Presented at Birmingham University, Sept 2000
Nottingham University, Computer Science Dept. Thurs 29th Nov 2001
Re-written October 2007

Supervenience, Implementation and Virtual Machines

(Philosophy and Software Engineering)

Aaron Sloman
http://www.cs.bham.ac.uk/axs
School of Computer Science
The University of Birmingham

With much help from Matthias Scheutz and Ron Chrisley

This talk is online as
http://www.cs.bham.ac.uk/research/cogaff/talks/#super

Related papers and slide presentations can be found at
http://www.cs.bham.ac.uk/research/cogaff/
http://www.cs.bham.ac.uk/research/cogaff/talks/
especially this talk on "information"
http://www.cs.bham.ac.uk/research/cogaff/talks/#inf

The importance of virtual machines

Many, and probably all, researchers in psychology, brain science, ethology, refer to states, events, processes and entities in virtual machines when they talk about experiences, decisions, intentions, thoughts, learning, feelings, emotions.

The concepts used are not definable in the language of the physical sciences but they refer to real phenomena which are are implemented or realised in the physical world.

By having a clearer view of what virtual machines are, what they can do, and under what conditions they exist, scientists may come up with better, more complete, more powerful, explanatory theories.

This requires adopting the “design stance”.

By clarifying the nature of virtual machines, their relationships to phenomena studied by the physical sciences, and especially their causal powers, we can clarify many old philosophical puzzles and explain why they arise naturally in intelligent, reflective, systems with human-like virtual machines.

I shall try to show how in what follows.

What's the problem?

WHAT SORTS OF THINGS EXIST?

- Social and economic objects, events and processes?
- Computational objects, events and processes?
- Mental objects, events and processes?
- Information, and events and processes involving information?
- Physical objects, events and processes?
- Others ???

  (I am not going to discuss gods, angels, devils, souls, ghosts and other paraphernalia of religion and superstition, only things that can be objects of analytical enquiry and scientific investigation.)

Normally we assume many different kinds of things exist and that their existence is real enough for them to have effects.

Many people, though not all, also normally think, numbers, theorems, proofs, and other mathematical objects (e.g. transfinite ordinals) exist, though not as causally active parts of the universe.

That kind of existence is important, but will not be discussed in detail here.

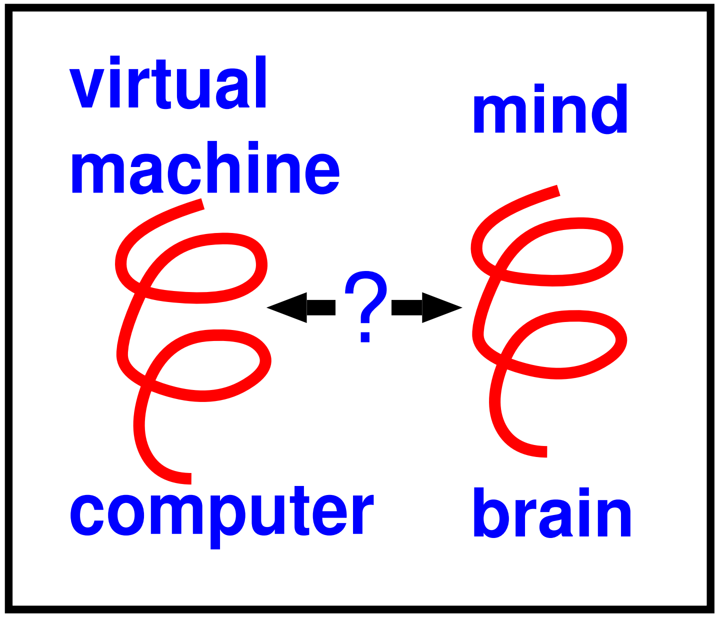
IT IS NATURAL TO HAVE A "PLURALIST" ONTOLOGY

(Until you start to philosophise.)

‘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 vitalist theories, sloppy thinking, wishful thinking, etc.

Thesis: engineers discussing implementation of a running virtual machine in a computer and philosophers discussing supervenience of a mind on a brain are talking about the same general ‘emergence’ relationship.



If we look closely at the kinds of ‘emergence’ found in virtual machines (VMs) 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.

However that emergence relationship has many complex, unobvious features.

And the two cases are different in many details.

But we need to understand the commonalities, before we can appreciate the differences.

Is this a new idea?

Yes and no: for decades many philosophers and scientists have suggested that the relation between mind and brain is something like the relation between program and computer.

But that familiar way of expressing things is both misleading and incomplete: when talking about virtual machines we are not talking about *programs*.

- A program is just a static collection of text (or if stored in a computer, a collection of bit-patterns, or states of a set of switches).
- A mind is something highly active, with internal processes that interact with one another, and which are influenced by and influence things in the environment, including other minds.
- For a better model of mind, we need something that is *abstract* in the way that a program is, but which is also *active* and includes many internal components constantly interacting with one another.
- The notion of a 'running virtual machine' (RVM), is much more appropriate than the notion of a 'program'.

  Running virtual machines are not programs, but are created when programs run on computers, producing abstract entities and interacting processes of many kinds.
- That is not a new idea, but its full meaning and its implications are rarely discussed.

Varieties of virtual machines

The phrase ‘virtual machine’ has become widely used mainly through its use in computer science and software engineering, where it refers to an abstract process running on one or possibly several computers, though this use can be confused with other uses described below.

Examples of running virtual machines (RVMs) include:

- The active run-time system of a programming language,
  e.g. a running Lisp VM, a running Prolog VM, a running Java VM.
- A running operating system, which may keep running non-stop for many months or years,
  e.g. Solaris, Linux
- A game-playing process: e.g. a chess VM, or Doom VM.
- A running word processor or spreadsheet package, or internet browser
  e.g. an electronic mail system, or a web browser left running for months.

Computer scientists and software engineers know a great deal about such virtual machines: they design them, they create them, they improve them, and they fix them when they have bugs.

They also have a lot of intuitive, but mostly unexpressed, knowledge about the relations between virtual machines and the physical computers on which they run.

But philosophers don’t discuss this much.

They prefer to discuss the far more complicated mind-brain relationship, but lack the conceptual tools required, because they cannot even handle the simpler cases: computer RVMs.

Engineers, Philosophers, Psychologists

Engineers and philosophers unwittingly talk about the same relationship: the virtual/physical relationship. However

- Engineers (especially software engineers)

  know a lot about it but lack the philosophical training required to articulate their ‘craft’ knowledge

  Moreover, there may be types of virtual machines about which they know nothing, because they have not yet been discovered/invented, or because their area of expertise is limited.

- Philosophers

  know very little about the relationship, because they have never designed and implemented a working example, but are very articulate and write a lot about it – and often get things wrong because of what they don’t know. (See below.)

- Psychologists and neuroscientists:

  Some psychologists and brain scientists think the mind-brain relationship is to be explained by searching for ‘neural correlates of consciousness’ (NCC).

  See: http://listserv.uh.edu/archives/psyche-b.html

  However, mere correlation explains nothing: we need to know *why* these brain processes produce those mental processes.

  Contrast the way in which software engineers, compiler designers, and computer scientists explain how physical processes in computers produce running virtual machines doing various things.

  They talk about compilers, interpreters, the digital electronic virtual machine, the operating system, etc.

  They do not need to talk about virtual-physical correlations: correlations are not what explain.

Some scientists ignore the problem because they have been badly trained philosophically and regard talk about mental events and processes or emergent levels as ‘unscientific’.

Physics also deals with different levels of reality

- The "observable" level with which common sense, engineering, and much of physics have been concerned for thousands of years:
  - levers, balls, pulleys, gears, fluids, solids, and other kinds of stuff, 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.

    How can we understand what we mean when we talk about unobservable things?

    A partial answer is offered here:
    `http://www.cs.bham.ac.uk/research/projects/cogaff/talks/#models`
- 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, normally assumed to be "fully implemented" in physics.

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

IS THERE A 'BOTTOM' LEVEL?

