The history, nature, and significance of virtual machinery

In natural and artificial systems

Aaron Sloman

School of Computer Science, University of Birmingham

http://www.cs.bham.ac.uk/~axs/

Related slides are available in my ‘talks’ directory:
http://www.cs.bham.ac.uk/research/projects/cogaff/talks/

and on Slideshare.net
http://www.slideshare.net/asloman/presentations

Robin Milner[*]

About 25 years ago, I remember Robin Milner giving an invited talk at a conference (an Alvey conference, I think) about how theory repeatedly follows practice in computing.

His main thesis was something like this:

> Engineers solve practical problems and in the process invent new things and do new things that they don't necessarily characterise explicitly.
>
> Theorists notice what's happening and invent formalisms and models to account for it, and mathematical techniques for reasoning about it.
>
> Meanwhile the engineers move on, and later the theorists follow, and so on.
>
> and the cycle continues.

I think my topic illustrates that cycle, though this particular development seems to me to be still in its early stages.

[*]Robin Milner died on Saturday 20th March 2010, and will be much missed.

ABSTRACT (Part 1)

Abstract:

Over the last six or seven decades there have been a lot of separate developments adding functionality of different sorts to computing systems including (in no significant order):

memory management, paging, cacheing, interfaces of many kinds, interfacing protocols, device drivers, adaptive schedulers, privilege mechanisms, resource control mechanisms, file-management systems, interpreters, compilers and run-time systems for a wide variety of types of programming language, garbage collectors, varied types of data-structure and operations on them, tracing and debugging tools, pipes, sockets, shared memory systems, firewalls, virus checkers, security systems, network protocols, operating systems, application development systems, etc. etc.

All this is very familiar to computer scientists and software engineers, though different experts know about different sub-sets of these developments and the whole package is not often described adequately.

In a way it is very familiar to millions of users, who are incapable of describing what they are using.

A consequence of all these developments is that we can now have, in addition to all the physical computing machinery that we use, varying collections of non-physical machinery made up of various kinds of interacting components with causal powers that operate in parallel with the causal powers of the underlying machines, and can help to control those physical machines, but with different kinds of granularity and different kinds of functionality from the physical machines.

These are running virtual machines (RVMs), as opposed to the mathematical abstractions that are sometimes called virtual machines (e.g. a universal turing machine, the java virtual machine, the/a linux virtual machine) whose instances are among the RVMs.

ABSTRACT (Part 2)

I shall try to characterise some of the forms of control and self-control that are made possible by the use of RVMs and suggest that biological evolution discovered the need for them long before we did and probably produced more complex and varied kinds than we have so far.

All this seems to me to have implications for the future of machine intelligence (and more generally for development of increasingly robust, autonomous systems).

It also has implications for the future of philosophy of mind, and looks likely to achieve the final removal of what T.H.Huxley called “the explanatory gap” between the physical and the mental, which was, and still is, a serious problem for Darwin’s theory of evolution.

To illustrate what I am talking about I’ll start with a simple demo (using the Pop-11 SimAgent toolkit.)

`http://www.cs.bham.ac.uk/research/projects/poplog/figs/simagent`

This is a request for help on my part:

I would like corrections if I make any incorrect historical or other factual claims,

and suggestions for improved ways of summarising and explaining the significance of all these developments,

especially for the benefit of people who do not work in computer science or software engineering.

KEY IDEA: Running Virtual Machine (RVM)

The idea of a running virtual machine (RVM) should not be confused with the abstract mathematical structure defining a type of VM, which can be thought of as a “Mathematical Model” (MM), about which theorems can be proved, etc., but which does not **do** anything, anymore than numbers do.

Physical processes:	**Mathematical models:**	**Running virtual machines:**
currents	numbers	calculations
voltages	sets	games
state-changes	grammars	formatting
transducer events	proofs	proving
cpu events	Turing machines	parsing
memory events	TM executions	planning

Distinguish: **PMs** **MMs** **RVMs**

Illustrate with demos.

E.g. Sheepdog demo. See Movie 5 here:
`http://www.cs.bham.ac.uk/research/projects/poplog/figs/simagent`

The “Explanatory Gap”

A problem that puzzled Darwin and fired up his critics:

- There’s lots of evidence for **evolution of physical forms**.
- There’s no evidence that human minds could be products of evolution.
- There seems to be no way that physical matter can produce mental processes.

This is the so-called “explanatory gap”

(T.H. Huxley – and various precursors).

- Until the last few decades, explanatory mechanisms linking physical and mental phenomena were not even conceivable to most scientists: hence Huxley’s “explanatory gap” and Chalmer’s “Hard problem of consciousness”, etc.
- Now, as a result of a great deal of work on hardware, software, firmware, and CS theory, we know how to make things that have some of the features required for working explanatory models of mental processes, with some of the key features of mental processes (including having causal powers, without being physical processes) – but only in very simplified form.
- Most philosophers, psychologists and neuroscientists have ignored or misunderstood this, and so have many AI/Computing/Software researchers.

  I’ll give some pointers, but not explain in detail, below.
- There are deep implications for philosophical analyses of causation.

  E.g. **downward** causation from mind-like events and processes to physical events and processes.

Let’s vote

Does **your** ontology include virtual machines?

Who agrees with the following?

- Ignorance can cause poverty?
- Over-zealous selling of mortgages can cause hardship for people in many countries?
- Voting can influence decisions?

If you AGREE with any of these, i.e. if you think such an effect CAN occur, then it appears that you (wittingly or unwittingly) have social and/or socio-economic virtual machines in your ontology.

What that means is another (hard) question, partially answered below.

In order to explain what a non-physical virtual machine is we need to:

- Explain what a machine is
- Define “physical machine” in terms of the sorts of concepts that suffice to describe the structure and operation of the machines.
- Provisionally define “virtual machine” as a machine that is not (fully) physically describable.
  (The phrase “virtual machine” is unfortunate, but it’s too wide-spread to change.)
- Later on generalise the notion “virtual” by defining it as relative.

What is a machine (natural or artificial)? (1)

The word “machine” often refers to a complex enduring entity with parts

(possibly a changing set of parts)

that **interact causally**[*] with other parts, and other “external” things, as they change their properties and relationships.

[*]Causation is discussed later.

The internal and external interactions may be

- **discrete** or **continuous,**
- **concurrent** (most machines), or **sequential** (e.g. row of dominoes, a fuse(?))
- if concurrent then **synchronised** or **asynchronous**

In Turing machines, everything is:

- Internal
- Discrete
- Sequential
- Synchronous

Concurrent and synchronized TMs are equivalent to sequential TMs.

I.e. parallelism in TMs adds nothing new.

But some machines have concurrent parts that are not synchronised, so they are not TMs, even if they have TM-like components.

And systems interacting with a physical or social environment are not TMs, since a TM, by definition, is a self-contained: machine table+tape.

What is a machine (natural or artificial)? (2)

The word “machine” often refers to a complex enduring entity with parts

(possibly a changing set of parts)

that **interact causally** with other parts, and other “external” things, as they change their properties and relationships.

The internal and external interactions may be

- **discrete** or **continuous,**
- **concurrent** (most machines), or **sequential** (e.g. row of dominoes, a fuse(?))
- if concurrent then **synchronised** or **asynchronous**

NOTEs

1. Machines, in this general sense, do not have to be artificial, or man-made, or deliberately designed to do what they do.

2. The perception of machines and how they work is one of the important functions of human visual perception, and haptic/tactile perception, (possibly also in some other species).

That includes the perception of structures, processes and causal relationships (proto-affordances).

This is generally ignored by vision researchers.

Perception of affordances is a special case of this. E.g. See

Architectural and Representational Requirements for Seeing Processes, Proto-affordances and Affordances,

`http://drops.dagstuhl.de/opus/volltexte/2008/1656`

Typical features of machines (natural and artificial):

Machines

- can have various degrees and kinds of complexity
  (often hierarchical – machines composed of machines)
- allow changes/processes to occur within them
  usually concurrent changes (e.g. gear wheels turning, ends of lever moving in opposite directions)
- can acquire, manipulate, use, produce, and/or transfer matter, energy or information.
- include processes that involve not mere change, but also **causation**
  - within the machine
    E.g. parts moving other parts, forces transmitted, information stored, retrieved, derived or transmitted, parts controlling or activating, other parts.
  - partly within the environment
    E.g. if there are sensors, motors, and communication channels
  - involving matter, motion, forces, energy, **information**, ... and more
- are usually embedded in a complex environment with which they interact. Often the boundary between machine and environment is different for different sub-systems of the machine.
  As every mathematician knows, you can use pen and paper as an extension of your mind.
  Sloman IJCAI 1971:
  `http://www.cs.bham.ac.uk/research/cogaff/04.html#200407`
- may include some internal processes whose effects are not externally detectable,
  e.g. a machine playing chess with itself and learning to play better as a result. In some cases the unobservable internal processes can be inferred indirectly. (More on this later)

What is a **physical machine** (PM)?

