Supervenience and Implementation: Virtual and Physical Machines

Aaron Sloman[1]

Abstract. How can a virtual machine X be implemented in a physical machine Y? We know the answer as far as compilers, editors, theorem-provers, operating systems are concerned, at least insofar as we know how to produce these implemented virtual machines, and no mysteries are involved. This paper is about extrapolating from that knowledge to the implementation of minds in brains. By linking the philosopher's concept of supervenience to the engineer's concept of implementation, we can illuminate both. In particular, by showing how virtual machines can be implemented in causally complete physical machines, and still have causal powers, we remove some philosophical problems about how mental processes can be real and can have real effects in the world even if the underlying physical implementation has no causal gaps. This requires a theory of ontological levels.[2]

1 Introduction

Consider these two questions:
(a) How can a virtual machine (X) be implemented in a physical machine (Y)?
(b) How can psychological processes (X), including conscious and unconscious parts, be implemented in physical brains of various kinds and possibly also computers and other future types of machines (Y).

The answer to question (a) is well understood where X is a software system, e.g. a word processor, the Prolog virtual machine, an operating system, a theorem prover, or even the internet, and Y is a computer or computer network. Regarding (b), the answer is not known: and there is strong disagreement as to whether X can be fully implemented in computers, or any other physical machines (e.g. brains).

Both are questions about levels of processes: at one level we have process Y which might involve only well understood physical mechanisms whereas the intrinsic nature of high level process X is very different. The ontology of X is not the ontology of physics, and the laws of behaviour of X are not physical laws.

Philosophers often refer to the relation between levels as "supervenience" (e.g. asking whether minds supervene on brains). Chalmers in [1] argues that consciousness cannot supervene entirely on physical processes. My claim is that such arguments depend on both (i) incorrect analyses of our pre-theoretic concepts of mind (including notions like "experience", "consciousness", and the more technical "qualia") and also (ii) a failure to understand how different ontological levels can exist and interact causally even if ultimately they are all implemented in physical mechanisms. For a more detailed defence of (i) see [8]. The remainder of this paper is mainly about point (ii).

Supervenience and implementation are closely related. Unfortunately many philosophers are not informed about how implementations work, and consequently make unjustified assumptions, which lead them to false conclusions (illustrated below). The cases understood by engineers, however, (e.g. compilers, word processors, robot control systems) are relatively simple, and their knowledge about implementation is mainly implicit practical know-how, whereas the cases studied by philosophers are far more complex, e.g. supervenience of minds on brains. By starting from simple, well-understood, kinds of implementations, and then (carefully) extrapolating from them, we can develop better ways of thinking about the latter, including different kinds of mental machines and different kinds of implementation machines, e.g. minds of different sorts of animals and humans in different cultures or different stages of development, or suffering from different types of brain injury or disease or degeneration. Any good theory should encompass all of this diversity.

2 Types of supervenience

Several types of supervenience can be distinguished. The one which is of importance here could be called *mechanism supervenience* since it involves a relationship between mechanisms. This is the kind of supervenience involved in the engineer's notion of *implementation*. Other kinds of supervenience involve relationships between patterns, or properties or mathematical structures, e.g. one mathematical structure modelled in another.

Implementation without an implementor

Although we normally talk about implementations as produced by a designer or implementor, we can extend the relationship to cases where there is no designer, such as the relationship between minds and brains, just as people often talk about designs without assuming there is a designer [2, 6] Anyone worried about using "implementation" to refer to cases

[1] School of Computer Science, The University of Birmingham, A.Sloman@cs.bham.ac.uk

[2] This is an extract from a much longer, evolving, paper, in part about the relation between mind and brain, and in part about the more general question of how high level abstract kinds of structures, processes and mechanisms can depend for their existence on lower level, more concrete kinds. For related work see [9, 8].

Submitted to ECAI 98
25 Jan 1998

where there is no implementor, can substitute "realisation". But I'll assume the implementation relation between X and Y does not presuppose any particular history.

Mechanism supervenience

"Mechanism supervenience" is the relation between X and Y if X and Y are mechanisms and X is implemented in Y. Being a mechanism involves having components and states and processes that interact *causally*. *Causal interaction* is an intuitive but indispensable concept that pervades all our thinking about processes, and I shall not attempt to analyse it here. It is not restricted to physical causation, as shown by:

- poverty causing crime,
- illiteracy causing poverty,
- insertion of a word in a wordprocessor causing a line, and possibly a page, to overflow,
- a syntactic error in a program causing compilation to abort,
- overloading an operating system causing it to thrash,
- evolutionary pressures causing a gene pool to change,
- remembering something causing a new desire to arise, which then causes an emotion (e.g. anxiety),
- an intention causing a physical action to occur,

and many more.

