Virtual Machines: Non-Reductionist Bridges between the Functional and the Physical

Matthias Scheutz

Abstract Various notions of *supervenience* have been proposed as a solution to the "mind-body problem" to account for the dependence of mental states on their realizing physical states. In this chapter, we view the mind-body problem as an instance of the more general problem of how a virtual machine (VM) can be implemented in other virtual or physical machines. We propose a formal framework for defining virtual machine architectures and how they are composed of interacting functional units. The aim is to define a rich notion of *implementation* that can ultimately show how virtual machines defined in different ontologies can be related by way of implementing one virtual machine in another virtual (or physical) machine without requiring that the ontology in which the *implemented VM* is defined be reducible to the ontology of the *implementing VM*.

1 Introduction

The notion of supervenience has become a major explanatory device for tackling the mind-body problem, in particular, and for relating higher-level properties to lower-level properties, in general. As Kim (1998, p. 9) puts it: "Supervenience is standardly taken as a relation between two sets of properties, the supervenient properties and their base properties" (p. 9). Especially for those who espouse non-reductive physicalism, supervenience seems to provide a solution to *the mind-body problem*: by claiming that mental properties *supervene* on physical properties, the dependence of mental properties on physical properties is secured, and the possibility of changes in the mental without changes in the physical are prevented. At the same time, this dependence does not necessarily lead to the reduction of

Matthias Scheutz
Department of Computer Science, Tufts University, Medford, MA 02155, USA
e-mail: matthias.scheutz@tufts.edu

mental properties to physical properties, as supervenience (at least in some of its readings) is compatible with substance dualism as well as token physicalism.

However, for "property supervenience" to be a potential candidate for a "solution" to the mind-body problem, two crucial assumptions about mental properties need to made: first, that mental properties are properties of physical systems (which means that "being in pain", for example, is a property of a particular patch of four-dimensional space-time occupied by a creature capable of having pain). And secondly, that the mind-body problem reduces to a problem about a relation between mental and physical properties. Yet, it is not obvious that either assumption is justified, since *prima facie* mind and brain seem to be more than a mere collection of properties, given that psychological entities such as thoughts, imaginations, experiences, etc. are of a very different nature from cells, chemicals, potentials, etc. Thus, to say that the quantifiers in a psychological theory range over the same entities as the quantifiers in a physical theory (which has to be effectively assumed by proponents of property supervenience) seems to be a claim that at the very least needs to be argued for.

We believe that such an argument might turn out to be very difficult, if not impossible, and that instead of restricting the notion of supervenience to properties, it might be more promising to talk about *whole ontologies* supervening on other ontologies (see Sloman 1998). And while we certainly do not know yet how this link is effected in the case of mind and brain, we can find paradigmatic examples of different causally interacting ontological levels in the computer science notion of a *virtual machine*, i.e., the formal specifications of computational architectures using a particular ontology. Using the notion of virtual machines being implemented in other virtual or physical machines, we will argue that the notion of *implementation* (of a virtual machine) might elucidate non-reductive dependence in at least one way that property supervenience cannot accomplish: *A* being implemented in *B* does not only establish a dependence of *A* on *B*, but it also shows *how*, in addition to *that*, *A* depends on *B*. Hence, we suggest that the notion of implementation might turn out to be able to do more work in attempts to solve the mind-body problem or how higher-level ontologies are realized in lower-level ontologies than any of the current notions of (property) supervenience, for notions of supervenience can ultimately only state the mind-problem, while the notion of implementation might be able to solve it.

2 Supervenience and the Mind-Body Problem

The mind-body problem posed as the question "how does the mind relate to the body?" is very general in nature. It does not *per se* imply anything about the underlying ontologies, neither the mental nor the bodily. One common answer to the above question is that the mental is determined by and dependent on the physical. This relationship is expressed in term of supervenience, saying that the mental supervenes on the physical. Usually supervenience is defined as a relation between

properties, but that shifts the original problem to "how do mental properties relate to physical properties". The so-obtained restriction, however, seems unnecessary, if not unwarranted, unless it can be shown that the relation between mental and physical ontologies somehow reduces to a relation between certain classes of properties. This, however, does not seem to be the case. Fist of all, relations cannot be completely reduced to properties (this is a well-known fact from logic). Furthermore, even if it is possible to view some mental relations in terms of a particular set of physical properties, the language will be unnecessarily complicated by avoiding to talk about physical relations. Finally, the general question remains why mental entities such as beliefs, plans, hunches, discoveries, etc. need to be treated as "properties". We believe that this restriction is an alleged consequence of the physicalist ontological substance monism, which does not allow for any non-physical entities. Alleged, because it does not apriori follow that entities have to be automatically treated as properties of physical space-time simply because they are physically realized/realizable. In fact, it would not make sense at all to treat numbers as properties of physical objects. As Cantor has reminded us, numbers, *qua* entities, are properties of sets, i.e., non-physical objects and as such they are not properties of any physical space-time entity. Yet, one does not usually dispense with numbers or sets for that matter simply because they are abstract entities (unless one is a committed nominalist, but then there are other difficulties). So, there is at least no apriori reason why mental entities have to be viewed as in some sense "reduced" to properties that supervene on physical properties and can be instantiated by them. We believe that keeping mental entities as such in our ontology does not do any harm to the physicalist program of viewing material objects with mentality as being entirely composed of "physical stuff".

2.1 Types of supervenience

Several types of supervenience have been distinguished in the literature, the most common ones being "weak supervenience", "strong supervenience", and "global supervenience".[1] Whereas weak and strong supervenience apply to sets of properties, global supervenience applies to properties of whole worlds. Fixing two sets of properties α and β, then these notions can be defined as follows: α-properties supervene on β-properties if and only if

(Weak supervenience) necessarily, for any mental property $A \in \alpha$, if anything has A, there exists a property $B \in \beta$ it has B, and anything that has B has A.