Some, but not all, machines satisfying the previous definition are physical.

If a machine and its operations (processes, and causal relationships) are fully describable using concepts of the physical sciences (plus mathematics), it is a physical machine (PM).

That's a first draft specification.
I'll contrast that with a kind of machine whose parts and processes are not fully describable within the language of the physical sciences: some additional concepts are required.

E.g. concepts like "winning" in a game or "correct spelling" in a document, or "attempting" to achieve something, which, I suggest, cannot be defined in the language of the physical sciences.

(There is probably a variant definition that does not mention concepts, but I am not sure.)

The contents of the physical sciences, expand over time,
so the broadest notion of "physical machine" must refer to the indefinite future.

Examples of physical machines include:

levers, assemblages of gears, mechanical clocks, audio amplifiers, electronic devices, wireless control systems, clouds, tornadoes, plate tectonic systems, atoms, bacteria, brains, and myriad molecular machines in living organisms.

There is much we don't know about what sorts of machine can be built out of chemical components. E.g. read recent issues of *Scientific American*.

Virtuality: absolute and relative

I shall first introduce a disinction between machines that are physical and machines that are not, even though the ones that are not are "fully implemented" in physical machines.

Later we'll see that that is one of many examples of "layering" – one aspect of reality is layered on another, so that certain things are virtual relative to others.

But first we start with a single distinction.

Our preliminary notion of a running virtual machine (RVM), refined later, is defined as a machine in the sense defined earlier, but is not a physical machine, in the sense of "physical machine" defined above:

i.e. a RVM can be complex, with parts that interact with one another and with things outside the machine, but describing the parts and their operations requires use of concepts that are **not definable in terms of concepts of the physical sciences.**

E.g. spelling checker, chess program, proof checker, winning, invalid inference.

This notion will now be elaborated.

It is relative to a concept of a physical science, a concept that has changed over centuries, making the distinction a fluid one.

Non-physically-definable (NPD) concepts

Certain states, processes and interactions of some machines

can be described fully **only** if we use concepts
that are not definable in terms of concepts
of the physical sciences
(E.g. not definable in terms of the concepts of physics, chemistry, plus mathematics.)

Information-processing machines are examples.

“Information” is not used here in Shannon’s sense,
but in the sense that includes “reference”, “meaning”,
with properties and relations like:

truth, consistency, implication, contradiction,

These concepts are not **definable** in terms of concepts of the physical sciences.

Though every information-using machine must be implemented (realised) in a physical machine.
See `http://www.cs.bham.ac.uk/research/projects/cogaff/misc/whats-information.html`

Non-physical machines include: socio-economic machines, ecosystems, many biological control systems, and many of the things that run in computers, including games, spelling checkers, email systems, time-managment systems, operating systems and networking systems.

More on non-physically describable machines

Non-Physically-Describable Machines (NPDMs) are the subject matter of common sense, gossip, novels, plays, legends, history, the social sciences and economics, psychology, and various aspects of biology.

An important common feature is use of non-physically-definable concepts.

Examples of such non-physically-definable concepts:

“information”, “inference”, “contradiction”, “strategy”, “desire”, “belief”, “mood”, “promise”, “contract”, “checking spelling”, “file access violation”, “sending email”, “playing chess”, “winning”, “threat”, “defence”, “plan”, “poverty”, “crime”, “economic recession”, “election”, “war”, ...

For now I’ll take that indefinability as obvious:
It would take too long to explain and defend.

This is connected with the falsity of “concept empiricism” and “symbol grounding theory”. See
`http://www.cs.bham.ac.uk/research/projects/cogaff/talks/#models`
`http://www.cs.bham.ac.uk/research/projects/cogaff/misc/whats-information.html`

In computer science and software engineering NPDMs are often called “virtual machines”
(terminology possibly derived from some of the earliest examples: virtual memory systems).

This terminology is unfortunate – since the word “virtual” can suggest that such machines do not really exist – like the entities represented in virtual reality systems.

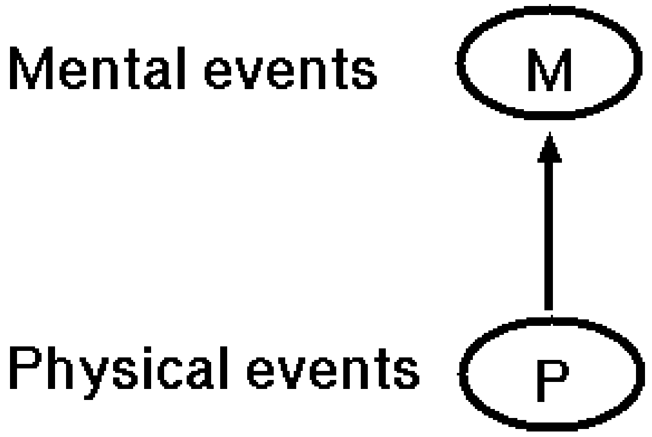
Nevertheless we are stuck with the word.

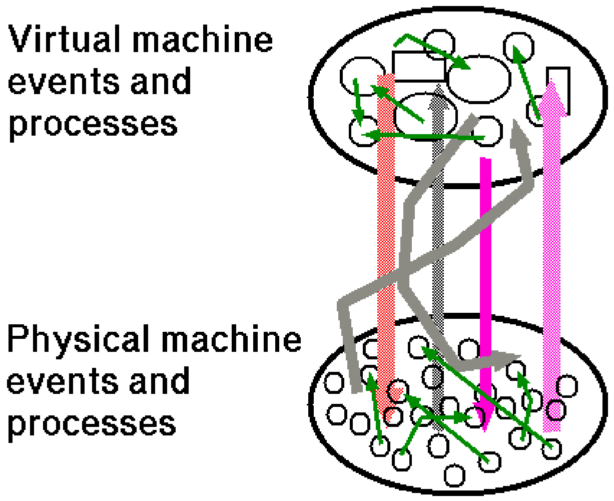
(Like the misleading phrase “Artificial Intelligence”, which labels a field that includes the study and modelling of natural intelligence.)

Later we’ll talk about layers of relative virtuality.

The 20th C Philosophical breakthrough: Virtual machinery

Brief introduction to the philosophical significance of the technology of virtual machinery (not virtual reality) developed over the last six decades: Processes and events in running virtual machines can be causes and effects, despite being implemented in deterministic physical mechanisms.




Virtual machine events and processes
Physical machine events and processes

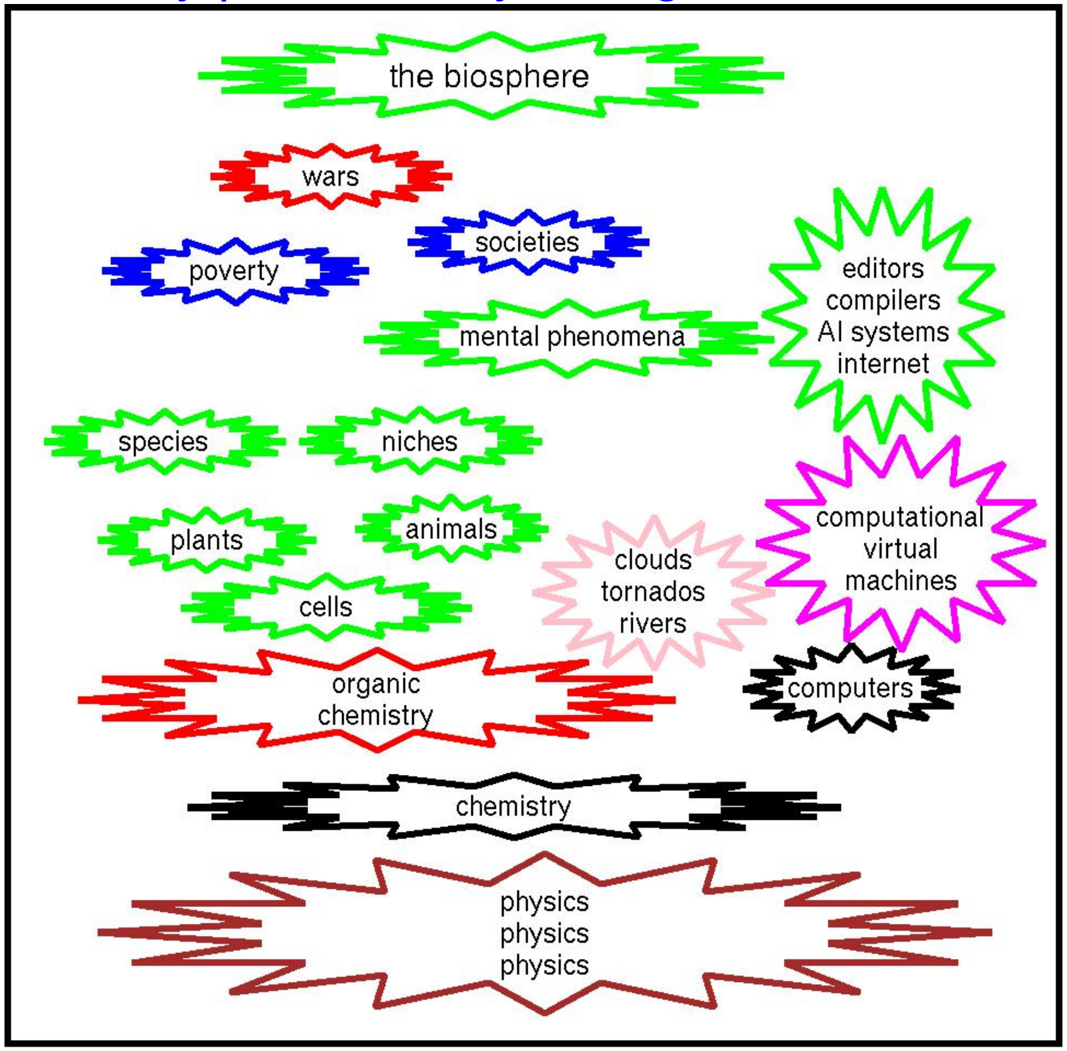
The erroneous picture on the left implies that there is only one-way causation from physical to mental (epiphenomenalism).

