Seminar, University of Birmingham 16 Oct 2008
The Great debate Newcastle 21 Oct 2008
Mind as Machine Weekend Course, Oxford 1-2 Nov 2008
Workshop on Philosophy and Engineering, RAE, London 10-12 Nov 2008

(This is work in progress. Comments and criticisms welcome.)

Virtual Machines in Philosophy, Engineering & Biology

Why virtual machines really matter –

for several disciplines

Aaron Sloman

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

School of Computer Science

The University of Birmingham

The latest version of these slides will be available online at
http://www.cs.bham.ac.uk/research/cogaff/talks/#wpe08

These slides on virtual machines and implementation are closely related:
http://www.cs.bham.ac.uk/research/cogaff/talks/#virt

Last revised January 9, 2009

Presentations based on versions of these slides

Long presentations

Based on: http://www.cs.bham.ac.uk/research/projects/cogaff/talks/#virt

1. Thur 16 Oct 2008: School of Computer Science Seminar, Birmingham

   Why virtual machines really matter – for several disciplines
   http://www.cs.bham.ac.uk/events/seminars/seminar_details.html?seminar_id=560

2. Tues 21 Oct 2008: The Great Debate, Newcastle

   What can biologists, roboticists and philosophers learn from one another? (Unnoticed connections)
   http://thegreatdebate.org.uk/UnnoticedConnections.html

3. Sat-Sun 1-2 Nov 2008: Weekend course Mind as Machine, Oxford

   Why philosophers need to be robot designers
   http://www.conted.ox.ac.uk/courses/details.php?id=O08P107PHR
   http://oxfordphilsoc.org/

Short presentation (this one!)

10-12 November 2008: Workshop on Philosophy and Engineering
Royal Academy of Engineering, London

Extended abstract: Virtual Machines in Philosophy, Engineering & Biology
http://www.cs.bham.ac.uk/research/projects/cogaff/08.html#803
Slides here http://www.cs.bham.ac.uk/research/projects/cogaff/talks#wpe08

The previous talks used the slides in this (PDF) presentation
http://www.cs.bham.ac.uk/research/projects/cogaff/talks/#virt
This presentation uses only a small subset of these slides.
(The presentation at the meeting used an even smaller subset.)

ACKNOWLEDGEMENTS

Thanks especially to

Matthias Scheutz, Ron Chrisley and Jackie Chappell

And users of our SimAgent toolkit, from whom I have learnt much.

Thanks also to questioners at previous presentations of these ideas

Many thanks to Linux/Unix developers:

I constantly use excellent virtual machines
that they have designed.

I am interacting with one now
and also several others running on it

Apologies for clutter: read only what I point at.
The slides are meant to be readable without me being available to present them.

Main Thesis

Philosophers, neuroscientists, psychologists, social scientists, among others, have identified puzzles regarding the existence and causal efficacy of non-physical states and processes

e.g. mental states and processes, socio-economic processes

But they have no good solutions.

- Philosophers have offered metaphysical, epistemological and conceptual theories about the status of such entities,

  for example in talking about dualism, "supervenience", mind-brain identity, epiphenomenalism, or simply denying that mental or other non-physical events can be causes.

- To the best of my knowledge the vast majority of such philosophers and scientists either know nothing about non-physically describable machines (NPDMs) running in computers, or ignore what they know.

  (There are exceptions, e.g. John Pollock, Ron Chrisley, and Peter Simons.)

- Some philosophers, e.g. Dan Dennett, mention virtual machines, but misdescribe them, e.g. as useful fictions, or wrongly compare them with abstractions such as centre of mass of a complex object.

Two claims

I claim

(a) most philosophers fail to notice major new insights readily available from the technology they use every day to write their papers, read or send email, browse the internet, etc.

(b) software engineers and computer scientists know enough about the technology to design, implement, analyse, debug, extend, and use such systems, but they lack the philosophical expertise to articulate what they know, and they have not fully appreciated the magnitude, the importance and the complexity of this 20th century development.

(c) There are also deep implications for biology, psychology, neuroscience, among others.

The key idea is “running machine”, of which there are physical examples and non-physical examples.

What is a machine?

A machine is a complex enduring entity with parts

(possibly a changing set of parts)

that **interact causally** with one another as they change their properties and relationships.

Most machines are also embedded in a complex environment with which they interact.

The internal and external interactions may be discrete or continuous, sequential or concurrent.

Different parts of the machine, e.g. different sensors and effectors, may interact with different parts of the environment concurrently.

The machine may treat parts of itself as parts of the environment (during self-monitoring), and parts of the environment as parts of itself (e.g. tools, external memory aids). (See Sloman 1978, chapter 6)

The machine may be fully describable using concepts of the physical sciences (plus mathematics), in which case it is a physical machine (PM).

Examples include levers, assemblages of gears, clocks, clouds, tornadoes, plate tectonic systems, and myriad molecular machines in living organisms.

Not all machines are physical machines

Some machines have states, processes and interactions

whose descriptions use concepts that cannot be defined in terms of those of the physical sciences

Examples of such concepts:

"checking spelling", "playing chess", "winning", "threat", "defence", "strategy", "desire". "belief", "plan", "poverty", "crime", "economic recession", ...

For now I'll take that indefinability as obvious: it would take too long to explain.

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

Those include information-processing machines.

Including socio-economic machines, ecosystems and many biological control systems.

Non-Physically-Describable Machines

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

They are frequently referred to in everyday life, including the press, political debates, etc.

They are also commonplace in computing systems, a fact that is widely ignored by philosophers trying to understand relations between physical and non-physical states and processes.

Too many philosophers are taught that if a problem is philosophical it can be addressed without knowing anything about discoveries or creations in science and technology, since philosophy (like mathematics) is defined by them to be a non-empirical discipline.

They either forget, or have never heard about, the enormously rich fertilization of mathematics by the empirical sciences and and by problems in technology and engineering, which is, presumably why, as they type their papers, or read and write email messages, or interrogate search engines, they do not ask:

> "How can all this text-processing, formatting, spelling correction, emailing, web-browsing work,
> and how does it relate to what is going on in
> the physical machine on my desk or lap or palm, or whatever?"

Even Immanuel Kant recognized that some "synthetic apriori knowledge" (significant non-empirical knowledge) had to be "awakened" by sensory experience.

Computing NPDMs are called “virtual machines”

Among computer scientists and engineers, NPDMs are normally referred to as “Virtual Machines” (VMs) a fact that can cause much confusion, since the word “virtual” often suggests something unreal, as in “virtual reality” systems.

A NPDM (or VM), like a PM, can have parts that interact with one another and with the environment.

This is commonplace in computing systems –

e.g. spread-sheets, word-processors, internet-browsers.

GIVE A DEMO.

Terminology:
Because the phrase “virtual machine” is so widespread, at least in computing circles, where a lot is known about them, I shall continue to talk about virtual machines (VMs), avoiding the mouthful: NPDM.

NB 1: “Virtual” in this context does not imply: “non-existent”, “unreal”, etc.

NB 2: Every VM must be implemented in a PM without which it cannot exist.