Causation is one of many subtle concepts required in everyday life linked to notions of "mechanism", "what would happen if", "control", "existence", etc., which philosophers have tried to analyse for centuries. As these notions are normally used their application is *not* restricted to purely physical phenomena, and any such restriction would cripple much of our everyday discourse and reasoning.

Implementation layers

Often there are several implementation levels between X and Y. This is commonplace in computing systems where coexisting virtual machines might include, for example, a theorem proving machine implemented in a Prolog virtual machine, implemented in a Sparc virtual machine, implemented in digital circuitry, implemented in physical mechanisms describable in the language of quantum mechanics. Social systems are partly implemented via the mental mechanisms of their inhabitants, which in turn are partly implemented in their physical brains. Besides such linear hierarchies there can be implementation *trees*, with X and X' implemented in Y, and Y and Y' implemented in Z, and so on.

It is even possible for implementation to be circular, e.g. where low level interrupt mechanisms are designed to invoke high level interrupt handlers to do their work, or where X includes mechanisms to reduce dependency on some details of Y e.g. using distortion reducing feedback and error-correcting algorithms. In such cases events in the low level system can cause events in the high level system, and vice versa: causation is "circular".

Is there a bottom level?

There is no clearly defined answer to the question whether there is a "bottom" implementation level. It might be thought that physical reality defines such a level. However, our idea of what physical reality is changes with the advance of science, and there's no telling what future physicists will be saying. Claiming that only the lowest level physical events can be causes or have causes suggests that the ancients knew of no causes, and perhaps we don't.

It is often assumed that the only *real* kind of causality is physical causality, but that seems plainly false. In fact it is not at all clear what sort of role causality plays in modern physics. Our ordinary notions of causation get a grip only at *intermediate* levels in the implementation layers of the universe. For instance there are economic causes, social causes, mental causes, as well as physical causes, and we make use of all of these types of causation in our everyday life. Whether all causation is *fully implemented* in physical causation is a separate question, not discussed here.

Full and partial implementation/supervenience

Supervenience, and implementation, may be full or partial depending on whether the implementation depends *only* on Y or whether something else is required. E.g. if some states of a virtual machine X can be defined only in terms of a relationship with the environment, then if Y does not include the environment, then X is not *fully* implemented in Y. Clearly being the richest person alive is a relational state, and so cannot be fully implemented in someone's physiology. Likewise, a clock's telling the local time correctly involves an external relationship and therefore cannot be fully implemented in the internal mechanisms. Since time-zones are relevant, even political facts can make a difference.

For a computing system X to contain and use information about towns in France, or parts of a chemical plant, involves a relationship between X and parts of the world outside X. No such relationship can be fully implemented in mechanisms which are entirely in the machine. This will be true of all virtual machines involving semantic content referring outside the machine. (Remote reference is analysed in [10].)

Consequently most human mental states cannot be fully implemented in, or fully supervenient on, their brain states, since they refer explicitly or implicitly to the environment. Likewise robots, office information systems, plant control systems. This leaves open whether it is possible to have an intelligent system whose mental states refer only to itself and to abstract entities like numbers, sets, algorithms, etc. which don't require an environment. I see no reason to rule out the possibility of a 'disconnected' pure mathematician (or philosopher), but will not argue that here. See [4, 5] for a discussion of reference within a computing system.

Limitations of current understanding

We understand many cases of X being implemented in Y, insofar as we know how to create those cases (e.g. we implement a software virtual machine in a computer), and we know how to use them, predict and explain their behaviour, debug them when they go wrong, and apply them in solving real practical problems. We also sometimes discover new ways of implementing X using another machine Y' which is cheaper, or faster, or more widely used.

But that is *practical* know-how, not *analytical* knowledge which answers philosophical questions about the general form of implementation or supervenience relations in natural or artificial systems, and we do not yet have a systematic overview of different types and their implications. There is not yet a clear theory of the conditions under which an implementation of X enables it to include semantic content (despite much discussion of Searle's 'Chinese room' argument [3]).