Virtual machines are everywhere

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

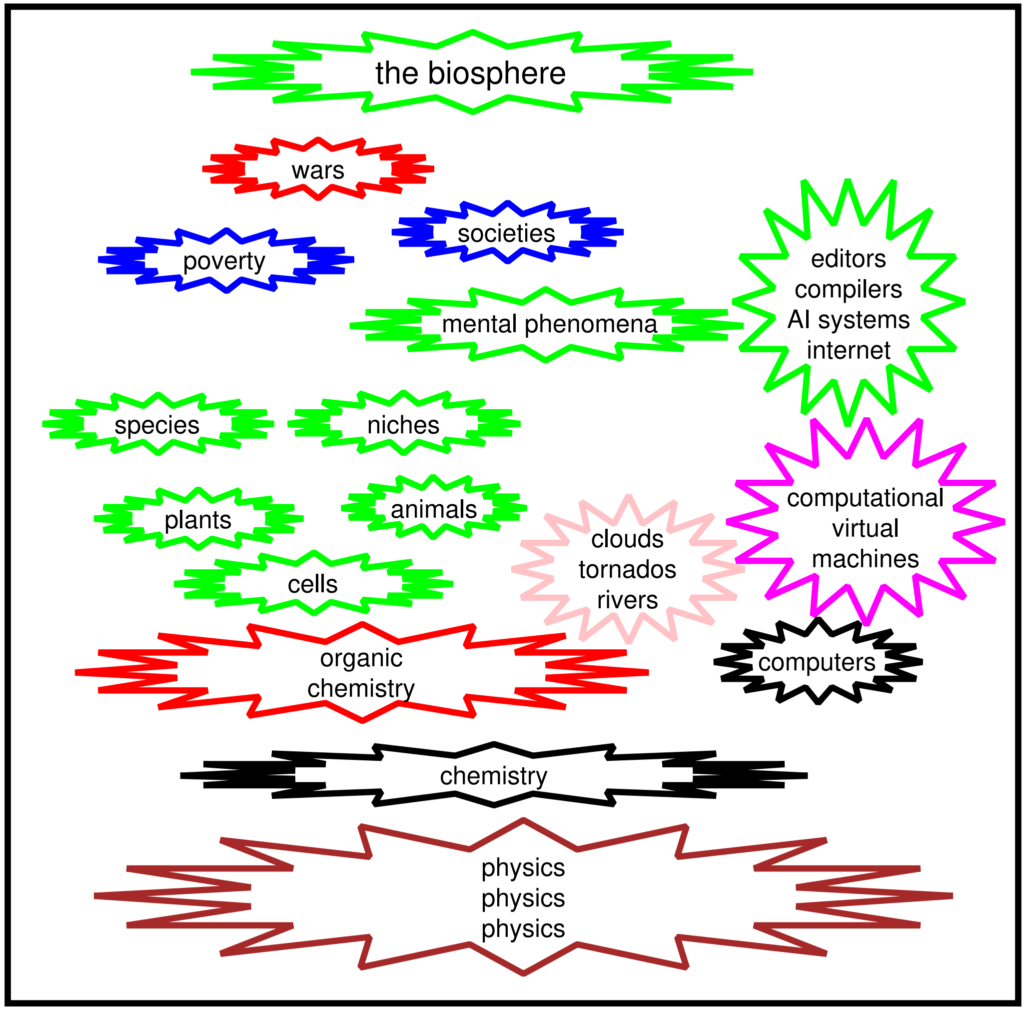
E.g. poverty can cause crime.

But they are all 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?)

Our emphasis on virtual machines is just a special case of the general need 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/`

How to make progress regarding what's real

Quine: "To be is to be the value of a variable"

He equates what people think exist with what their quantifiers (e.g. "All", and "Some") range over.

This raises many problems, including the problem of how we can quantify over sets of entities about which we know nothing, e.g. future events and people, entities discovered in future by scientists.

Me: "To be is to be capable of causing and/or being caused "

I suggest that we can make more progress if instead of talking about what exists, or is real, we talk about what is capable of being involved in causing things other than itself, or being an effect of other causes: TO BE IS TO INTERACT CAUSALLY.

Existence of numbers, sets, proofs, theorems, etc. is something different.

We may not know what causes a particular disease but if there is something that causes it then that something exists: Human knowledge is not a prerequisite for existence.

We can hypothesise that the cause exists, even if we do not know whether it is a chemical compound, a living organism, a kind of hitherto undetected radiation, some psychosomatic process, etc.

Of course, if it turns out that what we thought was one disease is several diseases with distinct causes, then what we thought existed doesn't, but other things closely related to it do exist.

Objects, events, states, processes in RVMs are capable of being causes and being influenced by other causes: That is why we find so many virtual machines running on computers so useful — and depend on them more and more.

Rigid Physicalism

What we could call 'rigid physicalism' states:

- there is only one level of reality, the 'fundamental' physical level,
- causes can exist only at that level of physics, and nowhere else,
- everything else is just a way of looking at those fundamental physical phenomena.

- Rigid physicalism leaves many questions unanswered: we have no idea what will be regarded as fundamental in physics in a hundred or a thousand years time, or perhaps is already so regarded by more advanced physicists on another planet,
- If the only 'real' causes are those that operate at the fundamental level of physical reality then our talk about one billiard ball *causing another to move* does not describe what is 'really' going on.

This extreme version of the theory that 'only fundamental physical causes are real' implies that all our common sense beliefs about existence and causation may be illusory.

Not only our belief that poverty can cause crime, that ignorance can cause poverty, that feeling sympathy can cause people to help others, that the spread of false information can cause social unrest, but also many everyday examples that we think of as physical causation such as ice on a road causing a crash would all be rejected as not being real causation.

A possible response is that what we are really talking about in these cases is fundamental physical entities and processes even when we think we are talking about something else.

This is the 'identity theory' of emergence: all the objects, events, states, processes, that exist just are physical events described as if they were something different.

An alternative view: Flexible physicalism

An alternative view is that there are different 'levels' at which causes can operate: causation is not restricted to the 'bottom' level of physical reality.

On that alternative many things, including sub-atomic particles and other recently discovered physical entities can interact causally, and so can older, more familiar things, such as

- billiard balls,
- clock-springs,
- tidal waves,
- tornadoes,
- weather-fronts,
- epidemics, etc.

And also many non-physical things:

For instance biological entities, social phenomena, economic phenomena, mental phenomena.

Economic inflation really can occur, and really can have effects.

On this 'flexible physicalist' view, many kinds of things exist and enter into causal relationships, even though in some sense their existence and their causal powers 'ultimately' depend on underlying physical states and processes.

Whether the difference between rigid and flexible physicalism is significant is something we'll turn to later: First we discuss information and information-processing.

A consequence of flexible physicalism

A consequence of flexible physicalism that some will find unpalatable is that 'downward causation' is possible.

- In other words, states, events and processes in a virtual machine V can not only produce effects in other parts of V, but can also produce effects in the physical or other lower levels in terms of which V is implemented.
- That follows from the fact that the changes produced in V cannot occur without other changes occurring at the implementation level (or possibly in the environment).

  That is a consequence of V being 'fully grounded' in the lower level machine, as defined below.
- The resistance to accepting downward causation is based on a failure to understand that causation is essentially a matter of a set of counterfactual conditional statements being true, as discussed later.

Beyond matter- and energy-manipulating machines

For centuries humans have studied, built, and used machines that

- manipulate matter, and
- manipulate energy

We need to understand a third kind of manipulation done by machines: There are information-processing machines, especially virtual information-processing machines.

THESE ARE WIDELY USED BUT NOT WIDELY UNDERSTOOD.

- Software engineers have a deep intuitive understanding of the new kind of machine, but often cannot articulate what they understand.
- Most philosophers know little about this, because of inadequacies in our educational system!
- Most people frequently interact with computer-based virtual information-processing machines, and indirectly depend on them, whether they know it or not:
  - spelling checkers
  - email programs
  - games software, e.g. a chess virtual machine
  - document formatters
  - spam filters
  - process-schedulers
  - file-system managers with privilege mechanisms
  - control systems for chemical plants or airliners.
  - income tax calculators and other online financial virtual machines.

Two notions of virtual machine

Some people object to the idea that causal interactions can occur in a virtual machine, or that events in a virtual machine can be caused by or can cause physical events,

- because they take literally the loosely worded claim that a virtual machine is a program, or
- because they think of a virtual machine as a mathematical object, about which theorems can be proved (e.g. proving that the Lisp virtual machine has the power of a Turing machine).

We have already noted that a computer program is a static object, e.g. text or machine structure, and needs to be distinguished from a running virtual machine (RVM) created when the program runs.

There is another important distinction between

- a VM which is an abstract mathematical object
  (e.g. the Prolog VM, the Java VM, the Linux VM)
- an RVM that is a running instance of such a mathematical object, controlling events in a physical machine, e.g. the linux virtual machine running on my computer as I type.

The difference between these two is very important.

The mathematical object does not do anything.

The running instances can do many things and can have causally interacting components
(e.g. scheduler, memory manager, file system manager, firewall, ...)