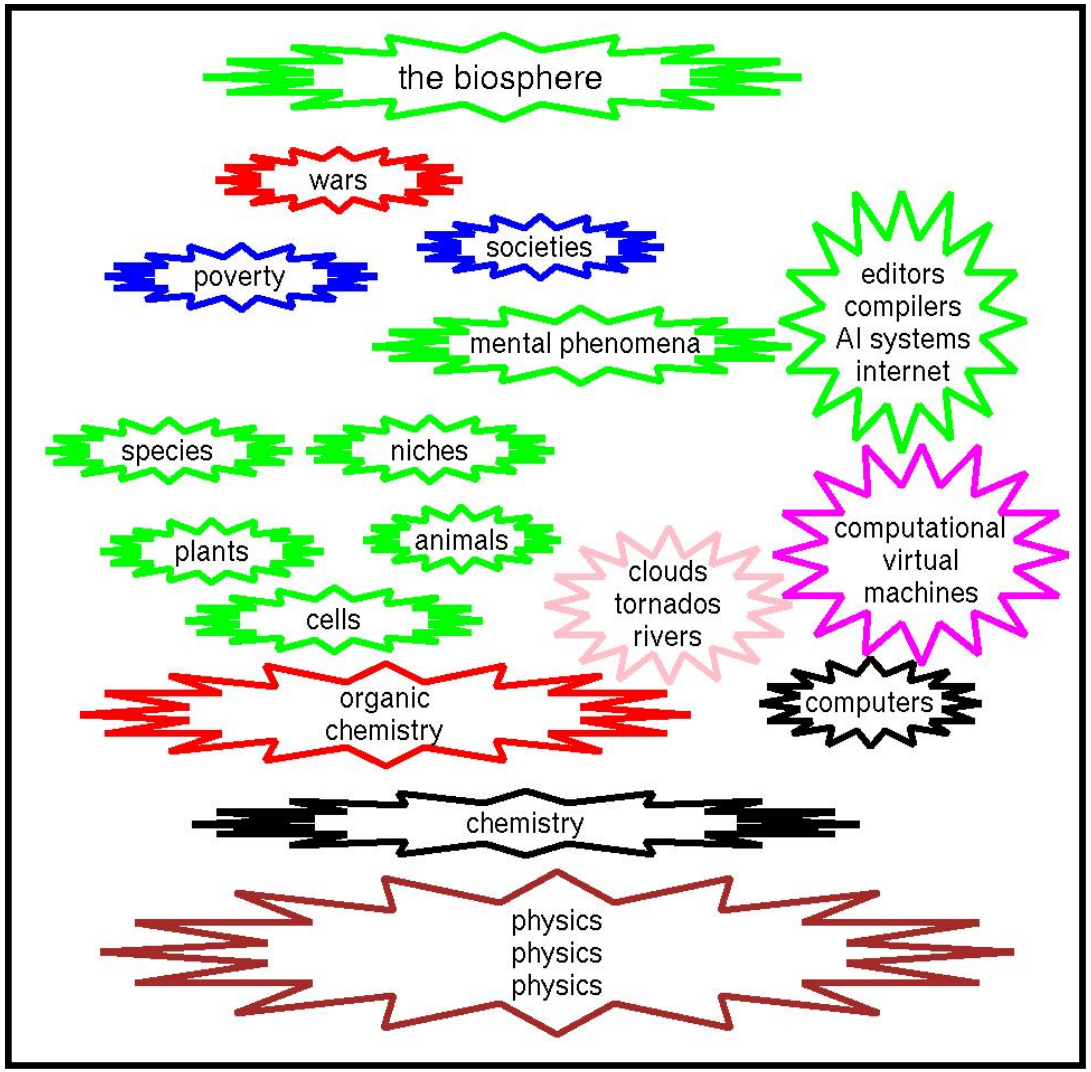
This is “causal dualism”, not “substance dualism”: virtual machines do not contain “stuff” that continues to exist when the physical machine is completely destroyed.

Virtual machines are everywhere

At all levels there are objects, properties, relations, structures, mechanisms, states, events, processes and also many CAUSAL INTERACTIONS.

E.g. poverty can cause crime.

- All levels are ultimately realised (implemented) in physical systems.
- Different disciplines use different approaches (not always good ones).
- Nobody knows how many levels of virtual machines physicists will eventually discover. (Uncover?)
- The study of virtual machines in computers is just a special case of more general attempts to describe and explain virtual machines in our world.



See the IJCAI'01 Philosophy of AI tutorial (written with Matthias Scheutz) for more on levels and causation:
`http://www.cs.bham.ac.uk/~axs/ijcai01/`

Physics also deals with different levels of reality

- The "observable" level with which common sense, engineering, and much of physics has been concerned for thousands of years:
  - levers, balls, pulleys, gears, fluids, and many mechanical and hydraulic devices using forces produced by visible objects.
- Unobservable extensions
  - sub-atomic particles and invisible forces and force fields, e.g. gravity, electrical and magnetic forces.
- Quantum mechanical extensions
  - many things which appear to be inconsistent with the previous ontology of physics

Between the first two levels we find the ontology of chemistry, which includes many varieties of chemical compounds, chemical events, processes, transformations, causal interactions.

The chemical entities, states, processes, causal interactions are normally assumed to be "fully implemented" (fully grounded) in physics.

We don't know how many more levels future physicists will discover.

IS THERE A 'BOTTOM' LEVEL?

In CS there are two notions of virtual machine

We contrast the notion of a PHYSICAL machine with two other notions:

- a VM which is **an abstract mathematical object** (e.g. the Prolog VM, the Java VM)
- a VM that is **a running instance of such a mathematical object**, controlling events in a physical machine, e.g. a running Prolog or Java VM.

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

VMs as mathematical objects are much studied in meta-mathematics and theoretical computer science. They are no more causally efficacious than numbers.

The main theorems of computer science, e.g. about computability, complexity, etc. are primarily about mathematical entities

They are applicable to non-mathematical entities with the same structure – but no non-mathematical entity can be proved mathematically to have any particular mathematical properties.

There's more on varieties of virtual machines in later slides.

More on virtual machines (VMs/NPDMs)

Recapitulation:

A virtual machine (VM) is a NPDM, a machine containing causally interacting components with changing properties and relationships, whose best description uses concepts that cannot be defined in terms of concepts of the physical sciences.

This implies (non-obviously) that there are states and processes in VMs that cannot be measured or detected using the techniques of the physical sciences (e.g. physics, chemistry), though in order to exist and work, a VM needs to be **implemented** in a physical machine.

An example is a collection of running computer programs doing things like checking spelling, playing chess, sorting email, computing statistics, etc.

"Incorrect spelling" cannot be defined in terms of concepts of physics, and instances of correct and incorrect spelling cannot be distinguished by physical measuring devices.

However, a second virtual machine that is closely coupled with the first, might be able to detect that the first is doing those non-physical things.

A socio-economic system is a more abstract and complex form of virtual machine: "economic inflation" and "recession" cannot be defined in terms of concepts of physics

Mental states and processes in humans and other animals can be regarded as states and processes in virtual machines, implemented in brains.

Erroneous philosophy on VMs

Much has been written about virtual machines by philosophers and others, but they are often mistaken,

e.g.

Most ignore the variety of types of VM and the complexity of the relations between VMs and PMs

(e.g. 2-way causation)

Oversimplified notions of VM used by many philosophers

Some philosophers who know about Finite State Machines (FSMs), use a simple kind of "functionalism" (atomic state functionalism) as the basis for the notion of virtual machine, defined in terms of a set of possible states and transitions between them.

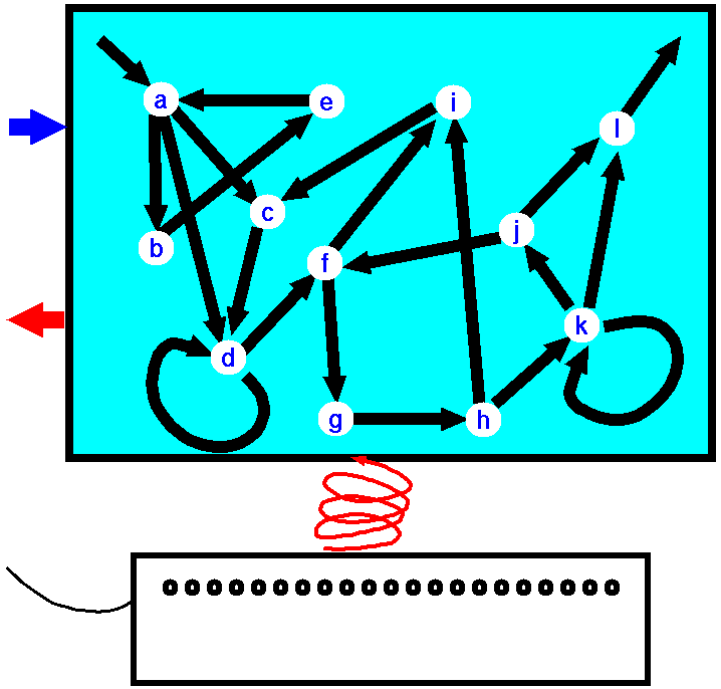
E.g. Ned Block, "What is functionalism?", 1996. (I think he has changed his views since then.)

