Avant. The Journal of the Philosophical-Interdisciplinary Vanguard
Volume II, Number 2/2011 www.avant.edu.pl
ISSN: 2082-6710

Evolution: The Computer Systems Engineer Designing Minds[8]

Aaron Sloman

Abstract:

What we have learnt in the last six or seven decades about virtual machinery, as a result of a great deal of science and technology, enables us to offer Darwin a new defence against critics who argued that only physical form, not mental capabilities and consciousness could be products of evolution by natural selection. The defence compares the mental phenomena mentioned by Darwin's opponents with contents of virtual machinery in computing systems. Objects, states, events, and processes in virtual machinery which we have only recently learnt how to design and build, and could not even have been thought about in Darwin's time, can interact with the physical machinery in which they are implemented, without being identical with their physical implementation, nor mere aggregates of physical structures and processes. The existence of various kinds of virtual machinery (including both "platform" virtual machines that can host other virtual machines, e.g. operating systems, and "application" virtual machines, e.g. spelling checkers, and computer games) depends on complex webs of causal connections involving hardware and software structures, events and processes, where the specification of such causal webs requires concepts that cannot be defined in terms of concepts of the physical sciences. That indefinability, plus the possibility of various kinds of self-monitoring within virtual machinery, seems to explain some of the allegedly mysterious and irreducible features of consciousness that motivated Darwin's critics and also more recent philosophers criticising AI. There are consequences for philosophy, psychology, neuroscience and robotics.

Keywords: Architecture, Causation, Cognition, Consciousness, Control, Darwin, Designer Stance, Evolution, Explanatory Gap, Mind, Self-monitoring, Universal Turing machine, Virtual Machinery.

[8] This is a previous version of an invited paper "Evolution of mind as a feat of computer systems engineering: Lessons from decades of development of self-monitoring virtual machinery" presented at: Pierre Duhem Conference (Society for Philosophy of Science),Tuesday 19th July 2011, Nancy, France.

Contents:

1. Virtual machines and causation

Have you ever wondered how a word-processor makes adjustments when you insert a new character in an already full line? If the extra character makes the length of the line exceed the specified text width the line is broken and some of the characters at the end are inserted at the beginning of the next line. This can cause the same thing to happen repeatedly for many more lines. The portion removed from the end of one or more of the affected lines may have to be moved onto the next page, which may cause that in turn to overflow. In some cases every subsequent page of the document, is altered, even if they only acquire new page numbers. Similar things can happen when a spelling checker runs, discovers an error, and corrects it by inserting one or more characters. If the corrected word is shorter, that can cause one or more lines, and possibly pages to shrink, in some cases also absorbing text from the next line or page.

The mechanisms producing such changes are generally very effective and reliable (most of the time) insofar as changes that occur in the document lead to required effects in other parts of the document, and also in various parts of the physical machinery, including active memory, hard drives, visual displays, and possibly also printed pages.

This all depends on interactions among very different technologies developed since the middle of the last century, some of which are constantly changing (e.g. the materials used and design and construction processes used to make computer processors, memories, interfaces, networks, displays, and other physical components). Some components can change while others remain unaltered – as a result of ingenious use of interfacing specifications that allow developers on one side of an interface to ignore what does or does not change on the other side. In particular, while some things change rapidly others may remain unchanged and still work, for instance the design and program specification of an old text editor, like the one I use, whose core code has been unchanged for nearly a quarter century.

The development of “cloud computing” (really a re-discovery of the usefulness of mechanisms some of us have been using for several decades) can lead to wide geographical dispersal of processes that previously all occurred in one box. As I type this text, sitting at a PC in my study at home I am using a text editor running on another computer some distance away, in my department. The editor was developed in the early 1980s, though much newer technologies (including networking technologies) allow each key I press to cause, almost instantaneously, effects in the remote machine, followed by changes to my screen display. If I switch to working on my own computer, the effects are indistinguishable to me, most of the time, though I then have to arrange my own backup processes, and later transfer the files produced to the university machines. Either way, there is no change in the correct description of what happens to *the document* when I insert a character or a spelling corrector changes the text.

That is because the document exists in a running virtual machine, or more specifically in a specific instance of a type of virtual machine, whose instances may have very different physical implementations. Opening up the computers involved and peering into them with the most sophisticated available physical sensing and measuring devices will not reveal any characters, words, pages, or spelling errors, let alone story plots, heroes, theories, debates, exhortations, arguments, etc. Data-mining processes that detect such things do not measure physical properties of the machines used.

Moreover, even the people who design the software will not be able to tell that it is running on these machines by opening them up and looking at patterns of physical activity. Often the core software was developed decades before the physical technology that now supports it, thanks to cooperation between designers of programming languages, compilers, interpreters, operating systems, a host of new physical devices, device drivers, networking protocols, and advances in materials science and electronic technology.

Most of the researchers and engineers involved in all that science and technology perceive only a tiny subset of the intricate web of mechanisms they contribute to. Probably nobody on this planet now understands the total systems we use. Things work because many different sorts of machinery have been designed to work together, some concerned with transmission of power, some with power-consuming physical changes in sub-microscopic components, some with physical links between subsystems, some with storage and transmission of information expressed in bit patterns (not numbers, as commonly supposed, since all the numerical computations are implemented in bit patterns, as are non-numerical processes like text manipulation).

Many portions of the technology are designed to preserve relationships while things change, or to propagate changes in information structures in very specific ways. Propagation of information structures, e.g. bit patterns, or more complex structures implemented in bit patterns, such as sentences or images, is different from propagating

energy, as happens in power transmission lines, or transporting matter as happens in water pipes[9].

Causation among patterns of change in information structures depends on, but is different from, causation involving changes in energy and matter. (For a more detailed discussion of the nature of information see (Sloman 2011).)

2. Layers of virtual machinery

When text-manipulation events occur in the document you are composing, there are also far more events that occur in the digital circuitry making up the computer you are using. Millions of transistors may change state causing bit patterns to be moved around in a linear array of bit patterns, triggering other collections of events that modify magnetic patterns on a hard drive and signals sent to the screen showing you what's going on, or in some cases driving a speech to text system, usually motivating you to make further changes using your mouse and keyboard.

But there is no *physical* linear array of bit patterns. That array, like the document and the documentation manipulation mechanisms, is a non-physical structure that is assumed and manipulated by mechanisms created by designers of the system. The structure can be treated as linear by mechanisms that impose an order on its parts. All those changes and movements of bit patterns, depend on and are implemented in even "lower level" physical structures and processes in which transistors change their state, electrical signals are transmitted, and power is consumed and dissipated. A quantum physicist might tell yet another, even more complex story about what is going on. What the physicists of future centuries will say remains unknown.