As the picture on right indicates, we need to think about running virtual machinery that co-exists with, and influences, underlying physical machinery, which it helps to control, even though the virtual machinery is all fully implemented in the physical machinery.

I.e. running software can cause changes in physical hardware, just as increases in poverty can cause increases in crimes involving movement of stolen objects.

Virtual machines are everywhere

Many produced by biological evolution



How many levels of (relatively) virtual machinery does physics itself require?
Compare the “Gaia hypothesis”

An example: memory management

A memory-management software process can detect a problem allocating a portion of contiguous memory required by a running process.

That detection event can then trigger a “garbage-collection” process which detects and merges unused portions of memory by re-locating portions in use, e.g. by

- reclaiming previously used memory locations that are no longer in use, and
- relocating many bit patterns,

  (by altering states of many switches in the computer’s memory)
- while altering many redirection addresses so that reference relations are preserved.

The relocation of patterns from one part of physical memory to another does not require movements of matter, but does involve transfer of energy and information, in signals that produce state-alterations in large numbers of bi-stable switches, e.g. when patterns are copied from one location to another and the old locations given new patterns.

Claiming that a software event (detection of a need for a big unused region of memory) causes certain physical changes is justified because there are many true conditional statements, including counterfactual conditionals, about what will happen if, what would have happened if, what would not have happened if, ... etc.

The truth of all those conditionals is the result of complex engineering design decisions.

This is not **translation** of (1) statements about VM events causing hardware changes into
(2) counterfactual conditional statements, but is a **partial explication** of (1):
truth of the conditionals helps to justify claims about causal influences from RVMs to physical changes.

Springs and levers vs enforced rules

There are some machines in which effects are propagated and constraints are enforced by means of purely physical mechanims that directly produce the required effects, for instance the system of levers and springs that ensures that

- if a boot lid is moved above a certain height then it is automatically pushed up further and held there,
- whereas if it is moved down below a certain height then it is pulled further down.

The propagation of effects of changes in a virtual machine and the enforcement of constraints on such changes is often of a very different kind, produced by quite different mechanisms,

- designed to maintain configurations of patterns in switch configurations,
- or to propagate changes in those patterns
- and sometimes to do that with patterns that are not composed directly of combinations of switch states,
- but are more abstract structures not directly mapped onto physical switch patterns
- for instance in the layers of structure used in network protocols, or a chess machine whose rules always ensure that attempts to check its king are blocked in advance.

In such systems the causal mechanisms at work are not visible objects like springs and levers but often rules that are compiled and run, or interpreted, by hardware designed to do whatever the installed rules specify!

Fixed and variable configurations of causes

The collection of constraints and stability patterns in the car boot lid system is fixed: the system remains the same indefinitely

(as long as it doesn't break or wear out.)

In contrast, collections of rules linking a set of structures, events and processes in a RVM can change frequently, if the system includes a rule interpreter or incremental compiler

contrast programming languages that require each program to be completely specified in advance, then compiled and run (perhaps run many times with different parameters).

For example, the SimAgent toolkit demonstrated in the sheepdog and "emotional" agent demos, allows rules to be edited, added, removed, or reordered while a program is running, and since the conditions and actions of rules can link quite complex structures a system implemented like that may be capable of making extensive changes to itself, which alter the networks of causation that define the nature of the virtual machinery.

Systems built on software mechanims that allow not just data structures, but also active rules and programs to be changed, can grow new architectures, or new VMs.

This can happen using

- either mechanisms in which lots of small, blind processes allow complex new things to emerge (as in patterns of swarming insects or birds)
- or mechanisms with richer semantic competences able to represent and achieve explicit re-structuring/reprogramming goals – triggered by a sophisticated motivational system.

Granularity differs at different levels

An indivisible virtual machine process, such as copying of a symbol from one abstract memory location to another can involve a large number of physical changes:

e.g. electrical pulses along conductors, and switches flipped from one state to another.

This is a common feature of mappings from virtual machine structures and events to the physical structures and events that underpin them: typically many distinguishable physical changes correspond to each “minimal” virtual machine change.

Another common feature is multiple realisability: the same VM process, if repeated, can make use of different physical machine processes on different occasions e.g.

- because the same abstract patterns are located in different parts of the machine at different times,
- because faulty physical components can be replaced by others using different technology, so that a new occurrence of a previous virtual machine event produces a new type of physical process, but interfacing mechanisms make the differences invisible to other parts of the system.

Some philosophers have mistakenly assumed that when a non-physical process “supervenes” on a physical process the two processes are isomorphic: ignoring the differences in granularity described above and other differences – containment can be circular in VMs but not in physical structures.

One reason why coarser granularity is sometimes **essential** for human developers and users, is to allow processes to be monitored and managed by human brains that would be unable to think or reason about full low-level details.

For that reason coarse-grained VMs may be needed in self-monitoring, self-modulating, self-debugging, self-extending systems. (Holes between the grains allow malware!)

Give some demos of RVMs

E.g.

- sheepdog,

  An interactive demo showing a virtual machine with a collection of concurrently active components (sheepdog, sheep) also causally linked to physical devices, e.g. computer mouse and screen (as well as internal registers, memory, etc.).

- “emotional agents”.

  Another interactive demo.

Some simple movies are here
`http://www.cs.bham.ac.uk/research/projects/poplog/figs/simagent`
(including showing non-interactive versions of the above – i.e. video-recordings of the RVMs not the running programs themselves):

The demos show that it is possible to generate a running program in which there are various sub-programs driving different entities, e.g. entities inhabiting a 2-D virtual space in which they move, that interact with one another in ways that are continually displayed on a screen, and with which a person can interact by using mouse or keyboard, directing actions selectively at different entities at different times: e.g. in the sheep-dog program, moving one of the sheep, one of the trees, or the dog, with consequent effects on other things in the virtual machine, because they sense the changes in the moved object (and other objects that move autonomously) and the sensed changes produce further reactions.

This sort of demo is not particularly impressive by current standards but depends on hardware and software developments in the last few decades: it would have been very difficult to produce such a collection of interacting pieces of virtual machinery four decades ago.

Some of the relevant technological advances

A small sample of technical developments supporting use of increasingly sophisticated RVMs in computing systems over the last half century:

- The move from bit-level control to control of and by more complex and abstract patterns.
- The move from machine-level instructions to higher level languages (using compilers that ballistically translate to machine code and especially interpreters that “translate” dynamically, informed by context).