On this model, a virtual machine that runs on a physical machine has a finite set of possible states (a, b, c, etc.) and it can switch between them depending on what inputs it gets. At each switch it may also produce some output. (The idea of a Turing machine combines this notion with the notion of an infinite tape.)

Finite, Discrete, State Virtual Machine:

Each possible state (e.g. a, b, c, ...) is defined by how inputs to that state determine next state and the outputs produced when that happens.

The machine can be defined by a set of rules specifying the state-transitions.



Implementation relation:

Physical computer:

As demonstrated by Alan Turing and others, this is a surprisingly powerful model of computation: but it is not general enough for our purposes.

A richer model: Multiple interacting FSMs

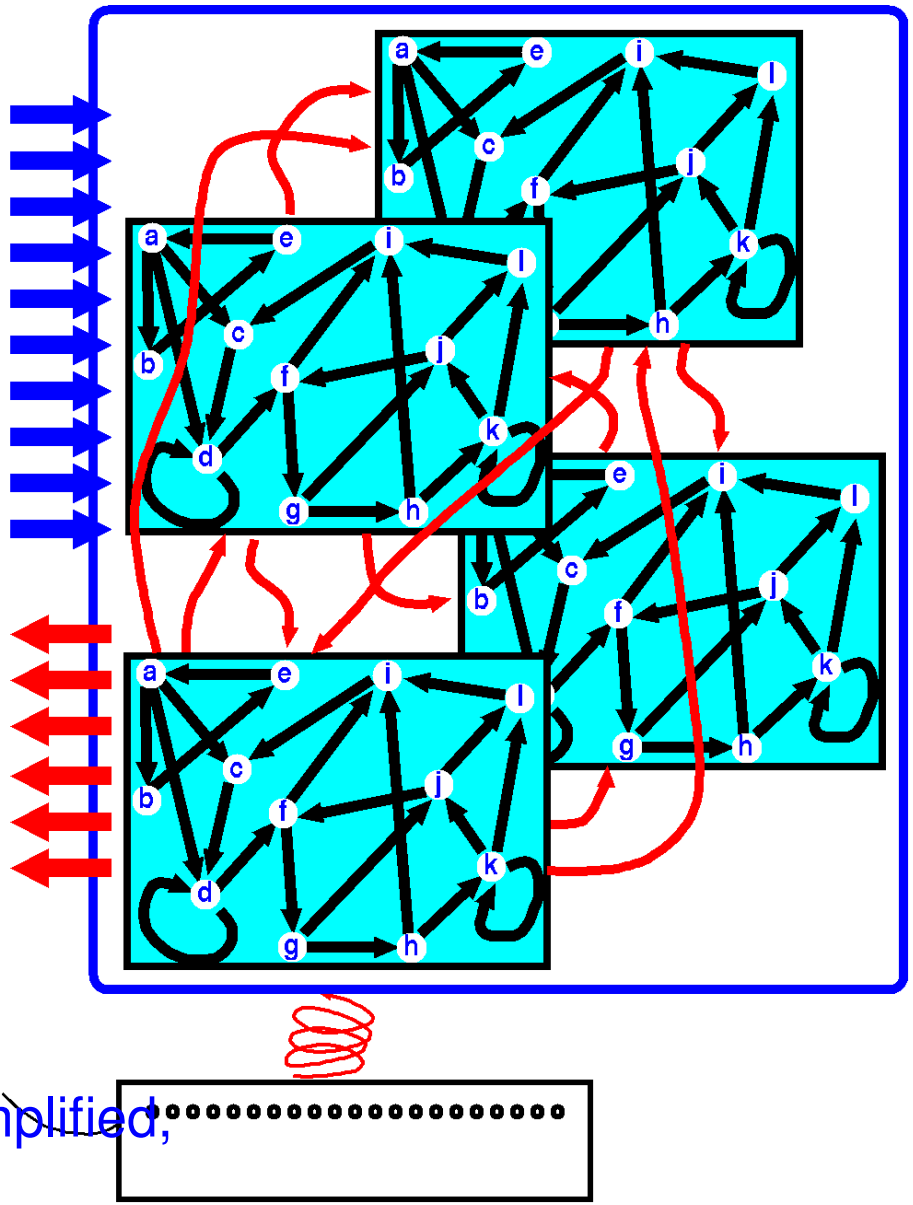
This is a more realistic picture of what goes on in current computers:

There are multiple input and output channels, and multiple interacting finite state machines, only some of which interact directly with the environment.

You will not see the virtual machine components if you open up the computer, only the hardware components.

The existence and properties of the FSMs (e.g. playing chess) cannot be detected by physical measuring devices.

But even that specification is over-simplified, as we'll see.



A possible objection: only one CPU?

Some will object that when we think multiple processes run in parallel on a single-CPU computer, interacting with one another while they run, we are mistaken because only one process can run on the CPU at a time, so there is always only one process running.

This ignores the important role of memory mechanisms in computers.

- Different software processes have different regions of memory allocated to them, which endure in parallel. So the processes implemented in them endure in parallel, and a passive process can affect an active one that reads some of its memory.

Moreover

- It is possible to implement an operating system on a multi-cpu machine, so that instead of its processes sharing only one CPU they share two or more.
- In the limiting case there could be as many CPUs as processes that are running.
- The differences between these different implementations imply that
  how many CPUs share the burden of running the processes is a contingent feature of the implementation of the collection of processes and does not alter the fact that there can be multiple processes running in a single-cpu machine.

A technical point: software interrupt handlers connected to constantly on physical devices, e.g. keyboard and mouse interfaces, video cameras, etc., can depend on some processes constantly "watching" the environment even when they don't have control of the CPU,

In virtual memory systems, and systems using "garbage collection" things are more complex than suggested here: the mappings between VM memory and PM memory keep changing.