3 Other kinds of supervenience

Besides *mechanism* supervenience there are other kinds which do not include causal mechanisms, such as *property* supervenience, e.g. where a family having an average height H supervenes on the heights of individuals, and *pattern* supervenience which occurs when there are observable patterns in a physical system, e.g. patterns seen in stars on a clear night, or larger patterns found in configurations of simpler patterns.

Properties and patterns do sometimes have causal powers, when they occur in a system which can detect and respond to them. (See below.)

Some related concepts

"Implementation" should not be confused with "instantiation", a relationship between a program or algorithm and a sequence of states involved in executing the program. This is a mathematical relationship between two structures, the structure of the algorithm and the structure of a process which could be generated by the algorithm (a "trace"). Interesting programs with choice points have many possible instantiations, depending on their inputs, sometimes infinitely many.

Many of the theorems of computer science are about structural properties of instantiations of algorithms (e.g. theorems about complexity, solvability, correctness of algorithms, etc.) and have nothing to do with the key properties of mechanisms: time and causation.

Similarly, when two types of machine M1 and M2 are described formally and it is shown that machines of type M1 can be implemented or modelled in a machine of type M2, this is also a relation between two mathematical structures, which need not actually be instantiated. E.g. they could be infinite machines.

Physically unimplementable machines

Given a specification of a type of virtual machine, (e.g. a type of game playing machine, or problem solver) it is an empirical question whether a physical implementation is possible. That depends on whether there is a possible configuration of physical objects, events and processes which constitutes an implementation.

Not all virtual machines can be implemented physically, e.g. a Turing machine with an infinite tape, or a type of machine which manipulates arbitrary infinite vectors, or which can complete an infinite sequence of actions (e.g. finding all the prime numbers) and then start another sequence, e.g. finding all pairs of "twin" primes. Thus mathematical specifiability does not imply physical implementability.

4 Ontological levels

When a virtual machine X is physically implemented in Y, X may use an ontology which is not *definable* in terms of the ontology of physics; and the laws of behaviour of X will not be *derivable* from the laws of physics. (Similar comments apply to any other implementation layer.)

Definability

Even though a chess playing virtual machine can be implemented in a physical computer, the concepts of chess involve notions like "pawn", "rook", "capture", "threaten", "pin", "win", which cannot be defined in terms of concepts of physics, for there is no particular physical implementation required for a game of chess to occur: it could use wooden pieces on a board, or messages transmitted electronically, or, in the case of experts, their thoughts and verbal communications of their moves. Likewise most concepts used in everyday life to think about causally related topics, e.g. family and social relationships, our jobs, crime, poverty, war, etc. cannot be defined in terms of concepts of physics. (There is insufficient space to justify this here.)

Some virtual machine concepts are precisely definable in terms of the relationships which can hold between their instances including the laws of behaviour within that ontology. This is true of chess machines and numerical calculators. Others not so definable will be discussed below.

Derivability *vs* sufficiency

The laws of X need not be derivable from those of Y. For instance laws constraining movements of chess pieces in a chess machine are internationally agreed rules of the game, which can be changed without having to change the laws of physics. Similarly, the conditions for guardianship of a person, or ownership of a piece of ground, depend on social, legal, political facts, and these are not derivable from laws of physics because they can change even though the laws of physics remain constant.

If the concepts of X are not definable in terms Y's ontology, then the laws of X will not even be derivable from those of Y together with a full description of Y expressed using the ontology of Y. Such "definitional disconnection" underpins many examples of "emergence". No set of laws of physics together with the physical specification physical machine Y can entail a statement such as "the pawns can never move backwards". That would require *bridging* premisses linking the two ontologies. (In a computing context this could involve specification of a compiler or an interpreter, for instance.) These will not be general laws of nature, but statements about how X was implemented in Y in this case. In the case of a formally specifiable virtual machine like a chess machine, a numerical calculator, or a Prolog virtual machine, it is possible to confirm that the low-level implementation has the required properties. Other cases, discussed below, are not so simple.

5 Types of equivalence

Because bridging premisses are required, the relationship between X and Y when Y is an implementation of X is not purely mathematical or logical. In particular, not all mathematically equivalent implementations are causally equivalent. Mathematically there's no difference between

(a) N programs running (synchronised) in parallel on N computers, and

(b) a simulation of the N programs running in one program on one computer.

No mathematical or logical problem can be solved by the multi CPU system that cannot be solved by the single CPU system.

