

"What Does a Computer Simulation Have to Reproduce? The Case of VMWare"

Introduction

The VMWare products and other virtualization and emulation technologies are not new (VMWare itself dates from the 1990s; emulation in a way dates from Turing's epochal 1936 paper. After all, what is the goal of what is now called the "Universal Turing Machine" but a computer that can simulate others?) However, the recent marketing and press blitz concerning "cloud computing" often invokes virtualization technologies as one approach to the supposed new model of providing computing resources.

But, simulations are never perfect, at least if they have to interact with something (a human, for example) outside the simulation¹. One way in which this (seemingly truistic) point has unforeseen consequences has been dealt with by Douglas (2008, 2010). In the following, I would like to move from the primarily epistemological questions of the aforementioned work into metaphysical considerations. I will address the question of my title, viz. what counts as an appropriate simulation of a system, with attention to the special case of simulation of computers by computers. Before I engage this subject, I

¹ There is no physical way in which even a nomologically perfect simulation would run with the same "time flow" as what is going on on the outside, due to the finite speed of any processing of the simulation. (Contrary to all known physics, one could have a machine that exactly "keeps up" only if it were infinitely fast.) Subsequently, any interaction of the simulation with the outside world will be slightly different from "the real thing" interacting in "the same way". When this is significant is, of course, a point at issue.

would first like to motivate this semi-metaphysical question both philosophically and practically, as befits a contribution to a conference on Computing and Philosophy.

First, then, Paul Churchland (2007 [2002], pp. 8-9), emphasizes in original:

“More generally, the idea that a machine, any machine, might be programmed to ‘simulate’ a [von Neumann] machine in particular makes the mistake of treating [von Neumann machine] as if it were itself a *special*-purpose piece of software, rather than what it is, namely an entirely *general*-purpose organization of hardware. In sum, the brain is not a machine that is capable of ‘downloading software in the first place, and a [von Neumann] machine is not a piece of ‘software’ fit for downloading in any case.”

Ignoring the bit about brains, which is off topic for the present paper, this remark seems at first glance to contain a mistake. A virtual machine (or other computer simulation of a computer by a computer) seems (*prima facie*) to be exactly a case of a von Neumann machine as a piece of software - after all, many organizations use them as if they were one, buy disks for them, etc. One might ask whether or not a class of entities can “exist” as both software and hardware, a question which will be implicitly discussed in what follows.

But it was not Churchland’s remarks that motivated me to write this paper: in fact, I was rereading his article **after** I started exploring the ideas that formed this paper, and after the work related issue discussed next was rendered moot by a management decision.

Practically, the work in the present paper in a way stems from a very interesting policy question I was provoked into considering. The author works for a Canadian federal government department which has strict policies about what software certain employees may use on official laptops and how to “harden” said laptops and their operating system, software, etc. against unwanted attack, misuse etc. These policies are in flux at

the time of writing due to various hardware additions being considered for the systems. One forthcoming use for some of our laptops is to run VMWare Server. So, I was confronted with the policy question: is a virtual server on a laptop subject to the same policies as the laptop itself; is it a machine of the appropriate kind, or of an analogous kind? If so, how does one implement the appropriate policies? Hence my question: what are simulations, and in particular, what are simulations of computing systems? Another practical question related to our present concern is virtual machine (hereafter, VM) detection / counterdetection. Malware (viruses, etc.) might wish to detect presence of a VM in order to defeat the “isolation” provided. As pointed out by Liston and Skoudis (2006), detection of virtualization by software has uses both for computing security and in compromising said security.

In what follows, four ways of understanding what an appropriate computer simulation of a system is will be addressed, with focus on the narrow subcase of what counts as an appropriate simulation of another computer. These ways are what I have called “naive functional equivalence”, “the gamer answer”, “isomorphism”, and what I will suggest is where to *look* for a possible answer, namely: “same laws”.