At a higher level of abstraction a linear array of text items may be imposed on the array of bit patterns, without directly mapping into an ordered subset of the patterns. For instance the implementation may have chunks of text scattered around the "memory" where the structure storing each chunk includes information about where the next chunk is (using memory pointers, or addresses). If the text processing system is able to use those cross links, it can treat all the text as linearly ordered. In that case there is virtual machine concerned with text processing implemented in an abstract virtual machine concerned with manipulation of bit patterns, which is implemented in digital circuitry, which is implemented in atoms molecules and other physical components. For an excellent account of some of the history of the technology of information, including use of signals based on fires many hundreds of years ago, see (Dyson 1997).

Does all this imply that someone developing a spelling checker has to know all about the movements and state changes of billions of sub-atomic particles, or of millions of bit

[9] For an excellent account of some of the history of the technology of information, including use of signals based on fires many hundreds of years ago, see (Dyson 1997).

patterns in digital systems implemented in physical machinery? No! If specification at that level of detail were required, the task would be impossible for a human mind, not least because the very same change of text in the very same place in the original file with the very same re-arrangement ramifications, and very similar alterations to what appears on the screen or on paper could occur while making use of very different transformations of bit patterns and physical machinery, even if the same editing process is repeated on the same computer later on, perhaps because the previously edited file was accidentally deleted.

So the designer of the spelling corrector does not need to be able to think about all the many possible changes in electronic, or worse, sub-atomic, structures and processes that can occur when a spelling mistake is detected so as to ensure that the right pattern of changes to remedy the fault occurs each time. The designer of the spelling checker need not know any physics at all.

A text editor with spelling checker is just one among very many different types of information processing machine that can be implemented as a virtual machine running on and alongside other machines, including possibly several layers of lower level machinery. One of the important facts about many such virtual machines is that they don't merely compute the answer to some logical or mathematical question, a task that can be performed without any interaction with the environment. Rather the virtual machines I have been talking about can interact with things in the physical environment – like a spelling checker that sometimes asks the user to choose between alternative corrections.

This ability to receive information from and to act on a physical environment is a feature of many kinds of information-based control system, including, for example flight control systems, and the information processing systems of robots. Later I'll present a claim that all living organisms make use of information in their control functions, and the more sophisticated ones need to use virtual machinery implemented in physical machines for that purpose, rather than merely using physical machines.

One of Turing's achievements was the specification of a *Universal Turing Machine* (UTM) within which any other Turing machine could be emulated by specifying its properties on the tape of a UTM (Turing, 1936). This led to proofs of important theorems, e.g. about equivalence, decidability and complexity. It can also be seen as a precursor of what we now call virtual machinery (not to be confused with virtual reality). I shall try to show how the combination of virtuality, causal interaction and (relative) indefinability can produce something new to science. The next few sections explain in more general terms what virtual machines (especially non-physically describable machines – NPDMs) are and why they are of much greater philosophical importance than is normally understood, and why some of them are not equivalent to any Turing machine. After that I'll present implications regarding evolution of mind and consciousness.

3. Virtuality

The UTM idea established that a computing machine can run by being implemented as a virtual machine in another machine. (I think the gist of this idea was understood by Ada Lovelace a century earlier.) The mathematical properties of a machine's trajectory through its state space will not depend on whether it is run directly in physical machinery or as a virtual machine implemented in another computation. This has proved immensely important for theorems of meta-mathematics and computer science and for some of the practicalities of using one computer for multiple purposes, including time-sharing. One of the consequences is that a Turing machine implementing another Turing machine can also be a virtual machine implemented in a UTM: so that layered implementations are possible.

In the following decades, engineering developments emerged in parallel with mathematical developments, with some consequences that have not received much attention, but are of great philosophical interest and potentially also biological import. I'll suggest later that biological evolution “discovered" many of the uses of virtual machinery long before we did. Unfortunately, the word “virtual" suggests something “unreal" or ”non-existent", whereas virtual machines can make things happen: they can be causes, with many effects, including physical effects. To that extent they, and the objects and processes that occur in them, are *real* not *virtual*!

4. Causation and computation

Causation is a crucial aspect of the engineering developments. For example, it is possible to take any finite collection of Turing machines and emulate them running in parallel, in synchrony, on a UTM. This demonstrates that *synchronised* parallelism does not produce any qualitatively new form of computation. The proofs are theorems about relationships between abstract mathematical structures including sequences of states of Turing machines- and do not mention physical causation. A running physical machine can be an instance of such an abstract mathematical structure. However, being physical it can be acted on by physical causes, e.g. causes that alter its speed. Sloman (1996) pointed out that theorems can break down for physical Turing machines that are not synchronised. For example, if TM T1 repeatedly outputs “0", and T2 repeatedly outputs “1", and the outputs are merged to form a binary sequence, then if something causes the speeds of T1 and T2 to vary randomly and they run forever, the result could (and most probably would) be a non computable infinite binary sequence, even though each of T1 and T2 conforms to theorems about Turing machines.

Likewise, if a machine has physical sensors and some of its operations depend on the sensor readings, then the sequence of states generated may not be specifiable by any TM, if the environment is not equivalent to a TM. So the mathematical “limit" theorems do not apply to all physically implemented information-processing systems.

Mathematical entities, such as numbers, functions, proofs, and abstract models of computation, do not have spatio-temporal locations, whereas running instances of computations do, some of them distributed across networks. Likewise, there are no causal connections, only logical/mathematical relationships, between the TM states that form the subject matter of the mathematical theory, whereas there are causal connections in the running instances, depending on the physical machinery used and the physical environment.

So, notions like “reliability" are relevant to the physical instances, but not the mathematical abstractions. From a mathematical point of view there is no difference between three separate computers running the same program, and a single computer simulating the three computers running the program. However an engineer aiming for reliability would choose three physically separate computers with a voting mechanism as part of a flight control system, rather than a mathematically equivalent, equally fast, implementation in a single computer (Sloman 1996).

Physical details of time-sharing of the three machines have causal consequences. When the three separate machines running in synchrony switch states in unison, nothing happens between the states, whereas in the time-shared implementation on one computer, the underlying machine has to go through operations to switch from one virtual machine to another. Such “context switching" processes have intermediate substates that do not occur in the parallel implementation. A malicious intruder, or a non-malicious operating system, will have opportunities to interfere with the time-shared systems during a contexts witching process, e.g. modifying the emulated processes, interrupting them, or copying or modifying their internal data.

Such opportunities for intervention (e.g. checking that a sub-process does not violate access restrictions, or transferring information between devices) are often used both within individual computers and in networked computers causally linked to external environments, e.g. sensing or controlling physical devices, chemical plants, air-liners, commercial customers, social or economic systems, and many more. In some cases, analog-to-digital and digital-to-analog devices, and direct memory access mechanisms now allow constant interaction between processes. See also (Dyson 1997).