Yet, for engineering purposes, multiple CPUs provide greater *reliability*, and therefore the difference could be crucial in a safety critical system. A firm may pay a lot of money for the extra reliability, even if over its lifetime the multiple CPU system behaves exactly like a single CPU system in the small firm down the road. The payment is for "what would happen if..." even if it doesn't actually happen. I.e. the two machines have different *causal powers* even if their behaviour is actually the same over their lifetime.

Two implementations Y and Y' of virtual machine X may behave the same in all contexts which actually arise, yet be importantly different, e.g. because one is more reliable than the other. Some differences are describable at the level of X, e.g. which range of problems X *could* solve, even if those which *actually* occur are handled by both implementations. Other differences are describable only at the level of Y and Y', e.g. because one implementation can work in a wider range of temperatures, or tolerate more component failures.

Specification-driven implementations

Some implementations make use of an explicit specification of X. Subtle and complex legal, social and psychological mechanisms allow an explicit specification of legal rules to produce (more or less rigid) conformity to the rules. By contrast, a word processor or chess machine may be created by a compiler, from an explicit specification, which thereafter has no influence. Where an interpreter is used, the specification controls behaviour at run-time, providing different causal potential, for instance ease of run-time modifiability or debugging, and potential for high level code changes to change behaviour. (Difference between interpreted and compiled implementations sometimes matter more to developers than "end users".)

6 Implementing configurations

Whether a type of machine is implementable or not depends on whether a configuration of physical components exists which has appropriate properties to constitute an implementation of the machine. What that means is not easy to specify. In particular, some of the requirements for supervenience often assumed by philosophers are definitely not requirements for implementation of software virtual machines.

Mistaken assumptions

In particular suppose X is implemented in Y.

1. X need not be isomorphic to Y or part of Y. Lazy evaluation (which allows components to be created 'invisibly' only when needed) and sparse arrays are counter examples. A lazy list or sparse array could have far more components than its physical implementation.

2. There need not be correlations between components of X and components of Y. Virtual memory systems and compacting garbage collectors, constantly change the mapping between virtual machine components and physical components. Such re-mapping in brains may be needed in 'working' memory. Correlation is neither necessary nor sufficient for implementation. A deeper causal connection is needed.

3. Part-whole relationships within X need not map into those of Y. E.g. two Lisp lists can each be elements of each other, whereas it is not possible for two physical components each to be a part of the other.

4. Implemented systems need not inherit features and restrictions of their implementation. A machine Y based on binary switches is often used to implement a virtual machine X with all sorts of non-binary data-types, e.g. integers, indefinite precision ratios, floating point numbers, functions, etc. A continuous virtual machine can be implemented in a digital machine (to any required degree of approximation, if it is not chaotic) by sampling its states at regular intervals. A discrete array can be interpreted as sampling a continuous image, with sub-pixel properties inferred when needed.

5. X can have causal powers, including the power to influence Y, even if the implementation level Y is *causally complete*. Events in a word processor involving a page overflowing can cause changes in the physical memory and on the screen, even though there are no causal gaps in the underlying physics. Similarly, events in many naturally occurring virtual machines can produce physical effects (e.g. political processes causing bombs to explode).

Consequently a mind could have physical effects without requiring quantum indeterminacy in brains.

The idea that if physics is causally complete then all non-physical levels of causation are causally redundant is based on an incorrect analysis of our ordinary notion of causation, using a model involving something like mechanical linkages; whereas our normal conception of causation is very abstract, and linked to the notion of "what would happen if", i.e. the truth of conditional statements. The existence of true conditional statements concerning the virtual machine X (including counterfactual conditionals) is perfectly consistent with Y being causally complete. This is obviously the case in familiar computational virtual machines.

6. It is a mistake to link criteria for identity of X to identity of components of Y. Even the ancients noticed that you can step into the same river twice even though the water has changed. Animals and plants constantly replace some of their component atoms. Ocean waves move horizontally whereas the water molecules in them oscillate vertically. Nations and species survive the individuals that compose them. Identity criteria for "high level" objects are often indeterminate: there's no clear number of components that have to be replaced before you've replaced your computer.

7 Describing configurations

In some more general sense than logical derivability, the existence of a configuration of type Y may be *sufficient* for the existence of a virtual machine of type X.

If X is implemented in a physical machine Y, then there must be something about that particular configuration of elements of Y, together with the laws of physics, which makes Y an implementation of X. However, exactly what it is that makes Y an implementation of X may not be describable at the level of Y i.e. in physical terms. There are several reasons.