Section 1 - Getting the Naive Answer out of the Way - Functional Equivalence

The “naive” answer to my question is simply the one might call the “logician’s answer”. (Not that logicians would endorse the answer if they know anything at all about computing or indeed the world generally, but for lack of a better phrasing I’ll make use of it.) The logician’s answer is simply that what one wants in a computer simulation of a computer is the same output given the same input. This has several obvious flaws, the

most important of which simply are timing considerations. It is true if one pretends that one's workstation has an unbounded amount of memory (or disk space) that it can be described as a universal Turing machine. However, there are three critical flaws with this: a simulation is usually created for some sort of "practical" purpose, even if it is just to play a game, relive one's childhood or the like. Subsequently, the timing consideration is possibly of some importance. Clearly, a simulation which interacts with the world in some way is going to be slower than the real thing (as mentioned in above, footnote 1). One might argue that a slow simulation is still a simulation, so I will leave that aside for now.

A more crucial problem concerns the user interface of the program. The claim is this: the user interface of a piece of software is as much a "part" or "aspect"² of it as anything else. Searle's (1990) famous wall is **not** running Wordstar, for one cannot press control-K, D to save a file using it. Note that a simulated XT computer running Wordstar still allows one to use this interface element. However, by somehow construing Wordstar and the simulated IBM PC XT it might be running on as a gigantic finite function obliterates this essential feature. (I take it as given that we want, at least sometimes, to **use** simulated hardware and relevant programs.)

Third, what might be called the "protocol" for understanding a simulation is necessary in order to couple it to other systems. This protocol actually consists of two layers of interest. One is the physical mapping of physical states to abstract states (e.g., the

² "Aspect" is vaguer but also in a way less dangerous in terms of misunderstanding. I find the term "part" more natural, but it carries mereological connotations I want to avoid at least in the present work.

famous set $\{0,1\}$). The second is the relevant encoding scheme for numerals, matrices, World of Warcraft characters, or whatever data we may be interested in. See Douglas 2003 for more on the necessity of this two pronged notion of “protocol” for understanding computing and its models.

From the above considerations, we thus conclude that the “logician’s answer” is unsatisfactory, though contains a grain of truth.

Section 2 - A Gamer’s Answer

One use of a computer emulator (or virtualization software) is to play games. The existence and popularity of programs like MAME (the Multi Arcade Machine Emulator: <http://mamedev.org/>), which can be used to play hundreds of arcade video games is evidence of this. Unsurprisingly, the gamer’s answer to our metaphysical question is a “practical” one. The gamer’s answer is thus: “One should simulate as much of a system as required in order to have fun playing the game one wants to play on it.”

I first note that this answer is, of course, subjective - it makes the notion of a simulation purely “what we say it is”. An Apple II emulator is good enough if it (for example) plays *Floppy* successfully. In a way, one could call this a species of simulation antirealism. It

has the strength that not all simulation need be “perfect”. *Floppy* is still a modestly interesting game, even if there is a way in which the emulation fails³.

Unfortunately, the gamer’s approach turns the intuitions of expertise (assuming, for the sake of charity, that there are experts for all possible classes of computer programs) into principles without specifying what they are. But I have no wish - at least initially - to turn metaphysics into a purely psychological investigation.

Section 3 - Isomorphism of States and Events

The existence of an isomorphism between a simulation’s states and those of the “real thing”, and similarly for all its events (state transitions; see Douglas 2001) is at first glance a promising avenue to answer our question of the present paper.

Analysis of this proposal should begin by a remark about what it is **not** proposing and why. That is, for two reasons, the proposal of this section is **not** that of Popek and Goldberg (1974 - hereafter: P&G). One is that P&G’s results, however foundational, do

³ Coincidentally enough, it also fails in timing related matters, at least in one emulator as can be seen with the classic game, *Floppy*. *Floppy* is an arcade style game which involves the typical “race against time” activity common to many games. On screen while playing is a timer; the player is rewarded, etc. based on the state of the timer at the end of each level of play. On a real Apple II, the timer counts down (visibly) in blocks of ten time units. By contrast, from time to time, using Virtual][(an Apple II emulator for Mac OS X), one can see the timer has not counted down by tens. This slight lack of fidelity does not generally detract from the gameplay, unlike (say) having a poorly emulated graphics mode with bad colour, etc. For example, in *Floppy*, the gameplay would be considerably more difficult (to the point of almost making the game impossible) if somehow the orange and blue colours were indistinguishable.

not address questions of timing and I/O, which have been argued elsewhere (Douglas 2008 and section 2, above) to be important for discussing successful simulation.