Two notions of virtual machine

Contrast the notion of a physical machine with:

- 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, e.g. about computability, complexity, etc. are primarily about mathematical entities (and non-mathematical entities with the same structure – but no non-mathematical entity can be proved to have any mathematical properties).

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

There are connections between VMs and RVMs

The situation can be confusing because theorems about mathematical structures can be applicable to portions of reality for which the mathematical structures are good models.

- For example, ancient Egyptians used Pythagoras theorem to help them solve practical problems relating to farmland.
- Likewise mathematical reasoning about virtual machines can be used to prove theorems such as that they will not need more than a certain amout of memory.
- This is no different from using differential and integral calculus to prove things about motions of bodies in the solar system.
- In both cases the proofs can be relied on only if the non-mathematical objects actually have properties of the sorts used to define objects about which the theorems are proved.
- In both cases the world can surprise us: e.g. because planets do not move exactly as Newton supposed, or because a program has an unnoticed bug or because the hardware fails while the program is running.
- Much of theoretical computer science is about mathematical objects that are assumed to correspond to RVMs, but in fact may not always not do so (e.g. when there is a bug in the firmware of a CPU, as happened to an Intel design.)

Despite these relationships, the RVMs really are different from the formal mathematical models: RVMs can make things happen or prevent things from happening.

Extending our ontologies and our thinking powers

Many people are taught to think about

- Matter-manipulating machines
- Energy-manipulating machines

But they do not learn to think about

- Information-manipulating machines.

So they often fail to notice important questions and fail to consider important classes of possible answers: like neuroscientists who study neurons, and psychologists who study behaviour.

We are in the very early stages of learning to think about important age-old products of evolution:

- Virtual machines:
  - with real causal powers
  - e.g. decisions change what happens.
- Much concurrency:
  so that it can be misleading to ask what IT (or she or he) is doing, or can do, or notices, perceives, feels, etc.
  - The answers may be different for different parts of the same system.

Functionalism ?

Functionalism is one kind of attempt to understand the notion of virtual machine, in terms of states defined by a state-transition table.

Arrows show possible transitions between labelled states. A full specification would indicate conditions under which optional transitions are followed and what outputs are produced when a transition happens.

These VMs have only a single ‘atomic’ state at any time, and all transitions are discrete.

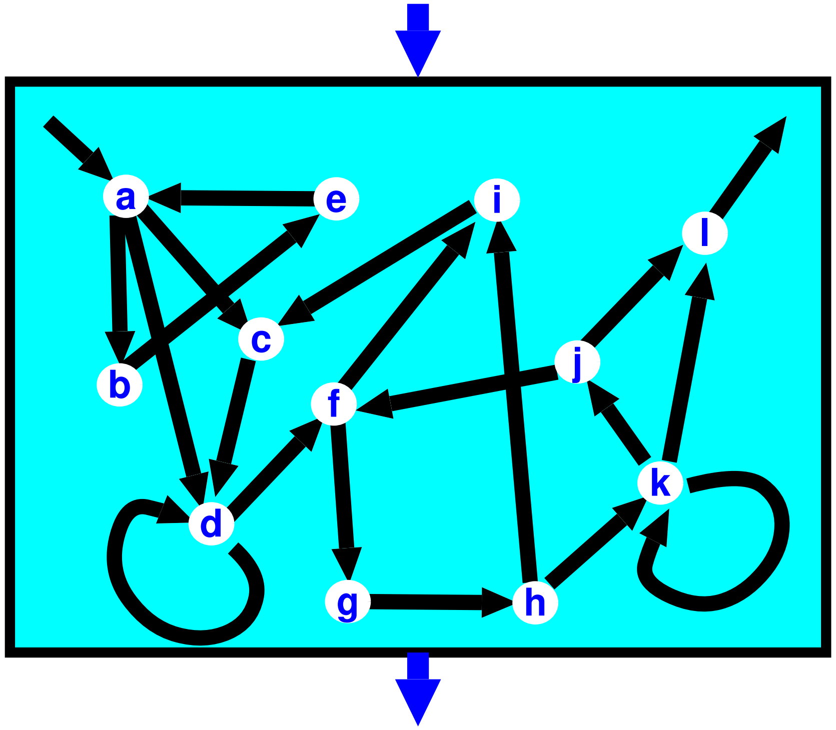
That is a very impoverished model.

This is how many people think of functionalism: there’s a total state which affects input/output contingencies, and each possible state can be defined by how inputs determine next state and outputs.

E.g. see Ned Block’s accounts of functionalism.

There’s a richer, deeper notion of functionalism, which I’ll now try to explain, at a high level of abstraction, namely:

Virtual Machine Functionalism.



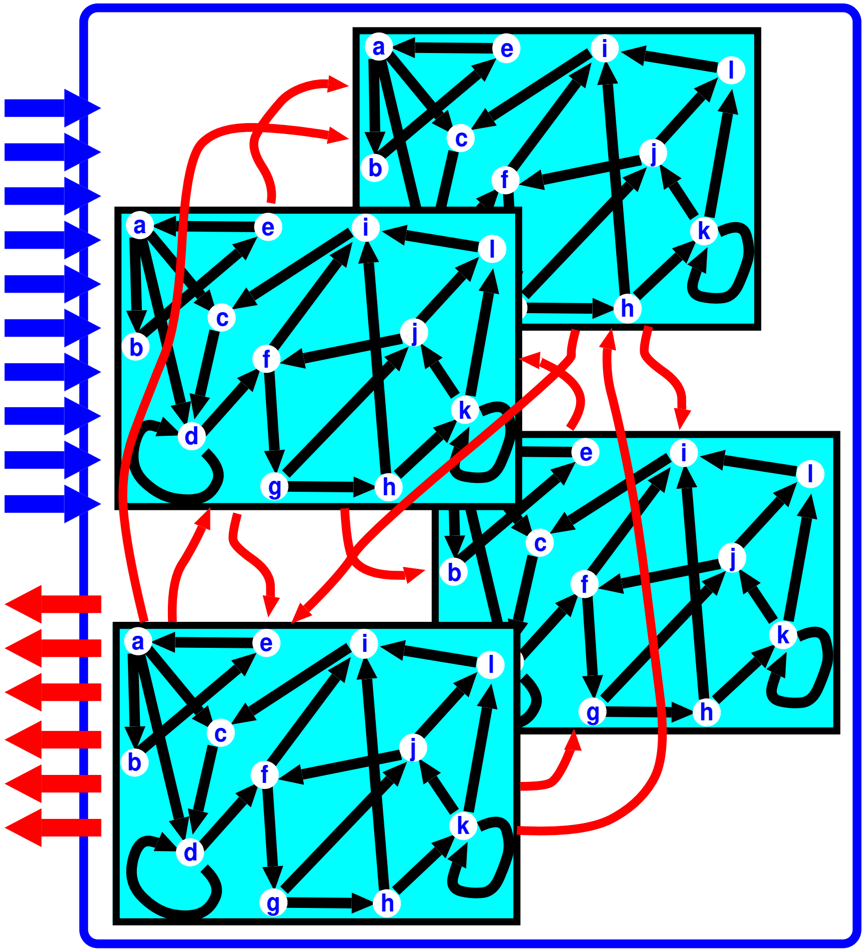
Another kind of Functionalism ?

Instead of having a single (atomic) state which switches when some input is received, a RVM can include many sub-systems with their own states and state transitions going on concurrently, some of them providing inputs to others.

The different states may change on different time scales: some may change very rapidly others very slowly, if at all.

They can vary in their granularity: some sub-systems may be able to be only in one of a few states, whereas others can switch between vast numbers of possible states (like a computer's virtual memory).

Some may change continuously, others only in discrete steps.



