

THE COMPUTER REVOLUTION IN PHILOSOPHY (1978)

Aaron Sloman

[Book contents page](#)

This chapter is also available [in PDF format here](#).

CHAPTER 8

ON LEARNING ABOUT NUMBERS:

PROBLEMS AND SPECULATIONS[*]

8.1. Introduction

The aim of this chapter is both methodological and tutorial. It should help to introduce readers to some computing ideas. It also includes some theoretical speculations about learning and memory. These speculations are fairly complex, yet it is clear that they are too simple-minded to be adequate accounts of how children perform their astonishing feats of learning. Many more questions will be asked than answered. And answers offered will be tentative and provisional. Unfortunately, experienced programmers will find some of the explanations below very tedious and over-simplified. I apologise to them, and hope that non-programmers will not find the same explanations too difficult!

Here is a typical conversation with a child aged between three and a half and five years.

Adult: Can you count up to twenty?

Child: One two three four five six seven eight nine ten eleven twelve thirteen fourteen fifteen seventeen eighteen twenty.

A: What comes after three?

C: One two three four --- four.

A: What comes after eight?

C: Four

A: What comes after six?

C: Don't know

A: What comes before two?

C: One

A: What comes before four?

C: Five

A: How many fingers on my hand?

C (counting fingers): One two three four five

A: What's two and three?

C (counting fingers): One two three four five. Five.

Does this child grasp number concepts? Perhaps there is something wrong with the question, because number concepts are not simple things which you have either grasped or not grasped?

What are number concepts? How is it possible for them to be learnt? How is it possible for them to be used? How is it possible to discover non-empirical facts about them? I believe we are not yet able to formulate adequate answers to these questions. What follows is offered as a preliminary exploration of some of the issues.

The method illustrated below is important. Previously (in Chapter 2), I argued that a major aim of science is to find out what is possible and explain how it is possible. We all know a great deal about what it is possible for adults and children to do with numbers. So, instead of collecting facts by doing experiments on children, we can generate requirements for explanatory theories by reflecting on the fine-structure of familiar human abilities. In other words, methods of conceptual analysis, typically practised by philosophers and linguists, can be an important source of data for psychology. (Compare chapter 4.)

I am not suggesting that conceptual analysis suffices to reveal everything we would like to know about, for example, ordinary counting abilities. The claim is only that it is foolish to embark on expensive empirical investigations before making a serious and systematic effort to articulate what you already know about the subject matter.

Here are some of the questions for which answers are lacking:

- What exactly is it that a child learns in learning about numbers?
- How is it possible for different fragments of the same number-system to be mastered by different people?
- How far does learning about numbers depend on very general learning abilities, and how far are the hurdles specific to numbers?
- How is it that a child who already seems to have the knowledge to answer a question or solve a problem, is often unable to use that knowledge?
- What enables the knowledge to be accessed at some later time?
- How can a child learn new truths about what she already knows, for instance learning that two of the numbers she has learnt add up to a third number she knows, or that two different additions generate the same result, or that some procedure (e.g. adding one) can be repeated indefinitely to yield larger and larger numbers?

I shall try to show how thinking about such apparently psychological questions can lead towards new answers to old philosophical problems about the nature of numbers, thereby providing further support for the claim that academic barriers between philosophy and science are artificial, (Some implications regarding information processing architectures for intelligent systems will emerge as a side-effect.)

[[Note added January 2002

I have just discovered the fascinating book *Wild Minds: What animals really think*, by Marc Hauser (Penguin Books 2001). Chapter 3, entitled "Number juggling", discusses and compares the understanding of numbers in very young children and in other animals. Hauser comes close to asking some of the questions asked here, and includes some speculations about possible mechanisms, but does not seem to be aware of the full variety of architectures and sub-mechanisms that might explain the observed evidence.

He repeatedly stresses the important point that it is very easy to assume that the observed behaviours of animals often suggest a unique interpretation, until we start exploring possible mechanisms that might produce those behaviours. He implicitly acknowledges that such mechanisms can be described at different levels of abstraction, not only at the level of brain physiology.

The common trap of anthropomorphism is often a product of a lack of understanding of the variety of possible information processing architectures. Some of them are explored in these recent online presentations:
<http://www.cs.bham.ac.uk/research/projects/cogaff/talks/>

Presentations particularly relevant to the nature of mathematical understanding include

- o [Talk 7: When is seeing \(possibly in your mind's eye\) better than deducing, for reasoning?](#)
- o [Talk 27: Requirements for visual/spatial reasoning](#)
- o [Talk 14: Getting meaning off the ground: symbol grounding vs symbol attachment/tethering](#)
- o [Talk 36: TWO VIEWS OF CHILD AS SCIENTIST: HUMEAN AND KANTIAN](#)
- o [Talk 6: Architectures for human-like agents.](#)

]]

8.2. Philosophical slogans about numbers

Here are some examples of philosophers' answers to the question 'What are numbers?', and related questions:

1. Numbers are non-physical mind-independent entities, existing in their own realm which is different from the world of spatial objects. (Platonists)
2. Numbers are perceivable properties of groups of objects. For example, the number three is what is *visibly* common to the two groups

* * *

and

\$ \$ \$

(Aristotle?)

3. Numbers are mental objects, created by human mental processes. Facts about numbers are discovered by performing mental experiments. (Kant, and the Intuitionist mathematicians)
4. Numbers are sets of sets, or predicates of predicates, definable in purely logical terms. An example of this view: the number one is the set of all sets capable of being mapped bi-uniquely onto the set containing nothing but the empty set. (Frege, Russell, and other logicists)

5. Numbers are meaningless symbols manipulated according to arbitrary rules. Mathematical discoveries are merely discoveries about the properties of this game with symbols. (Formalists)
6. Numbers are implicitly defined by a collection of axioms, such as, Peano's axioms. Any collection of things satisfying these axioms can be called a set of numbers. The nature of the elements of the set is irrelevant. Mathematical discoveries about numbers are merely discoveries of logical consequences of the axioms. (Many mathematicians)
7. There is no one correct answer to the question 'what are numbers?' People play a motley of 'games' using number words and other symbols, and a full account of the nature of numbers would simply be an analysis of these games (including the activity of mathematicians) and the roles they play in our lives. (Wittgenstein: *Remarks on the Foundations of Mathematics*)

For more details, see standard texts on philosophy of mathematics. I believe that more or less articulate versions of these philosophical theories, play an important role in many psychological and educational theories about numbers. (I have formed this opinion over many years, from wide but unsystematic reading and discussion, including attendance at lectures and seminars. So I am not in a position to document the claim. I shall continue in this chapter to make remarks about psychological theories -- if the disparaging ones are untrue I'll be delighted).

All the views listed above combine elements of truth with distortions and oversimplifications. I think that Wittgenstein's answer comes closest to encompassing the truth. In his writings he formulates many problems about mathematics, which are not answered by other theories, but his own solutions seem to me to be too shallow.

In particular, the anti-mentalism, or anti-psychologism, which pervades much of his writing prevents him from discussing mental processes in any depth. So he writes as if thinking about numbers were an essentially *social* process, consistently with his conclusion in *Philosophical Investigations* that *all* rule-following is an essentially social process, dependent on the existence of a public language.

This conflicts with a computational analysis of mental processes, according to which it is perfectly possible for a non-social mechanism to contain within itself rules which it can obey, for instance, programs transmitted genetically.

Wittgenstein's position also conflicts with any sensible account of the biological evolution of mental processes in precursors of *homo sapiens*.

I am not going to try to solve all the philosophical and psychological problems about numbers in one chapter. I shall merely try to show how we can get important new insights into the problems, and perhaps take some small steps towards formulating possible answers, if we think about the mental processes and mechanisms as if they were analogous to the processes and mechanisms involved in so-called 'list-processing' computer programs. Adequate exploration of these issues has been hampered by the current separation of philosophy and psychology, and the ignorance among most philosophers and psychologists of computing ideas.

I shall not be talking about events or processes or mechanisms in the human brain. Exactly how the brain works is as irrelevant to our problems as the detailed workings of a computer are to an explanation of a computer program written in a high-level programming language. There may be creatures on other planets, or robots, whose brains are totally unlike ours in their physiological details, yet such beings could well learn about numbers, and learn the same concepts as we do, just as two computers with quite different physical components can execute the same 'high-level' programs. (Incidentally, this undermines philosophical theories which claim that mental processes are identical

with brain processes. This is as inaccurate as the claim that computational processes in a computer are identical with physical processes.)

When I talk about mechanisms involved in using numbers, I am not talking about physiological mechanisms. I am talking about aspects of the way information is organised and represented, and about the kinds of symbol-manipulating processes which may be necessary for accessing and using various sorts of representations. In particular, such processes involve the following of rules, instructions, or plans, whether consciously or unconsciously.

This illustrates how the concept of 'mechanism' is extended by developments in computing.

8.3. Some assumptions about memory

Unfortunately, my speculations about mental processes will be intelligible only if I introduce some technical ideas and assumptions, already hinted at in previous chapters, especially chapter 6. If the assumptions are wrong, then quite different theories are required. At the moment, there does not seem to be any way of avoiding these assumptions, if we are trying to explain well-known facts about what people can do.