  A deep difference between compiled and interpreted programs: the compilation process makes the original program irrelevant, unlike an interpretation process: so altering interpreted program code at run time can have effects that would not occur if the program had been compiled and run.
- Memory management systems make physical memory reference context-dependent.
- Virtual memory (paging and cacheing) and garbage collection switch virtual memory contents between faster and slower core memories and backing store, and between different parts of core memory: constantly changing PM/VM mappings. (These support multiple uses of limited resources.)
- Networked file systems change apparent physical locations of files.
- Device interfaces translate physical signals into “standard” RVM signals and vice versa.
- Devices can themselves run virtual machines with buffers, memories, learning capabilities...
- Device drivers (software) handle mappings between higher level and lower level RVMs – and allow devices to be shared between RVMs (e.g. interfaces to printers, cameras, network devices).
- Context-dependent exception and interrupt handlers distribute causal powers over more functions.
- Non-active processes persist in memory and can have effects on running processes through shared structures. (It’s a myth that single-cpu machines cannot support true parallelism.)
- Multi-cpu systems with relocatable RVMs allow VM/PM mappings to be optimised dynamically.
- Multiplicity of concurrent functions continually grows – especially on networked machines.
- Over time, control functions increasingly use monitoring and control **of RVM states and processes.**

Different requirements for virtual machinery

The different engineering developments supporting new kinds of virtual machinery helped to solve different sorts of problems. E.g.

- Sharing a limited physical device between different users efficiently.
- Optimising allocation of devices of different speeds between sub-tasks.
- Setting interface standards
  so that suppliers can produce competing solutions, and new technology can be used for old functions.
- Allowing re-use of design solutions in new contexts.
- Simplifying large scale design tasks by allowing components to "understand" more complex instructions (telling them what to do, leaving them to work out how to do it).
- Letting abstract functionality be instantiated differently in different contexts (polymorphism).
- Improving reliability of systems using unreliable components. (E.g. error-checking memory.)
- Allowing information transfer/information sharing to be done without users having to translate between formats for different devices (especially unix since mid 1970s).
- Simplifying tasks not only for human designers but also for self-monitoring, self-modulating, self-optimising, self-extending systems and sub-systems.

These are solutions to problems that are inherent in the construction and improvement of complex functioning systems: they are not restricted to artificial systems, or systems built from transistors, or ...

Conjecture: Similar problems were encountered in biological evolution (probably many more problems) and some of the solutions developed were similar, while some were a lot more sophisticated than solutions human engineers have found so far.

Benefits of using running VMs

We don't know exactly what problems evolution faced, what solutions it came up with, and what mechanisms it used, in creating virtual machinery running in animals, to control increasingly complex biological organisms (and societies).

Perhaps we can learn from the problems human engineers faced, and the solutions they found.

- For example, a chess-playing computer can look for moves that achieve a win, or make a win likely.
- In performing that search it need not have the ability to represent the physical details of either the end result of such a search process or the physical details of the process of searching.
- Likewise a machine that aims to observe and understand processees running in itself is likely to do better by using a level of abstraction that has all the details needed for self monitoring, without representing all the physical details of the process.
- It may turn out to be essential for self-monitoring intelligent systems not to make any assumptions in advance about how they work.

Virtual machinery and causation

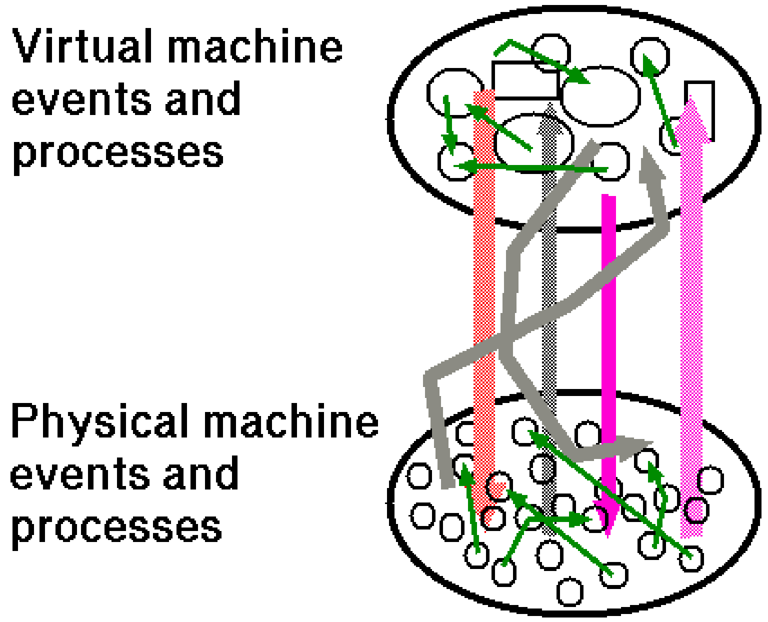
Virtual machinery works because “high level” events in a RVM can control both other virtual machinery and physical machinery.

Accordingly, bugs in the design of virtual machines can lead to disasters, even though nothing is wrong with the hardware.

As stated previously

Processes and events in running virtual machines can be causes and effects, despite being implemented in deterministic physical mechanisms.

Engineers (but not yet philosophers, psychologists, neuroscientists?) now understand how running virtual machinery can co-exist with, and influence, underlying physical machinery, which it helps to control, even though the virtual machinery is all fully implemented in the physical machinery.


Virtual machine events and processes
Physical machine events and processes

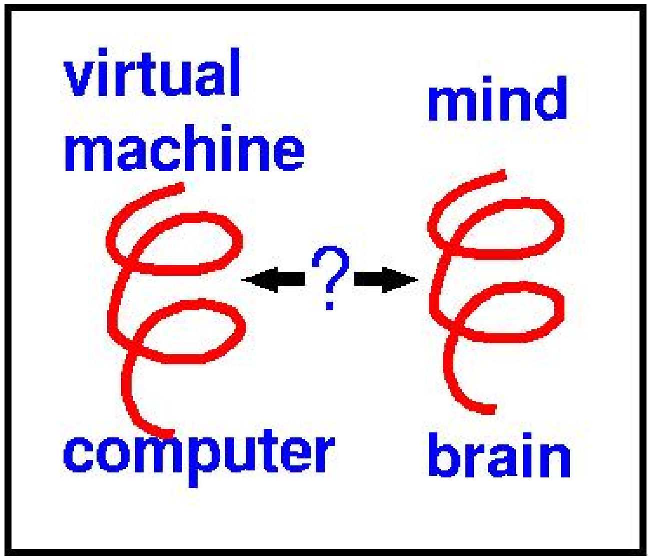
System designers who are concerned with providing the functionality, making it robust, or modifying it to cope with changing circumstances, often do not think only about the hardware involved.

They often think about, and write code to deal with, events like: information arriving, a context changing, a request being received, a plan being executed, a rule being followed with unexpected results, etc.

In doing that they depend on vast amounts of technology some of which they may not even be aware of, e.g. the use of garbage collectors, interrupt handlers, device drivers, etc.

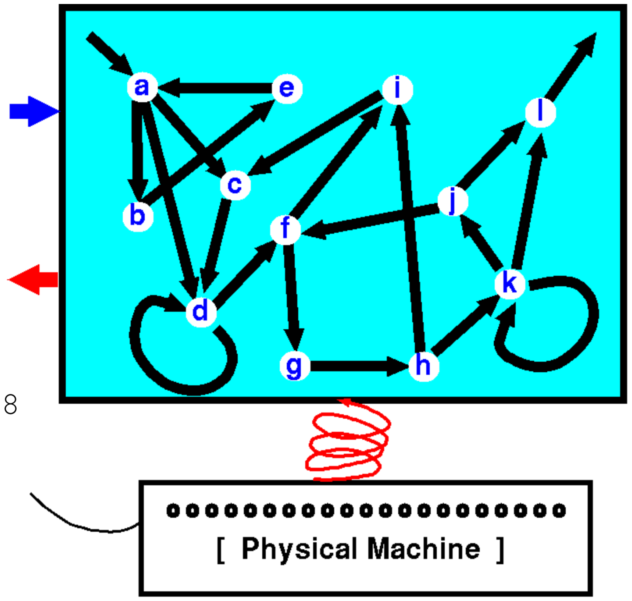
An old idea.

The idea of the analogy expressed in this diagram is very old, but we are only slowly understanding the variety of phenomena on the left hand side, extending our appreciation of what might be going on on the right.



The simple-minded notion that the left hand side involves a program in a computer is seriously deficient, (a) because it ignores the requirement for the program to be running and (b) because we know now that there are far more types of information-processing system than a computer running a single program, as explained in other slides.