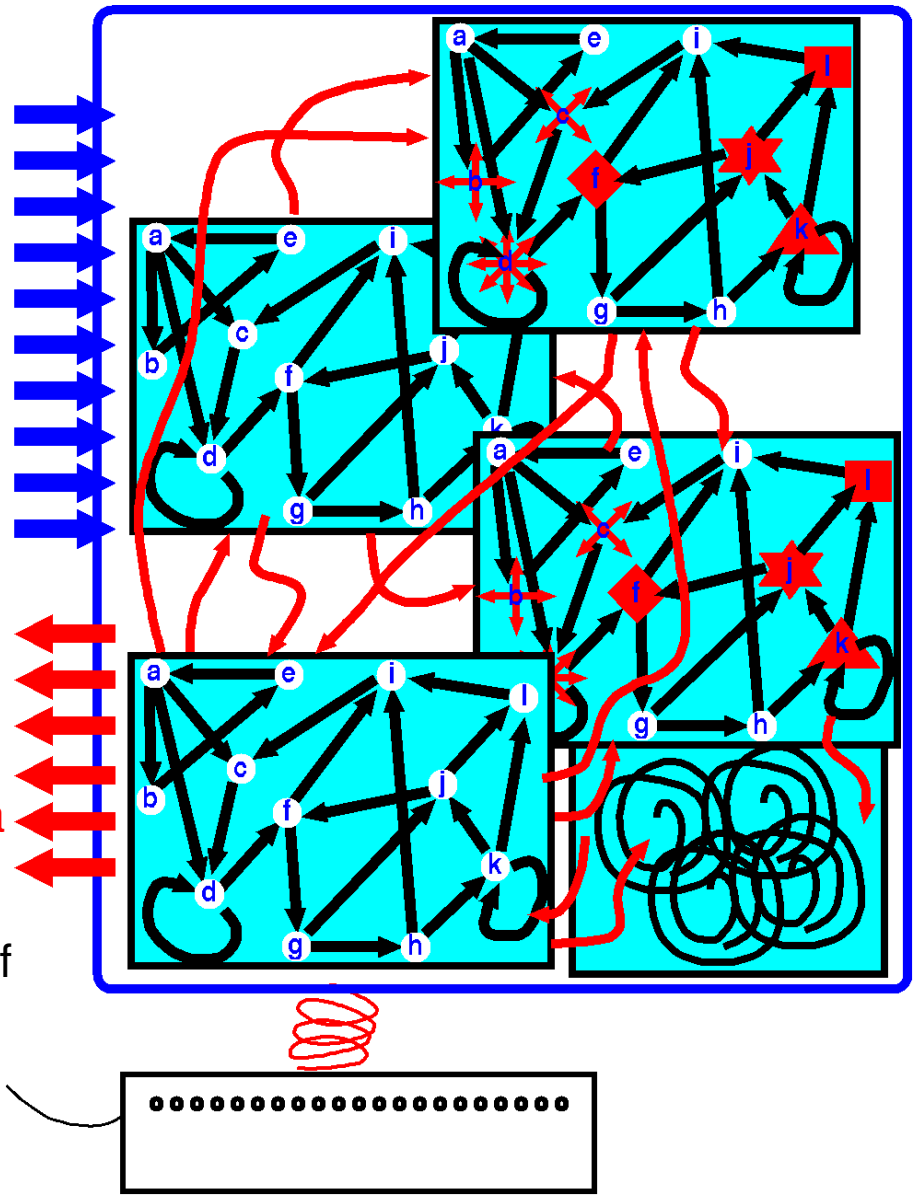
An even more general model

Instead of having a fixed set of sub- processes, many computing systems allow new VMs to be constructed dynamically,

- of varying complexity
- some of them running for a while then stopping,
- others going on indefinitely.
- some spawning new sub-processes...

The red polygons and stars might be subsystems where new, short term or long term, sub-processes (e.g. a new planning or parsing process) can be constructed within a supporting framework of virtual machines.

As indicated in the box with smooth curves, if analog devices are used, there can be VM processes that change continuously, instead of only discrete virtual machines.
Some VMs simulate continuous change.



More general?

Ron Chrisley has challenged my suggestion that the simplest finite state machine is less general than the other forms described here, since a Universal Turing Machine (UTM) is of the first type and all the others can be implemented in a UTM.

My minimal claim is that even if that is true, there are important differences between different designs, and I am concerned with the features of some of the designs.

I could argue that insofar as a system includes sub-VMs that are not synchronised and can vary in speed either randomly or under the influence of the environment and include continuous variation, the system cannot be modelled on a Turing machine.

But that is not the main point: the main point is that there are different sorts of VM whose differences are important in the current context.

Could such virtual machines run on brains?

We know that it can be very hard to control directly all the low level physical processes going on in a complex machine: so it can often be useful to introduce a virtual machine that is much simpler and easier to control.

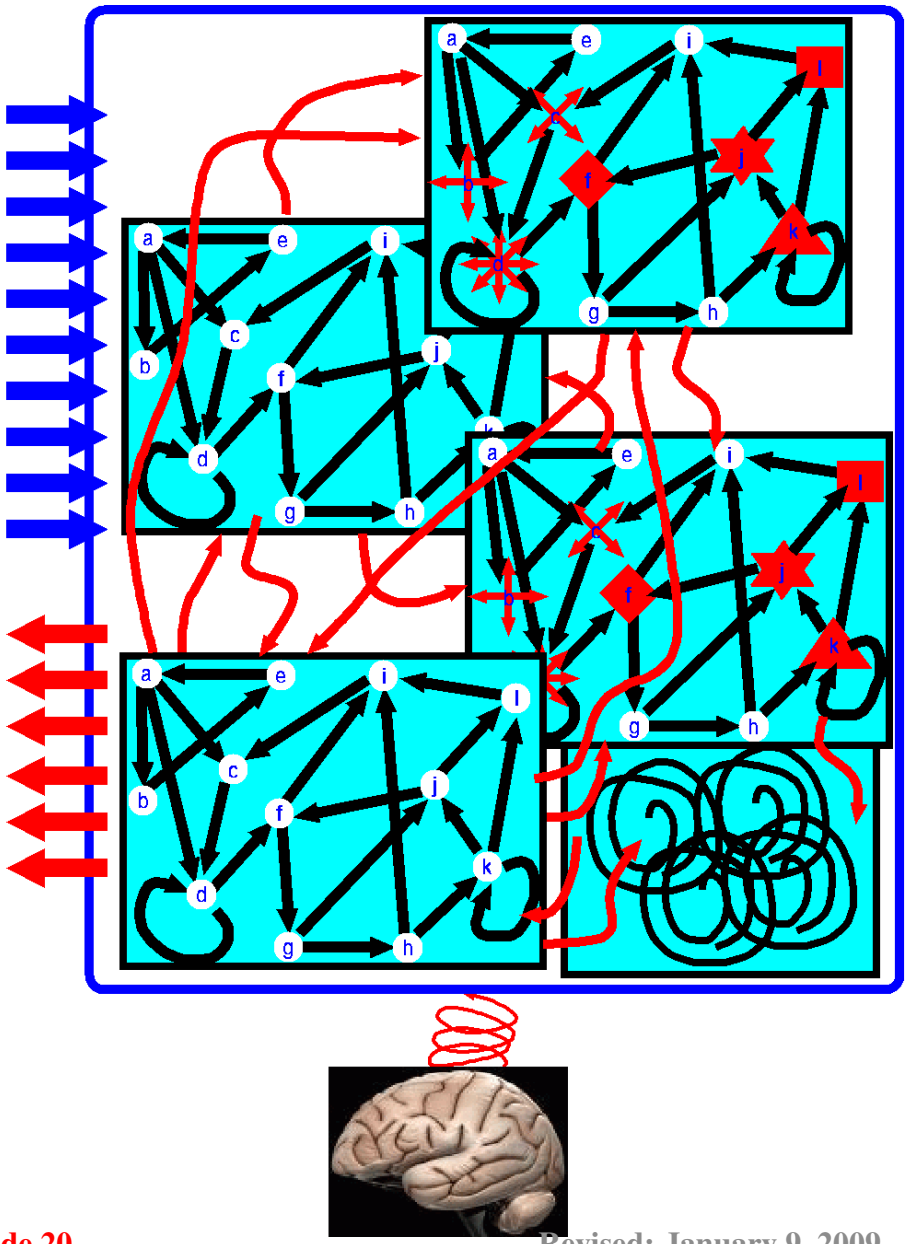
Perhaps evolution discovered the importance of using virtual machines to control very complex systems before we did?

In that case, virtual machines running on brains could provide a high level control interface.

Questions:

How would the genome specify construction of virtual machines?

Could there be things in DNA, or in epigenetic control systems, that we have not yet dreamed of?



VMs can have temporarily or partly 'decoupled' components

- "Decoupled" subsystems may exist and process information, even though they have no connection with sensors or motors.
- 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.)

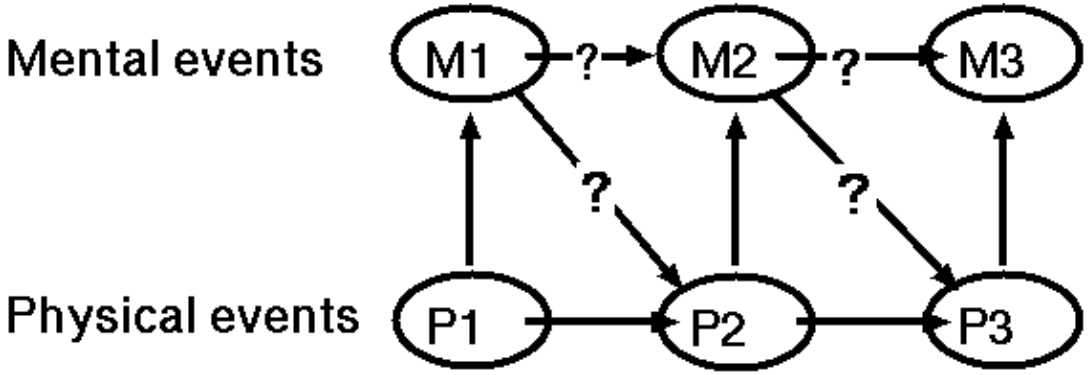
Problem: Supervenience and Causation

Mental states and processes are said to supervene on physical ones.