Infinite disjunctions

One possible reason may be that there are indefinitely many physical configurations which will suffice, like all the ways of implementing a chess machine, and no way of saying what they have in common except by describing the machine X which they all implement, which as explained above, involves going beyond the concepts of physics.

Because the rules of chess and the functioning of a computer are so precisely specified, the question whether a particular computer implements a chess machine is objectively answerable in principle, even though it may be very hard in practice, when it involves decompiling machine code without even knowing what sort of high level language was used. However, someone who has worked out what the language was, how it was compiled or how it is interpreted, how the machine's digital circuitry works, may be able to demonstrate, assuming the laws of physics, that the machine will always play chess correctly.

Thus certain physical systems are in principle demonstrably implementations of a chess virtual machine. When X is part of an autonomous agent, checking its presence may be more difficult. One reason is that a person who knows how to play chess, may have decided never to play again (e.g. for religious reasons). Then it may be impossible to demonstrate that he can, or to find which portions of the brain implement the ability. But we'll see that there are deeper obstacles, because adaptive brains create their own categories.

Culturally defined categories

Consider how a *culture* implicitly defines a concept. What makes something a letter of our alphabet depends in part on what we regard as a font, and that can change over time without the laws of physics changing. For instance, there is nothing *physical* that is common between all the 2-D patterns that we call instances of the letter "a", including upper and lower case, printed and handwritten text. There doesn't have to be any *logic* in the way we categorise shapes. It may just have turned out over the centuries that in our culture we divided the possible configurations of 2-D patterns in a certain way to form an alphabet, and in another part of the world they may have done it differently, so that if we go there we may at first find it hard to read their texts. (Compare the problems with handwritten "1" and "7" in different countries.) Moreover, the patterns accepted as particular letters need not be fixed: they could be constantly, though slowly, changing even within a culture. This may happen because the boundaries are slightly fuzzy, with a probability distribution which changes gradually.

Consider an adaptive machine which "tracks" the scripts used in a particular culture, so that it can read the characters humans read in that culture, and can follow changes in acceptable font styles, and which finds the same examples hard to read as people do. The machine could use neural nets or weighted rules modified by reinforcement learning.

That machine implements a character recogniser for written texts in that culture though there is no physical definition of what makes it such a recogniser. That is partly because the precise physical requirements are not fixed: they change as the culture changes. That in turn may depend on a particular set of happenings such as the particular physical characteristics of people who happen to be born in the culture, or the adoption of a particular type of flourish which happened to be used accidentally by someone and then was imitated by others.

An "autonomous" adaptive categoriser

Now consider a machine which has somehow developed categories of its own, for its own reasons. Pressure to categorise, albeit in fuzzy ways, could arise out of learning and adaptive mechanisms in an architecture which was not purely reactive but included a deliberative layer (as described in [7]) that can construct and evaluate plans for future action. Deliberation requires a categorisation of reality into chunks that can enter into learnt associations usable in planning (if I do A then B will follow whereas if I do A' them B' will follow, etc.). Such chunking could also be used for passive prediction: if event A is observed in situation S then S' will follow.

Which categories the machine develops and finds useful could depend on the particular sequence of contexts and problems it encounters. In effect, it develops its own "culture" containing a single individual. (Compare how human face recognition is tuned to the individual's environment. Getting to know a pair of almost identical twins may force a new chunking to be developed.)

The effect is implementation of a new virtual machine whose behaviour is accurately describable only by using the concepts that machine has developed for itself. These concepts need not be definable in terms of the concepts that were adequate when the machine was first built.

The high level cognitive processes in such a machine would be inherently incomprehensible to agents with different ontologies. (Consequently, social machines need cultural mechanisms to limit the amount of idiosyncratic adaptation.)

External observers may hypothesise that the machine is making use of perceptual categories for planning and prediction, but not know what those categories are. Checking precisely what is going on in the brains of such machines could be very difficult, especially if finding out what categories they use involves adopting a new set of concepts that is incompatible with one's existing set. This can be a serious problem for anthropologists and psychologists studying children and other animals.

For all these reasons it is to be expected that if intelligent agents do include adaptive virtual machines involved in perception, planning, goal formation, preferring, deciding, etc. they may differ in subtle ways which makes it impossibly difficult for any of them to be fully understood "from outside", even by the original designer, since the designer may have no conclusive way of telling precisely which ontology they have developed. This does not mean that it is totally futile to try to find out. It may be easier to check out general features of the architecture than precise details.