The main assumption is that we can speak of the human mind as storing information in a vast collection of locations'. They need not be spatial locations, like shelves in a library. Positions in any kind of symbolic space with appropriate mechanisms for storing and retrieving information will do. So the word location' is being used as a technical term. For instance, radio waves are often used to transmit information, different information being transmitted at different frequencies. So information could be stored in a collection of continually reverberating radio waves, with different symbols stored at different frequencies. Each possible frequency would then be a location in the sense required here.

Similarly, possible structures of a certain class of molecules could define 'addresses' in a space. Storing information at a certain address would mean attaching that information to molecules with the structure represented by the address. This could be done more or less simultaneously in many different physical places. But the information would still be stored in one symbolic place, just as a name occurs at only one symbolic location in a telephone directory, even though there may be millions of physically distinct copies of the directory containing the name. So from now on, when I talk about locations, this is neutral as to what sorts of locations they are.

I shall assume then that a mechanism is available which can store symbols in some 'space' of locations. Further, I assume that it is possible for some of the symbols to represent locations in this space. (For instance, a directory, or catalogue, can contain entries which refer to the location' of other entries, by page or section number.) Thus the space can contain information about itself.

A symbol representing a location can be called a 'pointer', or an 'address'. So the storage mechanism can be given an address and asked to produce the symbol located there. In other words, when given a pointer, it can determine what symbol is pointed at. What is pointed at may be a complex structure containing a symbol which is itself an address of some other location, that is a pointer to another symbol. (See [Figure 1](#).) So the space may contain chains of pointers. (In more elaborate systems, the addressing may be relative to a context or mode of operation. That is, which location is represented by a given symbol may depend on the current state of the accessing sub-mechanism. Some of the flexibility of behaviour of the system may depend on such systematic changes in the 'meaning' of symbols.)

The concept of a symbolic structure containing pointers into itself, and the investigation of processes in which such things are manipulated and used for solving problems, are among the important contributions of computing science. I shall try to show how these ideas help us to think about a child's ability to count, an ability which provides the substratum for a grasp of number concepts.

The first task is to make explicit some of our commonsense knowledge about the sorts of things we can do with number words and number concepts. Note the 'can': it is possibilities we most need to explain, not laws, that is not regularities or correlations. We know relatively few non-trivial laws of human behaviour. But we know of very many human possibilities, namely, many things at least some people can do. By thinking about possible mechanisms underlying fairly common abilities we can reveal the poverty of most philosophical and psychological theories about the nature of mathematical concepts and knowledge. These theories do not account for the fine-structure of what we all know. All this illustrates methodological points made in [chapter 2](#) and [chapter 3](#).

8.4. Some facts to be explained

Reflecting on even the simplest things we know children can learn (although not all children learn all of them) shows that children can somehow cope with quite complex problems of storing, using and manipulating symbols, that is, computational problems. Some of these problems are common to many forms of learning, others peculiar to counting.

I shall start with problems involved in learning number words. These problems are common to all words. Next, there are problems concerned with the fact that number words form a *sequence* to be memorised. Some of the problems are common to many other sequences, for instance letters of the alphabet, digits in telephone numbers, the letters used to spell a word, sequences of sub-actions making up a learnt action (a dance-routine, or a method for testing faulty engines). Finally, I shall mention some problems peculiar to numbers, without offering more than tiny steps towards solutions.

There are many facts about number concepts and the ways in which they are used that I shall not attempt to analyse or explain. For instance, I shall say nothing about our ability to learn to generate an indefinitely extendable set of number names in a systematic fashion, or our ability to learn to think algebraically about numbers, for example in proving general truths about adding, subtracting, multiplying, etc., without mention of particular numbers.

In unravelling some of the hidden complexities in even the simplest abilities, I hope to give a feeling for the even greater complexities still to be explored. The intellectual tasks accomplished by ordinary children in apparently simple activities are comparable in complexity to some of the mental processes of adult scientists, engineers and artists. The children merely have less knowledge to build on.

If children have these impressive powers, why don't they use them to learn about arithmetic, reading, music, painting, and so on, without formal schooling, just as they learn to walk, talk and manipulate objects without formal schooling? Perhaps the answer is that despite all the variations in parental behaviour and home environment, nearly all children are placed in situations where learning to talk, walk, etc. are essential for them to achieve things they are highly motivated to do (like eating and interacting with other people), and moreover there are well-structured opportunities for them to learn, even though they learn things at different rates and in different orders. By contrast, very few parents and teachers are able to provide similarly highly structured and highly motivating situations to generate learning about reading, writing, mathematics, science, music, history, etc. One of the difficulties of investigating such issues without good theories of learning is that there are so many different factors which can make a difference in subtle ways. (Selfe 1977 presents relevant evidence in the drawings of an autistic child.)

8.5. *Knowing number words*

How is it possible for a child to learn to recognise sounds, like 'one', 'two', 'three', etc.? A simple-minded answer is that repeated exposure causes the sounds to be stored so that they can be reproduced and new occurrences recognised by matching them with stored ones. Immediately all sorts of questions need to be asked. In what form is the sound represented, that is, what exactly is stored? Is the sound analysed into recognisable fragments, such as phonemes? How are they recognised? Is some symbolic description of the sound stored, for example a description of the components of the sound? How does the child cope with variations in the sound? For instance, we may hear the same word produced by different people, or by the same person with different intonation contours or different pronunciations.

Does the stored description cope with all variations by making use of relatively abstract specifications (whatever that means)? Or does the child store different descriptions corresponding to different ways the sound may be uttered? In the former case, how does the child learn to use descriptions with sufficient generality, and in the latter case how does she represent the fact that the different descriptions are of *the same* word?

Or is some other method used to cope with variations, such as storing a specific description (a description of a 'prototype' or 'template'), and using a flexible matching procedure so that things not quite like it will match anyway? (This kind of 'sloppy matching' is often useful in computer programs.) Or, as Kant suggested in his discussion of schemata, do we cope with variations by using rules or procedures for synthesis and analysis rather than stored templates or descriptions? For instance, a rule which says 'count the number of consecutive occurrences of "ho" in an utterance and if the result is above two then call it a laugh' can enable one to recognize laughs' of very varied lengths. (I am not suggesting, and neither was Kant, that these matching and testing processes are conscious. In any case, we know so little about the difference between what we are and are not conscious of, that we cannot draw any useful conclusions from the fact that they are mostly unconscious processes.)

For a brief introduction to further complexities of recognition, see [chapter 9](#). The artificial intelligence literature takes the topic much further.

8.6. *Problems of very large knowledge stores*

We know that children learn many things. It is arguable that, using any reasonable method of counting facts, they learn millions, or at least hundreds of thousands of facts about possible appearances of things, about sounds, about possible movements, etc., in the first year or two. Given that there is a vast store of known items in a child's mind when she hears a word, what sort of process can quickly decide (not necessarily consciously) whether the word just heard matches something previously stored? Clearly a linear search through a list is out of the question, unless human minds have mechanisms which can work at far greater speeds than computers, which seems very unlikely. When listening to something quite novel, how does one decide, without searching the whole collection of stored items, that the sound just heard *does not* match anything already known?

These problems can be dealt with if the child not only stores items, but also builds an appropriate index to what it knows. For instance we use alphabetically ordered indexes to help us search books, libraries, department-stores, etc. (How? Think about how you might teach a child or a computer to use an alphabetic ordering to avoid a complete linear search.) What sorts of indexing techniques do children use, and how are they able to use them? Are we born with some sophisticated indexing strategies? Is it possible that children unconsciously use some kind of ordered set of symbols, like an alphabet, and build 'alphabetically' ordered or tree-structured catalogues of what they know, to

minimise searches?

Librarians and computer scientists do not find it easy to design good methods of cataloguing and indexing. Children must be much more sophisticated, although unconsciously.

Why don't we (and children) learn things permanently as soon as we hear them? Why is repeated hearing sometimes needed for learning? One popular answer is that memory uses probabilistic mechanisms, and that repeated exposure to an item increases the probability of its being retrieved later. How this happens is rarely explained. In any case, it does not seem to be consistent with the fact that not all learning requires repetition. If someone tells you that he plans to leave you a fortune, you will probably remember it for a long time without his having to repeat it. And faces seen once for a short time are often recognized long after, even if nothing else is recalled about the context in which they were first seen, though not all shapes are so easily remembered. So we do have some abilities to store things quickly and permanently: why are they not applied to everything we experience?

Here is a sketch of a non-probabilistic explanation of the need for repetition in some cases: the child needs to experiment with different ways of analysing, describing, and indexing new experiences. For example, it may be necessary to experiment with different ways of describing the sounds of words, so as to develop a good way to cope with variations in the sound of a word. It may even be necessary to experiment with ways of analysing a total experience before a particular sound pattern can be noticed as a significant substructure in any experience. Many adults have already developed good ways of indexing information about likely disasters and benefactions, so that they can store important items and access them later without repetition.

A closely related problem is worth mentioning. At any moment a child's experience is rich and complex. How are some features selected to be stored? How does the child decide what is worth learning? More fundamentally, how are some aspects of the current experience selected as candidates for things to be recognised if possible? How is a chunk of sound selected from the whole stream of sounds for an attempt to find a match among known items? To say that the child selects what 'interests' her is no explanation, since she can only decide that something is interesting after it has already been recognized. (Or at least some parts or aspects of it have been recognized.) These questions are taken up again in the chapter on visual perception.