[1] There are various versions of these three notions depending on how the modalities are defined. Furthermore, there are other kinds of supervenience, which involve relationships between patterns, or mathematical structures, e.g. one mathematical structure modelled in another.

(Strong supervenience) necessarily, for any mental property $A \in \alpha$, if anything has A, there exists a property $B \in \beta$ it has B, and necessarily anything that has B has A.[2]
(Global supervenience) for any worlds w_i and w_j, if w_i and w_j are β-indiscernible, then w_i and w_j are α-indiscernible.

While the relation between strong and weak supervenience is clearly that of implication (i.e., strong implies weak), the relation of both kinds to global supervenience is not all that clear. Some have argued that global and strong supervenience are equivalent (e.g., Kim, 1984), others claim that global supervenience is even weaker than weak supervenience (e.g., Petrie, 1987). One of the main problems with global supervenience is that if two worlds differ at all (even if only in the slightest respect) than their supervening properties could be entirely different according to supervenient, which is rather counterintuitive. On the other hand, a similar more "local" problem might arise for the two other versions of supervenience as well. Assume that mental properties of humans supervene on physical properties of brains, e.g., that pain supervenes of C-fiber stimulation. Suppose D-fibers are very similar to C-fibers, in fact, they are functionally equivalent at a neural level, but do not have certain low-level physical properties regarding their cell structure, etc. Then D-fibers could have entirely different supervening properties and this would be consistent with weak as well as strong supervenience. From a functionalist point of view, however, the two kinds of fibers should have the same supervening properties, if they have the same functionality (at a relevant level of description). This goes to show that supervenience is not quite sufficient for the functionalist to explain how the mental is determined by the physical if it is possible that the same functional role could lead to different mental properties. Consequently, none of the three versions of supervenience seem to be sufficient to capture the functionalist view on how the mental relates to the physical, i.e. how the mental is realized or implemented in the physical.

2.2 Relation Supervenience

There are other problems with a notion of supervenience that only considers the relation between sets of properties. It has been argued, for example, that relational mental properties do not supervene on properties of brains or organisms alone, but on environmental properties in addition (e.g., see Papineau, 1995, or Heil, 1992). While it seems that this deficiency could be remedied by adding environmental properties, it still leaves the strange aftertaste that something important about the relation of individual and environment is lost in such a move. But one does not even have to include the environment to see that reducing relational properties (if possible at all) will have unwanted side effects. One most obvious implication of

[2] Note that the difference between strong and weak supervenience lies in the added necessity operator in the last conjunct.

such a restriction is that the language used to talk about mental architectures will be unnecessarily complicated if it has to be phrased it in terms of properties. For example, to say that "short term memory is connected to long term memory via recurrent excitatory connections" would translate the ternary relation "___connected to ___ via ___" into something like the property of "having a short term memory and a long term memory such that these memories are connected by recurrent connections". Of course, in the property reading it is not possible to "access" the parts of the expression that makes up the property definition, such as "recurrent connection", "connected by", "short term memory", etc. (One would have to introduce another property for all these terms...), hence it is not possible to state explicitly that the property really expresses a relation. Not only is this very clumsy, but it also not clear how many mental properties one would have to introduce to capture all the various theoretically interesting relationships among them.

Another complicating factor is that some of these properties might not supervene even weakly, as they might be entirely dependent on internal configurations of the mental architecture at a given time (e.g., the recurrent memory connection might be achieved by many different physical connections at different times, and the physical property that gave rise to the recurrent connection at some point, might give rise to something else at a different time).

It is not clear that all mental relations can be couched in terms of physical properties and their relation. Take, for example, Kim's suggestion of how to define the supervenience of a mental relation R on set of physical properties B (Kim, 1993, p. 161):

> For any n-tuples, $\langle x_1,\ldots,x_n\rangle$ and $\langle y_1,\ldots,y_n\rangle$, if they are indiscernible in set B, then $R(x_1,\ldots,x_n)$ if and only if $R(y_1,\ldots,y_n)$, where $\langle x_1,\ldots,x_n\rangle$ and $\langle y_1,\ldots,y_n\rangle$ are in indiscernible in B just in case x_i is indiscernible from y_i in respect of B-properties.

As Kim notes, while this might be sufficient for relations like "taller than", which hold of tuples because of their "intrinsic" properties alone, it will not suffice in general; just consider causal relations such as "earlier than" or "east of". In the latter case, the notion of indiscernability will have to be defined in a different way. Kim's suggestion for such an account of indiscernability, in case an n-ary relation R is present in the base set, is as follows:

> Two entities x and y are indiscernable with respect to R if and only if for all $x_1,..,x_{n-1}$ and $y_1,..,y_{n-1}$, $R(x,x_1,..,x_{n-1})$ if and only if $R(y,y_1,..,y_{n-1})$ and $R(x_1,x,x_2,..,x_{n-1})$ if and only if $R(y_1,y,y_2,..,y_{n-1})$ and ... and $R(x_1,..,x_{n-1},x)$ if and only if $R(y_1,..,y_{n-1},y)$.