The technology supporting the causal interactions includes (in no significant order): *memory management, paging, cacheing, interfaces of many kinds, interfacing protocols, protocol converters, device drivers, interrupt handlers, schedulers, privilege mechanisms, resource control mechanisms, file-management systems, interpreters, compilers, “run-time systems" for various languages, garbage collectors, mechanisms supporting abstract data types, inheritance mechanisms, debugging tools, communication channels within and between machines (“pipes" and “sockets"), shared memory systems, firewalls, virus checkers, software viruses, security systems, operating systems, application development systems, name-servers, password checkers*, and more. All of these can be seen as contributing to intricate webs of causal connections in running systems, including *preventing* things from happening, *enabling* certain things to happen in certain

conditions, *ensuring* that if certain things happen then other things happen, and in some cases *maintaining mappings* between physical and virtual processes. The word-processing example introduced above illustrates one of the simpler causal webs to be found in computing systems.

5. Varieties of virtual machinery

A running virtual machine can have many effects, including causing itself to change. Understanding how virtual machines can cause anything to happen requires a three-way distinction, between: (a) *Mathematical Models* (MMs), e.g. numbers, sets, grammars, proofs, etc., (b) *Physical Machines* (PMs), including atoms, voltages, chemical processes, electronic switches, etc., and (c) *Running Virtual Machines* (RVMs), e.g. calculators, games, formatters, provers, checking spellers, email handlers, operating systems, etc. MMs are static abstract structures, like proofs and axiom systems. Like numbers, they cannot do anything. They include Turing machine executions whose properties are the subject of a mathematical proofs. Unfortunately some uses of “virtual machine" refer to MMs, e.g. “the Java virtual machine". These are abstract, inactive, mathematical entities, not RVMs, whereas PMs and RVMs are active and cause things to happen.

Physical machines on our desks can now support varying collections of virtual machinery with various kinds of concurrently interacting components whose causal powers operate in parallel with the causal powers of underlying virtual or physical machines, and help to control those physical machines. Some of them are *application* RVMs that perform specific functions, e.g. playing chess, correcting spelling, handling email. Others are *platform* RVMs, like operating systems, or run-time systems of programming languages, which are capable of supporting many different higher level RVMs. Different RVMs have different *levels of granularity* and *different kinds of functionality*. They all differ from the granularity and functionality of the physical machinery.

Relatively simple transitions in a RVM can use a very much larger collection of changes at the machine code level and an even larger collection of physical changes in the underlying PM – far more than any human can think about. (That was not true of the earliest virtual machines running on single-process computers with at most hundreds or thousands of memory locations and no external interactions.) Apart from the simplest programs even machine-code specifications are unmanageable by human programmers. Automatic mechanisms (including compilers and interpreters) are used to ensure that machine-level processes support the intended RVMs.

6. Self-monitoring, self-control, and self-modification

Interpreted and compiled programming languages have important differences in this context. An interpreter ensures *dynamically* that the causal connections specified in the program are maintained. If the program is changed while running, the interpreter's behaviour (or potential behaviour under some conditions) will change. In contrast, a compiler *statically* creates machine code instructions to ensure that the specifications in the program are subsequently adhered to, and the original program plays no role thereafter. Changing it has no effect, unless it is recompiled (e.g. if an *incremental* compiler is used). In principle the machine code instructions can be altered directly by a running program(e.g. using the “poke" command in Basic) but this is usually feasible only for relatively simple changes and would probably not be suitable for altering a complex plan after new obstacles are detected, and modifying the physical wiring would be out of the question. So self-monitoring and self-modification are simplest if done using process descriptions corresponding to a high level virtual machine specified in an interpreted formalism and least feasible if done at the level of physical, structure. Processes monitoring and modifying compiled machine code instructions are an intermediate case.

There are two different benefits of using a suitable RVM: namely the already mentioned coarser granularity of events and states compared with a PM or low level RVM, and the use of an ontology related to the application domain (e.g. playing chess, making airline reservations). Both of these are indispensable for processes of design, testing, debugging, extending, and also for run-time self-monitoring and control, which would be impossible to specify at the level of physical atoms, molecules or even transistors (because of explosive combinatorics, especially on time-sharing, multi-processing systems where the mappings between virtual and physical machinery keep changing). The coarser grain, and application-centred ontology makes self-monitoring more practical when high level interpreted programs are run than when machine code compiled programs are run. This relates to the third aspect of some virtual machinery: ontological irreducibility.

7. Implementable but irreducable

The two main ideas presented so far are fairly familiar, namely (a) a VM can run on another (physical or virtual) machine, and (b) RVMs (and their components) running in parallel can interact causally with one another and with things in the environment. A third consequence of 20th Century technology is not so obvious, namely: *some VMs include states, processes and causal interactions whose descriptions require concepts that cannot be defined in terms of the language of the physical sciences: they are non-physically describable machines (NPDMs)*. Virtual machinery can extend our ontology of types of causal interaction beyond physical interactions.

This is not a form of mysticism. It is related to the fact that a scientific theory can use concepts (e.g. “gene", “valence") that are not *definable* in terms of the actions and observations that scientists can perform. This contradicts both the “concept empiricism" of philosophers like Berkeley and Hume, originally demolished in (Kant 1781), and also its modern reincarnation, the “symbol grounding" thesis popularised by (Harnad 1990), which also claims that all concepts have to be derived from experience of instances.

The alternative “theory tethering" thesis, explained in (Sloman 2007), is based on the conclusion in 20th Century philosophy of science that undefined symbols used in deep scientific theories get their meanings primarily from the structure of the theory, though a formalisation of such a theory need not fully determine what exactly it applies to in the world, since any formal system can have many different (Tarskian) models[10].

The remaining indeterminacy of meaning of a formally specified theory is partly reduced by specifying forms of observation and experiment (sometimes called “bridging rules" or “meaning postulates" (Carnap 1947)) that are used in testing and applying the theory, “tethering" the semantics of the theory to specific portions or aspects of the world. The meanings are never uniquely determined, since it is always possible for new observations and measurements (e.g. of charge on an electron) to be adopted as our knowledge and technology advance.

Ontologies used in specifying VMs, e.g. concepts like “pawn", “threat", “capture", etc. used in specifying a chess VM, are also mainly defined by their role in the VM, whose specification expresses an explanatory theory about chess. Without making use of such concepts, which are not part of the ontology of physics, designers cannot develop all the implementations that are currently found useful or entertaining, and users cannot understand what the program is for, or make use of it.

So, when the VM runs, there is a physical implementation that is also running (with changes of physical state, movements of physical matter, and transition and dissipation of energy), but the two machines are not identical: there is an asymmetric relation between them. The PM is an *implementation* of the VM, but the VM is not an implementation of the VM, and there are many other statements that are true of one and false of the other.