8.7. Knowledge of how to say number words

Children learn not merely to recognise familiar words, but also to say them. How is the ability to say the word represented in the child's mind? Is it a set of instructions for the appropriate muscles? Or is there some representation of how the word sounds, and a general procedure which can examine a description of a sound sequence and generate appropriate instructions for muscles? This may be compared with the difference between compiled and interpreted computer programs.

Clearly we need some general 'interpretative' procedure in order to be able to repeat a sequence of sounds which we do not recognise, for instance when imitating someone talking a foreign language, where there is no question of simply repeating something learnt previously.

Perhaps there are good reasons, if there is no shortage of space, for storing both explicit instructions for producing the sound and the specification which allows the sound to be recognised. But this raises new problems. If the knowledge of how to say the word is represented differently from knowledge of what it sounds like, how are the two items related? In particular, how is the *appropriate* knowledge found when needed?

This is just a special case of a much more general problem about how one piece of information can have other kinds of information linked to it, or associated with it. Suppose you have managed to find in your memory something matching a word you have just heard. How does that help you access your knowledge of how to reproduce the word yourself?

8.8. Storing associations

One might think that an answer could take the form: if two items need to be associated so that when one is found then the other will be found too, then store them in adjacent locations in the memory space, just as books on related topics are often stored in adjacent locations in a library. The trouble with this is that each stored item may have to be associated with not just one, but with very many other items, and the associations can change with time.

For example, a child has to associate the sound of a number word not only with how to say it, but also with a method of writing it down, and a method of reading written versions of it. She may even learn to say or write it in several different languages, or several different notations for numerals. Moreover, as she learns more and more about numbers, she will have to associate lots more information with each number name, including: the fact that it is a number name, that it is a word associated with certain games (e.g. chanting things in sequence), the fact that its successor is so and so, that its predecessor is so and so, the fact that it is or it is not a prime number, the fact that it is odd, or even, its 'multiplication table', its 'addition table', and so on.

Of course, not only number names generate this problem. Many known items each have to be associated with a large and growing collection of other items. For instance, in your mind your home town will be linked to very many facts which you know about the town, such as its name, its location, its direction and distance (roughly) from major towns, its population, many of its geographical details, and so on.

So we have some new problems. First, if you cannot tell when you first learn a word, say 'three', how many further items of information are to be associated with it as a result of further learning, you cannot tell how much space to reserve in the neighbourhood of the location at which a description of the sound is stored. If too little is reserved, you'll find a limit to what you can learn about the number. But people do not seem to have such limits. (For instance, think of all the things associated with the word 'word' in your mind, i.e. all the words you know.) Is there an upper limit? Some people can learn several languages!

Of course, the need for expansion could be dealt with by moving the whole collection of linked items to a larger unoccupied space if the initially reserved space overflows. Are we to assume that children have the ability to manage storage allocation like this? For instance, do they have ways of telling which of the 'free' locations have a large enough collection of free neighbouring locations? Large enough for how many additional items? Is it possible that extra chunks of space are allocated in minimal units, as in some computer storage-allocation procedures? Perhaps people solve the problem by using an abstract symbolic space of locations, like the space of decimal numbers: this has the advantage that new neighbours can always be generated for any given location. But this merely shifts the problem, for we now have to explain how information is stored about which symbols occupy which locations!

Philosophers love to analyse the concept of rationality, and to discuss rational ways of doing things. But I have yet to hear them discuss what it means to have a 'rational' way of organising and using a massive store of knowledge, subject to the constraint that in real life decisions often need to be taken fairly quickly. Attempting to design a working system forces one to address such issues.

8.9. Controlling searches

When a recognized item *is* associated with more than one other item, and some task requires one of the associated items, then how is the *right* one found? If you hear me say 'Please write down the word three', then how do you find the relevant bit of knowledge associated with the sound 'three'? That is, how do you find the specifications for writing it as opposed to saying it, or as opposed to what its successor is, or whether it is odd or not?

Obviously this search has to be controlled by the request or question. For instance, in this case, the hearer has to find something associated not only with 'three' but also with 'write down'. How is this done? There are many techniques for this sort of thing which have been explored by computer programmers, and some of them are quite sophisticated. Do children have the ability to perform the elaborate operations used by such programmers, or do they have special techniques not yet discovered by programmers?

The problem is compounded by the fact that having learnt about some structure, we can then learn about a larger whole containing it as a part. You probably recognise not only the individual words, but also the whole phrase here: 'three blind mice'. Which method is used for obeying an instruction like

'count to three'?

Has the whole instruction been memorised and stored (e.g. because it is encountered frequently), or is there a process by which something associated with one of the words (e.g. 'three') is found because it is also associated with one of the others, or does something much more elaborate than retrieving a stored specification go on?

Is it possible that analysis of the instruction is somehow used to generate an action-specification? Quite likely we can do both, namely analyse the instruction using general principles and recognise it as a familiar whole. So how do we, and children, decide (unconsciously) which to do?

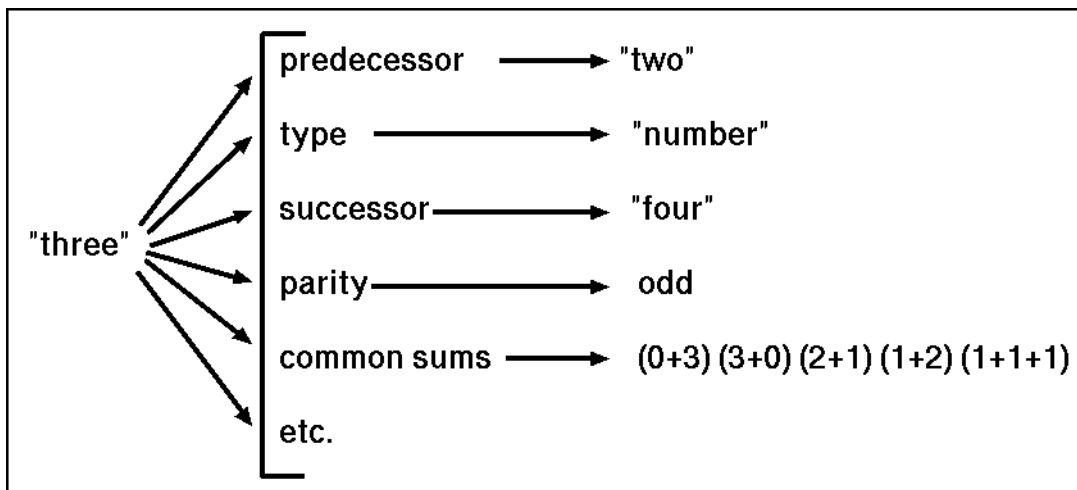
An explanatory theory, which purports to answer the questions raised here, must specify some kind of mechanism which is not merely able to hold learnt information in an efficiently accessible form, but is also capable of explaining how complex information structures are built up, how they are modified or replaced (e.g. when mistakes are discovered), and how they are used. I do not believe that educational psychologists have even the foggiest notion of what such a mechanism might be like, or what its limitations are, or what sorts of teaching strategies might interfere with its operation or facilitate learning. Some gifted teachers may have an intuitive grasp of some aspects of the mechanism, but they probably cannot articulate their implicit theories.

Computer scientists dealing with problems of managing complex collections of information in a flexible way seem to have unwittingly invented possible explanations some of which I sketch briefly below.

If we can find good theories, we may be able to do something about the large numbers of children who, for one reason or another, fail to learn so many things which might be useful or enriching to know. I believe that all normal children have the potential to learn a great deal of mathematics and other technical subjects painlessly, if only we knew how to prevent our teaching methods and attitudes to children (at home and in schools) from interfering with the learning process.

8.10. Dealing with order relations

A child can learn to answer the question 'What's after three?' How? The task is not merely to find something associated with both 'after' and 'three', since the word 'two' may also be associated with them, for three is after two. The child may also have learnt that 'five' and 'six' come after 'three' or, more specifically, that 'five' is two after 'three', 'six' is three after 'three', and so on.



Information to be stored about (associated with) "three"

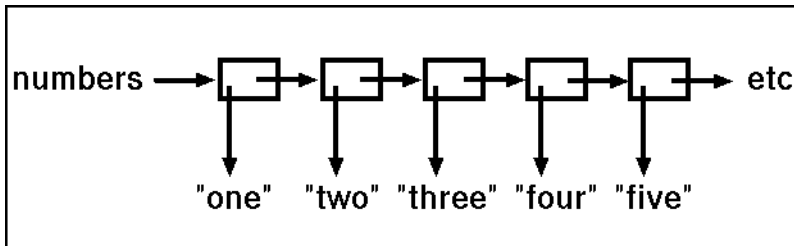
Figure 1

The problem cannot be solved by simply storing some such symbol as 'four is after three', that is, a representation of the required fact, since that would not always be the appropriate answer. For instance the question might be In the song *Ten green bottles* what comes after three? And if the context is unambiguous it is not even necessary to mention the song explicitly in the question.

So finding the required item of information may involve analysing the question in such a way as to control the search for *relevant* links in memory.