There are, however, problems with this requirement, as Kim points out, for suppose "we want to discuss whether a certain property P of wholes supervenes on the properties and relations characterizing their parts. Let X and Y be two distinct wholes with no overlapping parts, and suppose X consists of parts $x_1,..,x_n$ and Y consists of $y_1,..,y_m$. We would expect some properties of X and Y to depend on the relationships characterizing their parts–how these parts are organized and structured–as well as the properties of the parts. [...] What should we say about the conditions under X and Y may be said to be "mereological indiscernible"–that is, alike in respect of the way they are made up of parts? In a situation of this kind it

would be absurd to enforce [the above requirement for indiscernability]. For suppose that a dyadic relation, R, holds for two mereological parts of X, $\langle x_1, x_2 \rangle$; [the above requirement for indiscernability] would require mereological indiscernability of X and Y that some y_j be related to x_2! Obviously, what we want is that X and Y be characterized by *the same relational structure*. [...] If $x_1$has R to x_2, the some corresponding element of, y_j, of Y must have R to an appropriate y_h, not to x_2." (Kim, 1993, p. 164)

Kim then excludes isomorphism between X and Y as too strong a condition for mereological indiscernability, and suggests that "we may do well to work with similarity in the subvenient base set, rather than insist on indiscernability, when relations are present. In particular, the supervenience of the properties of wholes might be more appropriately explained in terms of their *mereological similarity* rather than their *mereological indiscernability*." (Kim, 1993, p. 164). Unfortunately, Kim then draws the conclusion that "strict relational supervenience–that is, relational supervenience satisfying [the above requirement for indiscernability]– may not be such a useful concept after all", mainly because of the problems with the above requirement for indiscernability, it seems. This, however, can be remedied, as we will show later, by using an extension of the notion of "bisimilarity" which underwrites the notion of implementation.

3 Virtual Machines and Virtual Machine Functionalism

In a sense, mereological supervenience is a special case of the general case of the supervenience of one ontology on another, namely the case where "higher level entities" can be seen as being composed of "lower level entities". The stratified view of our physical world is an example: molecules are made out of atoms, cells consist of many molecules, (higher) organisms are assemblies of many cells, etc. But while this mereological relationship seems to be generally true of "physical levels of description" (unless one wants to count some quantum decomposition phenomena as contradicting this claim), it is not true of all levels of descriptions, in particular, not of so-called implementation levels, i.e., levels of abstraction at which *virtual machines* are described.

The notion of "virtual machine" is prominent in cognitive science and even more so in computer science. By "virtual machine" we mean a specification of both an architecture and the kinds of processes that this architecture supports. The architecture specification typically includes basic entities or "parts" and their functional properties as well as descriptions of the various ways how these parts can be connected. Furthermore, it includes a specification of input and output kinds of the architecture and of operations that can be performed on them. If the virtual machine is a computational virtual machine, then it is convenient to think of its processes as being specified by some program, but there are also other ways of specifying interactions of various parts.

Example 1. When we talk about SCHEME being a virtual machine, for example, we certainly do not mean that it is indeed some sort of "machine" (in a standard sense of "machine")–compare this to the term "search engine", etc. Rather, there seem to be two parts involved, (1) the *programming language* SCHEME, which consists of symbols and compositions of these symbols, and (2) the SCHEME interpreter (i.e., the semantics of SCHEME), which evaluates SCHEME expressions. The later is what is considered the "virtual machine", as it does *perform* an operation, namely that of interpreting SCHEME expressions. To do this, it needs additional primitives and objects (such as environments, etc. most of which are accessible from within SCHEME) that allow it to store and retrieve, to evaluate and modify SCHEME expressions, and consequently (as implied by "store and retrieve", etc.) it needs some sort of scratch space ("working memory") and storage space ("long term memory") to operate with (tokens of) SCHEME expressions.

Virtual machine functionalism, then, is the view that all mental states (and this includes "phenomenal states") are states of processes of a virtual machine and mental concepts can be understood in terms of the concepts defining the respective virtual machine architecture. Virtual machine functionalism differs from other functionalist accounts in that it distinguishes between structural features of an architecture (such as its parts, the states that they can be in, and how they are related to other parts) and the processes supported by it (e.g., all possible processes of one subsystem and how they can influence the behavior of the processes of another subsystem). While parts of the virtual machine are generally viewed as serving particular functional roles in the overall architecture specification (analogous to functionalist accounts of functional states), states of the virtual machine, i.e., states of the processes "running on it", are taken to instantiate mental states.

If minds are then viewed as certain kinds of virtual machines, as we suggest, then it follows that supervenience, in order to be of any interest in addressing the mind-body problem (or the problem of how virtual machines relate to physical machines or other virtual machines), must be a relation between ontologies, i.e., it must include more than properties and relations. In particular, there will be a complex relation between the entities and concepts used in the specification of the supervenient virtual machine A and the entities and concepts used in the subvenient virtual machine or physical system B: while A entities will be related to B entities or conglomerations thereof, it might not be possible to relate A concepts to B concepts. Put differently, while A entities are *implemented* or *realized* in B entities, A concepts might not be reducible to B concepts. To couch this in terms of the mind-body problem: virtual machine functionalism suggests that mental entities are implemented in physical entities, while mental concepts are not reducible to physical concepts. This way the dependence and determination of the supervening virtual machine on the subvenient or implementing virtual machine is guaranteed without being reductionist about the supervenient ontology.

Even if one does not want to subscribe to the "virtual machine functionalist" stance on mind, the underlying notion of supervenience of an ontology is still more basic and general than any restricted notion of property or even relation supervenience. Any adherent of the latter notions will still have to explain why

"being a belief", for example, is a *property* of a physical object. First of all, it seems that "physics" *prima facie* does not supply the right kinds of objects that can have beliefs. If it does, then this needs to be pointed out. Furthermore, if my beliefs are inconsistent, for example, there does not have to be anything in the physical world that has that property of inconsistency and also has some physical property on which being a set of inconsistent beliefs supervenes. The burden of proof as to why such mental entities and their properties have to be reducible to physical properties is on the side of the adherent of property supervenience. And, furthermore, even if this difficulty could be resolved, an even trickier issue remains: as Kim (1993) points out at numerous places, "mind-body supervenience [...] does not state a solution to the mind-body problem; rather it states the problem itself" (ibid., p. 168). Rather, he believes that in order to obtain a substantive mind-body theory the dependence underlying the mind-body property covariance needs to be explicated and that mereological supervenience is a promising candidate. We agree with Kim that supervenience claims alone are insufficient to *explain* the mind-body dependence as they fail to show *how* the mind depends on the body. Any plausible candidate mind-body theory needs to explicate and explain the relationship between two ontologies, the mental and the physical, and one of the most promising candidates showing how the mental can be related to the physical is the virtual machine functionalism together with an appropriate notion of implementation.