8 Is being an implementation of X an objective property?

Some people argue that attribution of virtual machine X to physical machine Y is entirely in the eye of the beholder: it is just a matter of subjective interpretation not a matter of fact that machine X exists. I have tried to show that in some cases (e.g. a calculator or chess machine in a computer) it is an objective questions that may be settled by detailed analysis. In other cases it may be very difficult, e.g. where the machine is an autonomous agent which has decided never to perform certain tasks, or where its perception and deliberation use an ontology very different from that of the investigator.

The situation is made worse by the fact that the agent being studied may continue changing, and the very process of taking part in experiments to check out how it works could also change it.

Nevertheless, there are real questions even if answers are hard to check: which categories does the virtual machine use, what is the information processing architecture, and how does the system change itself? The mere fact that finding answers is very difficult does not mean that it is all a matter of what is in the eye of the beholder, like the preference for one style of painting over another, or the fact that a particular scene reminds one of a happy childhood event.

Moreover, even if we don't know precisely what sort of virtual machine ontology is in a particular agent we can still accept that events within that virtual machine have causal powers, especially planning and deciding events which can lead to actions being performed, and perceptual events which help to control the fine detail of the actions, or even lead to new desires which cause the original goal and its plan to be abandoned.

9 Conclusion

I have tried to show that mental processes may involve causal interactions with a complex self-modifying virtual machine architecture, that such a virtual machine can be implemented in a physical machine (possibly *via* several intermediate virtual machines), that the virtual machine can have causal powers even if the physical machine is causally complete, that the properties of the virtual machine need not be definable in the language of physics nor derivable from laws of physics or laws together with physical descriptions of the implementation.

I've also argued that whether a virtual machine of a particular sort is implemented in a particular physical system is an objective question, though in the case of an adaptive machine the question may be particularly difficult to answer, for several different reasons, and moreover the answer may be changing, and the change may be accelerated by attempting to find the answer.

In previous papers I have tried to argue that within certain types of information processing architecture, semantic content will be associated with some of the structures by the agent, not just observers, i.e. the agent can have what Haugeland called "original (non-derivative) intentionality", making the symbol-grounding problem a non-problem.

I have also argued elsewhere, that if a robot has the right sort of virtual machine, with appropriate perceptual, reactive, deliberative, and self-monitoring, capabilities then nothing more is needed for it to be conscious and have "qualia" (see [8]). Some such machines may even discover this fact about themselves and become embroiled in philosophical debate about it. This supports one version of the Strong AI thesis.

However, the discussion leaves open precisely what sorts of physical machines are capable of implementing all that functionality. Moreover it gives us reason to think that when we have built truly intelligent machines we shall not be able to understand them fully, any more than we understand each other fully, even if we can observe and measure every smallest detail of their physical functioning.

REFERENCES

[1] David J Chalmers, *The Conscious Mind: In Search of a Fundamental Theory*, Oxford University Press, New York, Oxford, 1996.

[2] Randall Davis, 'What are intelligence? and why?', *AI Magazine*, (1998). (Presidential Address to AAAI98, to appear).

[3] John R Searle, 'Minds brains and programs', *The Behavioral and Brain Sciences,*, **3**(3), (1980). (With commentaries and reply by Searle).

[4] A. Sloman, 'What enables a machine to understand?', in *Proc 9th IJAI*, pp. 995–1001, Los Angeles, (1985).

[5] A. Sloman, 'Reference without causal links', in *Advances in Artificial Intelligence - II*, eds., J.B.H. du Boulay, D.Hogg, and L.Steels, 369–381, North Holland, Dordrecht, (1987). Originally in Proceedings 7th European Conference on Artificial Intelligence, July 1986.

[6] A. Sloman, 'Explorations in design space', in *Proceedings 11th European Conference on AI*, Amsterdam, (1994).

[7] A. Sloman, 'What sort of architecture is required for a human-like agent?', in *Foundations of Rational Agency*, eds., Michael Wooldridge and Anand Rao, Kluwer Academic, (1998(forthcoming)). (Expanded version of invited talk at Cognitive Modeling Workshop, AAAI96 Portland, Oregon, August 1996).

[8] Aaron Sloman, 'The evolution of what?'. Draft version available online at ftp://ftp.cs.bham.ac.uk/pub/groups/cog_affect/ Sloman.consciousness.evolution.ps, 1998(in preparation).

[9] Brian C Smith, *On the Origin of Objects*, MIT Press, 1996.

[10] P. F. Strawson, *Individuals: An essay in descriptive metaphysics*, Methuen, London, 1959.