B?"). Properties and relations in that space could provide just as good a medium for implementing analogical relations as relationships in physical memory, or on a sheet of paper.

All this shows that in the very same system different forms of representation may be used which interact in a deep way, including both logical (applicative, Fregean) representations and analogical (pictorial) relationships.

## What defines a form of representation?

What defines a form of representation is the combination of syntax, pragmatics, semantics, and inference strategies supported by the form (see Sloman 1993a, 1994b). We can now see that there may be several different syntaxes and forms of inference at different levels in the same representational system. The use of resolution and logical inference when dealing with individual stored facts may be compatible with quite different modes of inference that make use of the structure in which items are stored, or the indexing schemes used to control access.

A system that uses a neighbourhood relation in the representing medium to represent some kind of neighbourhood relationship in the domain represented, is a special case of an analogical representation, even where the representing medium uses logical expressions. It would not be an analogical representation if the neighbourhood relation were represented by an explicit symbol, e.g. a function symbol or predicate such as '**next_to(x, y)**,' instead of a relation in the medium. However, neighbourhood relations implicit in one virtual machine could be based on explicit relations in a lower-level machine, like 2-D arrays implemented as logical assertions about values corresponding to different array coordinates.

## Mixing modes of representation

We have so far discussed ways in which the implementation of a particular set of representations may simultaneously use different formalisms at different levels in an implementation hierarchy. A subtly different point is that there may be *co-existing* systems some of which are used to *control* or *reason about* others. In these cases one sort of formalism F1 is used for some purpose and another sort F2 is used to refer to F1, for purposes of reasoning about F1 or in order to control manipulations of instances of F1. F2 may or may not be of the same general type as F1. Similarly yet another formalism F3 may be used to refer to F2. It is also possible for two different formalisms, F2a and F2b, to be used to refer to or manipulate F1 in different contexts, or for different purposes. And one or more formalisms may be used to manipulate F2a and F2b, and so on. The fact that such chains of reference (and control) can coexist with chains of implementation can be very confusing when thinking about how representations work. I'll now try to illustrate these points, in connection with representation of logical expressions and representations of list structures.

In several programming languages lists are used as a data-structure for many different purposes. For example, logical assertions could be implemented using lists (which in turn are implemented using lower level facilities for managing memory in the machine). If we use square brackets to denote lists, then the following three element list might represent the assertion that john is the parent of fred:

**[parent john fred]**

And the following might represent information about an event, of the sort described above:

**[event E2367 collision lorry car Thursday 0952]**

This use of lists to implement a logical form of representation (applying a predicate **event** to 6 arguments) could coexist with the use of lists to implement analogical forms of representation, where order in a list represents temporal or spatial order. I.e. a large list functioning as an analogical representation might contain many logical representations.

However, this analogical form of representation has properties that we (or a program) might need to talk about in deciding when and how to use lists. One way to talk about the properties of lists and operations on them is to use a list-processing programming language. Another is to use logic. In general a non-logical form of representation, such as lists or diagrams, may have properties that can be represented logically and reasoned about logically. An intelligent hybrid system might use logic for this purpose.

Consider the use of lists to represent analogically what happens when people stand in a queue. We might occasionally need to represent a process in which the person at the head of the queue moves to the back of the queue. We could represent this process by defining an operation called 'rotate' which takes a list and a number and moves that number of items from the front to the back, in order.

It is possible to define such operations recursively using logic, and then to use logic to reason about them, as I'll shortly illustrate. However, it turns out that such logical reasoning may sometimes compete with or be supplemented by spatial or some other form of non-logical reasoning, which I'll also illustrate.

## Logical reasoning about lists

Lists are used a great deal in AI programming, both as a basis for logical formulas and also as a basis for analogical trees, networks, or other representing structures. (See Woods 1975 for further discussion.)