4 Towards a Formal Specification of VM Architectures

The notion of "virtual machine architecture" or, more generally, "functional architecture" is central to the fields of cognitive science and artificial intelligence. It is thought to capture the basic information processing capabilities of natural or artificial information processing systems by specifying how functional "primitives" (or units), which cannot be explained in terms of decomposing them into smaller functional units, are related to each other to form a network of functions, all of which together define the information processing (and possibly cognitive) system. One of the interesting features of functional architectures is that they offer an escape from Ryle's Regress (sometimes also called "the homunculus fallacy") by using smallest non-decomposable functional units (these units are themselves explained by appealing to the properties of the systems implementing them).

Functional architectures can be used to explain mental states in terms of their functional (or causal) role, which can be decomposed into simpler terms (or states) until the smallest functional units are reached. If functionalism is right that mental states are defined solely by their causal role, then a functional specification of a cognitive architecture (i.e., providing a "functional architecture") is sufficient to study and explain mental phenomena at a certain level of description.

Some people have even linked functional specifications to defining a programming language: "Specifying the functional architecture of a system is like providing a manual that defines some programming language. Indeed, defining a

programming language is equivalent to specifying the functional architecture of a virtual machine" (Pylyshyn, 1984, p. 92).

In this section, we will work towards the definition of the notion of "functional unit", which is intended to capture the intuition that functional architectures consist of many different, yet connected functional parts or "subarchitectures", each of which can be in many different internal states. The basic idea of a "functional unit" is that it consists of input, inner, and output states as well as a transition function relating input and inner to output and inner states in time. Functional units will be allowed multiple inputs and outputs along what will be called "input or output channels". The notion of "channel" is to be understood in the sense of Shannon (1975) from an implementation point of view. From a logical point of view, however, it rather corresponds to the notion of "information channel", for example, as in Barwise and Seligman (1998).

4.1 Functional Units

Before tackling the complex case with multiple inputs and output, however, we shall start with the simple case of a functional unit with only one input and only one output. In a sense, such a functional unit is nothing but a finite state automaton without start state and final states, where transitions are labelled with a *duration*, i.e., the time it takes to transit from one state to the other. We will assume a (not necessarily finite) set Time, which comprises possible durations (e.g., "1 msec", "2 msec", etc.) and is closed under addition (i.e., any two durations can be added and will yield another duration that is again in Time. Note that no assumption is made as to *how* durations are specified, measured, etc. They are simply assumed to be given in advance. Also, there are issues regarding the nature of these durations that will not be addressed here (e.g., whether they are average durations).

Definition 1. A **simple functional unit** SFU is a tuple $\langle\langle Input\rangle, \langle Output\rangle, \langle Inner\rangle, trans\rangle$ where $Input$ is the set of input states, $Inner$ is the set of inner state, $Output$ is the set of output states, and $trans$ is the transition function $trans : Input \times Inner \longrightarrow \mathsf{Time} \times Output \times Inner$.

A special case is a functional unit with *no inner states*, where input states are directly related to output states. Such functional units will be called "atomic" (or "primitive" in Cummins' terms) and will become important later as basic building blocks in functional architectures.[3]

Definition 2. A **simple atomic functional unit** $SAFU$ is a *tuple* $\langle\langle Input\rangle, \langle Output\rangle, trans\rangle$ where $Input$ is the set of input states, $Output$ is the set of output states, and $trans : Input \longrightarrow \mathsf{Time} \times Output$ the function mapping inputs to durations and outputs.

[3] Ideally, a functional specification should be able to reduce the overall functional architecture to atomic functional units and their connections.

Obviously, a simple atomic functional unit is a functional unit in the sense that it can be written as $\langle Input, Output, \oslash, trans' \rangle$, where $trans' = \{\langle\langle a\rangle, \langle t, b\rangle\rangle | \langle a, \langle t, b\rangle\rangle \in trans\}$.

Everything said so far can be extended to functional units that have multiple input and multiple output channels. Instead of talking about "the input set" or "the output set", we simply consider finite sequences of such sets.

Definition 3. A **functional unit** FU is a tuple $\langle Input, Output, Inner, trans \rangle$ where $Input$ is a finite sequence (denoted as tuple) of sets of input states, $Inner$ a finite sequence of sets of inner states, $Output$ a finite sequence of sets of output states, and $trans$ is the transition function defined on sequences of sets as $trans : Input \times Inner \longrightarrow \mathsf{Time} \times Output \times Inner$.

An **atomic functional unit** AFU is a *tuple* $\langle Input, Output, trans \rangle$ where $Input$ is a finite sequence of input states, $Output$ is a finite sequence of output states, and $trans : Input \longrightarrow \mathsf{Time} \times Output$ the function mapping input sequences to durations and output sequences.

Note that this definition is very similar to what Chalmers' (1996) has called "combinatorial state automaton" (or CSA, for short). One minor difference is that CSAs can be infinite in that they can have an infinite sequence of inner states, which functional units cannot have. Another more important difference is that they incorporate the duration of their state transitions explicitly (and thus timing constraints implicitly).

Eventually we want to start building functional units from atomic functional units, i.e., from units of which we only know the IO-function, but where no inner states or state descriptions are available.[4] In this case, the connecting states, i.e., the output states of the first and input states of the second atomic functional unit will become the "inner states" of the new functional unit obtained by composition.