Simple-minded views about virtual machines lead to easily refutable computational theories of mind, e.g. the theory that virtual machines are simple finite state machines, as illustrated on the right ("Atomic state functionalism"). See



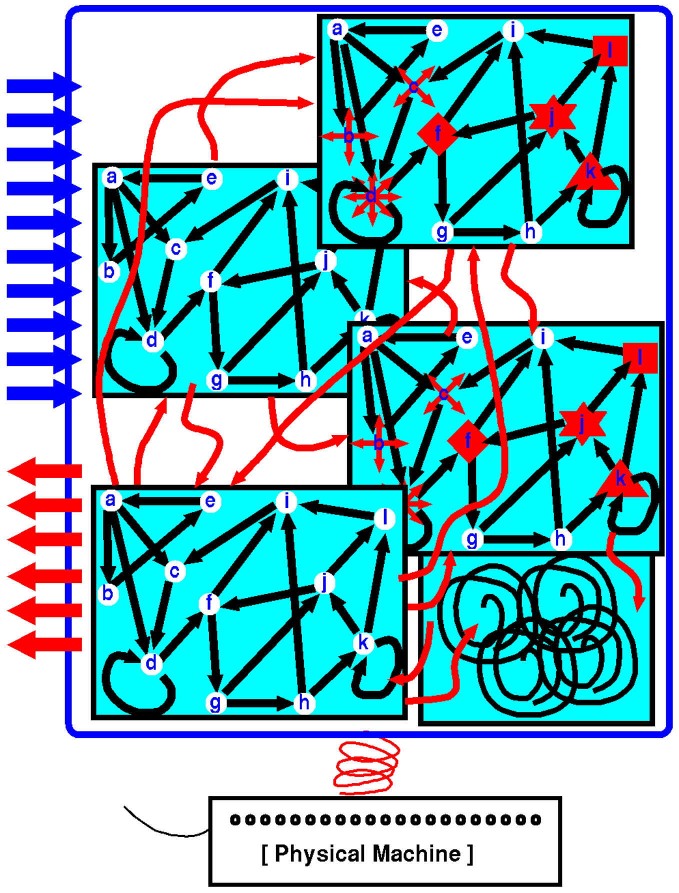
Ned Block: Functionalism `http://cogprints.org/235/`

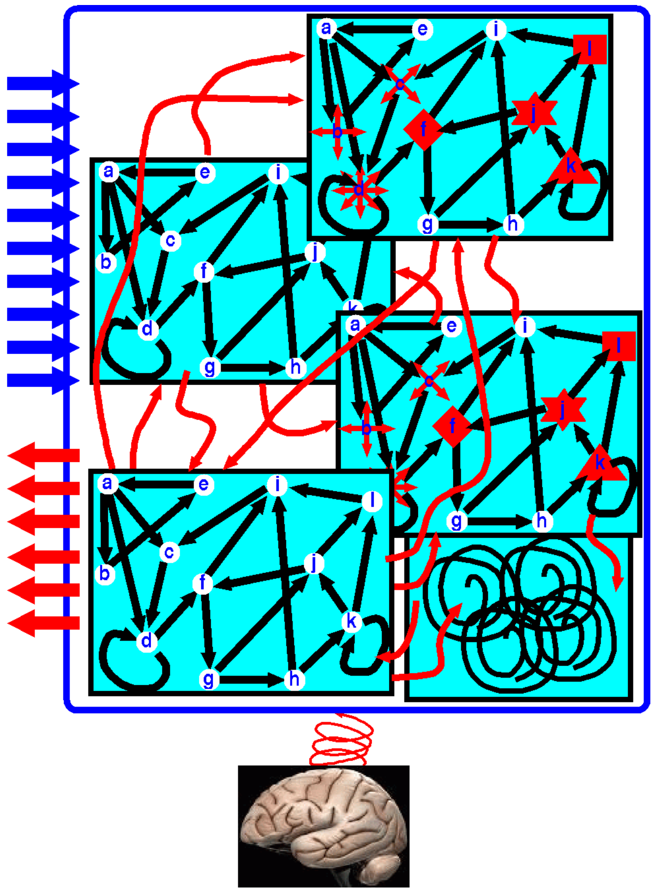
A. Sloman, The mind as a control system, 1993,
`http://www.cs.bham.ac.uk/research/projects/cogaff/81-95.html#18`

Forget what you have learnt about Turing machines: that's a simple abstraction which is surprisingly useful for theorising about classes of computations – but not so useful for modelling complex multi-component systems interacting asynchronously with a rich and complex environment.

More realistic models

As crudely indicated here we need to allow multiple concurrent inputs (blue) and outputs (red), multiple interacting subsystems, some discrete some continuous, with the ability to spawn new subsystems/processes as needed.





Artificial VM on artificial PM **Biological VM on biological PM**

In both cases there are multiple feedback loops involving the environment.

Two major kinds of running VM

There are various types of VM with different capabilities and different roles in larger systems.

Two important classes are:

Specific function/application VMs

E.g. a chess-playing VM, a word-processor, an email front-end.

Platform VMs

Capable of supporting a variety of other VMs

E.g. Operating systems (e.g. Linux, Unix, OSX, Windows, VMS, ...)

Language VMs (e.g. Lisp VM. Prolog VM, Java VM, Pop-11 VM)

NB: we are talking about running instances, not abstract specifications.

It seems that there are some biological platform VMs, which get extended in various ways.

An important research problem is to investigate the various types, their functions, which species use them, how they evolved, how they develop in individuals, etc.

One type of extension of biological VM capability involves learning new languages and notations.

E.g. learning a new programming language.

Natural and Artificial Platform VMs

- Platform VMs designed by human engineers provide a basis for constructing new VMs that are implemented in terms of the facilities provided by the platform VM:
  But most such extensions do not arise spontaneously.

  The operating system on your PC can be left running for many years and over time you and others may design and install different software packages, or attach new hardware devices along with device drivers for them.

  If left to itself, a normal operating system will just go on forever waiting for instructions, without initiating any major extensions, though some of them are designed to detect the availability of new versions of old subsystems and download and install them.

- Biological platform VMs, however, are not extended by external designers:
  They have to build and extend themselves

  (partly on the basis of external influences from both the physical environment and conspecifics).

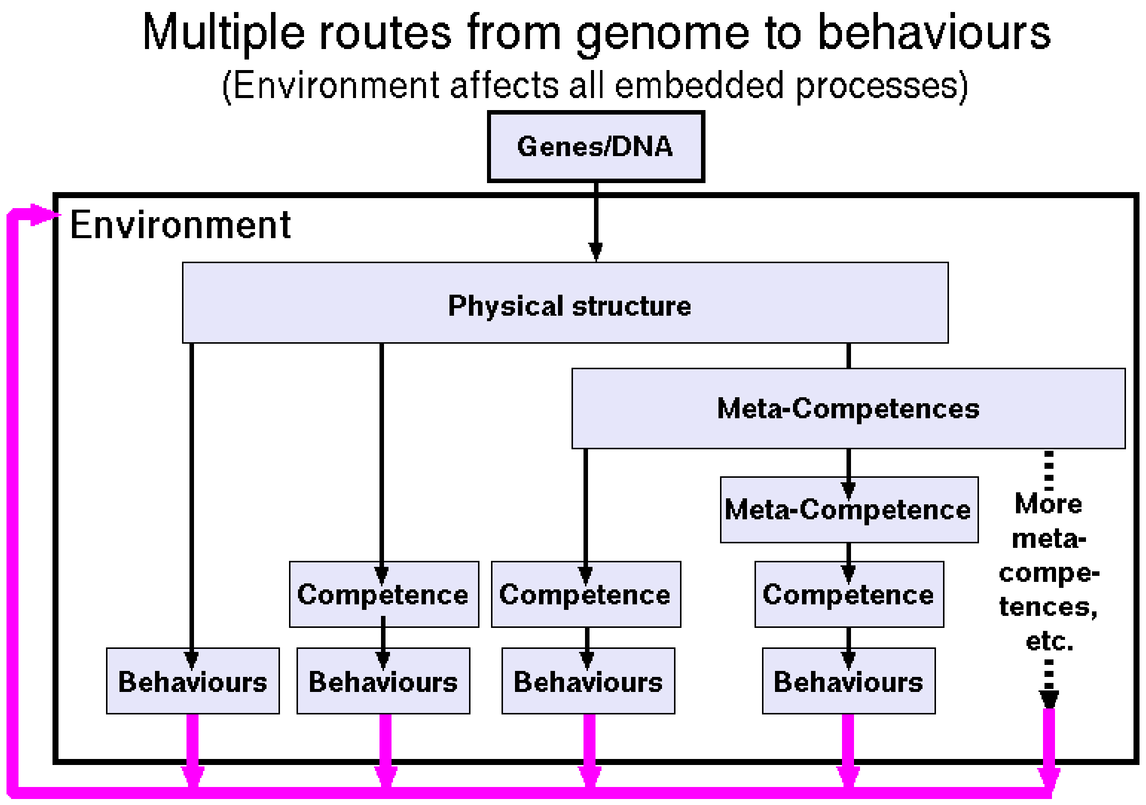
  The requirements to support this have never, as far as I know, been identified.