A running chess VM, but not the underlying PM, may include threats, and defensive moves. And neither “threat" nor “defence" can be defined in the language of physics[11]. So, not all the concepts used to describe objects, events and processes in a RVM are *definable* in terms of concepts of physics even though the RVM is *implemented* in a physical machine. The detailed description of the PM is not a specification of the VM, since the VM could be the same even if it were implemented on a very different physical

[10] For instance it is well known that any model of the axioms of projective geometry remains a model if lines and points are swapped, and the predicate and relation symbols are reinterpreted accordingly.
[11] This needs more discussion than I have space for.

machine with different physical processes occurring during the execution even of a particular sequence of chess moves.

The VM description is also not equivalent to any fixed *disjunction* of *descriptions* since the VM specification determines which PMs are adequate implementations. Programmers can make mistakes, and bugs in the virtual machinery are detected and removed, usually by altering a textual specification of the abstract virtual machinery not the physical machinery. When a bug in the program is fixed it does not have to be fixed differently for each physical implementation - different compilers or interpreters for the language can handle the mappings between virtual machine and physical processes in different physical machines, and those details are not part of the specification of the common virtual machine.

Neither can the VM machine states and processes be defined in terms of physical input-output specifications (as assumed in some varieties of functionalism), since very different technologies can be used to implement interfaces for the same virtual machine, e.g. using mouse, keyboard, microphone or remote email for input. Moreover, some VMs perform much richer tasks than can be fully expressed in input-output relations, e.g. the visual system of a human (or future robot!) watching turbulent rapids in a river. (Compare the critique of Skinner in (Chomksy 1959)).

The indefinability of VM ontologies in terms of PM ontologies does not imply that RVMs include some kind of “spiritual stuff" that can exist independently of the physical implementation machinery, as assumed by those who believe in immortal minds, or souls. Despite the indefinability there are close causal connections between VM and PM states, but that includes things like detection of a threat causing a choice of defensive move, which is a VM process that can cause changes in the physical display and the physical memory contents. We thus have what is sometimes referred to as “downwards causation", in addition to “upwards causation" and “sideways causation", within the RVM, or between RVMs running concurrently.

The complex collection of hardware, firmware, and software technologies, developed since Turing's time has enabled us to build information-processing systems of enormous complexity and sophistication performing many tasks that were previously performed only by humans and some that not even humans can perform. But perhaps more important in the long term is the new way of thinking about non-physically describable virtual machinery with causal powers that we have begun to develop. The new conceptual tools are relevant not only to what human designers can do but also to what self monitoring, self-controlling systems may be able to do. This has deep significance for our understanding of evolution.

8. Darwin's critics

Darwin's critics, some of whom are quoted in (Sloman, 2010a), argued that his evidence supported only the hypothesis that natural selection produces *physical* forms and behaviours. Nobody could understand how physical mechanisms can produce mysterious and externally unobservable mental states and processes: “The explanatory gap". Since Darwin's time the problem has been re-invented and re-labelled several times, e.g. as the problem of “Phenomenal Consciousness" (Block 1995) or the “Hard Problem" of consciousness (Chalmers” 1996). The topic was touched on and side-stepped by Turing in (1950). It remains unclear how a genome can, as a result of physical and chemical processes, produce the problematic, apparently non-physical, externally unobservable, personal experiences (qualia) and processes of thinking, feeling and wanting.

Earlier I presented Universal Turing Machines as theoretical precursors of technology supporting networks of interacting running virtual machines (RVMs) sensing and controlling things in their environment. Such RVMs are *fully implemented* in underlying physical machines (PMs) but the concepts used to describe the states and processes in some RVMs (e.g. “pawn" and “threat" in chess VMs) are not *definable* in the language of the physical sciences. We now develop the biological application of these ideas, explaining how self-monitoring, self-modifying RVMs can include some of the features of consciousness, such as qualia, previously thought to be mysterious, paving the way for a theory of how mind and consciousness might have evolved.

9. Epigenesis: bodies, behaviours, and minds

Turing was interested in both evolution and epigenesis and made some pioneering suggestions regarding the processes of morphogenesis – differentiation of cells to form diverse body parts during development. As far as I know, he did not do any work on how a genome can produce *behavioural competences* of the complete organism, including behaviours with complex conditional structures so that what is done depends on internal and external sensory information, though he briefly considered learning, in (Turing 1950)[12].

It is understandable that physical behaviours, such as hunting, eating, escaping predators, and mating, should influence biological fitness and that evolution should select brain and other modifications that produce advantageous behaviours. But there are internal non-behavioural competences whose biological uses are not so obvious: thinking, reminiscing, perceiving with enjoyment, finding something puzzling and attempting to understand it. It is not obvious how biological evolution could produce mechanisms that are able to support such mental processes.

Many species develop behavioural and internal competences that depend on the environment during development (e.g. which language a child speaks, and which mathemat-

[12] His suggestion about learning based on a *tabula rasa* can be criticised.

ical problems are understood), so the genome-driven processes must create some innately specified competences partly under the influence of the genome and partly under the influence of combinations of sensorimotor signals during development ((Held & Hein 1963; McCarthy 2008)). For humans at least, the internal processes of competence-formation though brain modification must go on long after birth, suggesting that the genome continues producing, or enabling, or constraining effects (including changes in sexual and parental motivations and behaviours) long after the main body morphology and sensory-motor mechanisms have developed.

10. Self-transformation in biological VMs

(Karmiloff-Smith 1992) presents many examples where, *after* achieving behavioural competence in some domain, learners (including some non-human species) re-organise their understanding of the domain in such a way as to give them new abilities to think and communicate about the domain. After children develop linguistic competences based on known phrases they spontaneously switch to using a *generative* syntax that allows *derivation* of solutions to novel problems, instead of having to learn empirically what does and does not work. (Craik 1943) pointed out the value of such mechanisms in 1943, suggesting that they could be based on working mental models[13]. (Grush 2004) and others suggest that such models could work as simulations or emulations. However, when used for reasoning purposes, as opposed to statistical prediction, a decomposable information structure is required, for instance when proving geometrical theorems (Sloman 1971).

The mental models we use to explain and predict, include things like gear wheels, bicycles, electric circuits and other mechanisms that are too new to have been part of our evolutionary history. So, at least in humans, the model construction process cannot all be encoded in the genome: the specific models need information obtained after birth from the environment, and, in the case of creative inventors, ideas thought up by the individual.

So, the genome specifies not only physical morphology and physical behavioural competences, but also a multi-functional information-processing architecture developed partly in species-specific ways, over an extended time period, partly under the control of features of the environment, and includes not only mechanisms for interpreting sensory information and mechanisms for controlling external movements, but also mechanisms for building and running predictive and explanatory models of structures and processes, either found in the environment or invented by the individual[14]. How can a genome spe-

[13] I have not been able to find out whether Craik and Turing ever interacted. Turing must have known about his work, since he was a member of the Ratio club, founded in honour of Craik, shortly after hedied in a road accident in 1945.