4.2 Composition of Functional Units

The next step is to use functional units to build more complex functional units. This can be achieved by connecting output channels of one functional unit to input channels of another (or possibly the same) functional unit. This connection will be effected by a "transducer function", which maps all possible values of an output channel in a 1-1 correspondence onto all possible values of an input channel.

Definition 4 (Composition of Functional Units). Let FU_1 and FU_2 be two functional units (not necessarily distinct) and let $Output_{FU_1}(i)$ be the set of values of the i-th output channel of FU_1 and $Input_{FU_2}(j)$ the set of values of the j-th input channel of FU_2. Then a bijective function f from $Output_{FU_1}(i)$ to $Input_{FU_2}(j)$ is called an "i,j-transducer" from FU_1 to FU_2. We then say that two functional units

[4] Note that this does not imply that the unit does not have internal states of any internal structure, only that we do not know what the details of its inner states or internal structure.

FU_1 and FU_2 are **i,j-composable** if there is an i,j-transducer from FU_1 to FU_2. And two functional units are said to be **composable** if they are i,j-composable for some i and j.

Transducers play an important role in connecting functional units; yet, they are mathematical constructions, and in the implementation of functional units links between units might have to be effected by a physical transducer. Hence, there are at least three different readings of "exists a bijection f": the general mathematical sense, the physical sense, and the practical sense, corresponding to three notions of possibility, the logical, physical, and feasible. For now, we will focus solely on the mathematical reading.

Merely connecting some input and output channels of functional units is not sufficient if we want to look at the resultant unit as a functional unit itself (as we have not defined what the "inner states" of the newly obtained unit are supposed to be). The idea then is to use a "product" of the inner states of both functional units in the sense that the new functional unit can at most have $n \cdot m$ different inner states, if the the first FU has n inner states and the second m.

Definition 5. Let FU_1 and FU_2 be two i,j-composable functional units and let f be an i,j-transducer from FU_1 to FU_2. Then the **f-composition** of FU_1 and FU_2 is the tuple $\langle Input_{FU_1}, Output_{FU_2}, Inner_{FU_1} \times Inner_{FU_2}, trans \rangle$ where $trans$ is the relation defined as the set of all tuples $\langle\langle in_1, \langle s_1, s_2\rangle\rangle, \langle t, out_2, \langle s_1', s_2'\rangle\rangle\rangle$ such that

1. $in_1 \in Input_{FU_1}$
2. $out_2 \in Output_{FU_2}$
3. $\langle\langle in_1, s_1\rangle, \langle t_1, out_1, s_1'\rangle\rangle \in trans_{FU_1}$ for some t_1
4. $\langle\langle in_2, s_2\rangle, \langle t_2, out_2, s_2'\rangle\rangle \in trans_{FU_2}$ for some t_2
5. $f(out(i)_1) = in(j)_2$
6. $t = t_1 + t_2$[5]

The way the above definition stands is not quite satisfactory. First of all, it is not clear what is going to happen with the outputs of the first functional units that are not connected to any inputs of the second, and vice versa with the inputs of the second unit that are not connected to any outputs of the first. Are they not used? One way to remedy this is to add the remaining input channels of the second to the input channels of the first, and to do the same for the output channels. However, this might lead to unwanted effects because of the time difference at which the input signals arrive at the second functional unit: the inputs that go through the first functional unit are by a factor of t_1 delayed as opposed to the ones that go directly to the second unit, and thus the overall output might be different from a scenario, in which the corresponding inputs are all applied at the same time to the second unit (e.g., by adding a delay units to all the input channels of the second functional unit that are not connected to outputs of the first functional unit, and to all output channels of the

[5] This is the overall duration between any input and the propagated effect through the one connected line to any output.

first that are not connected to input channels of the second, as the output channels of the second functional unit are delayed by a factor of t_2).

By adding all input lines and all output lines, we arrive at the definition of a complete f-composition:

Definition 6 (Complete f-composition). Let FU_1 and FU_2 be two i,j-composable functional units and let f be an i,j-transducer from FU_1 to FU_2. Furthermore, let $Input$ be the concatenation of the sequences $Input_{FU_1}$ and $Input_{FU_2}/j$ (which is the sequence reduced by the j-th component) and $Output$ be the concatenation of the sequences $Output_{FU_1}/i$ (the sequence reduced by the i-th component) and $Output_{FU_2}$.

Then the **complete f-composition** $FU_1 \Rightarrow_f FU_2$ of FU_1 and FU_2 is the tuple $\langle Input, Output, Inner_{FU_1} \times Inner_{FU_2}, trans \rangle$ where $trans$ is the relation defined as the set of all tuples $\langle\langle in_1, \langle s_1, s_2\rangle\rangle, \langle t, out_2, \langle s_1', s_2'\rangle\rangle\rangle$ such that

1. $in_1 \in Input$
2. $out_2 \in Output$
3. $\langle\langle in_1, s_1\rangle, \langle t_1, out_1, s_1'\rangle\rangle \in trans_{FU_1}$ for some t_1
4. $\langle\langle in_2, s_2\rangle, \langle t_2, out_2, s_2'\rangle\rangle \in trans_{FU_2}$ for some t_2
5. $f(out(i)_1) = in(j)_2$
6. $t = t_1$ for $in_1 \in Input_{FU_1}$
7. $t = t_2$ for $in_1 \in Input_{FU_2}/j$

As a corollary we get that both ways of combining functional units result in new functional units (the only difference being that some of the input and output channels are not available if the combination is not complete).