Secondly, P&G's results only apply to same-architecture virtualization - for example: IBM 360 virtualizing a smaller (less RAM, say) 360. It is my intention, if possible, to discuss more general principles. Second, because of the questions of timing, etc. I believe it is also necessary to consider the *sub instruction* level state. In P&G, only the characteristics of the instructions themselves are considered. Since on modern CPUs there is another "layer" of recognizable computational events (those involved in the so-called "microinstructions" - see, e.g., Hennessey and Patterson 1998), P&G's pioneering paper is not directly relevant.

But microinstructions can provide another important lesson on our theme. Namely, they show that "state" and "event" apply at many levels of analysis. In particular, to discover an isomorphism, microinstructions tell us that one would have to choose between microstate and the instruction level state to investigate. But then, a sort of "indifference argument" (Makin 1993) would apply. Why would one think that simulation is successful if of one level rather than another? Barring any additional principle, it would seem that this decision would be arbitrary. In the next section, then, I propose how to make this decision with less arbitrariness.

Section 4 - Obeying Same Laws

Invoking laws of nature to explain what VMWare Server *et al* have in common with real

hardware looks promising. The motivation for this approach has two roots. One is the idea that perhaps, under some circumstances, a simulation shares the (a?) same kind as the thing simulated. This principle is not always true; but I will assume it is correct (*ex hypothesi*, at least) for the kind “computer” or something near to it. After all, the simulation of a computer does have many properties of a real one - we use it do much of the work of a real one, etc. The other root is the metaphysics of David Wiggins (2001), which suggests that what individuates substances and substance kinds are relevant laws. Thus my thesis is: a simulation of a computer by another computer is successful to the extent that the simulation “obeys” the same laws.

This view recaptures three key insights from the previous sections. One, since a thing’s laws are (in part) how it behaves in interaction with others, they would include those of interface items, etc., solving both the “Searle’s Wordstar” problem and the “protocol” problem in section 1. Two, since, in principle, interaction with a human is also governed by laws, the gamer can be happy too for much the same reasons. This point is admittedly fairly weak. Three, laws specify a level at which isomorphism (or some other structural relation) of states and events is relevant: the level to which the relevant laws apply. It **might** turn out that in order to satisfactorily emulate computers one needs laws of physics, etc. but at present the success of emulation products suggests that this cannot be right.

The view I defend can be further fleshed out by addressing two problems with it. The first problem is a family of metaphysical problems similar to those discussed by Douglas (2009). Concretely: does something have to be of the same ontological category to

obey the same laws? For example, if a computer is a *thing* (in the sense of [say] Bunge 1977), does a simulated computer have to be a thing? If so, it looks like bad news for the viewpoint I have introduced, since it might seem pretty clear that a simulated computer just isn't a thing, whatever else it might be. (Someone fond of processes might claim that a simulated thing is a process.) Is this a fatal blow against the viewpoint I have advocated? No, for the following reason. A simulated computer still has hardware: the hardware of the host machine. Similarly, (assuming software is a process, for example) software on the guest is ultimately software on the host too and hence a process. Subsequently, the state space of the VM is a subspace of the host. This state space (or rather its real counterpart) is what is "constrained" by laws. One can see here that the "logician's answer" and the "isomorphism" viewpoint are actually not far wrong, since they do correctly capture the idea that there is some sort of "similarity" relation between a simulated machine and a real one. Spelling out the similarity relation is the work of the designer of the virtual machine software, etc.