For example, it may be that each item which is associated with several others somehow has links to those others which are labelled as represented in Figure 1. It is very easy to draw diagrams like this, but not so easy to describe mechanisms which can build and use such structures. A common method used by programmers is that shown in Figure 2. A 'property-list' or 'association-list' is made up of a *chain of links* where each link contains two storage cells treated as an association by the memory mechanism, for example because they are adjacent in the memory space.

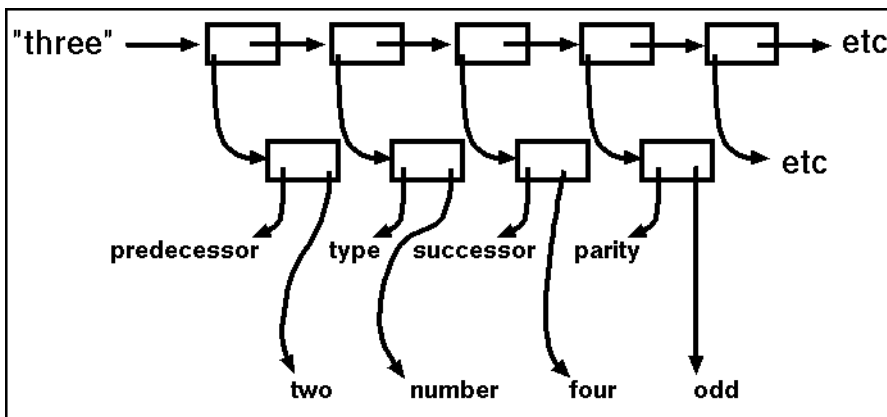
At least if the items associated with 'three' are all accessible through a linear list, then fairly obvious search procedures will enable the wanted item to be found, provided the location of the initial link of the list can be found easily.



Using a chain of two-element records to store information about order. Each link has two items of information (the main content of the link and where the next link is).

Figure 2

A chain of links may be attached to some item, for example the concept *numbers*, or the concept *three* with related items 'hung' from the chain by means of pointers giving their addresses. As Figure 3 shows, the items hung from the chain may themselves be associations, corresponding to the labelled links of Figure 1. Thus in the context of the chain attached to 'three', there is an association between 'predecessor' and 'two', whereas in a chain attached to 'four' (not shown) there would be an association between 'predecessor' and 'three'. Associations are relative to context.



Using two-element links to store the information in Figure 1.

Figure 3

Stored structures are not enough. Procedures are required for creating and finding associations in them. Such procedures are easily defined using modern programming languages. Suppose you want to search down a chain, starting from a specified link, looking for an association with a specified label (e.g. 'successor', or 'type'), because you want to find the item associated with that label in the chain. The obvious way is to see if the association pair pointed to by the given link starts with the required label, and if so to treat the second element of the pair as the desired result. Otherwise start again with the next link, whose address is in the BACK of the given link.

In a suitable programming language one could express this as Procedure-1, with the name ASSOC. (I am assuming that the subroutines FRONT and BACK when applied to a given pair produce the first thing and the second thing in the pair respectively. See Burstall *et al. Programming in POP2* for more details.)

Procedure ASSOC:

Given: initial LINK of chain, and target LABEL

Is FRONT of FRONT of LINK equal to LABEL?

If so, result is BACK of FRONT of LINK. STOP.

Otherwise, assign BACK of LINK to LINK, and restart, with LABEL as target.

Procedure-1

So ASSOC('THREE', 'TYPE') could represent the application of this procedure to a memory structure like Figure 3, with LINK starting as the first link in the chain called 'THREE', and LABEL having 'TYPE' as its value. The procedure would find a pointer to NUMBER as its result.

Similarly ASSOC('THREE', 'SUCCESSOR') would find the successor of 'three', namely 'four'. The same thing could then be used to find the successor of 'four', if that had been stored appropriately. By interleaving such searches with actions of saying what has been found, the child would have a procedure for counting, that is for reciting the numbers in their appropriate order. (More on the problems of interleaving later.)

Another way of thinking about this, is to say that information stored in a collection of structures like Figure 3, one for each known numeral, can be thought of as a sort of program for doing various things. The structure shown in Figure 2 is a much simpler program, and there is less that can be done with it. However using it as a program for counting is a simpler matter than using a collection of structures like Figure 3, since in Figure 2 all you need do in order to decide what to say next is find the link pointed to by the BACK of the current link in the chain, whereas in Figure 3 you first have to search for the 'successor' label, and then take the link it points to, and then start again from that link. We shall see later that different sorts of chains can coexist and be used for different purposes. (Figure 6)

Of course, there are many more structures and procedures that might be used for storing information about linear sequences in a computer, or in a mind. Different methods have their own advantages and disadvantages. For instance, the method of Figure 2, though simple and quick to use, has the disadvantage that when you get to the link involving 'three', there is no information stored there about items coming earlier in the chain. So using that structure makes it harder to answer questions like 'what comes before three?', though easier to answer questions like 'what comes after three?'

This is a space-time 'trade-off'. Other trade-offs involved in selecting representations include efficiency vs flexibility, simplicity of structures vs simplicity of procedures, and so on. Chapter 7 discussed trade-offs between Fregean and analogical representations. Investigations of such trade-offs between different representations is central to artificial intelligence but has hitherto been absent from philosophical discussions of rationality and most psychological theorising about cognitive processes.

Proposed explanations of a child's counting abilities must do much more than explain how the child manages to recite known numbers, or how the child answers simple questions. For example, it is necessary to explain also how the representation gets built up in the first place, how new items are added, and how mistakes are corrected. One may miss out an element of the sequence, or store some elements in the wrong order. So procedures are required for inserting new links, for deleting old ones, and perhaps for changing the order of existing links, when mistakes are discovered.

A more complex procedure is required for adding a new association: it will have to get a free link (how?) and insert it at a suitable place in the chain, with its FRONT pointing to the new association and its BACK pointing to the next link in the chain, if any.

If children do anything like this to store and use associations, then how do they build up such chains, and how do they come to know the procedures for finding required associations? Perhaps the ability to learn and use chains of associations, employing procedures something like ASSOC, is inborn? Clearly not all procedures for getting at stored information are innate. For instance, children have to *learn* how to count backwards or answer 'What's before "four"?' even though they may already know the order of the numbers. The same applies to other sequences children learn. (More about such tasks later.)

8.11. Control-structures for counting-games

All this points to the old idea (compare Miller *et al.*, 1960) that human abilities have much in common with computer programs. But further reflection on familiar facts shows that programs in the most common programming languages do not provide a rich enough basis for turning this from a thin metaphor into an explanatory theory.

For instance, people can execute unrelated actions in parallel, like walking and talking. Moreover, they apparently do not require their procedures to have *built-in* tests to ensure that conditions for their operation continue to be satisfied. Nor do they require explicit instructions about what to do otherwise, like instructions in a computer program for dealing with the end of a list. All sorts of unpredictable things can halt a human action at any stage (like learning one's house is on fire) and a decision about what to do can be taken when the interruption occurs, even if no explicit provision for such a possibility is built into the plan or procedure being executed.

These points suggest that models of human competence will have to use mechanisms similar to operating systems for multi-programmed computers. For instance, an operating system can run a program, then interrupt it when some event occurs although the program itself makes no provision for interruption. Similarly, if something goes wrong with the running of the program, like an attempt to go beyond the end of a list, the program breaks down, but the operating system or interpreter running the program can decide what to do, (for example, send a message to the programmer), so that there is not a total breakdown. Of course the operating system is just another program.

So the point is simply that to make the program metaphor fit human abilities we must allow not merely that one program can use another as a 'subroutine' but that some programs can execute others and control their execution, in a parallel rather than a hierarchic fashion. (For more on this, see chapters 6, 9 and 10.)

8.12. Problems of co-ordination

In counting objects, a child has to be able to generate different action sequences in parallel, keeping them in phase. Thus the process of saying number names, controlled by an internal structure, and the process of pointing in turn at objects in some group, the latter process being controlled by the external structure, have to be kept in phase. In a suitable programming language one could keep two processes in phase by means of a procedure something like the procedure COEXECUTE

Procedure COEXECUTE:

Given: step-by-step procedures P1 and P2,

Execute a step of P1.

Execute a step of P2.

**Has a stopping condition been reached?
If not, restart COEXECUTE (P1, P2).**

Procedure-2

Unfortunately, this is not an acceptable model in view of the familiar fact that children (and adults doing things in parallel) sometimes get out of phase when counting and (sometimes) stop and correct themselves. This suggests that keeping the two sequences in phase is done by a third process something like an operating system which starts the processes at specified speeds, but monitors their performance and modifies the speeds if necessary, interrupting and perhaps restarting if the sequences get out of phase. All this would be impossible with the procedure COEXECUTE. It is as if we could write programs something like the procedure RUNINSTEP.

Procedure RUNINSTEP:

Given: procedures P1 and P2,

DO (a) to (d) in parallel:

(a) repeatedly do P1

(b) repeatedly do P2

(c) observe whether (a) and (b) are getting out of step and, if they are, slow one down or speed up the other.

(d) if (a) and (b) are right out of step re-start P1 and P2

Procedure-3

The computational facilities required for this kind of thing are much more sophisticated than in COEXECUTE and are not provided in familiar programming languages.