Understanding how such programs work depends on being able to reason about the properties of operations on lists. An example is the operation **rotate(List, Num)** which takes a list and an integer, and and moves the first **Num** elements of the list (if there are any) to the end of the list. It can be defined recursively as follows, where **[]** is the empty list, sometimes called 'nil', **cons** is a binary operator that creates a new list from a new element and an old list, and <> represents the list concatenator operator, (sometimes called 'append'):

**rotate([], N) = []**

**rotate(cons(X, L), 1) = L <> cons(X, [])**

**rotate(L, N + 1) = rotate(rotate(L, N), 1)**

Note that <> can also be defined recursively:

**[] <> L = L**

**cons(X, L1) <> L2 = cons(X, L1 <> L2)**

Another operation on lists returns an integer giving the length of the list, and can be defined recursively thus:

**length([]) = 0**

**length(cons(X, L)) = 1 + length(L)**

On the basis of all this it can be proved by induction that if a list L has length N, then rotating it N times produces an equivalent list:

**Theorem: list(L) −> rotate(L, length(L)) = L**

A system capable of automated logical proof of this and other theorems about list processing is described in Ireland and Bundy, forthcoming. In order to prove this logically it is necessary to have an explicit logical definition of what a list is. This can be done by inductively defining lists as things that can be created by repeated application of the binary list constructor **cons**, starting from the empty list []. The predicate **list** can be defined inductively to say that [] is a list, and anything constructed by applying **cons** to an item and a list is a list, thus:

**1. list([])**

**2. list(L) & item(X) −> list(cons(X, L))**

**3. Nothing is a list unless it can be proved to be a list on the basis of the preceding axioms in a finite number of steps.**

(If updating the head or tail of a list were to be allowed, as in Lisp and Pop-11, the definitions would need to be made more complicated.)

On the basis of such recursive definitions, using structural induction, it is possible to prove many interesting theorems about lists and operations on lists, such as the rotate theorem. (Readers are invited to try proving it by induction.) So a form of representation that is used in analogical representations and in non-logical forms of reasoning (e.g. traversing a list to answer a question about ordering) may be reasoned *about* using logical representations and logical inference techniques. This is an illustration of the power and generality of logic.

## Logic is not the only way to reason about lists!

However, these logical proofs are quite hard for people to think of and to follow unless they are experienced logicians familiar with structural induction. Quite sophisticated logical apparatus is required to control the search for a proof. Nevertheless, even without having seen or understood the proof, many programmers (and non-programmers) can intuitively grasp the truth of (some of) the theorems, and confidently and justifiably use their intuitions in designing good programs. How?

Discussion with such people suggests that instead of always interpreting lists as recursively defined binary trees, they often think of them as spatial one-dimensional, totally ordered structures, for which iterative rather than recursive concepts are naturally applicable. They use their deep understanding of the properties of space and of spatial transformations to see that, for example, if you move an item at one end of a row to the other end, shifting everything else along to make space, and continue doing so until all items have been moved, then you will end up with exactly the same row of items, as illustrated here:

```
a  b  c  .  .  .  n
          |
          V
b  c  .  .  .  n  a
          |
          V
c  .  .  .  n  a  b
and so on.
```

Similar spatial intuitions can be used in reasoning about operators that treat a list as a stack, or as a queue, i.e. operators which, like **rotate**, can be defined and reasoned about logically.

Informal experiments with seminar audiences suggest that such ‘spatial’ proofs of the **rotate** theorem and similar theorems, are far easier for most people to discover and to understand than the logical proofs using structural induction. This appears to be true even of people who understand the logical modes of reasoning. However, little is known about what sorts of internal processes actually correspond to what people describe on the basis of introspection as ‘spatial’ reasoning. When we have machines that also find the spatial proof easier we may be a little closer to understanding human intelligence. (Whatever the mechanisms are we may still find it helpful to reason about them in part using logic, and in part using other means!)

This human ability to reason spatially about list structures depends on construing lists not as algebraic structures recursively built up by successive applications of a binary operator (cons) but rather as a flat, linear, spatially ordered structure. Because human programmers use this intuitive grasp of lists all the time in their programming, the list-processing syntax of the language Pop-11 (See Anderson 1989) was extended to include notations for list processing (including list construction syntax and a pattern matcher with segment variables) that supported this spatial view of lists, since for many problems this is of great help for both experts and beginners learning about list processing and AI. For example, the structure that might, from the Fregean viewpoint, be denoted thus:

**append(append(L1, cons(X, L2)), L3)**

or as

**L1 <> cons(X, L2) <> L3**

could be represented in Pop-11 as

**[ ^^L1 ^X ^^L2 ^^L3 ]**

where“^^” represents splicing in the elements of a list, and “^” represents insertion of only one element. In Common Lisp, this would be:

**‘( ,@L1 ,X ,@L2 ,@L3)**

Similarly, picture-like patterns of a list structure can also be used with a pattern matcher to extract information from lists. This makes many list processing operations much easier to get right in Pop-11 than in Prolog or Lisp without a pattern matcher, at least for beginners.