But there are many problems about that relationship:

Can mental process cause physical processes? (Sometimes called "downward causation".)

How could something happening in a mind produce a change in a physical brain?



(Think of time going from left to right)

If previous physical states and processes suffice to explain physical states and processes that exist at any time, how can mental ones have any effect?

How could your decision to come here make you come here – don't physical causes (in your brain and in your environment) suffice to make you come?

If they suffice, how could anything else play a role?

Explaining what's going on in VMs requires a new analysis of the notion of **causation**

The relationship between objects, states, events and processes in virtual machines and in underlying implementation machines is a tangled network of causal interactions.

Software engineers have an intuitive understanding of it, but are not good at philosophical analysis.

Philosophers mostly ignore the variety of complex mappings between VMs and PMs when discussing causation and when discussing supervenience,

even though most of them now use multi-process VMs daily for their work.

Explaining how virtual machines and physical machines are related requires a deep analysis of causation that shows how the same thing can be caused in two very different ways, by causes operating at different levels of abstraction.

Explaining what 'cause' means is one of the hardest problems in philosophy.

For a summary explanation of two kinds of causation (Humean and Kantian) and the relevance of both kinds to understanding cognition in humans and other animals see:

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

We often assume multiple varieties of causation

A person drives home one night after drinking with friends in a pub.

As he goes round a bend in the road he skids sideways into an oncoming car and and the driver in the other car dies.

In court, the following facts emerge.

- The driver had exceeded the recommended alcohol limit for driving, but had often had the extra glass and then driven home on that route without anything going wrong.
- There had earlier been some rain followed by a sharp drop in temperature, as a result of which the road was unusually icy.
- The car was due for its MOT test, and he had been given two dates for the test, one before the accident and one after. He chose the later date. Had he taken the earlier date worn tires would have been detected and replaced with tires that could have gripped ice better.
- There had been complaints that the camber on the road was not steep enough for a curve so sharp, though in normal weather it was acceptable.
- The driver was going slightly faster than normal because he had been called home to help a neighbour who had had a bad fall.
- A few minutes after the accident the temperature rose in a warm breeze and the ice on the road melted.

What caused the death of the other driver?

How VM events and processes can be causes

We need an explanation of how VM causes and PM causes can co-exist and both be causes of other VM and PM events and processes.

The crucial point is that the existence of causal links is equivalent to the existence of whatever makes certain sets of conditional statements (including counterfactual conditionals) true or false.

A misleading picture:

It oversimplifies.

Our previous diagrams implicitly supported a prejudice by showing a single upward pointing arrow from each physical state to the mental, or virtual machine state, above it.

This implied a simple one-way dependency relationship, where complex two-way relationships actually exist.

Requirements for complex VMs to work

A complete explanation of how VMs interact with PMs in computers would require tutorials on:

- Physical components used in computers
- Digital electronic circuits and their mechanisms
- Various kinds of interfaces/transducers linking computers to other devices (hard drives, displays, keyboards, mice, networks)
- Operating systems
- Device drivers
- File systems
- Memory management systems
- Compilers
- Interpreters
- Interrupt handlers
- Caches
- Programmable firmware stores

and other things I've forgotten to mention.

Note: my own understanding of many of those is incomplete.

A web of causal relationships and conditionals

The design of the various hardware and software items listed in the previous slide has a key feature:

the main effect of all the components is to ensure the simultaneous truth of a complex network of conditional statements relating what would happen if so and so occurred in physical or virtual machines.

The support for that network of truths, including counterfactual conditional truths, e.g. about what would have happened if ...., is equivalent to support for a complex web of causal connections.

As a result of all this, multiple virtual machines of different kinds each with many coexisting, causally interacting components, can coexist and interact and interact with one another and with physical components, in a single physical computer,

even a computer with only one CPU (plus a large number of memory locations).

The picture on the next slide crudely represents that web of interactions, involving both virtual and physical sub-machines.

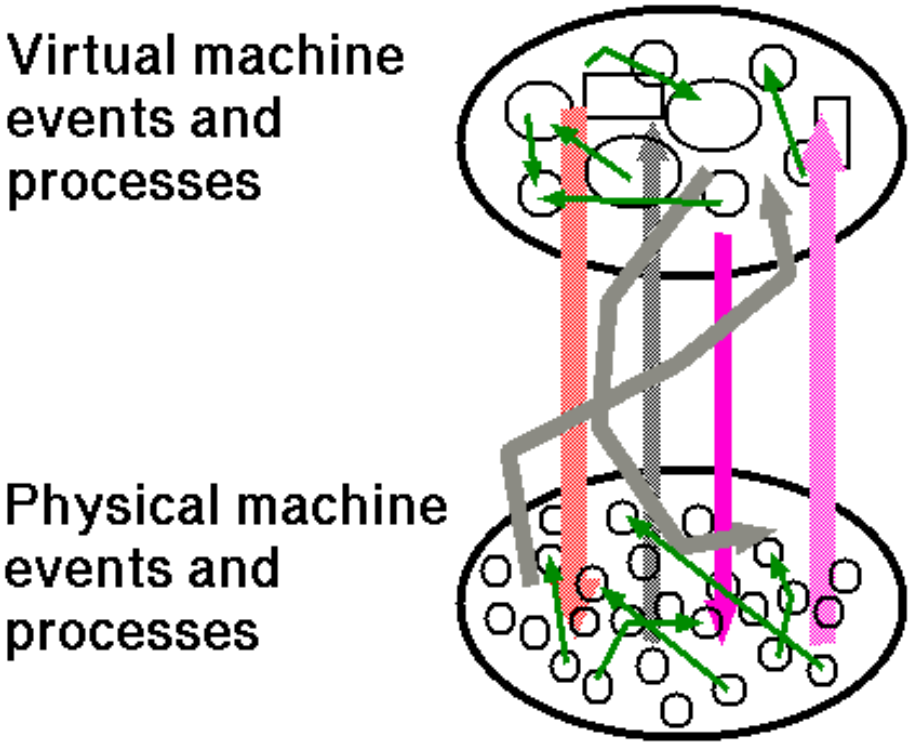
Supervenience of VMs: a complex relation

The two machines (PM and VM) need not be isomorphic: they can have very different structures.

There need not be any part of the PM that is isomorphic with the VM.

Not only static parts and relations but also processes and causal relations can supervene on physical phenomena.

The structure of the VM can change significantly (parts added and removed, and links between parts being added and removed) without structural changes occurring at the physical level –



though the physical states of millions of switches may change as the (much simpler, more abstract) VM changes and causal interactions occur.

The mappings between PM components and VM components may be complex, subtle in kind, and constantly changing.

A very large “sparse array” in the VM may contain many more locations than there are switches in the PM (as long as not all locations are actually occupied).

Distinct objects in the VM can have implementations that share parts of the PM.

Notions of Supervenience

We can distinguish at least the following varieties

- property supervenience

  (e.g. having a certain temperature supervenes on having molecules with a certain kinetic energy.)

- pattern supervenience

  (e.g., supervenience of various horizontal, vertical and diagonal rows of dots on a rectangular array of dots, or the supervenience of a rotating square on the pixel matrix of a computer screen.)