Corollary 1. *The (complete) f-composition of two f-composable functional units is a functional unit.*

Proof. Let $FU_1 \Rightarrow_f FU_2$ be the (complete) f-composition of two i,j-composable functional units FU_1 and FU_2 and let f be an i-j-transducer from FU_1 to FU_2. Obviously, $Input_{FU_1 \Rightarrow_f FU_2}$ $Inner_{FU_1 \Rightarrow_f FU_2}$ and $Output_{FU_1 \Rightarrow_f FU_2}$ are sets as required; to see that $trans_{FU_1 \Rightarrow_f FU_2}$ is of the right kind, observe that each tuple is of the form $\langle\langle in, \langle s_1, s_2\rangle\rangle, \langle t, out, \langle s_1', s_2'\rangle\rangle\rangle$, where $\langle s_1, s_2\rangle$ and $\langle s_1', s_2'\rangle$ are in $Inner_{FU_1 \Rightarrow_f FU_2}$ (as $Inner_{FU_1 \Rightarrow_f FU_2}$ is defined on $Inner_1 \times Inner_2$) and $t \in \mathsf{Time}$ given that Time is closed under addition.

So f-composition can be used to create more complex functional units from simpler ones. Note that timing (as mediated by "duration") plays a crucial role in constructing more complex functional units: without the explicit duration parameter it would not be possible to distinguish circuits that can be distinguished now (e.g., and XOR gate with a self-feedback loop).

Example 2. Suppose we are given the two functional units $FU_1 = \{\langle\{b\}\rangle, \langle\{c\}\rangle, \langle\{S\}\rangle, \{\langle\langle b, S\rangle, \langle 15, c, S\rangle\rangle\}\}$, and $FU_2 = \{\langle\{a\}\rangle, \langle\{e, o\}\rangle, \langle\{O, E\}\rangle, \{\langle\langle a, E\rangle, \langle 10, o, O\rangle\rangle, \langle\langle a, O\rangle, \langle 10, e, E\rangle\rangle\}\}$ and the

function $f = \{\langle c,a\rangle\}$ (Time is the set of positive integers). . Observe that both units are 1,1-composable and that f is a 1,1-transducer from FU_1 to FU_2. Hence, the complete f-composition is $FU_1 \Rightarrow FU_2 = \{\{b\},\{e,o\},\{\langle S,E\rangle,\langle S,O\rangle\},\{\langle\langle b,\langle S,E\rangle\rangle,\langle 25,o,\langle S,O\rangle\rangle\rangle,\langle\langle b,\langle S,O\rangle\rangle,\langle 25,e,\langle S,E\rangle\rangle\rangle\}\}$.

We also introduce another way of combining functional units which, different from the above combination, explicitly introduces a *new internal state*, namely the state of connection between the functional units as effected by the transducer. We will shall call "(complete) f-extension" to indicate that a new internal state was added to the functional unit.[6]

Definition 7 (Complete f-extension). Let FU_1 and FU_2 be two i,j-composable functional units and let f be an i,j-transducer from FU_1 to FU_2. Furthermore, let $Input$ be the concatenation of the sequences $Input_{FU_1}$ and $Input_{FU_2}/j$ (which is the sequence reduced by the j-th component) and $Output$ be the concatenation of the sequences $Output_{FU_1}/i$ (the sequence reduced by the i-th component) and $Output_{FU_2}$.

Then the **complete f-extension** $FU_1 \Rightarrow_f^+ FU_2$ of FU_1 and FU_2 is the tuple $\langle Input, Output, Inner_{FU_1} \times f \times Inner_{FU_2}, trans\rangle$ where $trans$ is the relation defined as the set of all tuples $\langle\langle in_1,\langle s_1,n,s_2\rangle\rangle,\langle t,out_2,\langle s_1',n',s_2'\rangle\rangle\rangle$ such that

1. $in_1 \in Input$
2. $out_2 \in Output$
3. $\langle\langle in_1,s_1\rangle,\langle t_1,out_1,s_1'\rangle\rangle \in trans_{FU_1}$ for some t_1
4. $\langle\langle in_2,s_2\rangle,\langle t_2,out_2,s_2'\rangle\rangle \in trans_{FU_2}$ for some t_2
5. $n,n' \in f(out_1(i)) = in_2(j)$
6. $t = t_1$ for $in_1 \in Input_{FU_1}$
7. $t = t_2$ for $in_1 \in Input_{FU_2}/j$

4.3 Functional Architectures and their Realization

Now that we have a way for functional units to be composed and extended, we also need to define a criterion for when we consider them "functionally equivalent", i.e., when they produce the same input-output mappings for all input patterns and times. Moreover, since we are interested in internal states as well, we want to have a way of distinguishing functionally equivalent units that also have, in some sense, the same internal states from those that might have a different internal state structure. This is, for example, important for the kinds of higher-level internal states that such a functional unit might realize (e.g., which mental states a particular functional architecture can instantiate). The idea here is based on Scheutz 2001 where we consider to computational systems the *same* if the internal states can be related through "bisimulation". Here we adapt the notion of bisimulation to functional units:

[6] Note that we can get an "incomplete" extension the same we got an incomplete composition by ignoring the input states of the second unit and the output states of the first that are not connected.