[14] It is argued in (Sloman 1979, 2008) that this requires types of “language" (in a generalised sense of the word, including structural variability and compositional semantics) that evolved, and in young humans develop, initially

cify ongoing construction processes to achieve that functionality? I don't think anyone is close to an answer, but I'll offer a conjecture: evolution discovered the virtues of virtual machinery long before human engineers.

Previous sections outlined the benefits of virtual machinery in human-designed computing systems and their advantages compared with specifying, designing, monitoring, controlling and debugging the physical machinery directly, because of the coarser granularity and the use of application-relevant semantics. Perhaps biological evolution also found the use of virtual machinery in animals advantageous for specifying types of competence at a relatively abstract level, avoiding the horrendous complexity of specifying all the physical and chemical details. The initial specification of behavioural competences in the genome might be far more compact and simpler to construct or evolve if a virtual machine specification is used, provided that other mechanisms ensure that that “high level language" is mapped onto physical machinery in an appropriate way. The use of self-monitoring processes required for learning and modifying competences, including debugging them, may be totally intractable if the operations of atoms, molecules or even individual neurones are monitored and modified, but more tractable if the monitoring happens at the level of a RVM.

So something like a compiler is required for the basic epigenetic processes creating common features across a design, and something more like an interpreter to drive subsequent processes of learning and development.

11. The evolution of organisms with qualia

We have seen that virtual machinery can be implemented in physical machinery, and events in virtual machines can be causally connected with other VM events and also with physical events both within the supporting machine and in the environment, as a result of use of complex mixtures of technology for creating and maintaining virtual/physical causal relationships developed over the last seven decades. Some of the events and processes in virtual machines are not identical with the underlying physical machinery and their description requires an ontology that is not definable in terms of the ontology of physics.

The use of such virtual machinery can enormously simplify the design, debugging, maintenance, and development of complex systems. Finally, and perhaps most importantly, in machines that need to monitor and modify *their own* operations, performing the monitoring and modifications at the level of virtual machinery can be tractable where the corresponding tasks would he intractably complex and too inflexible if done by monitoring and modifying physical machinery.

for *internal* information-processing, not for external communication. We can call these „generalised languages" (GLs).

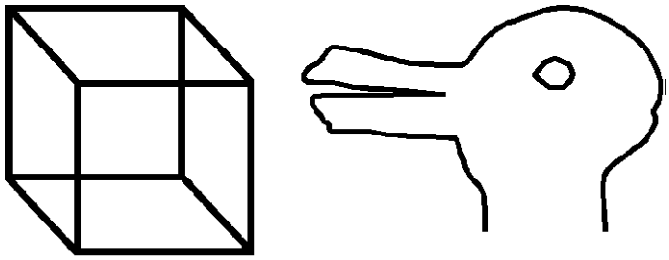
So, biological evolution could have gained in power, flexibility, and speed of development by using virtual machine descriptions in the genome for specifying behavioural competences, instead of descriptions of the physical details. Moreover if some of the virtual machinery is not fully specified in the genome, and has to be developed after birth or hatching by making use of new information gained by the individual from the environment, then that post-natal construction process will be much simpler to specify, control and modulate if done at the virtual machine level rather than specifying all the chemical and neuronal changes required. And finally self-monitoring, self-control, and self-modification in a sophisticated information-processing system needs to control virtual not physical, machinery.

12. Towards an understanding of qualia

For an intelligent organism perceiving, thinking about and acting on a rich and complex environment that contains enduring objects and processes at various locations not all constantly in perceptual range, it will be useful to store information about the environment using one or more appropriate virtual machines. Visual and haptic processes perceiving the same portion of the environment could include overlapping virtual machines dealing with different aspects of the environment processed at different levels of abstraction in parallel (Sloman 2009). Data-structures representing visible portions and features of the environment, e.g. visible portions of surfaces with colour, shape, orientation, curvature, speeds of motion or rotation, and relationships to other surface fragments (i.e. not the specific sensory signals), will then be components of virtual machines. If the information structures created during visual perception, are sometimes accessed by self-monitoring processes that attend not to what is in the environment, but to the content of what is currently being perceived, then we potentially have an explanation of the phenomena that have led to philosophical and other puzzles about the existence and nature of sensory qualia, which are often regarded as defining the most difficult aspect of mind to explain in functional terms, and whose evolution and development in organisms Huxley and others found so difficult to explain. See also (Sloman & Chrisley 2003).

To illustrate this point: when ambiguous figures, e.g. in Figure 1, are experienced as switching from one view to another, that will involve a change in the contents of some virtual machinery, and those contents will be represented at a virtual machine level, referring to different perceptual contents, including distance, direction of slope, bodyparts, direction faced, etc. More generally, we can attempt to identify the information contents needed in a wide variety of perceptual experiences that perform some function in enabling or controlling behaviour or testing theories, or generating hypotheses, surprises or questions.

Figure 1:



Necker Cube (a)

Duck-rabbit (b)

Figure 1: Each of the two figures is ambiguous and flips between two very different views. (a) can be seen as a 3-D wire frame cube. For most people it flips between two different views of the cube, in which the 3-D locations, orientations and other relationships vary. In (b), the flip involves changes in body parts, the facing direction, and likely motion – requiring a very different ontology.

Attempting to redesign, test and debug, working examples of such mental processes in robots will help us understand more clearly how it may be possible for the virtual machinery to be extended so as to include components that can detect, record, and make use of information about, what the contents of perceptual experiences are, i.e. what qualia are. As argued in (Sloman & Chrisley 2003), if the concepts used for recording such meta-information are not all pre-programmed but are produced by internal self-organising classifiers as proposed in (Kohonen 1989), then the resulting concepts, though useful for the individual concerned, may be inherently uncommunicable to others because their use implicitly refers to the discrimination mechanisms used in their application, an example of what (Campbell 1994) calls “causal indexicality".

Ryle, Dennett and others, have attempted to identify deep confusions in talk about consciousness and *qualia*, but such things clearly exist, though they are hard to characterise and to identify in other individuals and other species. Analysis of examples, including ambiguous figures, such as Figure 1, helps to determine requirements for explanatory mechanisms. Such pictures illustrate the *intentionality* of perceptual experience, i.e. interpreting something as referring to something else and the different *ontologies* used by different experiences. I suggest that that is only possible within running virtual machinery, since concepts like “interpreting", “referring", ‘intending" and “looking", are no more definable in the language of physics than ‘pawn" or “threat".

Many organisms can, I suspect, create and use such virtual entities without having the meta-semantic mechanisms required to detect and represent the fact that they do. One ofthe important facts relating to the diversity of kinds of mind, referred to in (Whittaker 1884), is that not all organisms that have qualia know that they have them! We can separate the *occurrence* of mental contents in an organism from their *detection* by the organism, which requires additional architectural complexity to support self-observation

and self-description mechanisms. I expect we shall need to experiment with a range of increasingly complicated working examples, using different kinds of mechanism, in order to understand better some of the questions to be asked about mental phenomena in biological organisms. This is very close to Arbib's research programme described in (Arbib 2003).