- mereological, or agglomeration, supervenience

  (e.g., possession of some feature by a whole as the result of a summation of features of parts, e.g. the supervenience of the mass of a stone on the masses of its atoms, or the supervenience of the centre of mass on the masses and locations of its parts, each with its own mass)

- mechanism supervenience

  (supervenience of one machine on another: a collection of interacting objects, states, events and processes supervenes on some lower level, often more complex, reality, e.g., the supervenience of a running operating system on the computer hardware – this type is required for intelligent control systems, as probably discovered by evolution millions of years ago?)

We are talking about mechanism supervenience.
The other kinds are less closely related to implementation of VMs on PMs.

Mechanism supervenience, far from being concerned with how one property relates to others, is concerned with how a complex ontology (collection of diverse types of entity, types of events, types of process, types of state, with many properties, relationships and causal interactions) relates to another ontology.

This could be called “ontology supervenience”. Perhaps “ontology instance supervenience” would be better.

A more general notion of supervenience

Supervenience is often described as a relation between **properties:** e.g. a person's mental properties supervene on his physical properties (or "respects").

> '[...] supervenience might be taken to mean that there cannot be two events alike in all physical respects but differing in some mental respects, or that an object cannot alter in some mental respect without altering in some physical respect.'
>
> D. Davidson (1970), 'Mental Events', reprinted in: *Essays on Action and Events* (OUP, 1980).

It's better described as a relation between *ontologies* or complex, interacting, parts of ontology-instances, not just properties.

The cases we discuss involve not just one object with some (complex) property, but a collection of VM components enduring over time, changing their properties and relations, and interacting with one another: e.g. data-structures in a VM, or several interacting VMs, or thoughts, desires, intentions, emotions, or social and political processes, all interacting causally – the whole **system** supervenes.

A single object with a property that supervenes on some other property is just a very simple special case. We can generalise Davidson's idea:

A functioning/working ontology supervenes on another if there cannot be a change in the first without a change in the second.

NOTE: the idea of "supervenience" goes back to G.E.Moore's work on ethics. A useful introduction to some of the philosophical ideas is: Jaegwon Kim, *Supervenience and Mind: Selected philosophical essays*, 1993, CUP.

Multiple layers of virtual machinery

The discussion so far suggests that there are two layers

- Physical machinery
- Virtual machinery

However, just as some physical machines (e.g. modern computers) have a kind of generality that enables them to support many different virtual machines

(e.g. the same computer may be able to run different operating systems – Windows, or Linux, or MacOS, or ....)

so are there some virtual machines that have a kind of generality that enables them to support many different "higher level" virtual machines running on them

(e.g. the same operating system VM may be able to run many different applications, that do very different things, – window managers, word processors, mail systems, spelling correctors, spreadsheets, compilers, games, internet browsers, CAD packages, virtual worlds, chat software, etc. ....)

It is possible for one multi-purpose VM to support another multi-purpose VM, which supports additional VMs.

So VMs may be layered:

VM1 supports VM2 supports VM3 supports VM4, etc.

The layers can branch, and also be circular, e.g. if VM1 includes a component that invokes a component in a higher level VM_k, which is implemented in VM1.

Not to be confused with control hierarchies

Layered virtual machines are not the same as

- Hierarchical control systems
- Brooks' "subsumption architecture"

Here the different layers implement different functions for the whole system, and can be turned on and off independently (mostly).

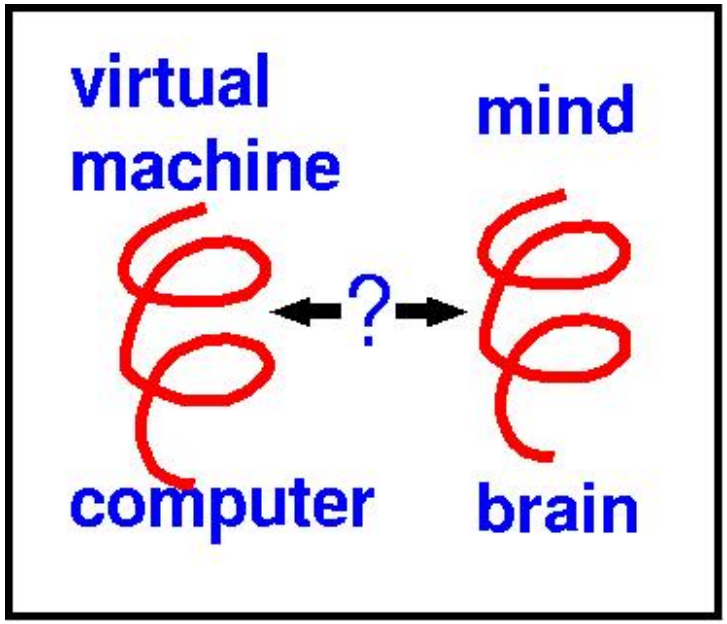
In contrast, a higher level VM provides functionality that is implemented in lower levels: the lower levels don't provide different competences that could be added or removed, e.g. damping.

Removing a lower level VM layer makes the whole thing collapse, unless it replaced by an equivalent lower level VM (e.g. a different operating system with similar functionality).

No time to explain the difference.

‘Emergence’ need not be a bad word

People who have noticed the need for pluralist ontologies often talk about ‘emergent’ phenomena.
But the word has a bad reputation, associated with mysticism, vitalist theories, sloppy thinking, wishful thinking, etc.



If we look closely at the kinds of ‘emergence’ found in virtual machines in computers, where we know a lot about how they work (because we designed them and can debug them, etc), then we’ll be better able to go on to try to understand the more complex and obscure cases, e.g. mind/brain relations.

Virtual machine emergence adds to our ontology: the new entities are not definable simply as patterns or agglomerations in physical objects (they are not like ocean waves).

My claim is that engineers discussing implementation of VMs in computers and philosophers discussing supervenience of minds on brains are talking about the same ‘emergence’ relationship – involving VMs implemented (ultimately) in physical machines.

NB. It is not just a metaphor: both are examples of the same type.

Why virtual machines are important in engineering

They provide "vertical" separation of concerns

Contrast "horizontal" separation: different kinds of functionality that can be added or removed independently, e.g email, web browsing, various games, spreadsheets – or the parts of such subsystems.

- Both horizontal decomposition and vertical decomposition involve modularity that allows different designers to work on different tasks.
- But vertical decomposition involves layers of necessary support.
- VMs reduce combinatorial complexity for system designers
- They can also reduce the complexity of the task of self-monitoring and self control in an intelligent system.
- Evolution seems to have got there first
- That includes development of meta-semantic competences for self-monitoring, self-debugging, etc.
- It can also lead to both incomplete self knowledge and to errors in self analysis, etc.

See also The Well-Designed Young Mathematician, *AI Journal*, December 2008
`http://www.cs.bham.ac.uk/research/projects/cosy/papers/#tr0807`

Concurrent, interacting virtual machines sharing a substrate

In a multi-processing computer the complexity would be totally unmanageable if software designers had to think about all the possible sequences of machine instructions.

Instead we use a VM substrate for handling multiple processes, with mechanisms for

- memory management
- context switching
- scheduling
- handling privileges and access rights, etc.
- filestore management
- various device drivers
- networking
- and in some cases use of multiple CPUs

Some VMs implement a particular kind of functionality (e.g. a chess playing VM) whereas others provide a platform of resources that can be combined in different ways to support multiple kinds of functionality (e.g. operating systems, and various kinds of software development toolkits).

How much of that did evolution discover?

Self-monitoring and virtual machines

Systems dealing with complex changing circumstances and needs may need to monitor themselves, and use the results of such monitoring in taking high level control decisions.

E.g. which high priority task to select for action.

Using a high level virtual machine as the control interface may make a very complex system much more controllable: only relatively few high level factors are involved in running the system, compared with monitoring and driving every little sub-process, e.g. at the transistor level.

The history of computer science and software engineering since around 1950 shows how human engineers introduced more and more abstract and powerful virtual machines to help them design, implement, test debug, and run very complex systems.