[[**Note added January 2002** The ability to monitor and modify two concurrently executed processes requires an information processing *architecture* which is not supported by the virtual machines defined by most programming languages, though it is a feature of operating systems. This is one of the sorts of tasks that might be required in a "meta-management" system, described in the presentations here: <http://www.cs.bham.ac.uk/research/projects/cogaff/talks/>]]

Notice also that there is a complex perceptual task involved in deciding whether two processes are getting out of step, and children sometimes find this difficult. Not only children: try counting rotations of a wheel with no clear markings on it, while it is turning quite rapidly!

Further, the child has to be able to apply different stopping conditions for this complex parallel process, depending on what the task is. So it should be possible for yet another process to run the procedure RUNINSTEP, watching out for appropriate stopping conditions. Alternatively, the procedure could be re-defined so as to have an additional 'given', namely a stopping condition, and an extra sub-process, (e), watching out for it. For instance, when the question is 'How many buttons are there?' use 'No more buttons' as main stopping condition, whereas in response to a request 'Give me five buttons', use 'Number five reached' as main stopping condition.

I say *main* stopping condition, because other conditions may force a halt, such as getting out of phase or running out of numbers or (in the second case) running out of buttons.

How do children learn to apply the same process with different stopping conditions for different purposes? How is the intended stopping condition plugged into the process? Notice that the perceptual tasks are further complicated by the need to detect different sorts of conditions, for example,

completion of the task, getting out of phase, running out of things to count, mistakes like counting the same thing twice, or leaving something out, and so on.

Some of this would be easy for a programmer using a high-level language in which a procedure (to test for the stopping condition) can be given as input to another procedure but do children have such facilities, or do they use mechanisms more like the parallel processes with interrupt facilities described here?

I believe we know very little about how children achieve these extraordinarily complex feats. Nor do we understand what sorts of teaching strategies can help or hinder their development. My own informal observations suggest that a tremendous amount of very varied practice is required, in an environment where teachers can use a deep analysis of failures to suggest variations in games and other learning activities. This analysis can be a challenging intellectual task. How many teachers are equipped for it?

The parallel mechanisms suggested above might explain the ability to learn to watch out for new kinds of errors. For example, after learning to count stairs, where there is little chance of counting an item twice, learning to count buttons or dots requires learning to monitor for repetition and omission.

Depending on the kind(s) of programming language(s) and operating system(s) used in a child's mind, it may be easier to add a new kind of monitoring process to run in parallel with previously learnt processes than to re-organise an existing procedure so as to include new tests at appropriate places, as would be required with a conventional programming language. Probably both sorts of learning occur.

Monitoring interactions between asynchronous parallel processes may be an important source of accidental discoveries (creativity) in children and adults. For example, ongoing (unconscious) comparisons between intermediate results of two different activities may lead one to notice a relation between the two which amounts to a new technique, concept, or theory. This whole discussion is centrally relevant to the analysis of concepts like consciousness, attention, and intention. We now have a basis for a complete rejection of a major theme of Ryle's pioneering work *The Concept of Mind*, namely its refusal to accept multiple inner mental processes.

We also have a basis for beginning to explore personality differences and mental disorders relating to problems of organising and controlling several different processes. By trying to design systems involving multiple interacting processes we gain a deeper understanding of the problems and possibilities.

8.13. Interleaving two sequences

If we consider what happens when a child learns to count beyond twenty, we find that a different kind of co-ordination between two sequences is required, namely the sequence 'one, two, three . . . nine' and the sequence 'twenty, thirty, . . . ninety'. Each time one gets round to 'nine' in the first sequence one has to find one's place in the second sequence so as to locate the next item. The same is true of counting backwards from a large number. (The rules in different languages are slightly different, but the general principles are apparently the same in most.)

A programmer would find this trivial, but how does a child create this kind of interleaving in his mind? And why is there sometimes difficulty over keeping track of position in the second sequence '... fifty-eight, fifty-nine, . . . um . . . er, thirty, thirty-one . . . '? Clearly this is not a problem unique to children. We all have trouble at times with this sort of book-keeping. But how is it done when successful? And what kind of mechanism could be successful sometimes yet unsuccessful at others?

My guess is that human fallibility has nothing to do with differences between brains and computers as is often supposed, but is a direct consequence of the sheer complexity and flexibility of human abilities and knowledge, so that for example there are always too many plausible but false trails to follow. When computers are programmed to know so much they will be just as fallible, and they will have to improve themselves by the same painful and playful processes we use.

8.14. Programs as examinable structures

We have noted a number of familiar aspects of counting and other actions which suggest that compiled programs in commonly used programming languages do not provide a good model for human abilities. A further point to notice is that we not only *execute* our procedures or programs, we also *build them up* in a piecemeal fashion (as in learning to count), *modify* them when they seem inadequate, and *examine* them in order to anticipate their effects without execution. We can decide that old procedures may be relevant to new problems, we can select subsections for use in isolation from the rest, and we may even learn to run them backwards (like learning to count backwards).

This requires that besides having names and sets of instructions, procedures need to be associated with specifications of what they are for, the conditions under which they work, information about likely side-effects, etc. The child must build up a *catalogue* of his own resources. This is already done in some A.I. programs, e.g. Sussman, 1975.

Further, the instructions need to be stored in a form which is accessible not only for execution but also for analysis and modification, like inserting new steps, deleting old ones, or perhaps modifying the order of the steps, as is done in Sussman's program. Such examination and editing cannot be done to programs as they are usually stored, after compilation.

List structures in which the order of instructions is represented by labelled links rather than implicitly by position in memory would provide a form of representation meeting some of these requirements (and are already used in some programming languages). Thus, as already remarked, Figure 2 can be thought of either as a structure storing information about number names (an analogical representation of their order), or else as a program for counting. The distinction between data structures and programs has to be rejected in a system which can treat program steps as objects which are related to one another and can be changed. We now explore some consequences of this using counting as an example.

8.15. Learning to treat numbers as objects with relationships

There are several ways in which understanding of a familiar action sequence may be deficient, and may improve. One may know a sequence very well, like a poem, telephone number, the spelling of a word, or the alphabet, yet have trouble reciting it backwards. One may find it hard to start from an arbitrary position in a sequence one knows well, like saying what comes after 'K' in the alphabet, or starting a piano piece in the middle. But performance on these tasks can improve.

A child who counts very well may be unable easily to answer 'What comes after five?'. Later, he may be able to answer that question, but fail on 'What comes before six?', 'Does eight come earlier or later than five?' and 'Is three between five and eight?'. He does not know his way about the number sequence in his head, though he knows the sequence.

Further, he may understand the questions well enough to answer when the numbers have been written down before him, or can be seen on a clock. (There are problems about how this ability to use what you see to answer such questions is learnt, but I shall not go into them.)

Later, the child may learn to answer such questions in his head, and even to count backwards quickly from any position in the sequence he has memorised. How? To say the child 'internalises' his external actions (an answer I have often been given in the past) is merely to label the problem, if all that is intended is the claim that one can learn to represent in one's mind actions previously performed externally.

Moving back and forth along a chain of stored associations is quite a different matter from moving up and down staircases or moving one's own eye or finger back and forth along a row of objects. The latter is a physical movement through space, whereas the former is movement through a set of computational states, not necessarily involving physical movement. Lack of reversibility in one case may be accounted for by physical structures, like ratchets or uni-directional motors, whereas in the other case the explanation is more likely to be lack of information. For instance in Figure 2, the link pointing to 'four' contains information about the next link, but does not contain information about the preceding link.

Learning to overcome physical impediments to reversibility need have nothing in common with learning to overcome computational problems. The child who has learned to move his eye or finger back and forth around a clock face to answer 'what comes before four' is not thereby provided with a mechanism which could somehow be used internally. At most, it provides him with a model, or analogy, which may be helpful in grasping what the task is. But how the analogy is used is totally unexplained.

8.16. Two major kinds of learning

There are at least two important kinds of development of knowledge about a previously stored structure (which may be a program), namely

- a) learning new procedures for doing things with the structure
- b) extending the structure so as to contain more explicit information about itself.

The former will tend to be involved in learning to do new kinds of things with the stored information, whereas the latter may simply be a matter of improving the efficiency with which old tasks are performed. But this in turn may facilitate the learning of quite new tasks which depend on rapid and skilful execution of sub-tasks using previous skills. (This bears on the debate about formal and informal methods of teaching in schools.)

A very simple procedure enables a chain like that in Figure 2 to be used to generate a sequence of actions, for example the procedure RECITE.

Procedure RECITE.

Given. a chain starting from LINK.

Utter FRONT of LINK.

If BACK of LINK isn't empty, make it LINK and restart.

Otherwise stop.

Procedure-4

Going down the chain starting from a given link is thus easy, and a procedure to find the successor of an item would use a similar principle. But answering 'What's before item X?' is more sophisticated, since on getting to a particular location (e.g. the link whose FRONT points to X), one does not find there any information about how one got there. Somehow the last item found must be stored

temporarily. One method is illustrated in the procedure PREDECESSOR, as it might be defined in a programming language.

Procedure PREDECESSOR:

Given target X in chain starting at LINK:

Create temporary store TEMP, with undefined value.