13. Supervenience, realisation, identity and levels

This section is an attempt to connect with some of the suggestions analytical philosophers have made about the mind body relation and how that compares with relationships between computers and their computations. The topic is vast with many detailed ramifications, and all I can do here is indicate some differences between the ideas presented here and a subset of previous attempts to characterize mind-matter relationships. (I don't claim to have surveyed everything written or said on this topic).

The main point is that, as far as I know, no other philosopher has attempted to characterise in detail the relationships between contents of running virtual machines in complex computing systems and the physical technology on which they depend. Insofar as computation has been mentioned at all in this context it is usually assumed to be either(a) just the execution of a single program which takes some initial input and later produces a result (e.g. calculating the value of a mathematical function for certain arguments, or answering the question whether a proof of a specific formula exists in some formal system), or (b) the operation of a finite state machine that at various points can select the next action on the basis of some input received, as described in (Block 1996), for example.

Such systems do not have the characteristics I have described in running virtual machines, namely multiple concurrently active non-physical (software) mechanisms influencing one another's behaviours while some of them are also connected with specific internal hardware subsystems and also things going on in the environment, sensed by or controlled by processes in the running virtual machines. Such a system can be thought of as a complex network of causally connected subsystems in which many control and communication functions are exercised in parallel, some of them including links to structures and processes that are not part of the system. Moreover, during the operation of such a system the number and connections between sub-systems can change, as can the underlying physical machinery, e.g. because processes get relocated in the machine's memory, or because some physical components are repaired or replaced. Many programming systems even allow the program instructions to be changed at run time, either by changing interpreted source code or by using an incremental compiler to change code at run time[15]. This could enable future robots to grow their virtual machinery as they interact with the environment, as human infants and toddlers appear to do. Virtual machine supervenience is a much richer and deeper relationship than superve-

[15] As in Poplog http://www.cs.bham.ac.uk/research/projects/poplog/freepoplog.html

nience of states or properties. The former has considerable implications for science as well as philosophy of mind and metaphysics.

Moreover when changes in virtual machines occur they need not all be changes inmeasurable quantities, such as size, orientation, distance, current, voltage, magnetic fields, and so on. That's because the processes can include things like construction, transmission, and analysis of complex *structured* entities such as words, sentences, paragraphs, problems, theories, explanations, diagrams, parse-trees, graphs, topological maps, logical formulae, proofs, intentions, plans, questions, answers to questions, proposals, decisions, and more. (Some of these are information-bearing structures, others information contents, and some are both.) Insofar as the variations in running virtual machines are not all quantitative, the causal relations cannot be expressed in algebraic formulae, as often happens in the physical sciences and some branches of engineering. For example, causal relationships in a spelling corrector or chess playing virtual machine are expressed in the form of algorithms and databases, not equations. A corollary is that much philosophical discussion of how to detect causal connections by comparing rates of change is irrelevant since in many cases there is no well-defined notion of *amount* of change and therefore no measure of rate of change - though in some cases there are partial orderings - e.g. if one set of changes subsumes another, while other sets merely overlap. Such changes have to be described rather than *measured*. This has implications for the form of psychological theories.

For some of the contents of virtual machinery, such as contents of sophisticated visual systems in animals, or the changes that occur as a mathematician looking at a diagram notices a way of modifying it to construct a proof in euclidean geometry, we do not yet know what the entities are that the various mental subsystems construct and manipulate. I'll assert without argument here that what goes on in most forms of animal visual processing remains a mystery, although many physical and physiological details are known, including which parts of brains are involved.

All of that complexity is usually ignored when philosophers discuss mind-brain relationships. For example, one of the themes in recent philosophy has been discussion of how mental states or mental properties relate to, or supervene on, brain states (or, more generally, physical states including aspects of the environment). That's a pale shadow of the question I have been posing about how a complex mental machine performing a host of perceptual, learning and control functions, is related to and supervenes on physical machinery.

One of the important ideas goes back to a concept introduced by G.E. Moore in connection with ethics around 1903. He said that ethical properties of actions, such as their goodness or badness “supervene" on their non-ethical properties, such as what was done by whom to whom and with what intention and what consequences. The relation of supervenience does not allow the ethical properties to be deduced logically from the other properties. It is a weaker relation, namely that it is impossible for two actions to differ ethically, e.g. one being good and the other bad, unless they also differ in a non-

ethical way. This idea was transferred to philosophy of mind by D. Davidson around 1970. He asked whether mental properties and states supervene on physical properties and states in the sense that it is impossible for a person's mental properties or state to change unless there is also a physical change. This and related ideas have been elaborated by various philosophers in the last few decades, e.g. (Kim 1993, 1998).

The idea of supervenience is useful, but covers significantly different cases. For example, what I have been talking about can be described as “virtual machine supervenience" since what supervenes on a physical system, or on lower level virtual machinery is not a state or a property but a running and changing virtual machine with interacting internal components. We can contrast this with other sorts of supervenience.

“Pattern supervenience" occurs when a pattern defined by a collection of relationships necessarily exists when some other pattern exists: E.g. vertical columns of dots supervene on a collection of horizontal rows of equally spaced dots.

“Agglomerative supervenience" (which could also be called “part-whole-supervenience"), exists when some property or entity is defined in terms of the collective contribution of many parts of an object. Examples include aspects of a physical object such as its mass, centre of gravity, angular momentum, or kinetic energy which can be computed from the properties and arrangements of the parts. These are sometimes described as “useful fictions", by philosophers who misunderstand their role in science. For example, the centre of gravity of a rigid object is not a fiction: it is a real location defined by the distribution of matter of the object. Forces directed through the centre of gravity (or centre of mass) produce different effects from other forces. A change in the centre of gravity of an object can cause it to fall over.

“Mathematical supervenience" occurs if whenever something has property P1 it also has property P2 because having P2 is mathematically derivable from having P1. For example having an odd number of legs can supervene on having five legs. Being a polygon with five vertices supervenes on being a polygon with five sides. In this case the supervenience is symmetric. It is not known whether being the sum of two prime numbers supervenes on being an even number.

The existence of shadows illustrates a kind of supervenience. A shadow cannot exist without a light source, a partly illuminated surface and an intervening object that casts the shadow. If the shadow changes in any way then some aspect of the light source, the intervening object or the partly illuminated surface must have changed. This sort of thing might be described as “causal supervenience".