When this happens the human designers of high level systems need to know less and less about the details of what happens when their programs run.

Making sure that high level designs produce appropriate low level processes is a separate task, e.g. for people writing compilers, device drivers, etc. Perhaps evolution produced a similar "division of labour"?

Similarly, biological virtual machines monitoring themselves would be aware of only a tiny subset of what is really going on and would have over-simplified information.

THAT CAN LEAD TO DISASTERS, BUT MOSTLY DOES NOT.

Robot philosophers

The simplifications in self-monitoring VMs could lead robot-philosophers to produce confused philosophical theories about the mind-body relationship – e.g. theories about "qualia".

Intelligent robots will start thinking about these issues.

As science fiction writers have already pointed out, they may become as muddled as human philosophers.

So to protect our future robots from muddled thinking, we may have to teach them philosophy!

BUT WE HAD BETTER DEVELOP GOOD PHILOSOPHICAL THEORIES FIRST!

The proposal that a virtual machine is used as part of the control system goes further than the suggestion that a robot builds a high level model of itself, e.g. as proposed by Owen Holland in

`http://cswww.essex.ac.uk/staff/owen/adventure.ppt`

For more on robots becoming philosophers of different sorts see

Why Some Machines May Need Qualia and How They Can Have Them:
Including a Demanding New Turing Test for Robot Philosophers

`http://www.cs.bham.ac.uk/research/projects/cosy/papers/#tr0705`
Paper for AAAI Fall Symposium, Washington, 2007

Biological evolution probably “discovered” all this (and more) first

Even though biological evolution does not need an intelligent designer to be involved, there are strategies that could be useful for evolution for the same reason as they are useful for designers.

That includes the use of virtual machines, for example.

- More precisely – it could turn out that a modification of a design for an organism that gives it a kind of self-understanding its competitors lack, could make it more successful.
- E.g. it may monitor its own reasoning, planning, and learning processes (at a certain level of abstraction) and find ways to improve them.
- If those improved procedures can also be taught, the benefits need not be rediscovered by chance.

Why the same considerations are relevant to biology

Conjecture: biological evolution "discovered" long ago that separating a virtual machine level from the physical level made it possible to use the VM as a platform on which variants could be explored and good ones chosen,

e.g. different behaviours, or different control mechanisms, different mechanisms for choosing goals or planning actions, or different mechanisms for learning things.

- Long before that, the usefulness of "horizontal" modularity had already been discovered, with different neural or other control subsystems coexisting and controlling different body parts, or producing different behaviours, e.g. eating, walking, breathing, circulating blood, repairing damaged tissue.
- But developing new parts with specific functions is different from developing new behaviours for the whole organism.
- If each new behaviour has to be implemented in terms of low level states of muscles and sensors that could be very restrictive, making things hard to change.
- But if a VM layer is available on which different control regimes could be implemented, the different regimes will have much simpler specifications.
- This allows one genome to support multiple possible development trajectories, depending on environment (as in altricial species).

  Conjecture: this allows common functionality to exist following different trajectories (in different individuals with that genome) e.g. doing mathematics or physics in English or Chinese?

A first draft ontology for architectural components

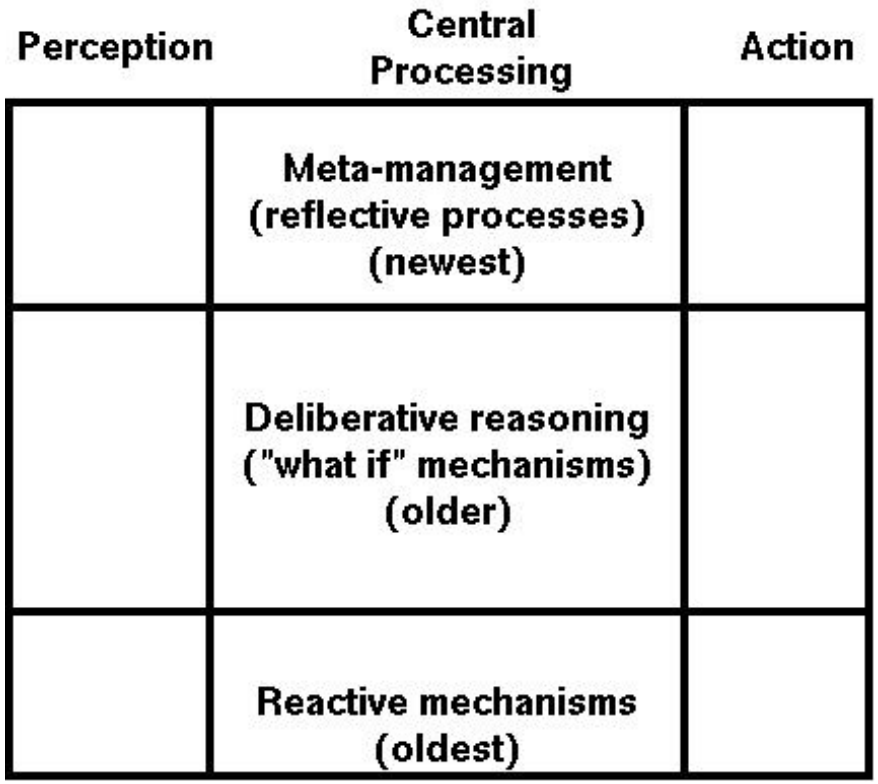
THE COGAFF ARCHITECTURE SCHEMA

For now let's pretend we understand the labels in the diagram.

On that assumption the diagram defines a space of possible information-processing architectures for integrated agents, depending on what is in the various boxes and how the components are connected, and what their functions are.

So if we can agree on what the types of layers are, and on what the divisions between perceptual, central and motor systems are, we have a language for specifying functional subdivisions of a large collection of possible architectures, ...

even if all the divisions are partly blurred or the categories overlap.



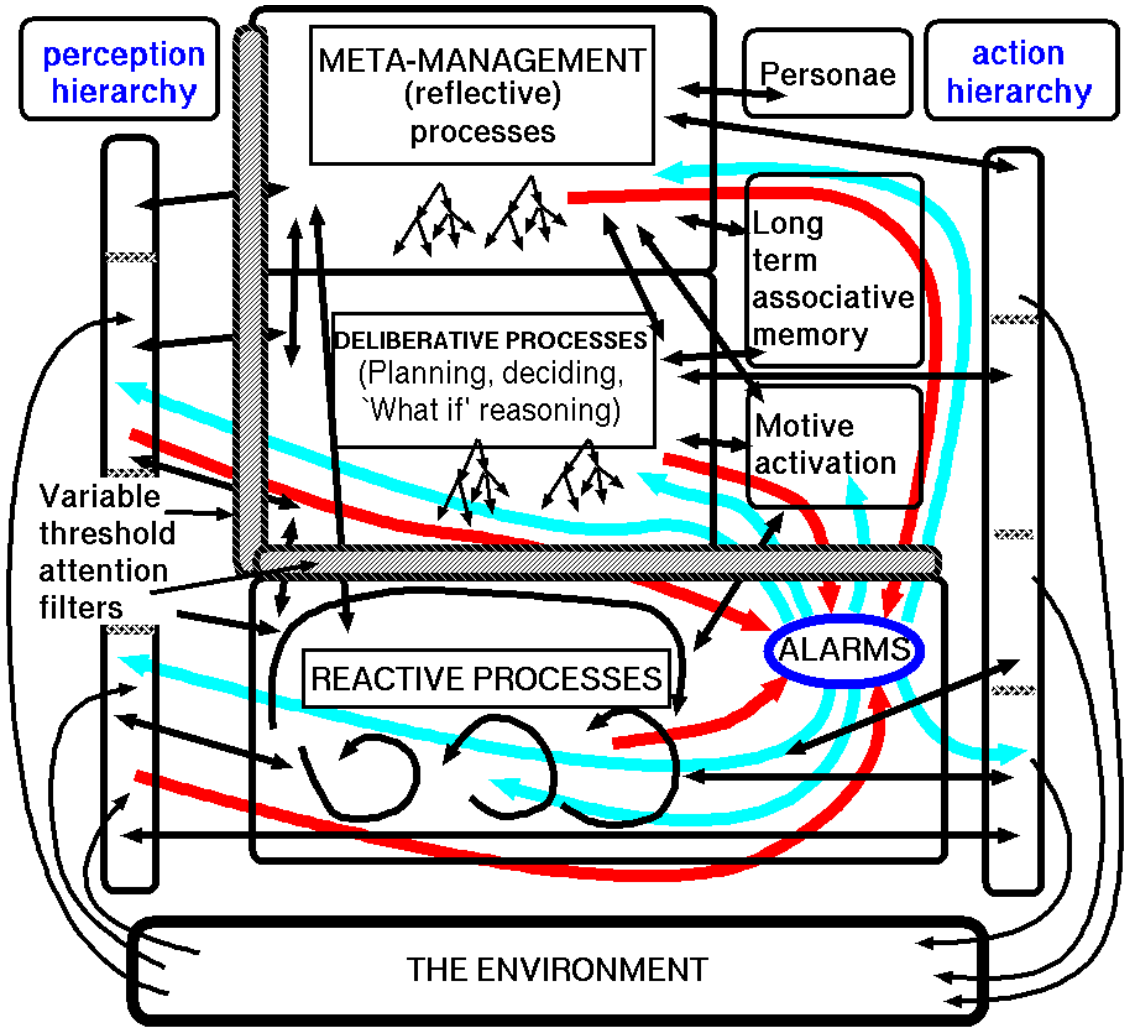
Note: Marvin Minsky's book *The emotion machine* uses finer-grained horizontal division (six layers). There's largely because he divides some of these cogaff categories into sub-categories, e.g. different sorts of reactive mechanisms, different sorts of reflective mechanisms.