Second, it might be rejoined that Wiggins' work is only intended to apply to natural kinds and their instances. Wiggins holds that artefacts have no laws *qua* artefacts, hence simulation of artefacts cannot be about laws. I think this part of Wiggins' views is simply mistaken. While there may be no laws of hammers *qua* hammers, etc. (I am willing to grant that *ex hypothesi*) I do not think Wiggins has paid enough attention (at least in the work I draw upon) to all kinds of artefacts. In particular, I think some artefacts refute the thesis. Computers are one of them: my computer is an instance of a natural kind of a sort: this category might be called *artificial natural* or *constructed natural*. This category is not *ad hoc* (i.e., introduced just for the sake of computers). I would also place novel

chemical elements and compounds in this category. It seems clear enough that there are laws governing new elements and compounds created by particle physics and chemistry respectively⁴. What are the laws which govern the kind “computer” (or, to be safe, as Siegelmann 1999 would remind us, Turing computer)? The most firmly established laws are those “captured” by the basic results of computability theory; for example, the Turing-unsolvability of the Turing machine halting problem by Turing

⁴ It might be rejoined that novel compounds (for example) do not have their own laws, but merely those of an appropriate class (or classes) of compounds of the natural sort(s) in question; for example, that a novel sort of alkane “follows” the laws of alkanes. This might be a plausible view for some compounds, but what about chemical species of kinds which are unlikely to exist anywhere in nature, like compounds containing the (short lived) five-nitrogen ion N_5^+ first synthesized approximately 11 years ago (Browne 1999)? Rhetorical questions aside, I do recognize that without a well established view of laws this issue is perhaps unanswerable. The view I am adopting tacitly is compatible with (at least) that of Bunge (1977) and Armstrong (1997, 2004). The minimal ingredients for a view of laws required are: (1) a reminder that we are talking about laws, not law **statements** (the conceptual reconstruction of laws in thought or language); (2) that laws are some sort of genuine relation between universals. (The precise nature of the relation is unimportant for the present purpose.); (3) that laws can be “multiply realizable” in some yet to be specified sense. This is vital for the microinstruction concern mentioned above. It should not be that a “correct” simulation at a given “level” requires simulation “all the way down” to whatever elementary particles (if any) make up a system. Nor for a computer system to simulate another should it be necessary for the microinstructions to be simulated - or the converse.

machine computers. Or, given certain classes of machines⁵, there are other reasonable candidates for laws referred to by the axioms and theorems of computational complexity theory (cf. Dean 2007 for a slightly different but partially compatible viewpoint).

However, the most important obstacle standing in the way of the Wiggins-style approach is actually that we know very few such computational laws. In a way, this obstacle can be stated as follows: how much “universalism” is required in laws (witness debates over biological laws, etc. See, e.g., Sterelny and Griffiths [1999], particularly pages 365-370)? Moreover, should we discover that there are very few possible laws that would govern both real computers and various sorts of computer simulation, will that refute the plausible Wigginsian approach I have sketched?

Thinking about whether there are computational laws will (perhaps) provide more material for the debate in general. In my view there are very few laws which really do seem to apply **everywhere and at all times** (i.e., as the “received” view has it). For example, chemical reactions require bound electrons, the relevant environment (e.g.,

⁵ I note in passing that the relativization of the results of computational complexity theory (see Cormen, Leiserson and Rivest [1996]) to (disjoint, nontrivial) classes of machines also gives another way to see computers as something like a natural kind: it allows categorizing computers into what might be called different *species* where the overall kind abides by some laws and there are laws for each species.

Moreover, various classes of machines have what look like lawlike relations between them. For example, a large class of “physically realizable” models can simulate each other in subpolynomial time (see, e.g., Arora and Barak 2007). A parallel would be how the laws of chemical bonding are the same for all elements (e.g., pairs of electrons are required for covalent bonding), but in addition specific classes of elements (e.g. alkali metals, halogens, noble gases) each have their own additional laws.

atmosphere) and time scale. Many chemical species exist momentarily, at least according to the theory of reaction mechanisms. Subsequently any chemical laws are relative to such scale, assuming the referents of laws statements governing of chemical bonding and stability of compounds are to be candidates for laws. Even particle physics is similar. Perhaps only conservation laws are truly general and thus only they might count as laws. I do not know, nor will I argue any further on this matter here. If so, I “bite the bullet” and stipulate: the views in the present paper are mistaken if the “narrow” view of laws mentioned is correct.