  The problem is not addressed by research in developmental psychology on which concepts, knowledge or competences are innate.

  Some of the requirements for a “well-designed child” were discussed by McCarthy in this paper `http://www-formal.stanford.edu/jmc/child.html` (written 1996 and published in 2008).

- In humans, biological platform VMs seem to grow throughout infancy and childhood, and for some people (e.g. academics) continue being extended until late in life.
  The extensions support new competences of many kinds, including manipulative and perceptual competences, linguistic, musical and artistic competences, mathematical competences, extended ontologies, new planning and reasoning capabilities, new forms of motivation, new control regimes,

Biological VMs have to grow themselves



Humans and some other species require layered construction of competences and meta-competences in a layered architecture.

Not core competences as normally construed: core architecture-building competences, metacompetences, ...

Work done with Jackie Chappell (IJUC, 2007) – Chris Miall helped with diagram.
“Natural and artificial meta-configured altricial information-processing systems”
`http://www.cs.bham.ac.uk/research/projects/cosy/papers/#tr0609`,

Warning from biology:

Don’t expect a **sharp** divide between systems using only physical machines and those also using virtual machines: biology provides intermediate cases for most distinctions,

e.g. is a homeostatic control loop a VM?

Neither biology nor engineering needs to respect philosophers’ desires for simple classification schemes:

there tend to be many small discontinuities rather than just a few big ones.

But differences across multiple steps can be huge.

RVMs with temporarily or partly ‘decoupled’ components

A challenge for philosophy of science and psychological methodology.

- “Decoupled” subsystems may exist and process information, even though they have no connection with sensors or motors.
- Theories referring to them cannot be decisively proved or refuted.

  Compare Lakatos on methodology of scientific research programmes
- For instance, a machine playing games of chess with itself, or investigating mathematical theorems, e.g. in number theory.
- Some complex systems “express” some of what is going on in their VM states and processes through externally visible behaviours.

  However, it is also possible for internal VM processes to have a richness that cannot be expressed externally using the available bandwidth for effectors.

  Likewise sensor data may merely introduce minor perturbations in what is a rich and complex ongoing internal process.

This transforms the requirements for rational discussion of some old philosophical problems about the relationship between mind and body:

E.g. some mental processes need have no behavioural manifestations, though they might, in principle, be detected using ‘decompiling’ techniques with non-invasive internal physical monitoring.
(This may be impossible in practice, or at best only a matter of partly testable conjecture.
Compare theoretical physics.)

How does all that work?

- We have learnt how to set up physical mechanisms that enforce constraints between abstract process patterns (unlike mechanisms that merely enforce constraints between physical or geometric relations).
- Chains of such constraints can have complex indirect effects linking different process-patterns.
- Some interactions involve not only causation but also meaning: patterns are **interpreted** by processes in the machine as including descriptive information (e.g. testable conditions) and control information (e.g. specifying what to do).

  See "What enables a machine to understand?" (IJCAI 1985)
  `http://www.cs.bham.ac.uk/research/projects/cogaff/81-95.html#4`

Could biological evolution have solved similar problems?

BIOLOGICAL CONJECTURE: Over time, control functions increasingly used monitoring and control of VM states and processes.

CONJECTURE:

- The pressures for such developments that drive human engineers towards more and more "virtual" machinery of different sorts, were just as strong, in biological evolution
- as biological machines and their control functions became more and more complex
- with increasingly complex decisions taken "at run time"
- about how to process sensory information, what ontologies to use, what information to store, how to use the information, how to generate hypothese, goals, plans, etc.
- how to use them
- how to detect bugs in them, and debug them, ...

Controlling astronomically large numbers of interacting tiny physical subsystems directly is far too difficult.

But the solution involving RMVs depends crucially on finding the right, re-usable, levels of abstraction, and modules.

Whether there are any, and what they are depends on the type of environment.

Physical $\Longleftrightarrow$ virtual interfaces at different levels

Starting with simple physical devices implementing interacting discrete patterns, we have built layers of interacting patterns of ever increasing spatial and temporal complexity, with more and more varied functionality.

- Physical devices can constrain continuously varying states so as to allow only a small number of discrete stable states (e.g. only two)

  (e.g. using mechanical ratchets, electronic valves (tubes), aligned magnetic molecules, transistors etc.)
- Networks of such devices can constrain relationships between discrete patterns.

  E.g. the ABCD/XY example: a constraint can ensure that if devices A and B are in states X and Y respectively then devices C and D will be in states Y and X (with or without other constraints).
  So, a device network can rule out some physically possible combinations of states of components, and a new pattern in part of the network will cause pattern-changes elsewhere via the constraints.
  Compare: one end of a rigid lever moving down or up causes the other end to be moving up or down.
- Such networks can form dynamical systems with limited possible trajectories, constraining both the possible patterns and the possible sequences of patterns.
- A network of internal devices can link external interfaces (input and output devices)
  thereby limiting the relationships that can exist between patterns of inputs and patterns of outputs, and also limiting possible sequences of input-output patterns.
- Patterns in one part of the system can have meaning for another part, e.g.
  - constraining behaviour (e.g. where the pattern expresses a program or ruleset) or
  - describing something (e.g. where the pattern represents a testable condition)
- Such patterns and uses of such patterns in interacting computing systems may result from design (e.g. programming) or from self-organising (learning, evolving) systems.
- Some useful patterns need not be describable in the language of physics.

The “gap” closing

The last half century’s advances in development of **running** virtual machines at last provide a basis for closing what has been called “the explanatory gap”.

(But only if the right notion of virtual machine is used: a concept many people understand only dimly.)

- This gap was identified as a real problem for Darwin’s theory of evolution, even among people who were convinced by evidence for evolution of physical forms

  (e.g. T.H. Huxley, though the physicist Tyndall had earlier discussed it at as a mystery)
- The problem for Darwinians was that there was plenty of evidence for gradual evolution of physical forms between various species, including humans, but the transition from non-human to human minds seemed to involve such a big discontinuity that there was no comparable evidence.
- Further, some people thought it was inconceivable that an explanation of how physical bodies could produce minds would ever be forthcoming.
- Only now are we on the threshold of understanding what evolution might have had to do.

What follows from all this?

- There are many empirical facts about human experience (most of them easily checked) that support claims about the existence of introspectively accessible entities, often described as privately accessible contents of consciousness.

  Various labels are used for these entities: “phenomenal consciousness”, “qualia” (singular “quale”), “sense-data”, “sensibilia”, “what it is like to be/feel X” (and others).
  For a useful, but partial, overview see `http://en.wikipedia.org/wiki/Qualia`

- What is not clear is what exactly follows from the empirical facts, and how best they can be described and explained.

- In the following slides I'll demonstrate some of the empirical facts about contents of consciousness that raise a scientific problem of explaining how such entities arise and how they are related to non-mental mechanisms, e.g. brains (and future machines).

- Philosophers and scientists, understandably (in the past) ignorant of what we now know about virtual machinery have referred to “the explanatory gap” later redescribed by Chalmers as “the hard problem” of consciousness: a problem for Darwin

- Unfortunately, some philosophers, trying to characterise what needs to be explained have ended up discussing something incoherent.

  E.g. qualia, or phenomenal consciousness **defined** as incapable of causal/functional relations.
  For more on this see my online presentations:
  `http://www.cs.bham.ac.uk/research/projects/cogaff/talks/`

Reactive vs deliberative interacting patterns

A Conway machine uses real or simulated concurrency: behaviour of each square depends only on the previous states of its eight neighbours and nothing else.

On a computer the concurrency is achieved by time-sharing, but it is still real concurrency.

Consider what happens when two virtual machines running on a computer compete in a chess game, sharing a virtual board, and interacting through moves on the board, each can sense or alter the state of any part of the (simulated) chess board.

- In general, programs on a computer are not restricted to local interactions.
- In some cases, the interacting processes are purely reactive: on every cycle every square immediately reacts to the previous pattern formed by its neighbours.
- If two instances of a chess program (or instances of different chess programs) interact by playing chess in the same computer, their behaviour is typically no longer purely reactive. Good ones will often have to search among possible sequences of future moves to find a good next move – and only then actually move.

  In addition, one or both of the chess virtual machines may do some searching in advance while waiting for the opponent's next move.
- Then each instance is a VM with its own internal states and processes interacting richly, and a less rich interaction with the other VM is mediated by changes in the shared board state (represented by an abstract data-structure).