Repeat the following:

If FRONT of LINK = X then result is TEMP, stop.

Otherwise, assign FRONT of LINK to TEMP and BACK of LINK to LINK and restart.

Procedure-5

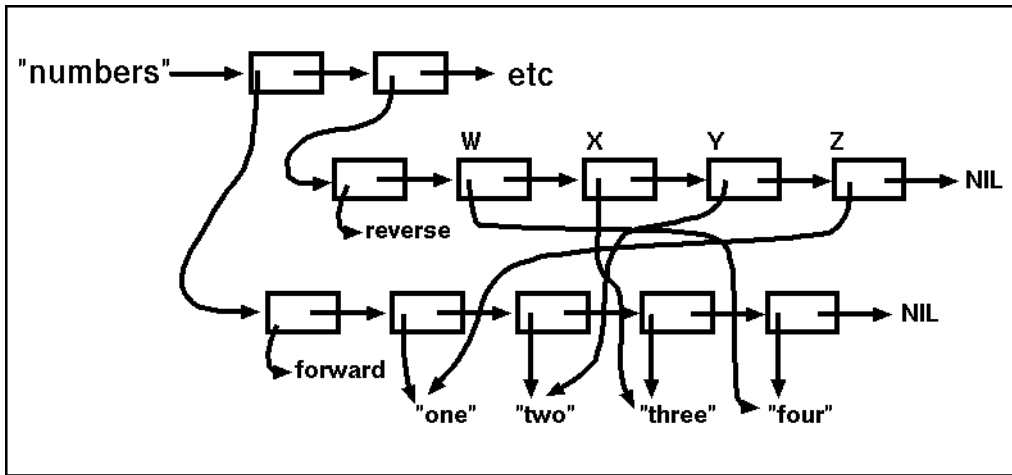
How could a child learn to create a procedure like this? Does he start with something more specialised, then somehow design a general method which will work on arbitrary chains? Perhaps it has something to do with manipulating rows of objects and other sequences outside one's head, but to say this does not give an explanation, since we do not know what mechanisms enable children to cope with external sequences, and in any case, as already remarked, chains of associations have quite different properties. For a child to see the analogy would require very powerful abilities to do abstract reasoning. Maybe the child needs them anyway, in order to learn anything.

My observations suggest that the child's learning task (at least between the ages of three and four, or later) is very different from the task of designing a procedure like Procedure-5. This is because the child is already able to remember steps he has just executed. So if he is asked to count to 'four' and does so, and then is immediately asked what came before 'four' he can answer. He does not have to allocate special-purpose temporary storage, like the 'local variable' TEMP. His problem is *to think of counting up to four* as a way of answering the question 'What comes before four?'

He does not, presumably, have a representation of the fact that if he recites some sequence he can remember the final fragment immediately after stopping. Adults have learnt this and can use it to answer a question about the predecessor of a letter of the alphabet, even if they do not have the information explicitly stored. However this technique is very tedious for reciting the whole of a learnt sequence backwards, and is useless if the process is to be done quickly. (The general ability to remember what one has just done is useful for the reasons given in chapter 6. The reasons apply to intelligent artefacts as well as to humans. This self-monitoring is not usually a built-in feature of programming systems, but there is no difficulty of principle in incorporating it.)

8.17. Making a reverse chain explicit

Merely being able to invent some procedure for finding the predecessor of a number is not good enough. For some purposes, such as counting backwards quickly, we want to be able to find the predecessor or successor of an item much more quickly than by searching down the chain of links until the item is found.

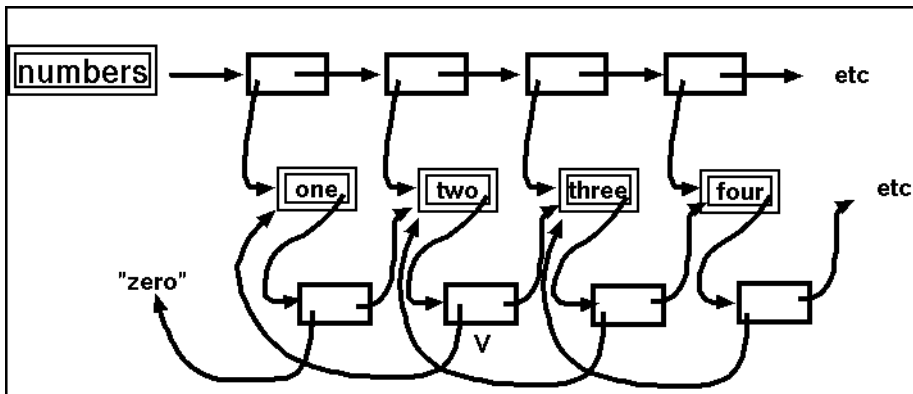


Two co-existing chains record different orders for the same set of items.

Figure 4

If a child knew only the first four numbers, then he could memorise them in both directions, building up the structure of Figure 4 instead of Figure 2. Notice that this use of two chains increases the complexity of tasks like 'Say the numbers', or 'What's after three?', since the right chain has to be found, while reducing the complexity of tasks like 'Say the numbers backwards' and 'What's before three?'. (Another example of a computational trade-off.)

However, when a longer sequence had been learnt, this method would still leave the need to search down one or other chain to find the number N in order to respond to 'What's after N?', 'What's before N?', 'Count from N', 'Count backwards from N', 'Which numbers are between N and M?', etc., for there is only one route into each chain, leading to the beginning of the chain. For instance, when one has found the link labelled 'X' in Figure 4, one knows how to get to the stored representation of 'three'. But it is not possible simply to start from the representation of 'three' to get to the links which point to it in the two chains. So we need to be able to associate with 'three' itself information about where it is in the sequence, what its predecessor is, what its successor is, and so on.



A chain represents the order of number names.
 Additional links make predecessor and successor information explicit.
 E.g. box V has pointers to predecessor and successor of "two".
 (Double boxes are directly accessible from a central index of names.)

Figure 5

A step in this direction is shown in Figure 5, where each number name is associated with a link which contains addresses of both the predecessor and the successor, like the link marked V, associated with 'two'. The information that the predecessor is found in the FRONT and the successor found in the BACK would be implicit in procedures used for answering questions about successors and predecessors. However, if one needed to associate much more information with each item, and did not want to be committed to having the associations permanently in a particular order, then it would be necessary to label them explicitly, using structures like those in Figure 1 and Figure 3, accessed by a general procedure like ASSOC, defined previously.

To cut a long story short, the result of explicitly storing lots of discoveries about each number, might be something like the network of associations in Figure 6, which is highly redundant, in the sense that a lot of the information there could in principle be derived from other information in the network. On the left there are (vertical) chains of links available for use in counting rapidly forwards or backwards, analogous to the chains in Figures 2 and 4. In addition, associated with each number is a great deal of information about it in a chain of attribute/value pairs analogous to Figure 3, except that some of the values are new sub-chains. Included in the chain hanging from each number is a pointer into the 'fast-forward' chain and a pointer into the 'fastback' chain, making it possible to count quickly forwards or backwards from that number without first having to search for the number in the relevant 'fast' chain.

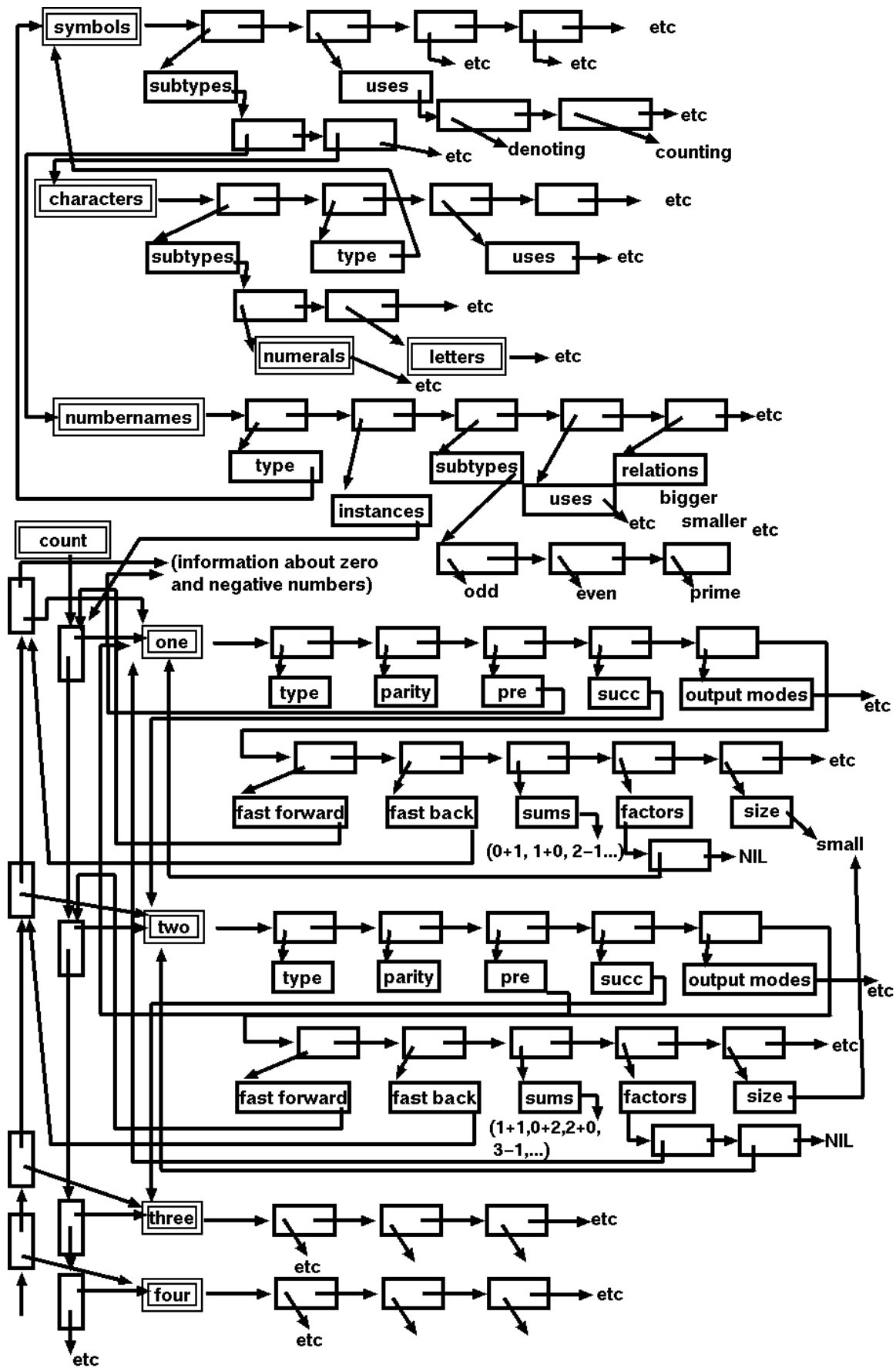
In the light of the previous remarks about the need to blur the distinction between information-structures and programs, we can see how a structure like that depicted in Figure 6 can be thought of as containing several different programs embedded within it, such as programs for counting forward from various numbers, programs for counting backwards from various numbers, programs for answering questions, and so on. The different programs share common sub-structures.