If not, however, we have thus seen a way of understanding what makes for a good simulation of a computer by a computer.

Conclusion

Simulations of computers are like real computers in so far as they “obey” the same laws. Since this entails that they have some of the same real world consequences, the policy question also has a fairly easy answer: computer simulations should, again to the extent that they “obey” the same laws, be constrained by the same policies. Further philosophical work would be to continue the investigation of the nature of laws presupposed in this paper and also to examine the relationship between computational models and actual computing systems (a case of, at first glance, a non-numerical idealization). But these questions, policy-oriented and philosophical, are other stories for other times.

References

Armstrong, David. 1997. *A World of States of Affairs*. Cambridge: Cambridge University Press.

Armstrong, David. 2004. *Truth and Truthmakers*. Cambridge: Cambridge University Press.

Arora, Sanjeev and Barak, Boaz. 2007. *Computational Complexity: A Modern Approach*. Draft available at < <http://www.cs.princeton.edu/theory/index.php/Compbook/Draft> > . Accessed December 30 2009.

Browne, Malcolm. 1999. "New Nitrogen Ion Carries Warning: Handle With Care". New York Times, February 2 1999. Available at < <http://www.nytimes.com/1999/02/02/science/new-nitrogen-ion-carries-warning-handle-with-care.html> > . Accessed December 19 2009.

Bunge, Mario. 1977. *The Furniture of the World*. (Volume 3 of *Treatise on Basic Philosophy*). Dordrecht: Reidel.

Churchland, Paul. 2007 [2002]. "Functionalism at Forty". Republished in *Neurophilosophy at Work*. New York: Cambridge University Press.

Cormen, Thomas, Leiserson, Charles. and Rivest, Ronald. 1996. *Introduction to Algorithms*. Cambridge: MIT Press.

Dean, Walter. 2007. "What Algorithms Could Not Be". Unpublished Ph.D. dissertation, Rutgers University.

Douglas, Keith. 2001. "A Special Davidsonian Theory of Events". Unpublished M.A. Thesis, University of British Columbia.

Douglas, Keith. 2003. "Super-Turing Computation: A Case Study Analysis". Unpublished M.S. Thesis, Carnegie Mellon University.

Douglas, Keith. 2008. "Through an Event Log, Darkly". Unpublished paper presented at NACAP 2008, Indiana University, July 2008. Revised and expanded version available in *The Information Society* 26(1), 2010.

Douglas, Keith. 2009. "Prolegomena to any Future Metaphysics of Computer Networking". Unpublished paper presented at NACAP 2009, Indiana University, June 2009.

Hennessey, John and Patterson, David. 1998. *Computer Organization and Design: The Hardware/Software Interface* (2e). San Francisco: Morgan Kaufman.

Liston, Tom and Skoudis, Ed. 2006. "On the Cutting Edge: Thwarting Virtual Machine Detection". Available at < http://handlers.sans.org/tliston/ThwartingVMDetection_Liston_Skoudis.pdf > . Accessed October 17 2009.

Makin, Stephen. 1993. *Indifference Arguments*. Oxford: Blackwell.

Popek, Gerald and Goldberg, Robert. 1974. "Formal Requirements for Virtualizable Third Generation Architectures." *Communications of the ACM*, vol. 17, no. 7.

Searle, John. 1990. "Is the brain a digital computer?" *Proceedings and Addresses of the American Philosophical Association* 64:21-37.

Siegelmann, Hava. 1999. *Neural Networks and Analog Computation: Beyond the Turing Limit*. Boston: Birkhäuser.

Sterelny, Kim and Griffiths, Paul. 1999. *Sex and Death: An Introduction to Philosophy of Biology*. Chicago: University of Chicago Press.

Turing, Alan. 1936. "On Computable Numbers, With an Application to the Entscheidungsproblem." Reprinted in Davis, Martin (ed.) 1964. *The Undecidable*. Hewlett: Raven Press.

Wiggins, David. 2001. *Sameness and Substance Renewed*. Cambridge: Cambridge University Press.