  For more on varieties of deliberation see:
  `http://www.cs.bham.ac.uk/research/projects/cosy/papers/#dp0604`

Intentionality in a virtual machine

A running chess program (a VM) takes in information about the state of the board after the opponent moves, and builds or modifies internal structures that it uses to represent the board and the chess pieces on it, and their relationships, including threats, opportunities, possible traps, etc.

- In particular it uses those representations in attempting to achieve its goals.

  So, unlike the interacting Conway patterns mentioned earlier, some of the patterns in the chess virtual machine are treated by the machine as representations, that refer to something.

- During deliberation, some created patterns will be treated as referring to non-existent but possible future board states, and as options for moves in those states.

  They are treated that way insofar as they are used in considering and evaluating possible future move sequences in order to choose a move which will either avoid defeat (if there is a threat) or which has a chance of leading to victory (check-mate against the opponent).

- In this case the chess VM, unlike the simplest interacting Conway patterns, exhibits intentionality: the ability to refer. (NB. The programmer need not know about the details.)

  Since the Conway mechanism is capable of implementing arbitrary Turing machines, it could in principle implement two interacting chess virtual machines, so there could be intentionality in virtual machines running on a Conway machine – probably requiring a very big fairly slow machine.

- The intentionality of chess VMs is relatively simple because they have relatively few types of goal, relatively few preferences, and their options for perceiving and acting are limited by being constrained to play chess:

  For a human-like, or chimp-like, robot the possibilities would be much richer, and a far more complex architecture would be required. See
  `http://www.cs.bham.ac.uk/research/projects/cogaff/03.html#200307`

Adding meta-semantic competences

If a virtual machine playing chess not only thinks about possible board states and possible moves, and winning, moving, threats, traps, etc. but also thinks about what the opponent might be thinking, then that requires meta-semantic competences: the ability to represent things that themselves represent and use information.

- It is very likely that biological evolution produced meta-semantic competences in some organisms other than humans because treating other organisms (prey, predators, conspecifics to collaborate with, and offspring as they learn) as mere physical systems, ignoring their information-processing capabilities, will not work well (e.g. hunting intelligent prey, or avoiding intelligent predators).
- Another application for meta-semantic competences is self-monitoring, self evaluation, self-criticism, self-debugging: you can't detect and remedy flaws in your thinking, reasoning, planning, hypotheses etc. if you are not able to represent yourself as an information user.
- It is often assumed that social meta-semantic competences must have evolved first, but that's just an assumption: it is arguable that self-monitoring meta-semantic competences must have evolved first

  e.g. because an individual has relatively direct access to (some of) its own information-processing whereas the specifics of processing in others has to be inferred in a very indirect way (even if evolution produced the tendency to use information about others using information).
  See A. Sloman, 1979, The primacy of non-communicative language,
  `http://www.cs.bham.ac.uk/research/projects/cogaff/81-95.html#43`

Emerging varieties of functionality

Computer scientists and engineers and AI/Robotics researchers have been learning to add more and more kinds of control, kinds of pattern, and ways of interpreting patterns of varying levels of abstraction.

- A simple machine may repeatedly take in some pattern and output a derived pattern,
  e.g. computing the solution to an arithmetical problem.
- More complex machines can take in a pattern and a derivation-specification (program) and output a derived pattern that depends on both.
- Other machines can continually receive inputs (e.g. from digitised sensors) and continually generate outputs (e.g. to digitally controlled motors).
- More sophisticated machines can
  - solve new problems by searching for new ways of relating inputs to outputs, i.e. learning;
  - interpret some patterns as referring to the contents of the machine (using a somatic ontology) and others to independently existing external entities, events, processes (using an exosomatic ontology)
  - extend their ontologies and theories about the nature and interactions of external entities
  - perform tasks in parallel, coordinating them,
  - monitor and control some of their own operations – even interrupting, modulating, aborting, etc.
    (Including introspecting some of their sensory and other information contents: qualia.)
  - develop meta-semantic ontologies for representing and reasoning about thinking, planning, learning, communicating, motives, preferences, ...
  - acquire their own goals and preferences, extending self-modulation, autonomy, unpredictability, ...
  - develop new architectures which combine multiple concurrently active subsystems.
  - form societies, coalitions, partnerships ... etc.
- Biological evolution did all this and more, long before we started learning how to do it.

Causal networks linking layered patterns

How can events in virtual machines be causes as well as effects, even causing physical changes?

The answer is

through use of mechanisms that allow distinct patterns of states and sequences of patterns to be linked via strong constraints to other patterns of states and sequences of patterns (as in the ABCD/XY example, and the Conway machines, mentioned above). (Some VMs may use probabilistic/stochastic constraints.)

What many people find hard to believe is that this can work for a virtual machine whose internal architecture allows for divisions of functionality corresponding to a host of functional divisions familiar in human minds, including

- interpreting physical structures or abstract patterns as referring to something (intentionality)
- generation of motives,
- selection of motives,
- adoption of plans or actions,
- perceiving things in the environment,
- introspecting perceptual structures and their changes,
- extending ontologies,
- forming generalisations,
- developing explanatory theories,
- making inferences,
- formulating questions,
- and many more.

Biological unknowns: Research needed

Many people now take it for granted that organisms are information-processing systems, but much is still not known, e.g. about the varieties of low level machinery available (at molecular and neuronal mechanisms) and the patterns of organisation for purposes of acquiring and using information and controlling internal functions and external behaviours.

Steve Burbeck's web site raises many of the issues:

"All living organisms, from single cells in pond water to humans, survive by constantly processing information about threats and opportunities in the world around them. For example, single-cell E-coli bacteria have a sophisticated chemical sensor patch on one end that processes several different aspects of its environment and biases its movement toward attractant and away from repellent chemicals. At a cellular level, the information processing machinery of life is a complex network of thousands of genes and gene-expression control pathways that dynamically adapt the cell's function to its environment."

`http://evolutionofcomputing.org/Multicellular/BiologicalInformationProcessing.html`

"Nature offers many familiar examples of emergence, and the Internet is creating more.

The following examples of emergent systems in nature illustrate the kinds of feedback between individual elements of natural systems that give rise to surprising ordered behavior. They also illustrate the trade off between the number of elements involved in the emergent system and the complexity of their individual interactions. The more complex the interactions between elements, the fewer elements are needed for a higher-level phenomenon to emerge. ... networks of computers support many sorts of emergent meta-level behavior because computers interact in far more complex ways than air and water molecules or particles of sand ... Some of this emergent behavior is desirable and/or intentional, and some (bugs, computer viruses, dangerous botnets, and cyber-warfare) are not."

`http://evolutionofcomputing.org/Multicellular/Emergence.html`

Closing Huxley’s Explanatory Gap

If we can learn more about:

- varieties of virtual machinery and their roles in generating, controlling, modulating and extending behaviours in organisms;
- how they are implemented in various types of biological organism;
- how their features can be specified in a genome (e.g. the control mechanisms for mating, web-making, and eating in a spider seem, for many species to be genetically determined, although specific behaviours are adapted to the precise details of environment);
- how in some species the virtual machinery instead of being fully specified genetically is built up within an individual as a result of operating of genetic, environmental and cultural processes (see Chappell and Sloman, IJUC, 2007, mentioned above);
- how and why self-monitoring mechanisms came to include mechanisms able to focus on intermediate information-structures within sensory/perceptual sub-systems

  (e.g. how things look, how they feel, how they sound, etc.)

then we may be able to understand how a Darwinian evolutionary process that is already demonstrably able to explain much of the evolution of physical form might be extended to explain evolution of information processing capabilities, including the phenomena that lead to philosophical theories of consciousness.

But we should not expect there to be **one** thing, one **it** that evolved.

Darwin and his contemporaries knew nothing about virtual machines, alas.

Biological conjecture

I conjecture that biological evolution discovered those design problems long before we did and produced solutions using virtual machinery long before we did – in order to enable organisms to deal with rapidly changing and complex information structures (e.g. in visual perception, decision making, control of actions, self-monitoring etc.).