Definition 8 (Bisimulation between Functional Units). Let $FU_1 = \langle Input, Output, Inner_1, trans_1 \rangle$ and $FU_2 = \langle Input, Output, Inner_2, trans_2 \rangle$ be two functional units where $Input$ and $Output$ are two finite sequences of sets of input and output states, respectively, $Inner_1$ and $Inner_2$ are two sequences of sets of inner states, and $trans_1$ and $trans_2$ are the transition functions defined on sequences of sets as $trans_1 : Input \times Inner_1 \longrightarrow \mathsf{Time} \times Output \times Inner_1$ and $trans_2 : Input \times Inner_2 \longrightarrow \mathsf{Time} \times Output \times Inner_2$. The two functional units are said to be *bisimilar* if there exists a non-empty relation R (called "bisimulation") defined on $Inner_1 \times Inner_2$ such that the following four conditions hold:

1. If $\langle s_1, s_2 \rangle \in R$ and $\langle \langle i, s_1 \rangle, \langle t, o, s_1' \rangle \rangle \in trans_1$), then there exists $s_2' \in Inner_2$ such that $\langle s_1', s_2' \rangle \in R$ and $\langle \langle i, s_2 \rangle, \langle t, o, s_2' \rangle \rangle \in trans_2$)
2. If $\langle s_1, s_2 \rangle \in R$ and $\langle \langle i, s_2 \rangle, \langle t, o, s_2' \rangle \rangle \in trans_2$), then there exists $s_1' \in Inner_1$ such that $\langle s_1', s_2' \rangle \in R$ and $\langle \langle i, s_1 \rangle, \langle t, o, s_1' \rangle \rangle \in trans_1$)
3. For every $s_1 \in Inner_1$ there exists a $s_2 \in Inner_2$ such that $\langle s_1, s_2 \rangle \in R$
4. For every $s_2 \in Inner_2$ there exists a $s_1 \in Inner_1$ such that $\langle s_1, s_2 \rangle \in R$

The bisimulation relation bins internal states in each functional unit and relates them such that "redundant" but different internal states are grouped together (into equivalence classes where each state in a class is, from the perspective of the bisimulation relation, indistinguishable from the other members). As a corollary we get that there is a smallest bisimilar functional unit which has the minimal number of internal states necessary to effect the requisite input-output mapping given the internal state structure (compare this to the notion of characteristic automaton in Scheutz 2001).

As mentioned in the beginning, functional architectures are made up of functional units (simple or complex), each of which serves a (causal) role in the overall architecture:

Definition 9 (Functional Architecture). A *functional architecture* is a tuple $\langle Arch, Parts, Labels \rangle$ where $Arch$ is a (usually composite) functional unit, $Parts$ is a *finite* set of functional units which are *components* of $Arch$ (in that they, if composed appropriately, will become *bisimilar* to $Arch$), and $Label$ is a function assigning each functional unit in $Parts$ a unique label (which can be used, for example, to describe the causal role of the labeled part; or it could be labeled with a predicate expressing the mental property realized by the part, etc.).[7]

Note that $Parts$ can be empty (in case $Arch$ is not made up of any components, i.e., if it is an atomic functional unit), or it may contain different kinds of functional units. If it contains only simple atomic functional units, then the advantage is that all "inner states" in $Arch$ can be explained as and arise from connections between units in $Parts$. Even if a functional architecture is not specified by virtue of only simple functional units (without inner states), it is always possible to decompose

[7] Cp. this with Copeland's 1996 notion of architecture, which is based on the idea that one can label parts of a physical system, and with Gandy 1980, who uses hereditarily finite sets to form such hierarchies of parts of systems.

any functional architecture into atomic functional units, i.e., the given architecture can be specified by an *equivalent architecture* such that the set *Parts* consists solely of atomic functional units (this requires a straight-forward extension of the notion of bisimulation between two functional units to functional architectures such that the two *Arch* functional units are to be bisimilar, but not necessarily the functional units in *Parts*). This is argument is supported by the following decomposition theorem:

Theorem 1 (Decomposition). *Every functional unit with inner states can be decomposed into atomic functional units without inner states.*

Proof (Sketch). Let FU be the functional unit given by $\langle Input, Output, Inner, trans\rangle$ where $Input$ is a finite sequence of sets of input states, $Inner$ is a finite sequence of sets of inner states, $Output$ is a finite sequence of sets of output states, and $trans$ is the transition function defined on sequences of sets as $trans : Input \times Inner \longrightarrow \mathsf{Time} \times Output \times Inner$. Then define two i,j-composable atomic functional units $AFU_1 = \langle Input \cup Inner, Inner \cup Inner, trans_1\rangle$ – the "state updater" – and $AFU_2 = \langle Input \cup Inner, Output, trans_2\rangle$ – the "output producer" – where $trans_1 : Input \times Inner \longrightarrow \mathsf{Time} \times Inner \times Inner$ and $trans_2 : Input \times Inner \longrightarrow \mathsf{Time} \times Output$ such that $\langle\langle in, inner_m\rangle, \langle t, inner_n, inner_n\rangle\rangle \in trans_1$ and $\langle\langle in, inner_m\rangle, \langle t, out\rangle\rangle \in trans_2$ iff $\langle\langle in, inner_m\rangle, inner_n, out\rangle \in trans$. First, we recursively produce a new atomic functional unit $FU_1' = \langle Input, Output, trans\rangle$ by recursively connecting all output lines for one set of inner states to the input lines for inner states using the identity function as a transducer (this functional unit now maps input states onto inner states of FU). Then recursive i,i-composition of the i-th output line from FU_1' with the corresponding i-th input line for inner states for FU_2 for all i output lines of FU_1 (leaving only input lines for $Input$ states in FU_2) eventually leads to a functional unit that is bisimilar to FU. Finally, to use a new functional "split" unit which performs the function $split(X) = \langle X, X\rangle$ for all inputs sequences X, to duplicate the set of input channels in two and connect one set of output channels to FU_1', and the other to FU_2.