What I am heading towards: H-Cogaff

The H-Cogaff (Human Cogaff) architecture is a (conjectured) special case of the CogAff schema, containing many different sorts of concurrently active mutually interacting components.

The papers and presentations on the Cognition & Affect web site give more information about the functional subdivisions in the proposed (but still very sketchy) H-Cogaff architecture, and show how many different kinds of familiar states (e.g. several varieties of emotions) could arise in such an architecture.

This is shown here merely as an indication of the kind of complexity we can expect to find in some virtual machine architectures for both naturally occurring (e.g. in humans and perhaps some other animals) and artificial (e.g. in intelligent robots).



The conjectured H-Cogaff (Human-Cogaff) architecture

See the web site: `http://www.cs.bham.ac.uk/research/cogaff/`

MORE TO BE SAID BUT NO MORE TIME

Summary:

The idea of a virtual machine (or NPDM) is deep, full of subtleties and of great philosophical significance, challenging philosophical theories of mind, of causation, and of what exists.

The use of virtual machines has been of profound importance in engineering in the last half century, even though most of the people most closely involved have not noticed the wider significance of what they were doing –

especially the benefits of vertical separation of concerns, and the complexity of what has to be done to make all this work.

Biological evolution appears to have “discovered” both the problems and this type of solution long before we did, even long before humans existed.

Despite the benefits the use of virtual machines can bring problems and some of those problems may afflict future intelligent machines that are able to think about themselves.

See also `http://www.cs.bham.ac.uk/research/projects/cogaff/talks/#virt`

There are lots more slides and more on the web

Give to google: ”Aaron Sloman” talks

My talks page has several related presentations.

Further reading: Background to these slides

For many years, like many other scientists, engineers and philosophers, I have been writing and talking about "information-processing" systems, mechanisms, architectures, models and explanations, e.g.:

My 1978 book *The Computer Revolution in Philosophy*, now online here:
`http://www.cs.bham.ac.uk/research/cogaff/crp/` (especially chapters 6 and 10)

A. Sloman, (1993) 'The mind as a control system,' in *Philosophy and the Cognitive Sciences,* Cambridge University Press, Eds. C. Hookway & D. Peterson, pp. 69–110.
Online here: `http://www.cs.bham.ac.uk/research/cogaff/`

Since the word "information" and the phrase "information-processing" are both widely used in the sense in which I was using them, I presumed that I did not need to explain what I meant. Alas I was naively mistaken:

- Not everyone agrees with many things now often taken as obvious, for instance that all organisms process information.
- Some people think that "information-processing" refers to the manipulation of bit patterns in computers.
- Not everyone believes information can cause things to happen.
- Some people think that talk of "information-processing" involves unfounded assumptions about the use of representations.
- There is much confusion about what "computation" means, what its relation to information is, and whether organisms in general or brains in particular do it or need to do it.
- Some of the confusion is caused by conceptual unclarity about virtual machines, and blindness to their ubiquity.

Further Reading

A very stimulating and thought provoking book overlapping with a lot of this presentation is

George B. Dyson *Darwin among the machines: The Evolution Of Global Intelligence* 1997, Addison-Wesley.

Papers and presentations on the Cognition and Affect & CoSy web sites expand on these issues, e.g.

- A. Sloman & R.L. Chrisley, (2003),
  Virtual machines and consciousness, in *Journal of Consciousness Studies*, 10, 4-5, pp. 113–172,
  `http://www.cs.bham.ac.uk/research/cogaff/03.html#200302`
- A. Sloman, R.L. Chrisley & M. Scheutz,
  The Architectural Basis of Affective States and Processes, in *Who Needs Emotions?: The Brain Meets the Robot,* Eds. M. Arbib & J-M. Fellous, Oxford University Press, Oxford, New York, 2005.
  `http://www.cs.bham.ac.uk/research/cogaff/03.html#200305`
- A. Sloman and R. L. Chrisley,
  More things than are dreamt of in your biology: Information-processing in biologically-inspired robots, *Cognitive Systems Research*, 6, 2, pp 145–174, 2005,
  `http://www.cs.bham.ac.uk/research/cogaff/04.html#cogsys`
- A. Sloman
  The well designed young mathematician. In *Artificial Intelligence* (2008 In Press.)
  `http://www.cs.bham.ac.uk/research/projects/cosy/papers/#tr0807`
- "What's information?"
  `http://www.cs.bham.ac.uk/research/projects/cogaff/misc/whats-information.html`
- Presentations `http://www.cs.bham.ac.uk/research/projects/cogaff/talks/`

M. A. Boden, 2006, *Mind As Machine: A history of Cognitive Science* (2 Vols), Oxford University Press

There are many other books in philosophy of mind and cognitive science.

For more on all this

Some computer scientists and AI researchers have appreciated the importance of these ideas, and are investigating ways of giving machines more self awareness, in order to make them more intelligent.

John McCarthy, “Making robots conscious of their mental states”.
`http://www-formal.stanford.edu/jmc/consciousness.html`

John L. Pollock (a computationally informed philosopher)
“What Am I? Virtual machines and the mind/body problem”, *Philosophy and Phenomenological Research*, 2008, 76, 2, pp. 237–309,
`http://philsci-archive.pitt.edu/archive/00003341`

Alos work by Dave Clark at MIT on ‘The knowledge layer’ in intelligent self-monitoring networks.

Cognition and Affect Project and CoSy Project papers and talks:
`http://www.cs.bham.ac.uk/research/cogaff/`
`http://www.cs.bham.ac.uk/research/cogaff/talks/`
`http://www.cs.bham.ac.uk/research/projects/cosy/papers/`

The Tutorial presentation by Matthias Scheutz and myself on Philosophy of AI at IJCAI’01.
`http://www.cs.bham.ac.uk/research/cogaff/talks/#talk5`

Collaborative work with Jackie Chappell on cognitive epigenesis
Jackie Chappell and Aaron Sloman, ‘Natural and artificial meta-configured altricial information-processing systems,’ in *International Journal of Unconventional Computing*, 3, 3, pp. 211–239,
`http://www.cs.bham.ac.uk/research/projects/cosy/papers/#tr0609`