The growth of this kind of network would be an example of the second type of learning, namely extending an information store to contain explicitly what was previously implicit in it. This often involves trading space for time. That is, much redundant information is stored explicitly so that it does not have to be re-computed every time it is needed. This includes information on how to do things. It seems that a great deal of early learning about numbers has this character, as well as much of the development of skill and fluency in thinking and acting.

"Progressive" educational procedures which attempt to do without any rote learning may be depriving children (or adult learners!) of opportunities to build up some structures which are useful for rapid access -- unless the old formal methods are replaced with carefully structured play situations, to achieve the same effect (which they could probably do much more effectively, since they would be more highly motivating.) Children need a lot of practice at 'finding their way about' their own data-structures.

The structure of Figure 6 may look very complex, yet using it to answer certain questions requires simpler procedures than using, say Figure 2. For, having found the link representing a number, one can then find information associated with that number by simply following forward pointers from it, for example, using a procedure like ASSOC; whereas in Figure 2 or 5, finding the predecessor and successor of a number requires using two different procedures, and each requires a search down a chain of all the numbers to start with. Of course, a structure like Figure 6 provides simple and speedy access at the cost of using up much more storage space. But in the human mind space does not seem to be in short supply!





**Using chains of associations to represent what a child must learn about numbers.
(Double boxes are directly accessible from a central index of names.)**

Figure 6

A further problem is that each time new information is added to a chain, the increased length increases the average time for searching along the chain. So if an item in a structure like Figure 6 has a very long chain of associations, it might be preferable to replace the linear chain with a local index to avoid long searches. So, instead of 'three' being linked to a linear list of associations, it would have some kind of structured catalogue. Someone who knew a very large number of things about 'three' might find that this saved time searching for information. This would require the procedure ASSOC to be replaced by something more sophisticated, and would probably also require more space. Alternatively, by switching pointers, one could easily bring a link to the front of the chain each time the association hanging from it is used: this would ensure that most recently and most frequently used information was found first, without the help of probabilistic mechanisms, often postulated to explain such phenomena.

8.18. Some properties of structures containing pointers

Notice that in a structure like Figure 6, normal 'part-whole' constraints are violated: information about numbers is part of information about 'three', and *vice versa*. So by using pointers (addresses) we can allow structures to share each other. In a rich conceptual system circular definitions will abound. If much of our knowledge is non-hierarchic, as this suggests, then perhaps strictly cumulative educational procedures designed to achieve complete clarity at every stage are quite misguided. So also are philosophical investigations of knowledge in the tradition of Descartes, trying to show how everything can be based on completely rational chains of inference starting from self-evident, or at least minimally doubtful, premisses.

(Perhaps only trivial things can be taught without generating a great deal of confusion. Infants learning to speak experience a great deal of confusion, but this does not usually make them give up! Only later on do we teach them to give up too soon, by labelling them as 'stupid', for example, or perhaps by helping them too often when they are in difficulty.[1])

This kind of circularity (or mutual recursion) is especially common in our mental concepts. For instance, the concept of 'belief cannot be analysed without reference to the concepts of desire and decision, and these cannot be analysed without reference to each other and the concept of belief. Yet ordinary people learn to use these concepts in their ordinary life (for instance, when they explain someone's action in terms of a belief: 'He did it because he believed I was out to get him'). We learn to use mutually recursive concepts without being at all aware of their complexity.

In my experience philosophers and psychologists tend to get very confused about how to deal with this kind of circularity, for example in discussing varieties of Behaviourism. Analogies with recursive computer programs and data-structures can help to clarify the issue. One can distinguish varieties of behaviourism according to whether they will tolerate recursion (especially mutual recursion) in their definitions of mental concepts. Ryle's book *The Concept of Mind* was more sophisticated in this respect than most other forms of behaviourism, since it implicitly allowed mutually recursive definitions of mental concepts, implying that mentalistic concepts cannot all be eliminated by analysis in terms of dispositions to respond to stimuli. This, presumably, explains why Ryle did not see himself as a behaviourist.

The kind of structure depicted in Figure 6 does not need a separate index or catalogue specifying where to look for associations involving known items, for it acts as an index to itself, provided there are some ways of getting quickly from outside the structure to key nodes, like the cells containing 'three' and 'number'. (This might use an index, or content addressable store, or indexing tricks analogous to hash coding, for speedy access.)

The use of structures built up from linked cells and pointers like this has a number of additional interesting features, only a few of which can be mentioned here. Items can be added, deleted, or rearranged merely by changing a few addresses, without any need for advance reservation of large blocks of memory or massive shuffling around of information, as would be required if items were stored in blocks of adjacent locations (another trade-off: space against flexibility).

The same items can occur in different orders in different structures which share information (see Figure 4 for a simple example). Moreover, the order can be changed in one sequence without affecting another which shares structure with it. For instance, in Figure 4 the addresses in links W, X, Y, and Z can be changed so as to alter the order of numbers in chain labelled 'reverse' without altering the chain labelled 'forward'.

As we saw in connection with Figure 2, when the rest of the mechanism is taken for granted, a structure of the kind discussed here looks like a program for generating behaviour, but when one looks into problems of how a structure gets assembled and modified, how parts are accessed, how the different stopping conditions are applied, etc., then it looks more like an information structure used by other programs.

8.19. Conclusion

Further reflection on facts we all know reveals many gaps in the kinds of mechanisms described here. For instance, very little has been said about the *procedures* required for building, checking, modifying, and using a structure like Figure 6. Nothing has been said about the problems of perception and conception connected with the fact that counting is not applied simply to bits of the world but bits of the world individuated according to a concept (one family, five people, millions of cells but the same bit of the world counted in different ways).

I have offered no explanation of the ability to answer 'How many?' questions by recognising a visual pattern, without explicit counting. Obviously, there is a lot to be said about the development of new perceptual abilities related to numbers, for example the ability to perceive groups of three objects without counting, by matching against a structural definition, much as one recognises arches, letters and horses (see chapter 9 and Winston, 1975).

Nothing has been said about how the child discovers general and non-contingent facts about counting, such as that the order in which objects are counted does not matter, rearranging the objects does not matter, the addition or removal of an object must change the result of counting, and so on. How does a child come to grasp the fact that in principle counting can go on indefinitely, so that its stock of number names may need to be extended, or replaced by a rule with unlimited generative power?

(Philosophers' discussions of such non-empirical learning are usually so vague and abstract as to beg most of the questions. Piagetian psychologists comment on some of the achievements, but provide no means of analysing or explaining the underlying mental processes discussed here.)

I cannot explain these and many more things that even primary school children learn. I do not believe that anybody has even the beginnings of explanations for most of the things we know they can (sometimes) do: all we find is new jargon for labelling the phenomena.

I have offered all this only as a tiny sample of the kind of exploration needed for developing our abilities to build theoretical models worth taking seriously. In the process our concept of mechanism will be extended and the superficiality of current problems, theories and experiments in psychology and educational technology will become apparent.