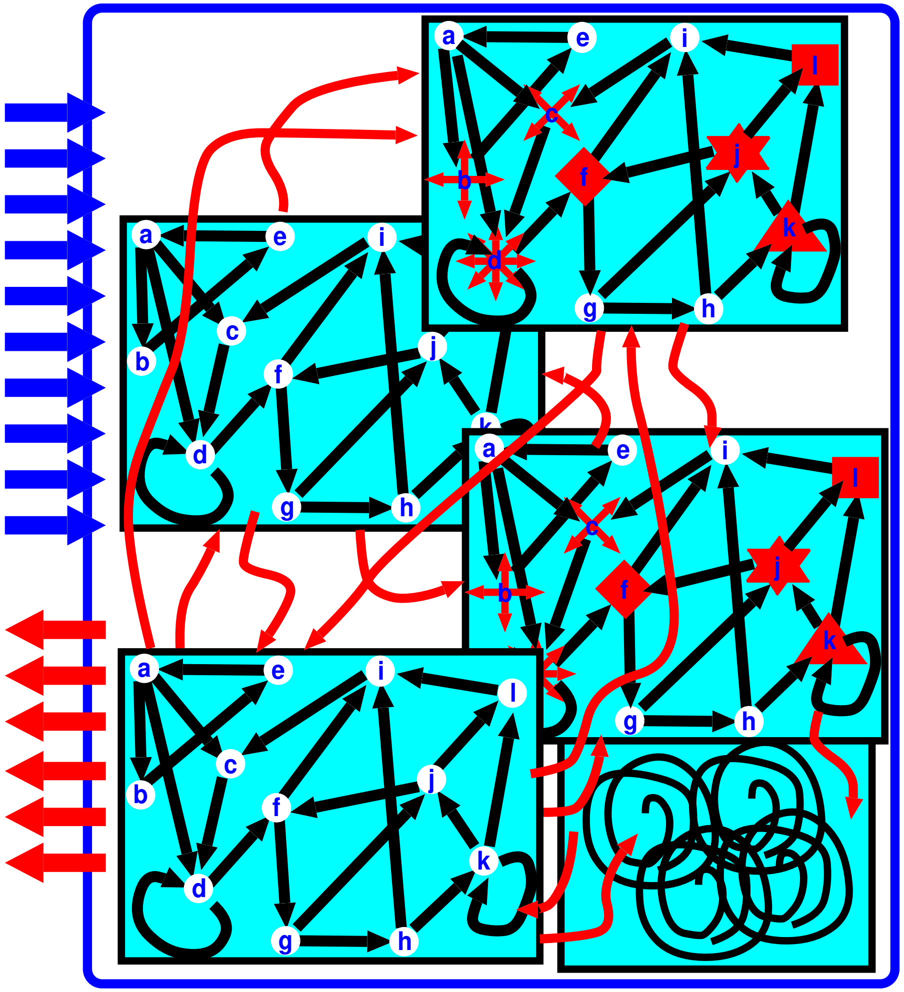
Some may be directly connected to sensors (blue arrows) and effectors (red arrows), whereas others have no direct connections to inputs and outputs and may only be affected very indirectly by sensors or affect motors only very indirectly (if at all!).

The previous picture is still misleading

Because it doesn't allow for continuously changing subystems, and it suggests that the total state is made up of a fixed number of discretely varying sub-states:

We also need to allow systems that can grow structures whose complexity varies over time, as crudely indicated in some boxes on the right,
e.g. trees, networks, algorithms, percepts, plans, thoughts, etc. built at different types, and sometimes later discarded

And systems that can change continuously, such as many physicists and control engineers have studied for many years, as crudely indicated bottom right
e.g. for controlling movements.



The label 'dynamical system' is applicable to all these types of sub-system and to complex systems composed of them.

VMF: Virtual Machine Functionalism

We use “Virtual Machine Functionalism” (VMF) to refer to the more general notion of functionalism, in contrast with “Atomic State Functionalism” (ASF) which is generally concerned with finite state machines that have only one state at a time.

Full VMF allows multiple concurrently active, interactive, sub-states changing on different time scales (some continuously) with varying complexity, and allows the VM to grow itself over time.

VMF also allows that the Input/Output bandwidth of the system with multiple interacting internal states may be too low to reveal everything going on internally.

There may still be real, causally efficacious, internal virtual machine events and processes that cannot be directly observed and whose effects may not even be indirectly manifested externally.

Even opening up the system may not make it easy to observe the VM events and processes (decompiling the physical processes can be too hard).

NOTE: If some links between systems can be turned on and off by internal processes, then during some states: some of the subsystems may not have any causal influence on outputs.

Those running sub-systems still exist and can include internal causal interactions within and between themselves: scientific investigations will have to allow for this possibility.

The notion of a “Turing test” as something that can determine what is going on, fails to take account of many of the possibilities for natural and artificial virtual machines.

(Turing did not propose the test as a test, only as a thought experiment to provoke discussion.)

Could evolution produce de-coupled VM sub-systems?

It is sometimes argued that sub-systems that do not have externally observable effects on behaviour would never be produced by evolution, because they could never provide any biological advantage.

This assumes an over-simplified view of evolution:

It ignores the fact that many neutral or harmless mutations can survive because they don't make sufficient difference to the survival chances of individuals. This could be because the environment is not sufficiently harsh or because more able individuals help less able ones or for other reasons.

A consequence is that a succession of changes that do not directly produce any great benefits (or disadvantages) may eventually combine to produce something very beneficial.

In some cases the benefits are insignificant until there's a major change in the environment requiring some new capability.

E.g. a succession of changes producing a mechanism for "thinking ahead" may be of no real benefit to members of a species until the environment changes so that food is not plentiful and actions to find food have to begin before the food is needed.

Likewise in individual development: virtual machines may change in (partly genetically programmed) ways that have no immediate benefit and show no behavioural consequences, but later on link up with other sub-systems and give the individual considerable advantages, e.g. mathematical thinking capabilities.

Good educational systems have always allowed for this: the child may be incapable of understanding the benefit of what is being learnt until long after it has been learnt.

Further complications

All the RVMs discussed are dynamical systems, but not all dynamical systems have state spaces and trajectories definable in terms of physics; that's just an implementation level.

(E.g. "capture" and "winning" in games VMs are not definable in terms of the concepts of physics.)

Implementations of the same VM can change: e.g. programs that ran on computers 20 years ago can still run on new computers that are physically very different.

- The changes may not have consequences detectable at a certain VM level.
- Where the changed implementation does affect VM processes they may be merely quantitative (e.g. faster, or run out of memory less often, or more reliability) or they may also be qualitative e.g. quite different needs for energy replenishment or temperature control.
  E.g. some programs can now run for hours on a portable, battery driven computer that previously would have require kilowatts of mains power throughout.

Partly "decoupled" subsystems, e.g. produced by neutral mutations, could provide a basis for major new evolutionary developments later, or possibly for certain kinds of learning and development.

Some VMs instead of being fully specified in programs, can build themselves while running, altering their structure and contents according to results of interacting with the environment. E.g. learning French or English or Urdu, ...

Different models of development and learning come from different starting points: Altricial/Precocial species (and machines);

Many of the costs and constraints of biological systems are non-obvious: e.g. evolutionary history may or may not include opportunities for something to have evolved.

Affect in VMs depends on complex interacting subsystems capable of many sorts of processes acting on other processes, including initiation, termination, modulation, arbitration, evaluation ...

Tangled causal webs

Later we give a characterisation of what it is for a VM to be grounded in a physical machine: that is a very abstract notion, similar to what many philosophers have said about ‘supervenience’ or ‘realisation’.

But explaining how a VM works and how the implementation machine makes it work is an entirely different matter: it requires a detailed specification of a a great many causal connections.

The causal connections can form a fairly tangled network involving:

- causal connections between components and processes in the implementation machine (e.g. the physical machine)
- causal connections between components and processes in the virtual machine
- causal connections linking components and processes in both the implementation machine and the virtual machine

As an example: if some part of the VM is capable of being in any one of N distinct states, then normally it should be necessary for the underlying physical machine to be capable of being in at least N distinct states.

But there does not need to be a subdivision of the physical machine into components responsible for implementing specific components of the VM.

The implementation can be distributed, with different VM components sharing physical components for their implementation.

And, as already indicated, the mappings between components at different levels can change over time.

How can we define VM ontologies?

If the concepts used to describe a VM are not definable in terms of the concepts used to describe the lower level machines, and if, as indicated above, they are also not definable in terms of input-output specifications for the whole system (i.e. logical behaviourism is untenable), that raises a question:

HOW CAN WE UNDERSTAND THE CONCEPTS?

I think the answer is the same as for all the theoretical concepts of science referring to unobservables (e.g. 'gene', 'electron', 'electromagnetic radiation', 'economic inflation', etc.)

- The concepts are mostly determined by their role in the theory of the machine, including the specification of their mutual causal interactions, but also partly determined by a small set of links to possible experiments that can be performed and their outcomes.
- How this is possible is spelled out in more detail in this presentation:
  `http://www.cs.bham.ac.uk/research/projects/cogaff/talks/#models`
  Talk 49: Introduction to key ideas of semantic models, implicit definitions and symbol tethering.
- The ideas will be mostly familiar to anyone who has studied 20th Century philosophy of science and who understands Tarski's notion of formal system having a model.
- However as explained in the presentation we need to generalise those ideas beyond standard logical means of expressing theories.

Implications for testable theories

Virtual Machine Functionalism (VMF) implies that theories about systems using virtual machines can be very hard to test directly.