However, like many spatially based representations this is less general than the logical format. There are problems that require recursing both along ‘tails’ of a list and down the ‘heads’, for which this particular spatial view can be very limiting. Moreover, I am not claiming that it is impossible to use logic to characterise this spatial view of lists. After all, one can express many of the key ideas of Euclidean geometry in a logical form. All I claim is that the human brain finds the spatial representation easier to use for certain sorts of problems, and I suggest that this is because for those problems a spatial representation (possibly occurring in a high level virtual machine that could be implemented using quite different representations) *intrinsically* has more heuristic power. Defending that claim would require a lengthy discussion for which there is no space here, though several papers in this volume address the topic.

## Precision and spatial representations

To finish, I'd like to raise some questions about the nature of spatial representations in animal vision, provoked by reading an early draft of the paper by Ken Forbus in this volume.

Forbus discusses representations and forms of reasoning that use not only exact numerical values but also assertions about relative magnitudes in some cases where exact values are not known. The latter is often called "qualitative" reasoning in the AI literature (which can be confusing for those who are used to reserving "qualitative" for non-numerical information, e.g. information in family trees). As I understand him, he claims that natural visual systems, like artificial visual systems that capture TV images in regular 2-D arrays, start from *exact* metric information rather than being restricted to "qualitative" information. On reflection I do not find this claim about natural vision obvious: it is not clear how best to describe visual processes in animals.

One can assume that because visual information is projected in a precise way onto the retina and to some extent processed there, exact information about location of features on the retina is available to the visual system. However, if retinal receptive fields vary in precision, as is the case with many animals, the information available to the brain about certain features may actually have considerable imprecision, especially near the periphery of the visual field. This makes the visual system very unlike a computer that projects an image directly into a uniform resolution 2-D array.

Many researchers have taken it for granted that visual systems use 'exact' coordinate representations for image structures, and this is indeed commonplace in machine vision using 2-D arrays to represent images. But it is not obvious that location on an animal's retina is treated that way by the animal's brain. It may be, for example, that receptive fields are represented by neurons that fire when those receptive fields are suitably stimulated, and that 'links' between neurons represent neighbourhood relations between receptive fields without representing absolute distances. Then relations between remote portions of the image would be represented by chains of neighbourhood relationships between receptive fields of varying sizes (the fields are much smaller near the fovea than further out), as described in Young 1989. Such chains of relationships would not correspond to exact measures of distance.

Such a system would directly map images onto structures that are more like so-called semantic nets (networks of nodes and links) than like 2-D arrays. Of course, it is clear that somehow our visual system computes very precise metrical relationships (required for accurate throwing for example), but it may do so by making complex inferences from very imprecise ('qualitative') topological relationships which are constantly recalibrated according to context. One reason for doing things this way may be that a collection of receptive fields of varying sizes arranged in concentric circles is more useful than a regular rectangular grid for supporting certain transformations such as rotation, expansion or contraction, as shown in Funt 1980.

Another reason for not attempting to make direct use of exact metrical information regarding image structure could be that although within the visual subsystem there are many transformations from one part of the brain to another that preserve retinal ordering and neighbourhood relations, it is very difficult to arrange for metrical properties to be preserved. Further, such preservation may be pointless because constant saccadic motion causes the geometry of the projected images of a particular object to be constantly changing.

Insofar as precise metrical information is required (e.g. for judging time to contact) it is not image measurements, but measurements in the 3-D environment that are needed. Thus it is possible that the brain largely ignores transient and irrelevant exact geometric measurements in the *2-D image* and instead computes exact geometrical relationships in the *3-D scene* as and when needed, using the inexact image relationships found in several views.

A possible objection to this might be that our visual system uses precise measures of binocular disparity to compute depth information, as shown by random dot stereograms (e.g. in Frisby 1979). However, it is interesting that some of the more difficult random dot stereograms (e.g. those depicting scenes with gradually changing depth rather than large discontinuities) do not fuse properly at first; and only after several seconds, or even minutes, during which the image does not need to be fixated on the retina, does the depth information emerge. This suggests that there is not a simple computation of disparities between two retinal images, since saccadic motion ensures that the images are constantly changing while the perceiver waits for the 3-D structure to appear.