It then follows that every functional architecture has an equivalent functional architecture consisting of only atomic functional units as parts and all internal states are the states on the connection lines. This is important because we already have a formal criterion for what it means to realize atomic functional units in something physical, say, based on the notion of "realization of a function" developed in Scheutz 1999:

Definition 10 (Realization of Atomic Functional Unit). An atomic functional unit $AFU = \langle Input, Output, trans\rangle$ with a finite sequence of input states $Input$, a finite sequence of output states $Output$, and a state transition function $trans : Input \longrightarrow \mathsf{Time} \times Output$ mapping input sequences to durations and output sequences is realized by a physical system S (describable in a theory P) if and only if the following conditions hold:

1. There exists a (syntactic) isomorphic mapping I from the "input domain" of S to $Input$[8]
2. There exists a (syntactic) isomorphic mapping O from the "output domain" of S to $Output$
3. There exists a function F that describes the physical property (=behavior) of S for the given input-output properties (i.e., F is a mapping from the "input domain" of S to its "output domain" described in the language and by the laws of P) such that for all $x \in Input$, $O(F(I^{-1}(x))) = trans(x) \in Output$.

Given that we know what it means to realize the atomic functional units, we can provide at least one way to realize any functional unit by simply using the same construction as in the *Decomposition Theorem* and applying the above definition to the two decomposed units. And, as a result, we also can postulate at least one way of realizing a whole functional architecture, by realizing its *Arch* functional unit (according to the above definition), and by ensuring that the same physical system also realizes all atomic functional units. Of course, the "better way" of realizing a functional architecture in a physical system (or another functional architecture) is to ensure that all atomic functional units are realized in (distinct) parts of the physical system and that those physical parts are connected in the same way (i.e., the connection graphs among realized units in the two architectures are isomorphic) that the atomic functional units are connected to give rise to the whole functional architecture. This requires us also to check that the inputs and outputs of all connected physical parts match up (otherwise physical transducers will have to be introduced to ensure that parts can be connected properly). Moreover, we also have to pay attention to meet the timing constraints imposed by the parts of the functional architecture in the physical system (as different transition times might give rise to different functional systems). Note that the above construction not only works for physical systems, but also for other virtual machine architectures, thus allowing us to define a very general notion of one virtual machine being implemented in another virtual machine.

5 Discussion

The above sketch of how one could define a general notion of implementing one virtual machine in another by way of showing how their functional architectures can be related is, of course, only a start. For one, the notion of virtual machine used in the above discussion was very informal, and we attempted to make it more formal by introducing the notion of a functional unit as constituent part of a virtual machine. However, in this discussion, we glossed over the ontological status of input and output states, i.e., what kinds of entities those ranged over. For example, the inputs to one functional unit might be chess pieces, while the inputs and outputs

[8] See Scheutz 1999 about the "qualifier "syntactic isomorphism" (for all practical purposes we can simply consider it here an isomorphic mapping).

to another might be polygons or words. As a result, there is an important open problem left to tackle, namely how to map entities onto each other when they do not belong to the same domain (e.g., real numbers). What is needed, effectively, is a theory of transduction and encoding that shows how entities of one kind can be obtained by putting together structures of entities of another kind (e.g., see Pollock (2008) for a promising approach). For example, in word processors implemented in von Neumann machines, words are made out of sequences of letters encoded in 7-bit binary ASCII binary codes. If one input line in a component of the word processing VM can hold any word (of at most 15 letters, say), this line would have to be mapped on 15*7 binary input lines in the implementing VM. Similarly, we might be able to encode some entities not only in terms of spatially separate codes, but temporally (e.g., in the sequential way the Morse code is realized). All of this will require more fleshing out of details in the above definitions to allow for more general mappings between different ontological entities. However, the overall structure of the relationship between components of an architecture, their connections, and the implementing structures will overall remain the same. I.e., the higher-level VM that is to be implemented in a lower-level VM will be defined by its architecture that either can or cannot be realized, in the above sense, in the architecture of the lower-level VM. As a result, the notion of implementation of virtual machines in other virtual or physical machines shows one way in which whole ontologies could be related, assuming that the problem of relating and encoding higher-level into lower-level entities can be defined in general enough terms (the details of this definition have to be left for another occasion). This also entails that *any relation* defined in terms of the states of connections among atomic functional units realized in one virtual machine are also realized in the implementing virtual machine. Moreover, the mapping of VM parts of the higher-level VM onto VM parts in the lower-level VM shows *how* the relation is realized. But note that the mapping does not mean that the *concepts* – the intensions – associated with those entities and relations *can be reduced*. In other words, VM implementation provides a way of showing how higher-level entities can be encoded in, made out of, or related to possibly complex structures of low-level entities *without* forcing one to *equate* or *identify* higher-level entities with those realizing structures, thus providing non-reductive bridges that preserve the conceptual autonomy of higher-level concepts in the ontology without having to subscribe to spooky metaphysical theories (such as various forms of dualism) in order to explain the dependence of higher-level on lower-level virtual machines.

Acknowledgments

This paper would not have been possible without the many discussions with Aaron Sloman the author was fortunate to have over the years, although Aaron is by no means to blame for any errors or potential problems with the specific content.

References

Copeland J (1996) What is Computation? Synthese 108:335–359
Kim J (1984) Concepts of Supervenience. Philosophy and Phenomelogical Research 14:153–176
Kim J (1993) Supervenience and Mind: Selected philosophical essays. Cambridge University Press, Cambridge
Kim J (1998) Mind in a Physical World. MIT Press, Cambridge, Mass
Papineau D (1995) Arguments for supervenience and physical realization. In: Savellos E, Yalcin U (eds) Supervenience, Cambridge University Press, Cambridge
Petria B (1987) Global Supervenience and Reduction. Philosophy and Phenomelogical Research 48:119–130
Pollock J (2008) What Am I? Virtual Machines and the Mind/Body Problem. Philosophy and Phenomelogical Research 76:237–309
Pylyshyn Z (1984) Computation and Cognition. MIT Press, Cambridge
Scheutz M (1999) When Physical Systems Realize Functions... Minds and Machines 9:161–196
Scheutz M (2001) Causal vs. Computational Complexity? Minds and Machines 11:534–566
Sloman A (1998) Supervenience and implementation. Technical Report, School of Computer Science, University of Birmingham.