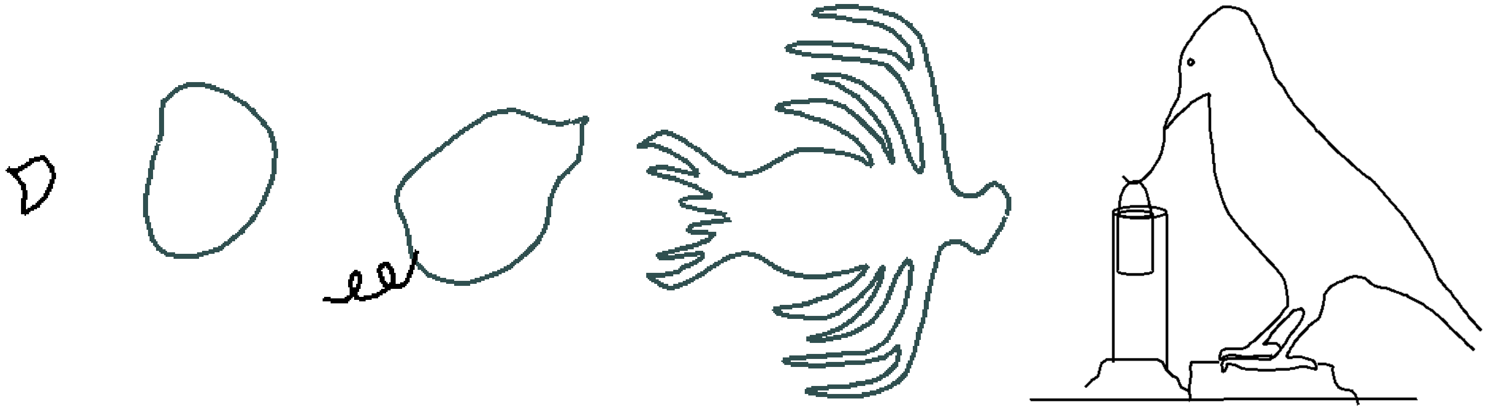
- You can't rapidly rewire millions of neurons when you look in a new direction
- or when you switch from approaching prey to deciding in which direction to try to escape from a new predator, using visible details of the terrain.
- So there's no alternative to using virtual machinery.
- But we know very little about biological virtual machinery.

  Nobody knows how brain mechanisms provide virtual machinery that supports proving geometric theorems, thinking about infinite sets of numbers, or algebra, or wanting to rule the world.
- The visual competences demonstrated here remain unexplained
  `http://www.cs.bham.ac.uk/research/projects/cogaff/misc/multipic-challenge.pdf`

We know that humans can use very different languages to say and do similar things (e.g. teach physics, discuss the weather); but evolution could not have produced special unique brain mechanisms for each language (since most are too new) – it's more likely that language learning creates specialised VMs running on more general physiological mechanisms.

Some conjectures about evolution of language are here:
`http://www.cs.bham.ac.uk/research/projects/cogaff/talks/#glang`

All organisms are information-processors but the information to be processed has changed and so have the means



Types of environment with different information-processing requirements

- Chemical soup
- Soup with detectable gradients
- Soup plus some stable structures (places with good stuff, bad stuff, obstacles, supports, shelters)
- Things that have to be manipulated to be eaten (e.g. disassembled)
- Controllable manipulators
- Things that try to eat you
- Food that tries to escape
- Mates with preferences
- Competitors for food and mates
- Collaborators that need, or can supply, information.

The role of the environment

Ulric Neisser:

“We may have been lavishing too much effort on hypothetical models of the mind and **not enough on analyzing the environment** that the mind has been shaped to meet.”

Neisser, U. (1976) *Cognition and Reality*, San Francisco: W. H. Freeman.

Compare: **John McCarthy**: “The well-designed child”

“Evolution solved a different problem than that of starting a baby with no a priori assumptions.

....

“Instead of building babies as Cartesian philosophers taking nothing but their sensations for granted, evolution produced babies with innate prejudices that correspond to facts about the world and babies’ positions in it. Learning starts from these prejudices. What is the world like, and what are these instinctive prejudices?”

`http://www-formal.stanford.edu/jmc/child.html` Also in AI Journal, December 2008

All biological organisms are solutions to design problems that cannot be specified without specifying in detail the relevant features of the environment.

Turing, surprisingly got this wrong: he thought human-like learning was possible from a “clean slate”.

How to look at the environments of organisms

All biological organisms are solutions to design problems that cannot be specified without specifying in detail the relevant features of the environment. (This does not imply that there is a designer.)

In order to understand which features of the environment are capable of influencing designs (or more precisely producing "pressures" to alter designs) we have to understand what the problems are that a team of engineers would have to solve – including hardware and software engineers.

That means understanding things like

- what information the organism needs in different parts of the environment while in different states (hungry, thirsty, escaping, competing, playing, exploring, etc.)
- what forms of representation of that information can be useful for the purposes of influencing internal and external processes including physical behaviours and information-processing (at the time or in future). in future).
- what information processing mechanisms can make use of the information
- what sort of architecture can combine a variety of forms of information processing possibly running concurrently.

Turing, surprisingly got this wrong: he thought human-like learning was possible from a "clean slate".

J.J. Gibson understood the general point, but missed many important details.
Other relevant authors: Piaget, Fodor, Chomsky, Mandler, Keil, Gopnik, Tenenbaum, Thomasello, Karmiloff-Smith, Spelke, ... (Not all seem to understand how to think about requirements and designs.)

To gain deep understanding, don’t study just one species

Trying to do AI as science by developing only one type of system
(e.g. “human level AI”)

is like trying to do physics by investigating only how things behave near the leaning tower of Pisa.

Or studying only the motion of planets.

we need to understand spaces of possibilities
and the tradeoffs between alternative designs: so look at different species.

Don’t assume all effective information-processing mechanisms have to be rational
(as in Dennett’s “intentional stance”, Newell’s “knowledge level”.)

Engineers need to build reflexes into complex machinery to cope with the unexpected.

Likewise, evolution provided useful reflexes: reflexes are neither rational nor irrational.

Some useful reflexes are cognitive. Some even meta-cognitive.

Including reflexes that generate goals/motives warnings, things to remember, etc.

Not all goals are chosen because the individual knows associated rewards/costs/benefits:

See
`http://www.cs.bham.ac.uk/research/projects/cogaff/misc/architecture-based-motivation.html`
Architecture-based motivation vs reward-based motivation.

Work to be done:
Biology, psychology, neuroscience, robotics

There is much work still to be done.

That includes finding out precisely what the problems were that evolution solved and how they are solved in organisms, and why future intelligent robots will need similar solutions.

There are more slide presentations on related topics here:

`http://www.cs.bham.ac.uk/research/projects/cogaff/talks/`

Many of the papers in the Birmingham CogAff project (Cognition and Affect) are relevant, especially papers on architectures.

`http://www.cs.bham.ac.uk/research/projects/cogaff/`

But the problem of explaining how a genome can specify types of virtual machinery to be developed in individuals, including types that are partly determined by the environment at various stages of development is very difficult.

We need to understand much more about the evolution and development of virtual machinery.

See Jackie Chappell and Aaron Sloman, “Natural and artificial meta-configured altricial information-processing systems”
`http://www.cs.bham.ac.uk/research/projects/cosy/papers/#tr0609` (IJUC, 2007)

Importance and implications of VMs

There are additional slides available on Virtual Machines, e.g.

`http://www.cs.bham.ac.uk/research/projects/cogaff/talks/#mos09`
Virtual Machines and the Metaphysics of Science Expanded version of presentation at: Metaphysics of Science'09)

Topics include:

- Explanation of the importance of virtual machines in sophisticated control systems with self-monitoring and self-modulating capabilities.
- Why such machines need something like "access consciousness"/qualia – and why they too generate an explanatory gap – a gap bridged by a lot of sophisticated hardware and software engineering developed over a long time.
- In such machines, the explanations that we already have are much deeper than mere correlations: we know **how** the physical and virtual machinery are related, and what difference would be made by different designs.

More to read

We need much better understanding of nature-nurture issues, and requirements for educational systems.

John McCarthy on “The well-designed child”.
`http://www-formal.stanford.edu/jmc/child.html`

Chappell and Sloman on “Natural and artificial meta-configured altricial information-processing systems”
`http://www.cs.bham.ac.uk/research/projects/cosy/papers/#tr0609`

Expand on varieties of metacognition, and differences between introspection and other aspects of metacognition.
`http://www.cs.bham.ac.uk/research/projects/cosy/papers/#tr0803`

See other presentations in

`http://www.cs.bham.ac.uk/research/projects/cogaff/talks/`

and CogAff and CoSy papers:

`http://www.cs.bham.ac.uk/research/projects/cogaff/`

`http://www.cs.bham.ac.uk/research/projects/cosy/papers/`

Further Reading

Novelists have traditionally regarded themselves as the experts on consciousness, with some justification. See for example, David Lodge’s essays and his novel on consciousness:
David Lodge, *Consciousness and the Novel: Connected Essays,* Secker & Warburg, London, 2002.
David Lodge, *Thinks ....*, Penguin Books, 2002.

A huge and important topic: disorders of consciousness, self-consciousness and control.

We need to explain development of kind of self-awareness that enables people to tell the difference between what they treat as empirical generalisations and what they understand as (mathematically) provable – e.g. facts about topological relations, geometry, mechanics, and numbers. (The roots of mathematical thinking.)