Philosophers have much to learn from this sort of exercise too, concerning old debates about the nature of mind, the nature of concepts and knowledge, varieties of inference, etc. Consider my short survey of answers they have given to the question 'What are numbers?' The answers do not begin to match the complexity of what a child has to grasp in learning about numbers. They do not account for the fact that number concepts are used in a variety of activities. They perhaps take the uses for granted, but make no attempt to explain how they are possible. Philosophers of the future, who have a much better grasp of what such explanations might look like, will be in a better position to formulate adequate analyses.

Similarly, when they have learnt about possible mechanisms underlying processes of inference and discovery, they will be in a better position to discuss the nature of mathematical discovery and other forms of *a priori* learning. The most that can be said at present is that it will probably prove helpful to think of mathematical discovery by analogy with a program which discovers new facts about itself by a combination of executing parts of itself and examining some of its instructions. In the process it might decide that some things could be done more quickly in a different way. Or it might discover, by analysing its own structure, that instead of executing bits of programs, it can work out their effects by reasoning about them.

More importantly, it may discover ways of generalising and extending its procedures to accomplish more tasks of the same sort, or new kinds of tasks. Programmers often discover unexpected ways of elaborating and generalising their programs, in the course of examining and using them, much as an artist learns more about what he can and should do by examining an incomplete work. A program which builds its own programs can do this too. Sussman's 'Hacker' program (1975) builds programs, and, in some cases, generalises them.

I believe that similar ideas are to be found in Piaget's writings. Computer models turn such thoughts from vague speculations to testable theories. See Young, 1976.

These sorts of second-order discoveries about one's own procedures do not fit the normal definition of 'empirical'. For example, they need not involve the use of the senses to gain information about the world. And a kind of necessity seems to be involved in the truths so discovered which is not normally thought to be compatible with empirical learning: if experience can lead us to a hypothesis can it not also produce a refutation of the hypothesis? But it seems that no experience can refute the claim that adding two lots of two things produces the same result as adding one thing to a group of three things, or the claim that there is no largest number. And the same is true of many other discoveries about properties of the procedures we use. Yet such mathematical discoveries involve a kind of exploration of possibilities which is closely analogous to empirical learning.^[2]

We need a richer set of distinctions than philosophers normally employ. There is learning from sensory experience and learning from symbolic experience. The latter seems to include the processes generating what Kant called 'synthetic *a priori* knowledge'. However these processes require a great deal of further investigation. In particular, it is important to note that symbolic experiences may occur either entirely within the mind, or else may use external symbolisms, as when we use diagrams or

calculations on paper. The use of our senses to examine our symbols and our procedures for manipulating them should not be confused with the use of our senses to examine the behaviour of objects in the world.

The task of designing programs which simulate these sorts of human learning to a significant extent is at the frontiers of current research in artificial intelligence. Until further progress has been made, philosophical speculation about non-empirical knowledge is likely to remain as unproductive as it has been through most of history.

The old nature-nurture (heredity-environment) controversy is transformed by this sort of enquiry. The abilities required in order to make possible the kind of learning described here, for instance the ability to construct and manipulate stored symbols, build complex networks, use them to solve problems, analyse them to discover errors, modify them, etc., all these abilities are more complex and impressive than what is actually learnt about numbers! Where do these abilities come from? Could they conceivably be learnt during infancy without presupposing equally powerful symbolic abilities to make the learning possible? Maybe the much discussed ability to learn the grammar of natural languages (cf. Chomsky, 1965) is simply a special application of this more general ability? This question cannot be discussed usefully in our present ignorance about possible learning mechanisms.

Finally a question for educationalists. What would be the impact on primary schools if intending teachers were exposed to these problems and given some experience of trying to build and use models like Figure 6 on a computer? Our experience of teaching philosophy and psychology students computing in the Cognitive Studies Programme at Sussex University, and similar experiences at other centres, such as Edinburgh University and Massachusetts Institute of Technology, suggests that it can produce a major transformation of outlook including a new respect for the achievements of children. Here is a tremendous opportunity for educational administrators and teacher-training institutions. Will they grasp it?[3]

NOTE added Oct 2001:

At the time I was writing this chapter I was aware that there were many people trying to explain learning in terms of probabilistic associations. Although this seemed to be a good description of some of the intermediate stages in learning about numbers, e.g. before a child is reliably able to recite the number sequence, probabilistic associations did not seem to characterise adequately the rich and precise grasp of structure that comes with a deep understanding of the world of numbers, including not only the ability to answer the simple sorts of questions discussed in this chapter, but also the ability to think about infinite sets (e.g. the set of even numbers or the set of prime numbers) the ability to discover new regularities in the structure, the ability to invent new, provably correct procedures e.g. for doing long multiplication or finding square roots, and so on.

Part of all this is the ability to understand the difference between the *necessary, exceptionless*, truths of number theory, such as that seven plus five equals 12, or that the cardinality of a set is independent of the order in which the elements are counted and merely *contingent* truths, such as that if you put one rabbit in an empty hutch and then later add another rabbit, and do not put any additional rabbits in the hutch there will be only two rabbits in the hutch thereafter.

In other words I was convinced that although the processes involved in learning some of the basic features of the number system, including the names for the numbers and their order, might use probabilistic mechanisms (such as the neural nets that became popular in the two decades following publication of this book), this could not be the key either to the nature of our mathematical knowledge, or to many other features of our knowledge of the world, such as understanding how a clock works, or why turning a handle can enable a door to be opened, or why it is necessary to open the door in order to go into a room. When we understand these things we do not merely understand probabilistic associations, we understand *structural relationships*.

By the early 1970's there had already been some deep work in AI investigating structure-based learning and understanding, e.g. the papers in Minsky's 1968 collection, and Sussman. (Progress was very slow, however, because of the extremely limited speeds and memory capacities of computers of the time, but more importantly because the sheer difficulty of the problems.)

When I wrote this chapter, I was attempting to generalise some of that early work by exploring the notion that a human's ability:

- a) to construct and then inspect and manipulate list structures (or similar structures found in computational virtual machines)
 - b) to inspect and manipulate the procedures for operating on those structures
 - c) to run processes in parallel, including processes observing and modifying other processes,
- could explain a wider range of phenomena than mere learning of associations could.

I also suggested that if some of the list structures did not have a fixed order but were re-linked, e.g. bringing more recently accessed items closer to the front, then that could explain some of the variability in performance that others had assumed must be explained by probabilistic mechanisms.

In retrospect, it seems that a mixture of the probabilistic and deterministic approaches is required, within the study of *architectures* for complete agents: a more general study than the investigation of algorithms and representations that dominated most of the early work on AI (partly because of the dreadful limitations of speed and memory of even the most expensive and sophisticated computers available in the 1960s and 1970s).

There are many ways such hybrid mechanisms could be implemented, and my recent work on different processing layers within an integrated architecture (combining reactive, deliberative and meta-management layers) indicates some features of a hybrid system, with probabilistic associations dominating the reactive layer and structure manipulations being more important in the deliberative layer. For recent papers on this see the Cogaff papers directory <http://www.cs.bham.ac.uk/research/cogaff/> and my "talks" directory: <http://www.cs.bham.ac.uk/research/projects/cogaff/talks/>

More specific though less comprehensive models have been proposed by other researchers, one of the most impressive being the ACT-R system developed by John Anderson and his collaborators. See <http://act.psy.cmu.edu/>.

End notes

[*]Note: This is a modified version of a paper with the same title presented to the AISB Summer conference, in July 1974, at The University of Sussex.

Most of the content was inspired by my interactions with Benjamin Sloman while he was learning to think about numbers, aged about 5 years, during a year (1972-3) when I was visiting the University of Edinburgh, aged about 36. I was learning to think about information structures, programs and architectures while he was learning to think about numbers (and many other things.)

We both learnt an enormous amount that year. Trying to understand his development, and ways in which it could be influenced (programmed?) helped to convince me that AI was at least *beginning* to produce theories of the right general sort, though still lacking in detail and comprehensiveness. (Ben was born in November 22nd 1967 and died 2nd February 2002)

(1) For instance, it might be the case that because of the differences between human nipples and the rubber variety, infants who are breast-fed develop a better grasp of some basic principles of physics and mechanics at a very early age because they have to surmount more obstacles to get their milk! This could also develop perseverance, independence, the ability to cope with frustration, etc. Such differences may, however, easily be counteracted by other factors in the environment.

(2) See Pylyshyn 1978, Sloman 1978.

(3) **[[Note added in 2001, about 25 years after the above was written:**

Alas it seems that too much of education regarding use of computers in schools is now just another case of teaching students to be passive users of complex systems others have produced, like teaching them to drive cars. The opportunity to use computers in the way that meccano sets can be used for educational purposes, has largely been ignored.

If it had not been ignored, children would be learning how to design, implement, test, debug, analyse, describe, explain and criticise increasingly complex systems. Their minds are not being trained to deal with all the complex systems they will encounter in real life, including social systems, political systems, economic systems, computing systems, and human minds, including their own. If it had not been ignored, people starting training to be psychologists would have had experience of building and testing systems that manipulate and use information structures far more complex than the one depicted in Figure 6.

I often use the phrase "mouse potato" by analogy with "couch potato" used to describe passive watchers of television programmes. The generation of couch potatoes is educating a generation of mouse potatoes.]]

[Book contents page](#)

[Next: Chapter 9](#)

Last updated: 4 Jun 2007; 19 Sep 2010