Instead we have to learn to work like physicists investigating sub-atomic entities, events and processes, where only very indirect testing is possible, and the most one can ever say of any theory is:

“This theory at present is better than any of its rivals”

When studying a particular RVM, it is always possible that a new, better, deeper, explanatory theory will turn up than we have discovered at any time, as happened when relativity and quantum mechanics replaced older theories.

This does not make truth relative, only very hard to discover, in some cases.

Mental states and processes on this view are not mere “attributions” – they are real aspects of virtual machines, insofar as they can interact causally with one another and with things in the environment.

Finding the right ontology for describing what’s going on can be very hard: we still have much to learn about this.

Example: The ontology of biology

Biology introduces several non-physical extensions to our ontology:
E.g.

- Organisms
- Reproduction
- Growth, development and learning
- Disease, injury and death
- Species and societies
- Genes and inheritance
- Information (acquired and used by individuals or by genomes)
- Evolution, etc. ....

These are non-physical in that they have properties that are not physical properties, and are not definable in terms of physical concepts, and are not observable or measurable using physical instruments (scales, calipers, voltmeters, thermometers, etc. etc.)

We normally (apart from vitalists and some theologians) assume that, just as chemical phenomena are grounded in physics, so also:

> Biological objects, events, processes are
> "fully grounded (realised)" in physics and chemistry,

The grounding/realisation relation

A FIRST DRAFT DEFINITION

Phenomena of type X (e.g. biological phenomena) are fully grounded in, or realised in, or implemented in phenomena of type Y (e.g. physical phenomena) if and only if:

(a) phenomena of type X *cannot exist without* some entities and processes of type Y.

(i.e. it is necessary that something of type Y exist for anything of type X to exist)

(b) certain entities and processes of type Y *are sufficient for* the phenomena of type X to exist – they constitute the implementation.

(The actual implementation is not necessary: there can be alternative implementations.)

NB: This has nothing to do with “symbol grounding” theory.

Example: if computational virtual machines are fully grounded in physical machines then

(a) computational machines cannot exist without being embodied

(b) their physical embodiments suffice for their existence - no extra independent stuff is needed. no computational spirit, soul, etc.

(c) the particular physical embodiment of a RVM M is not necessary for M, insofar as M can in principle be implemented in different ways. (E.g. some parts could be replaced by superior ones while M is running.)

Later we ask *how* it suffices.

And consider some philosophically varied answers.

Example: a Chess Virtual Machine

The user of a chess playing computer knows that a game of chess is being played when the system runs. The ontology of chess is appropriate for describing what is going on (kings, queens, bishops, pawns, rows, columns, diagonals, captures, threats, pins, etc.)

(Here I assume it is obvious that the concepts of chess, including 'winning', 'playing', 'pawn', 'capture', are not definable in terms of the concepts of the physical sciences. That requires argument.)

Moreover that ontology is instantiated *because* of what is going on in the physical computer when it runs. However:

It is not logically provable that the ontology will be instantiated

The concepts of chess cannot be *defined* in terms of those of physics and therefore it will not be possible to produce *logical* relations between the physical descriptions and the chess descriptions.

(The connection may be mathematical in some sense – see below.)

It is not a subjective interpretation

Chess players would object if the virtual machine started doing something other than playing legal chess.

It is not a mere contingent, empirical fact.

Designers know why those physical processes produce the chess-playing virtual machine processes.

The same chess machine is multiply realisable

Instances of exactly the same type of chess virtual machine could be implemented in different physical systems, e.g. old and new sparcs, pentiums, vaxen, etc. The VM is multiply realizable.

Virtual machines can endure across physical changes

A running instance might have its physical basis altered (e.g. memory replaced) and remain the same chess process: for instance, garbage collections and paging algorithms can relocate the physical implementation.

More on“fully grounded” (realised/implemented)

RECAPITULATION: Phenomena of type X are fully grounded in, or realised (implemented) in, phenomena of type Y *if and only if:*

(a) Necessity condition: phenomena of type X *cannot exist without* certain entities and processes of type Y.

(b) Sufficiency condition: those entities and processes of type Y *are sufficient for* the phenomena of type X to exist.

(Multiple realisability implies that quite a broad subset of phenomena of type Y will sometimes be sufficient to produce X – but there are constraints, mentioned below.)

The *kind* of sufficiency referred to in (b) needs clarification.

If I have a computer running a certain chess virtual machine, we normally think of the physical machine as implementing the chess machine, and the precise state of the physical machine as being sufficient for the chess VM (though not necessary as there are alternative implementations).

But the sufficiency depends on certain conditions, e.g. that the power supply is not turned off, that the transistors in the CPU continue to operate properly, there is no destructive bombardment with gamma rays, that no hitherto unknown physical phenomena disrupt the performance of the machine, that no software intruder tampers with the operating system or some of the machine code, etc.

The sufficiency depends on a range of conditions that may be impossible to specify exhaustively. Nevertheless it is a more than ‘accidental’ sufficiency.

More examples of virtual machines

Recent extensions to our ontology come from Software Engineering, Computer Science and AI. Chess is just a special case.

Software engineers commonly think about objects, events and processes in computational virtual machines.

EXAMPLES:

- a word processor running on a computer
- a spreadsheet propagating values
- the operating system on my computer (linux)
- a parser building a parse tree
- a graph-matcher comparing two networks
- a compiler converting source code to machine code
- a chemical plant control system
- most of the internet (e.g. an email message travelling from you to another country).

Some virtual machines are implemented in others.

Not all are directly implemented in physics: there can be layers of implementation.

Even tangled networks, in addition to simple implementation hierarchies.

These virtual machines can contain:

- enduring objects, properties, relationships, events, processes,
- causal interactions: e.g. too much email can slow down networks.

NOTES on grounding of a VM (1)

- **Multiple realisability of a VM**

  A specific thing of type Y that produces a particular instance of type X *may not be necessary* for X: alternative objects and processes of type Y could also suffice to produce that X. The Y-mechanisms could change even while X continues running, e.g. in a memory upgrades in a 'hot-swap' system.

- **The sufficiency is not a subjective matter**

  That an instance of X exists when the appropriate instance of Y exists is a matter of fact. We constantly depend on such facts when we use computers, e.g. to control airliners.

- **The sufficiency need not be *logical* (in the narrow, precise sense)**

  The specific Y phenomena that suffice for X need not be *logically* sufficient.
  I.e. it may not be possible to *prove from definitions and logical relationships* that if those Y phenomena exist then the X ones will.
  E.g. it is impossible to derive by logic alone the description of a chess virtual machine (CVM) from a physical description of a machine that implements CVM.

- **The sufficiency is not a contingent matter**

  The link between Y and X is intrinsic to the structures of the processes, not just an empirical correlation. However, hardware faults, bombs, and other extreme cases can disrupt sufficiency – it holds only in certain contexts.

  Nobody may know enough to specify the contexts without circularity, since the variety of ways in which a physical machine can be made to alter its behaviour always goes beyond what is known to science at any time.

NOTES on grounding of a VM (2)

- **Mathematical derivations of VMs from physical descriptions**

  An idea from Matthias Scheutz

  ('When physical systems realize functions...' *Minds and Machines*, 9:161–196, 1999):

  If a virtual machine is defined entirely formally, e.g. a chess VM or a Lisp VM, we can (a) start from a detailed physical description of the implementing machine, expressed mathematically, then (b) derive a more abstract mathematical description leaving out many physical details, then (c) show a structural correspondence with the required VM (e.g. modelling rules of chess).

- **Mathematical derivability is not always possible**

  Such derivations are not not possible for VMs with an ontology introduced as part of an explanatory theory of some part of reality, rather than an invented game or formally specified computational mechanism – e.g. a biological or mental ontology.

  These are implicitly defined in part by their causal links with a larger ontology. E.g. the percepts, beliefs, intentions, in an animal or robot go beyond mere formal processes. However the notion of "causation" is very hard to analyse.

- **Some of the implementation may be extended in space and time.**

  As argued in P.F. Strawson's Individuals 1959 (explained below) it may be impossible for individual X to intend to refer to individual Y (e.g. Julius Caesar or the future death of our sun) without making use of an extended chain of causal connections in the environment, without which Y would not be identified as the referent.