The point of all this is first of all to show that there is much we don't know about spatial representations in animal brains, and secondly to show that the fact that some physical structure (e.g. the precise pattern of intensity of illumination on the retina) exists in sensory stimulation does not imply that all the information it contains is used directly. The brain moves in mysterious ways its wonders to perform.

## Conclusion

I have tried to show that the study of forms of representation goes back a long way, and that some of the very oldest controversies in philosophy (e.g. about the nature of mathematical knowledge) remain unresolved by developments in AI. Perhaps the time is ripe for new attempts to investigate and simulate how human mathematicians think about infinite sets and continuous shapes and transformations in Euclidean spaces, and how we use these spatial capabilities in reasoning about abstract and discrete algebraic or logical spaces.

Work done so far on diagrammatic reasoning identifies some of the problems, and provides fragments of evidence about how people work, and how some of their capabilities might be modelled on computers, yet there is still much to be done to understand the variety of forms in which information can stored and manipulated in intelligent control systems, natural or artificial. We may be better able to understand the full range of possibilities and the trade-offs between different design decisions if we keep an open mind in our exploration of design space.

## References


Anderson, J.A.D.W. ed., 1989. *POP-11 Comes of Age: The Advancement of an AI Programming Language* Chichester: Ellis Horwood.

Bobrow D.G. and Collins A.M. eds. 1975. *Representation and Understanding: Studies in Cognitive Science*, New York: Academic Press.

Brachman, R.J. and Levesque, H.J. (eds). 1985 *Readings in knowledge representation* Los Altos, CA: Morgan Kaufmann 1985.

Bundy, A. 1973 Doing Arithmetic with Diagrams, in *Proceedings 3rd International Joint Conference on AI*

Feigenbaum, E.A. and Feldman, J., eds. 1963 *Computers and Thought* New York: McGraw Hill.

Forbus, K. 1994. Qualitative spatial reasoning: framework and frontiers, this volume.

Frisby, J.P. 1979. *Seeing: Illusion, Brain and Mind* Oxford: Oxford University Press.

Funt, B.V. 1980. Problem-Solving with Diagrammatic Representations. *Artificial Intelligence*, 13(3), 201-230. Reprinted in Brachman & Levesque 1985, and in this volume.

Gelerntner, H. 1959. Realization of a Geometry-Theorem Proving Machine. *Proceedings of an International Conference on Information Processing*, 273-282. Paris: UNESCO House. Reprinted in Feigenbaum and Feldman 1963.

Glasgow, J. I. 1993, The Imagery Debate Revisitied: A Computational perspective. *Computational Intelligence: Special issue on Computational Imagery*, Vol. 9, No. 4 309-333

Hayes, P. J. 1974. Some problems and non-problems in representation theory. *Proceedings of the 1974 AISB Summer Conference* University of Sussex. Reprinted in *Readings in knowledge representation* eds. R.J. Brachman & H.J. Levesque. pp. 4-21. Los Altos, CA: Morgan Kaufmann 1985.

Hayes, P. J. 1979. The Logic of Frames. In *Frame Conceptions and Text Understanding*, 46-61, ed. D. Metzing. Berlin: Walter de Gruyter & Co Also reprinted in Brachman & Levesque 1985, 288-295.

Hayes, P. J. 1985. The Second Naive Physics Manifesto. In *Formal Theories of the Commonsense World*, 1-36, eds. J.R. Hobbs & R.C. Moore. Norwood, NJ: Ablex Publishing Corp. Also reprinted in Brachman & Levesque 1985, 468-485.

Ireland, A. and Bundy, A. 1994 Productive Use of Failure in Inductive Proof, in preparation for submission to *Journal of Automated Reasoning*, Special issue on mathematical induction, forthcoming. Also available as Department of AI technical report Edinburgh University August 1994.

Kowalski, R.A. 1980. Contribution to a survey on knowledge representation reported in *Special Issue on Knowledge Representation, SIGART Newsletter No 70*, eds R.J. Brachman and B.C. Smith.

Lindsay, R.K. 1963. Inferential memory as the basis of machines which understand natural language. In Feigenbaum and Feldman, 1963.

McCarthy, J. 1968. Programs With Common Sense. In *Semantic Information Processing*, 403-418, ed. M.L. Minsky, Cambridge MA: MIT Press.

McCarthy, J. and Hayes, P.J.. 1969. Some Philosophical Problems From The Standpoint Of Artificial Intelligence. *Machine Intelligence 4*, eds, B. Meltzer and D. Michie. Edinburgh: Edinburgh University Press.