For some kinds of supervenience there is a defensible claim that the supervenience is a type of identity. For example if a pattern consisting of horizontal evenly spaced rows of evenly spaced dots exists, then a pattern of equally spaced vertical columns of evenly spaced dots also necessarily exists. It could be argued that the two patterns are the same thing viewed differently using selective attention, and described differently.

Some philosophers have attempted to solve the puzzle of how mental events can cause physical events if the physical world is causally closed, by arguing that mental states, properties, events, etc. not only *supervene* on physical entities but are *identical* with them, so that mental causation just is physical causation.

The identity claim is undermined by the existence of virtual machine processes whose descriptions require concepts (such as “attack" or “threaten" in chess, “incorrect spelling")that are not definable using the concepts of the physical sciences, along with the claim that the relationship between the virtual machinery and the physical machinery is not symmetric(they are not each supervenient on the other), However there is no space here for a full discussion.

Another kind of attempt to rebut claims that both physical and non-physical events can be causes argues that the virtual machine events and processes are “epiphenomenal" ,i.e. they are incapable of being causes. However, in the case of events in virtual machinery running in computing systems, that is just false: the whole point of designing and constructing many virtual machines is to ensure that certain things go on in them that cause other things to happen, and there are now vast numbers of virtual machines running on this planet because of such effects, which are very hard to produce using only physical machinery. Their production in virtual machinery is also non-trivial. But a great deal of effort and ingenuity by hardware and software engineers has made that possible.

I don't claim to have settled all the philosophical problems and disputes, though I hope it is clear that the phenomena of virtual machine supervenience described above are different in important ways from the much simpler cases philosophers have previously discussed. They are different in their complexity, the richness of content of what supervenes, and in the mappings between virtual machines and the underlying physical machinery, which in some cases are rapidly changing relationships. I have tried to argue that this was very important for evolution of information processing systems in organisms, and conjectured that the evolution of virtual machines that can inspect, evaluate, remember and otherwise make use of some of their own virtual machine contents (e.g. the intermediate forms of representation used in visual information processing) will eventually be shown to explain the phenomena that gave rise to philosophical discussions of qualia. We can now specify that qualia are potentially introspectible components of virtual machinery, and by analysing functions of vision we can see why the existence of self monitoring mechanisms with access to the contents of qualia are, in some situations, biologically useful.

One type of philosophical functionalist attempts to define various types of mental state in terms of sets of input-output relationships. A common, and much debated objection to this is the zombie argument: anti-functionalists claim that they can imagine zombies, namely entities whose external features and visible behaviours in all circumstances make them indistinguishable from human beings, even though they lack all mental states and processes, and in particular lack consciousness. I have no doubt that many

people can imagine that and in principle such machines could be implemented - with behaviours indistinguishable from human behaviours, but without the internal virtual machinery I have been describing. They might, for instance, at least in principle, have huge lookup tables determining behaviour in all possible circumstances, instead of the kind of virtual machinery working out what to do that I have been referring to.

But trying to use the zombie argument against virtual machine functionalism of the sort presented here requires imagining that all the *internal*, invisible, non-physical causal interactions in a human can be replicated without any mental states and processes, in a zombie devoid of consciousness is another matter. The claim to be able to imagine that is just philosophical bluster: no human is capable of imagining all the detailed virtual machine processes required to replicate human functionality, and the claim to be able to imagine it all happening in a mindless zombie should be taken no more seriously than the incoherent claim to be able to imagine the whole universe moving west, or the claim to imagine that it is now noon at the centre of the earth. People can imagine that they imagine it, but that doesn't prove they really can imagine it. In any case, the history of mathematics shows clearly that what people think they can imagine is not a proof of possibility.

14. Implications for the future of philosophy

Experience shows that, for many thinkers, this talk of virtual machinery whose main features cannot be described in the language of physics will be rejected as ridiculous mumbo-jumbo even though exactly this sort of talk has proved essential for the design and development of ever more sophisticated information processing systems used for various practical purposes, and may be increasingly important as we require our systems to become better at knowing what they are doing and how they do it. As far as I know, the only educational process for producing a deeper understanding of the issues is to let people have personal experience of trying to build machines that can do things humans and other animals can do instead of trying to discuss these topics only on the basis of general ideas about what computation is.

Experience also shows that for thinkers of a different sort none of this will shake belief in an unbridgeable mind/body explanatory gap. As argued in (Sloman 2010b), some cases of opposition will be based on use of incoherent concepts (e.g. a concept of “phenomenal consciousness" *defined* to involve no causal or functional powers - which leaves as a mystery how people come to talk about them.). Working systems that show how different robot designs correspond to different products of evolution may help. But it is likely that no form of argument or practical experience of designing working systems will convince people who simply do not wish to believe that the workings of human minds can be understood in terms of information processing mechanisms, including possibly some that work in ways that are very different from present day computers.

Moreover, current achievements in AI vision, motor-control, concept-formation, forms of learning, language understanding and use, motive-generation, decision-making, plan formation, problem-solving, and many others, are still (mostly) far inferior to those of humans and other animals, in part because designers typically consider only a small subset of the requirements for biological intelligence. For example many researchers aim only to produce robots with the kind of competence that (Karmiloff-Smith 1992) refers to as “behavioural mastery", and totally ignore the other kinds of abilities humans develop, including the ability to reflect in advance about possible combinations of actions that are suited to this particular environment and have never previously been proposed.

One of the abilities young humans, and also a subset of other species, develop is the ability to *work out* in advance what will happen if some action is performed instead of having to find out by *trying to perform* it, which could be fatal, as noticed by (Craik 1943).Some other species also seem to have this sort of capability. In humans it appears to be closely related to the development of mathematical competences, which allow problems to be solved by reasoning about abstract structures, instead of having always to examine objects in the environment, as required for the physical sciences. At present none of the designs being explored by roboticists (as far as I know) would allow the robots to start noticing mathematical features of time, space and motion and begin to discover something like euclidean geometry or even elementary topology.

Even if we omit uniquely human competences, current robots are still far inferior to other animals. There is no easy way to close those gaps, but there are many things to try, as long as we think clearly about what needs to be explained.

It has been clear to philosophers for some time that studying logic is necessary for a study of philosophy of mathematics, philosophy of science, philosophy of language, and also for epistemology. The time has come for practical experience in designing, testing, debugging, criticising and analysing working models of various kinds of human and animal competence to be regarded as a necessary component of the education of professionally competent philosophers of mind, philosophers of biology, and philosophers interested in aspects of metaphysics concerned with varieties of causation[16]. Perhaps one side effect of such an expansion of philosophical education will be investigation of the need for new forms of physical information processing machinery required to support aspects of biological information processing that don't match the kinds of virtual machinery that can be supported by current computer technology.

For example, I suspect that we need forms of information processing that allow for more kinds of causal interaction between virtual machines, including continuous opposition and competition, features which are now only represented indirectly using (somewhat artificial)numerical measures of strength or importance in selection among competing alternatives.