Hierarchies of implementation

- **Multiple VMs may be realised by the same physical machine**
  When an instance of X exists the lower level processes of type Y that realise it may simultaneously realise virtual machines at different levels of abstraction: e.g. a sparc virtual machine, a lisp virtual machine, a chess virtual machine, etc.
- **Virtual machines may be implemented in other virtual machines**
  If a chess virtual machine is implemented in a Lisp VM, then the implementing ontology is not physics: VMs in computers often use several layers of implementation above the physical level(s).
  It is likely that mental objects, events, processes, causal interactions, are likewise not directly implemented in brains, but use intervening virtual machines with their own non-physical ontologies – e.g. information-processing mechanisms.
- **Implementation does not require simple structural correspondence**
  Many examples from software engineering show that correspondences between implemented and implementing ontologies may contradict naive assumptions.
  - A virtual machine may have more components than the physical machine implementing it, for instance if sparse arrays or lazily evaluated functions are included,
  - Interpreters, unlike most compiled programs, produce dynamically changing correspondences.
  - The generalisations linking causes and effects need not map onto generalisations linking physical conditions and physical effects.

  (More examples refuting common intuitions are needed.)

Information processing virtual machines

- Philosophers discuss supervenience as a relationship between mental and physical properties.
- Software engineers and AI researchers are particularly interested in the design and implementation of virtual machines that are *information processing* machines,

  i.e. machines that acquire, store, analyse, transform, combine, search for, create, or use information of various kinds, including

  – information about their environment
  – information about their own information processing.

What we know, or can learn, about the implementation of such information-processing VMs is crucial for understanding how mental phenomena supervene on physical phenomena.

A full study of this topic requires analysis of different *architectures* for information processing virtual machines, including architectures in which different sorts of virtual machines (reactive, deliberative and meta-management processes) run in parallel, interacting with one another.

Grounding is not reduction: X and Y things are not identical

If Xs are fully grounded in Ys does that mean Xs are really nothing more than Ys? Does it imply that every entity of type X is identical with some entity of type Y?

People, including philosophers, often argue about what is identical with what, as if it were a question of *fact*. But ancient philosophers noticed problems with this.

Is it a matter of fact that

- This river is the same river as the one I stepped in yesterday, despite the fact that all the water is new?
- This axe I give to you is the one you lent me, despite the fact that the blade was replaced and then later the handle was?
- You are the same thing as the helpless baby that was given your name many years ago, despite the fact that you are so physically different and almost all the physical atoms have been replaced?

Suggestion: identity is not a factual relation but a matter of ethical, social, political, practical convenience: we could carve up reality in different ways.

Does a chess process survive garbage collections?

New technology, e.g. Star-trek transporters, can lead to changes in the criteria for identity. Compare something that made two copies of you.

See also: “New bodies for sick persons: Personal identity without physical continuity (1971)

`http://www.cs.bham.ac.uk/research/projects/cogaff/07.html#702`

Virtual/physical identity?

- We can ask questions about a RVM without thereby specifying which physical components are referred to.

  E.g. how long would that search process have taken if we had replaced the memory with the latest technology?

- If virtual machines were identical with their physical realisers then:
  - Implementation would be symmetrical: but it isn't:

    (X can be grounded in, or realised in Y, without Y being realised in X.)

  - The same predicates ought to be applicable to X instances and Y instances, but they are not.

    E.g.

    – The predicates of chess (e.g. 'pawn', 'threatened', 'illegal', etc.) are not applicable to the electronic components and processes of a chess computer, and the chess entities in the virtual machine do not have physical properties such as mass and electrical resistance.

    – Font CHANGES and speelling mistaiks can exist in a document in a word processing virtual machine, but nothing in the physical machine is a font or a spelling mistake, or even a word of English, mis-spelt or not.

Construing implemented entities as identical with their realisers solves no problems.

There is no *factual* question whether implemented entities and their realisers really are identical or not.

Just saying they are identical in order to avoid having to explain the relationship has no explanatory power.

- Such an identity claim does not say anything about how an entity that persists across changes in its implementation can have causal powers – possibly changing powers, e.g. when a learning system or an adaptive mechanism gets better at its job.
- To explain such powers and how they change we need to refer to the enduring entity which has the powers, the information and mechanisms it uses (including rules or algorithms) and what changes occur within the entity, for instance acquiring new information, or improving its rules, or discovering a new concept.
- Consider the difference between what computer science students have to learn about operating systems in order to understand them, and what a physicist could tell such students about the underlying physical processes: the latter would not help a student design, implement, debug, maintain, or use an operating system.

  Why not?

Causation in virtual machine ontologies

Our common sense view of the world includes 'high level' ontologies that involve *causal interactions* between components of a virtual machine. E.g.

- In a compiler, a parser could interact with

  an error handler and

  a code generator,

  among other things.

- In an operating system the process scheduler

  interacts with memory manager

  and interrupt handler, among other things.

All this talk of "interaction" presupposes a notion of "causation".

Analysing the concept of causation is probably the hardest problem in philosophy.

[[A longer discussion would need to say something about Humean theories of causation and why they won't suffice – saying that X causes Y is not just saying that 'X precedes Y' is an instance of some true generalisation.

The link with counterfactual conditionals (what would have happened if) goes beyond generalisations.

See also presentations on varieties of ways of understanding causation in humans, other animals and machines, here:
`http://www.cs.bham.ac.uk/research/projects/cogaff/talks/wonac`]]

Causation and Computer Science

Causation is irrelevant for (most of) mathematical computer science

A computation can be regarded as just a mathematical structure (possibly infinite), something like a proof.

Such "computations" need not occur in time, nor involve causation:

- E.g. a Gödel number encoding a sequence of Gödel numbers can be regarded as a computation. It is a timeless, static model that accurately reflects all mathematical properties of the computation.
- Talking about 'time' in this context is just a matter of talking about position in a (possibly infinite, possibly branching) ordered set.
- State transitions are then not things that happen in time, but relations between adjacent components in the ordered structure.
- The notions of space complexity and time complexity in theoretical computer science refer to purely syntactic properties of a 'trace' of a program execution: another mathematical structure.

Perhaps we should say: theoretical computer science does not study computations, only mathematical models of computations.

Do such models capture important facts about causation, and the possibilities of causal interactions with an environment?

How?

Implementation as a mathematical relation

To say that a system X is mathematically implemented in or realised in Y is simply to say that a structural mapping exists.

Either

- X just is a subset of Y

  or

- There is a mapping from the whole of X to some part of Y which preserves certain relationships. In particular the ordering relations called state transitions in X are mapped into suitable relations in Y.

This is just a structural mapping: like the mapping of the set of positive integers onto the set of multiples of 5.

In this sort of implementation

- there is no causation
- nothing happens: only mathematical structures exist
- No energy is used.
- The issue of *reliability* does not arise, as it does for physical implementations.

Notes on mathematical implementation of X in Y

- X may be able to be mapped into Y in more than one way.
  (Multiple realisability)
- The relationship is sometimes symmetric:
  X and Y are implementable in each other (e.g. a Turing machine and a production system interpreter) if mappings exist in both directions.
  For a *working* implementation such symmetry is not possible.
- If Y implements an *interpreter* for X, things are rather more complex.
  The mapping is then not just a relationship between the two structures specified independently of their state, but ‘unfolds’ in a sequence of ‘state transitions’ within Y.
  - I.e. the mathematical model needs to represent different sequences of states of X and Y and the mappings between them, instead of being simply a mapping between X and Y.
  - If X is a non-terminating program which can be affected by external input, the model will have to take account of different possible inputs at different times, leading to mappings between branching infinite sequences.
  - However, the code for the interpreter abbreviates all of that, in something like the way axioms and inference rules abbreviate a set of proofs generated by those axioms and rules.

Causation in Computer Applications

People who use computers require more than structural mappings: the machine must be able to *do* things.

There must be causal interactions, happening *reliably* in real time.

So, for software engineers, robot designers, and computer users,

computation involves a process in which

– things exist
– events and processes happen
– what causes what (e.g. an effect of a bug) can be important

Software engineers want to make some things happen and prevent others

– Some of what happens, or is prevented, is in the virtual machine (e.g. preventing one user's program from accessing other user's files.)
– Some of it is in the environment, under the control of the virtual machine (e.g. an airliner landing without crashing)

So the engineering notion of implementation/realisation goes beyond the mathematical notion of structural mapping. It requires production of causal interactions in the virtual (implemented) machine, and usually its environment.

Moreover, energy is consumed (dissipated).

Mathematical vs useful implementation

In a purely mathematical implementation

- There is no time (though there may be an abstract mathematical model of time)
- There is no causation
- No energy is consumed
- It does not cause anything in the world to change
- It supports no counterfactual conditionals about what would happen if, since nothing happens.
- It cannot be unreliable (e.g. subject to unknown causes of failure)

In contrast, in the physical implementation of a useful virtual machine

- Processes occur in time and objects endure in time
- There are causal interactions
- Energy is consumed when this happens
- There are many true counterfactual conditionals:
  e.g.
    if this pawn had been moved, that piece would have been captured,
    if this variable's value had been less than 10, the stack overflow would not have happened
- The system can (sometimes) be used to control part of the physical world
- Issues of reliability of the implementation can arise.