Minsky, M.L., 1961. Steps Towards Artificial Intelligence. *Proceedings of the Institute of Radio Engineers*, vol 49, pp 8-30. Reprinted in Feigenbaum & Feldman 1963.

Newell, A. 1980. Physical Symbol Systems. *Cognitive Science,* vol 4, no 2. 135-183,

Narayanan, N.H. 1993. (ed) Taking Issue/Forum: The Imagery Debate Revisited. *Computational Intelligence*, Vol. 9, No. 4 303-435

J.M. Nicholas, ed. 1977. *Images, Perception, and Knowledge*. Dordrecht-Holland: Reidel.

Penrose, R. 1989. *The Emperor's New Mind: Concerning Computers, Minds and the Laws of Physics*. Oxford: Oxford University Press.

Penrose, R. 1990. Precis of *The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics*. *The Behavioral and Brain Sciences* 13,4 pp 643-705, followed by 37 commentaries and author's response.

Peterson, D.M., ed. Forthcoming. *Forms of representation* Oxford: Intellect press 1995

Pylyshyn, Z.W. 1973 What the Mind's Eye Tells the Mind's Brain: A Critique of Mental Imagery, in *Psychological Bulletin* 80, 1-24. Reprinted in Nicholas, 1977.

Sloman, A. 1971. Interactions Between Philosophy And A.I.: The Role Of Intuition And Non-logical Reasoning In Intelligence, *Proceedings 2nd International Joint Conference on Artificial Intelligence* London 1971. Reprinted in *Artificial Intelligence*, pp 209-225, 1971, and in Nicholas 1977.

Sloman, A. 1975. Afterthoughts on Analogical Representation, in R. Schank and B. Nash-Webber (eds), *Theoretical Issues in Natural Language Processing*, (proceedings of a conference held at MIT in June 1975) Association for Computational Linguistics, reprinted in Brachman and Levesque 1985.

Sloman, A. 1985. Why We Need Many Knowledge Representation Formalisms. In *Research and Development in Expert Systems,* ed. M Bramer, pp 163-183, Cambridge University Press.

Sloman, A. 1989. On Designing a Visual System: Towards a Gibsonian Computational Model of Vision. *Journal of Experimental and Theoretical AI* 1,4, pp 289-337.

Sloman, A. 1992. The Emperor's Real Mind: critical discussion of Penrose 1989. *Artificial Intelligence* vol 56, pp 355-396.

Sloman, A. 1993a. Varieties Of Formalisms For Knowledge Representation. *Computational Intelligence, Special issue on Computational Imagery*, Vol. 9, No. 4, pp 413-423

Sloman, A. 1993b. The Mind As A Control System. In *Philosophy and the Cognitive Sciences*, (eds) C. Hookway and D. Peterson, pp 69-110. Cambridge: Cambridge University Press,

Sloman, A. 1994a, Explorations in Design Space. In *Proc ECAI94, 11th European Conference on Artificial*

*Intelligence,* 578-582, ed. A.G.Cohn, Chichester: John Wiley.

Sloman, A. 1994b, Towards A General Theory Of Representations. In D.M.Peterson (ed) *Forms of representation* Intellect Press. Forthcoming.

Sloman, A. 1994c Semantics In An Intelligent Control System. In Proc. British Academy and Royal Society Conference: Artificial Intelligence and The Mind: New Breakthroughs Or Dead Ends? To be published in *Philosophical Transactions of the Royal Society*. Forthcoming.

Winston, P.H. 1975 Learning Structural Descriptions from Examples. In P.H. Winston (ed) *The Psychology of Computer Vision* pp 157-209, New York: McGraw-Hill

Woods, W.A. 1975. What's In A Link?. In *Representation and Understanding: Studies in Cognitive Science*, 35-82, eds. D.G. Bobrow & A.M.Collins, New York: Academic Press. Also reprinted in Brachman and Levesque 1985, 218-241.

Wahlster, W. 1994. Computational Models of Multimodal Communication. Invited address to 11th European Conference on AI, *ECAI94* Amsterdam.

Young, D.S., 1989. Logarithmic sampling of images for computer vision. In *AISB89: Proceedings 7th Conference of Society for the Study of Artificial Intelligence and Simulation of Behaviour*, 145-150, ed. A.G.Cohn, London: Pitman & Morgan Kaufmann.