[16] I rashly made this claim a long time ago in (Sloman 1978). But philosophical education changes more slowly than I had anticipated.

This fails to distinguish clearly the difference between *producing* a result of competition and *predicting* that result. Different virtual machine components cannot directly oppose each other at present, though they can control physical machines that oppose each other, e.g.by pushing in opposite directions. This may make it difficult to produce accurate models of conflicting motivation or irresistible impulses (e.g. the impulse to look at something in the environment or to laugh, cry, sneeze or scratch an itch). I have begun to explore such issues and have incomplete discussions on my web site[17].

One thing is clear: insofar as we are talking about machines that instead of merely deriving new information structures from old ones, also interact continuously with the physical and social environment and which control performances of different functions in parallel, such as walking, talking, enjoying the view and eating a sandwich, we are not talking about Turing machines, even if some of Turing's ideas are relevant to the task. Turing machines were designed to study possible forms of transformation of information structure according to fixed rules. What we now need are machines designed (like biological information processing systems) to have a variety of *control* functions.

I believe we have so far only begun to study a small subset of the variety of information processing systems that are used by plants and other animals. It may take a surprisingly long time to design robots that learn and develop as humans do, including developing strong interests and competences in mathematics and philosophy. No doubt when we produce them, they will disagree as much among themselves about answers to philosophical questions as we do, including the question whether machines can have minds.

References:

Arbib, M. A. 2003. Rana computatrix to Human Language: Towards a Computational Neuroethology of Language Evolution. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, 361 (1811): 2345-2379. Http://www.jstor.org/stable/3559127.

Block, N. 1995. On a confusion about the function of consciousness. *Behavioral and Brain Sciences*, 18: 227-47.

Block, N. 1996. What is functionalism? Http://www.nyu.edu/gsas/dept/philo/faculty/block /papers/functionalism.html (*The Encyclopedia of Philosophy Supplement*. Macmillan. 1996).

Campbell, J. 1994. *Past, Space and Self*. Cambridge: MIT Press.

Carnap, R. 1947. *Meaning and necessity: a study in semantics and modal logic*. Chicago: Chicago University Press.

Chalmers, D. J. 1996. *The conscious mind: In search of a fundamental theory*. New York, Oxford: Oxford University Press.

Chomksy, N. 1959. Review of skinner's Verbal Behaviour. *Language*, 35: 26-58.

[17] http://www.cs.bham.ac.uk/research/projects/coga_/talks/

Craik, K. 1943. *The nature of explanation*. London, New York: Cambridge University Press.

Dyson, G. B. 1997. *Darwin Among The Machines: The Evolution Of Global Intelligence*. Reading, MA: Addison-Wesley.

Grush, R. 2004. The emulation theory of representation: Motor control, imagery, and perception. *Behavioral and Brain Sciences*, 27: 377-442.

Harnad, S. 1990. The Symbol Grounding Problem. *Physica D*, 42: 335-346.

Held, R., Hein, A. 1963. Movement-produced stimulation in the development of visually guided behaviour. *J. Of Comparative and Physiological Psychology*, 56 (5): 872-876.

Kant, I. 1781/1929. *Critique of pure reason*. Transl.: Norman Kemp Smith. London: Macmillan.

Karmilo-Smith, A. 1992. *Beyond Modularity: A Developmental Perspective on Cognitive Science*. Cambridge, MA: MIT Press.

Kim, J. 1993. *Supervenience and Mind: Selected philosophical essays*. Cambridge: Cambridge University Press.

Kim, J. 1998. *Mind in a Physical World*. Cambridge, MA: MIT Press.

Kohonen, T. 1989. *Self-Organization and Associative Memory*. Berlin: Springer-Verlag.

McCarthy, J. 2008. The well-designed child. *Articial Intelligence*, 172 (18): 2003-2014.

Sloman, A. 1971. Interactions between philosophy and AI: The role of intuition and non-logical reasoning in intelligence. W: Proc 2nd ijcai: 209-226. London: William Kaufmann. Http://www.cs.bham.ac.uk/research/coga/04.html#200407.

Sloman, A. 1978. *The computer revolution in philosophy*. Hassocks, Sussex: Harvester Press (i Humanities Press).

Sloman, A. 1979. The primacy of non-communicative language. Red. M. MacCafferty i K. Gray. The analysis of Meaning: Informatics 5 Proceedings ASLIB/BCS Conference, Oxford, March 1979:1-15. London: Aslib. Http://www.cs.bham.ac.uk/research/projects/coga/81-95.html#43.

Sloman, A. 1996. Beyond turing equivalence. Red. P. Millican, A. Clark. *Machines And Thought: The Legacy Of Alan Turing*, vol I: 179-219. Oxford: The Clarendon Press. (Presented at Turing 90 Colloquium, Sussex University, April 1990).

Sloman, A. 2007. Why symbol-grounding is both impossible and unnecessary, and why theory-tethering is more powerful anyway. Research Note No. COSY-PR-0705. Birmingham, UK. Http://www.cs.bham.ac.uk/research/projects/coga/talks/#models.

Sloman, A. 2008. Evolution of minds and languages. What evolved first and develops first in children: Languages for communicating, or languages for thinking. *Generalised Languages*: GLs. Research Note No. COSY-PR-0702. Birmingham, UK.

Sloman, A. 2009. Some Requirements for Human-like Robots: Why the recent over-emphasis on embodiment has held up progress. Red. B. Sendho, E. Koerner, O. Sporns, H. Ritter i K. Doya. *Creating Brain-like Intelligence*: 248-277. Berlin: Springer-Verlag.

Sloman, A. 2010a, August. How Virtual Machinery Can Bridge the Explanatory Gap. Red. S. Doncieux i in. *Natural and Artificial Systems*. Proceedings SAB 2010, LNAI 6226: 13-24. Heidelberg: Springer.

Sloman, A. 2010b. Phenomenal and Access Consciousness and the “Hard” Problem: A View from the Designer Stance. *Int. J. Of Machine Consciousness*, 2 (1): 117-169.

Sloman, A. 2011. What's information, for an organism or intelligent machine? How can a machine or organism mean? Red. G. Dodig-Crnkovic, M. Burgin. *Information and Computation*: 393-438. New Jersey: World Scientic.

Sloman, A., Chrisley, R. 2003. Virtual machines and consciousness. *Journal of Consciousness Studies*, 10 (4-5): 113-172.

Turing, A. 1950. Computing machinery and intelligence. *Mind*, 59: 433/460 (przedruk w: E.A. Feigenbaum, J. Feldman. Red. *Computers and Thought.* McGraw-Hill, New York, 1963: 11-35).

Turing, A. M. 1936. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Soc*., 42 (2): 230-265.

Whittaker, T. 1884, April. Review of G.J. Romanes Mental evolution in animals. *Mind*, 9 (34): 291-295.