Concurrency and reliability

In some safety-critical systems a VM is implemented in three different physical systems which are run in parallel with constant checking that their decisions agree. If one of the systems has a flaw the chances are that the other two do not have the same flaw at the same time, so a simple vote can determine which decision is wrong.

- From a mathematical point of view:
  There is no difference between such a system, and a system where all three processes are run via emulations on a single time-shared CPU which is three times as fast as each of the others.
- From the engineering point of view
  There is a significant difference in their causal powers. The system using three physically distinct processors is more reliable, especially if the processors are separated in space. There is an even more subtle difference if the three CPUs are not fully synchronised.

We can build a mathematical model of both systems, but in order to explain why the second is more reliable we are forced to take account of additional factors, such as possible disruptive interactions with the environment, or flaws in physical materials.

VMs are not defined by input-output relations

The philosophical functionalist view of mind treats mental states and entities as ‘functionally’ defined.

This is normally assumed to imply that they are defined in terms of some particular relationships between inputs and outputs. BUT

- A virtual machine can run without any inputs or outputs (e.g. computing primes).
- Different virtual machines can have the same input-output relations
- The ‘defining’ causal relations of a VM involve *internal* states, events and processes
- The actual implementation need not have input and output transducers with sufficient bandwidth and adequate connections to check out all the distinct VM states and processes that can or do occur.

So a virtual-machine functionalist analysis of mental concepts is very different from the conventional functionalist analysis.

NOTE:

States that cannot be identified through input-output relationships might in principle be capable of being identified through direct measurement and observation of the internal physical states and processes.

But this may be physically impossible without disrupting the system.

There may also be some states that are only describable using an ontology that the machine has generated for itself to describe and control itself: these descriptions may be intelligible only to the machine.

See Sloman and Chrisley (2003) *Virtual machines and consciousness*,
`http://www.cs.bham.ac.uk/research/projects/cogaff/03.html#200302`

Non-redundant multi-level causation

All this implies that some events, including physical events like a valve being shut in an automated chemical plant, may be multiply caused - by physical and by virtual machine events.

NOTE:

This is not like the multiple causation discussed by Pearl and others where removing one of the physical causes would have left a sufficient set of causes

(e.g. death from cancer and heart disease, or being killed by several members of a firing squad).

VM causation is in an important sense non-redundant: there's no way to simply remove the VM cause of effect E without changing the world in such a way that the physical world ceases to cause E.

Of course the VM cause can be replaced by another: if the bishop had not captured the pawn the rook would have.

There is also the ever-present possibility of total destruction of the whole system, e.g. by a bomb, or a hardware fault that disrupts the virtual machine.

VM counterfactuals depend on a 'normality' condition.

A good implementation tends to preserve normality, e.g. by error detection and error compensation. But there are always limits.

Virtual Machine events as causes

Most people, including scientists and philosophers in their everyday life, allow causal connections between non-physical events. E.g.

– Ignorance can cause poverty.
– Poverty can cause crime.
– Crime can cause unhappiness.
– Unhappiness can cause a change of government.
– Beliefs and desires can cause decisions, and thereby actions.
– Detecting a threat may cause a chess program to evaluate defensive moves.

How can that be, if all these non-physical phenomena are *fully implemented* in physical phenomena?

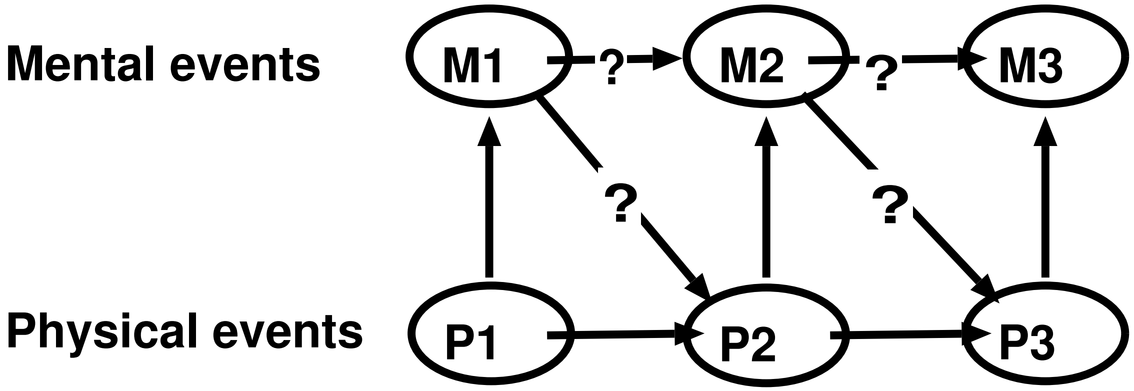
For, unless there are causal gaps in physics, there does not seem to be any room for the non-physical events to influence physical processes. This seems to imply that all virtual machines (including minds if they are virtual machines) *must be epiphenomena*.

Some philosophers conclude that if physics has no causal gaps, then human decisions are causally ineffective. Likewise robot decisions.

This is a seriously flawed argument, but exposing the flaws means solving the problem of analysing our concept of causation.

Must non-physical events be epiphenomenal?

Consider a sequence of virtual machine events or states M1, M2, etc. implemented in a physical system with events or states P1, P2, ...



If P2 is caused by its physical precursor, P1, that seems to imply that P2 cannot be caused by M1, and likewise M2 cannot cause P3.

Moreover, if P2 suffices for M2 then M2 is also caused by P1, and cannot be caused by M1. Likewise neither P3 nor M3 can be caused by M2.

So, according to this reasoning

the VM events cannot cause either their physical or their non-physical successors.

E.g. poverty cannot cause broken windows or crime.

This would rule out all the causal relationships represented by arrows with question marks in the diagram, leaving the M events as epiphenomenal.

The flaw in the reasoning?

THIS IS HOW THE ARGUMENT GOES:

IF

- physical events are physically determined

  E.g. everything that happens in an electronic circuit, if it can be explained at all by causes, can be fully explained according to the laws of physics: no non-physical mechanisms are needed, though some events may be inexplicable, according to quantum physics.

AND

- physical determinism implies that physics is 'causally closed' backwards

  I.e. if all caused events have physical causes, then nothing else can cause them: any other causes will be *redundant*.

THEN

- no non-physical events (e.g VM events) can cause physical events

  E.g. our thoughts, desires, emotions, etc. cannot cause our actions.
  And similarly poverty cannot cause crime, national pride cannot cause wars, and computational events cannot cause a plane to crash, etc.

ONE OF THE CONJUNCTS IN THE ANTECEDENT IS INCORRECT. WHICH?

It’s the second conjunct

Some people think the flaw is in the first conjunct:

i.e. they assume that there are some physical events that have no *physical* causes but have some other kind of cause that operates independently of physics, e.g. a spiritual or mental event that has no physical causes.

The real flaw is in the second conjunct:

i.e. the assumption that determinism implies that physics is ‘causally closed’ backwards.

Examples given previously show that many of our common-sense ways of thinking and reasoning contradict that assumption.

Explaining exactly what is wrong with it requires unravelling the complex relationships between statements about causation and counterfactual conditional statements.

A sketch of a partial explanation can be found in the last part of this tutorial:

`http://www.cs.bham.ac.uk/~xs/ijcai01`

‘Emergent’ non-physical causes are possible

Problems with the ‘monistic’, ‘reductionist’, physicalist view that non-physical events are epiphenomenal:

- It presupposes a layered view of reality with a well-defined ontological bottom level. IS THERE ANY SUCH BOTTOM LEVEL?
- There are deep unsolved problems about which level is supposed to be the real physical level, or whether several are.
- It renders inaccurate or misleading much of our indispensable ordinary and scientific discourse, e.g.
  - Was it the government’s policies that caused the depression or would it have happened no matter which party was in power?
  - Your anger made me frightened.
  - Changes in a biological niche can cause changes in the spread of genes in a species.
  - Information about Diana’s death spread rapidly round the globe, causing many changes in TV schedules and news broadcasts.

As previously remarked

Saying that the non-physical phenomena are identical with physical ones

(a) does not explain anything,
(b) contradicts the asymmetry in the realisation relation.

But most importantly

- The argument that virtual machine events cannot have causal powers is based on ignorance of how actual implementations of virtual machines work, and the ways in which they produce the causal powers, on which so much of our life and work increasingly depend.

  More and more control systems depend on virtual machines that process information and take decisions.

- The ideas presented here are only a first draft attempt to clarify these topics:

  Detailed explanations would require analysis of how compilers, interpreters, operating systems, and digital circuits interact, and different ways of linking them to sensors and effectors.

- There is much more that is intuitively understood by engineers which has not yet been clearly articulated and analysed.

Strictly the people who use this kind of understanding should be called craftsmen rather than engineers.

This is a special case of the general fact that craft precedes science.

Philosophers and psychologists need to learn the craft, and the underlying science, in order to avoid confusions and false assumptions.

A more general notion of supervenience

Philosophers normally explain supervenience as a relation between properties: e.g. a person's mental properties supervene on his physical properties.

'[...] 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', repr. in: *Essays on Action and Events* (OUP, 1980).

In contrast we are concerned with a relation between *ontologies* or parts of ontologies, not just properties.

The cases we discuss involve not just one object with some (complex) property, but large numbers of objects enduring over time, changing their properties and relations, and interacting with one another: e.g. data-structures in a virtual machine, or thoughts, desires, intentions, emotions, or social and political processes, all interacting causally.

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

An ontology supervenes on another ontology if there cannot be a change in the first ontology without a change in the second.

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 – omitting subjective aspects of having a temperature, etc..)
- pattern supervenience
  (e.g., the supervenience of a rotating square on the pixel matrix of a computer screen, or supervenience of various horizontal, vertical and diagonal rows of dots on a rectangular array of dots.)
- mereological, or agglomeration, supervenience
  (e.g., possession of some feature by a complex whole object as the result of a summation of features of parts, e.g. the supervenience of a pile of sand with a certain mass on a collection grains of sand each with its own mass)
- mechanism supervenience
  (supervenience of a collection of interacting objects, states, events and processes on some lower level reality, e.g., the supervenience of a running operating system on the computer hardware – this type is required for intelligent control systems)

We are talking about mechanism supervenience.
The other kinds are not so closely related to implementation.

I believe philosophers have ignored mechanism supervenience.

The Physical Realization Theory of mind

The PRT states:

The mental is realised, or fully grounded, in the physical

or put differently,

if a collection M of mental objects/properties/states/events exists they have to be fully grounded in some physical system.

I.e. There must be some physical system P such that

– The existence of P is sufficient for M

– If neither P nor any other physical system sufficient for M exists then M does not exist (i.e. there cannot be disembodied mental objects events, processes, etc.)

P need not be unique: multiple realisation is possible

All this fits the engineer’s notion of implementation, though engineers know a lot more about the details of how various kinds of implementations work. (They understand mechanism supervenience.)

How does it relate to philosophical ideas about supervenience?

NOTE

The physical realisation thesis is probably what Newell and Simon meant by their “physical symbol system” hypotheses.

Their terminology is very misleading because most of the symbols AI systems deal with are not *physical* symbols, but *symbols in virtual machines*.

They should have called it
  the physically implemented virtual symbol system
hypothesis.

Realisation (grounding) entails supervenience

IF

M is fully grounded in P (as defined above),

THEN IT FOLLOWS LOGICALLY THAT

no feature, event, property or relation of M can change unless something changes in P.

Note: this would not be true if some changes in M are produced by spirits or souls that are not implemented in any physical systems.

PROOF:

If M is fully implemented in P, then every facet of M is explained by P.

If there's a change in M, that would introduce a new facet not explained by P (since P was not sufficient for it before the change).

Therefore: something must have changed in P which explains the change in M.

I.e.

Physical realisation entails supervenience.

Difference in physical machines does not imply difference in VMs, but difference in VMs implies physical differences.

(Actually we need to discuss more cases — another time.)

Surprising aspects of implementations.

– The relation may be “partial” (if considered at a particular point in time) if there are entities in the VM which do not correspond to physical parts.

  A partial implementation of a large array might have cells that will be created only if something needs to access their contents. A collection of theorems might exist *implicitly* in mathematical virtual machine, but not be explicitly recorded until needed. It may be partial in another way if there are physical limitations that prevent some VM processes occurring, e.g. memory limits.

– A VM may contain a huge ‘sparse array’ with more items in it than there are electrons in the computer implementing it.

  (e.g., cells containing items with some computable or default value are not explicitly represented)

– Individual VM entities may map on to physical entities entities in different ways.

  (e.g., some VM list structures might be given distinct physical implementations, while others share physical memory locations because they have common ‘tails’.)

– In a list-processing language like Lisp or Pop-11, there can be two lists each of which is an element of the other, whereas their physical counterparts cannot be so related.

– If the virtual machine uses an interpreted programming language, then the mapping between high level objects or events and physical entities is constantly being (re-)created by the interpreter.

– A learning, self-optimising machine may change the way its virtual entities are implemented, over time.

Non-features of mechanism supervenience

Sometimes conditions are proposed for supervenience that are violated by examples from computing.

E.g. the following must be rejected as *necessary* for supervenience.

- Components of a supervenient system must correspond to fixed physical components which realise them:
  Counter-examples were mentioned above.
- *Types* of VM objects or events must always be implemented in the same *types* of physical objects or events.
  Refuted by the recent history of computing: a running system can have its memory replaced piecemeal by newer faster physical components.
- The structural decomposition of a VM (i.e. the part-whole relations) must map onto isomorphic physical structures: NO.
  Counter-examples were given above.
- The same temporal relations hold between VM events and physical events as hold between physical events.
  No. E.g. VM time may be discrete while physical events occur in continuous time.

All this mean that searching for so-called “neural correlates of mental events requires great conceptual clarity and care.

More likely, it is just misguided: we should be looking for implementations, not correlates.

Some VM states need external objects

VM events may depend on, be implemented in, “external”, even remote, physical events.

- Information in VM X about the external object O can switch from being accurate to being inaccurate simply because O has changed.
- Whether a database is up to date, or complete, is not determined solely by the contents of the physical machine that implements it.
- Reference to particular spatio-temporally located objects requires some external relationship with those objects.

  E.g. for a VM to contain information about the particular individual Julius Caesar, it must have some sort of causal connection with that object, e.g. through informants, or records, etc.

  Otherwise the VM contains only a description of a possible object *similar* to the tower, or to Caesar. (Strawson 1959)
- So not ALL mental states of a person or a robot able to relate to an environment are fully implemented *within the body* of that person or robot. Supervenience need not be a “local” relation.

Studying only relations between mind and brain ignoring the physical (and social) environment (methodological solipsism), is a mistake.

More on Causation, Virtual Machines and Levels

These notes are incomplete: further topics to be discussed.

- Causation and counterfactuals
- Varieties of emergence
- Physical determinism does not imply backward causal closure
- Our normal concept of causation allows over-determination of effects.
- This explains how both virtual machine phenomena and the physical substrate can be causally efficacious.
- How can virtual machine states have semantic content?
- Mind and computation (different notions of computation).
- The 'design stance', directed to virtual machine architectures, makes the 'intentional stance' unnecessary.
- How machines can have qualia: requires an appropriate virtual machine architecture (discussed in Sloman&Chrisley 2003).
- How we can have theoretical concepts referring to virtual machines.
  See `http://www.cs.bham.ac.uk/research/projects/cogaff/talks/#models`

See also notes on the concept of "information"

`http://www.cs.bham.ac.uk/research/cogaff/talks/#inf`
`http://www.cs.bham.ac.uk/research/projects/cogaff/misc/whats-information.html`

These notes may be revised and extended.

Last updated October 8, 2007

I should add some references

E.g. books and papers by Jaegwon Kim, Donald Davidson, Matthias Scheutz, and many others.

In particular

Ansgar Beckermann, (1997),

“Property physicalism, reduction and realization,
`http://www.uni-bielefeld.de/philosophie/personen/beckermann/prpph_ww.pdf`

He doesn’t mention running virtual machines, alas.

Counterfactual conditionals

This analysis of causation is incomplete until we find a good analysis of conditionals, including counterfactual conditionals.

However that is a problem that needs to be solved independently of our analysis of causation.

The notion of “what would happen if” is more general than the notion of causation, and is presupposed by our concept of causation.

[To be completed]

Biological and artificial VMs

Need to add things about the diversity of VMs produced in evolution.

See the suggestions about evolution of self-monitoring VMs in

http://www.cs.bham.ac.uk/research/projects/cosy/conferences/mofm-paris-07/sloman

The special case of humans

Biological virtual machines that

– grow themselves

– learn to monitor themselves

– develop theories about other VMs (in other organisms)

etc.

Currently AI is far behind this.

[To be completed]