Marcin Miłkowski

# Computation and Multiple Realizability

## Abstract

Multiple realizability (MR) is traditionally conceived of as *the* feature of computational systems, and has been used to argue for irreducibility of higher-level theories. I will show that there are several ways a computational system may be seen to display MR. These ways correspond to (at least) five ways one can conceive of the function of the physical computational system. However, they do not match common intuitions about MR. I show that MR is deeply interest-related, and for this reason, difficult to pin down exactly. I claim that MR is of little importance for defending computationalism, and argue that it should rather appeal to organizational invariance or substrate neutrality of computation, which are much more intuitive but cannot support strong antireductionist arguments.

I want to undermine the conviction that multiple realizability (MR) is particularly important in understanding the nature of computational systems. MR is held to be fundamental as far as it is considered indispensable in arguing for irreducibility of theories that appeal to the notion of computation. This is why this conviction is especially common among proponents of antireductionism who want to defend the autonomy of psychology (for example, (Block, 1990, p. 146)), even if it's not so important for theorists of computability, if not completely alien to them.[1]

Recently, it was also argued that mechanistic theories of physical computation (Piccinini, 2007, 2010) were not compatible with multiple realization (Haimovici, 2013), which was supposed to show that mechanism is wrong. Indeed, in defending my mechanistic account of computation, I denied that multiple realization is an essential feature of computation, and that there are no facts of the matter that could easily establish that a given computational capacity is actually multiply realized or not (Miłkowski, 2013, Chapter 2). I want to develop this point in detail here and show that there are multiple ways of carving the computational capacity but whenever it is made precise, there are either too many examples of MR or scarcely any in cases that have been considered paradigmatic. For this reason, it does not seem to be an essential feature at all; I suggest that it can be easily replaced with another similar feature of organizational invariance or substrate neutrality.

The organization of the paper is as follows. First, I introduce the notion of MR, and show it is quite vague. I argue for additional restrictions on the notion to make it more precise. Then, I analyze one vivid example from the history of computing. In the following discussion, I distinguish MR from substrate neutrality and suggest that the latter is indeed more important. At the same time, substrate neutrality does not lend credibility to antireductionist arguments, and remains compatible with type-identity theory.

## 1. Multiple realization introduced, criticized, and made more precise

The argument from multiple realizability to irreducibility is usually attributed to Jerry Fodor (1974) and to Hilary Putnam (1975). Elliot Sober summarizes the argument in the following way:

1. Higher-level sciences describe properties that are multiply realizable and that provide good explanations.

[1] I owe this observation Aaron Sloman.

2. If a property described in a higher-level science is multiply realizable at a lower level, then the lower-level science will not be able to explain, or will explain only feebly, the phenomena that the higher-level science explains well.

3. If higher-level sciences provide good explanations of phenomena that lower-level sciences cannot explain, or explain only feebly, then reductionism is false.

---

Reductionism is false. (Sober, 1999, p. 558)

However, it needs to be noted that neither Fodor nor Putnam uses the term "multiple realization" or its cognates. In the subsequent discussion, it was used mostly informally, usually without a definition but with the help of Fodor's example: money. Money can be realized as coins, banknotes, credit cards, sea shells, and what not. There is simply no physical kind that encompasses all realizations of money, Fodor claims, and the lower-level, physical description of the bearers of monetary value is bound to be wildly disjunctive (Fodor, 1974). The same is supposed to happen with realizations of computer algorithms, but not simply because they are cultural entities, like money: it has been claimed that one can make a computer of silicon chips, but also of toilet paper or Swiss cheese, and they would realize the same algorithm and be different realizations. Simply speaking, a functional capacity is multiply realized if and only if it its occurrence is owing to different realizing structures. But this initial definition remains fairly vague; it does not decide whether anything else but functional capacities is multiply realized; it also does not set any standards on when functional capacities count as exactly of the same type, and what the standards for sameness of realizers are. It turns out that making the notion precise is extremely difficult. In addition, according to Keeley (2000), neuroethology uses both behavioral evidence and physiological evidence to justify its claims; and MR might be compatible with reductionism after all. But whether the reductionism is correct or not is not the question I am interested in this paper.

It's fair to say that after initial enthusiasm, philosophers have become much more skeptical about MR, and many argued that it need not be so frequent or essential. For example (Bechtel & Mundale, 1999) argued that, contrary to appearances, different brains are at least sometimes viewed as exactly the *same* realizers of a given psychological capacity: Neuroscience frequently uses animal brains as models of the human brain, and there would be no animal models for human brains if animal brains were actually so wildly different as antireductionists suppose. For example, the rat's brain is not really a different realization of dopamine-related capacities when it uses dopamine. Just like Bechtel and Mundale, (Polger, 2004, 2008; Shapiro, 2000, 2004) consequently argued that there might be merely illusory MR in cases where the function has been described in a generic way and its realization quite specifically. Just as different colors of a corkscrew do not make them different physical realizations (after all, the color is irrelevant to the function of the corkscrew), not all specific details count when it comes to distinguishing different realizations. But even not all *causally* relevant detail is equally important: for example, for opening a wine bottle, one could easily use a corkscrew with five or four threads on their screws. There is a certain level of changes that we usually consider irrelevant (in the engineering contexts, one speaks of tolerance levels). What makes an amount of possible differences irrelevant? Are there any principled answers to this question? Before I go on, an important caveat is in place.

Some would deny that corkscrews are ever multiply realized. First, they are artifacts, and some deny functional status to artifacts or claim that artifacts only have functionality derived from the functions of their biological users (Davies, 2001). Here, I will assume that derived functions are equally functional so as not to beg the question against the proponents of MR with respect to computation. Second, and more importantly, functional capacities of corkscrews and properties of realizations of their capacities to open the bottles might be both ascribed simply to corkscrews, rather than parts of

corkscrews. That leads to a general question. Is it possible for a functional capacity to be a property of the same entity as the properties of the realizers? A positive answer is the so-called 'standard', or 'flat' view on MR, whereas the 'dimensioned view' stresses that instances of properties of realizers are not the properties of the same entity, which introduces more 'dimensions' to the definition of MR (Aizawa & Gillett, 2009; Gillett, 2002). The dimensioned view has its roots in the functionalist tradition: Cummins' (1975) conditions for functional analysis require that functional capacities be ascribed to higher-level structures, and their realizing structures be on the lower-level; thus on the dimensioned view, MR is an ontological inter-level relationship. The defenders of the flat view have argued, however, that the dimensioned view leads to absurdity (Polger & Shapiro, 2008). My point in this paper is however largely independent from this debate (but for a reply, see (Gillett, 2011)), and the main problem for MR of computation is essentially the same for both dimensioned and flat views. In what follows, I will still peruse the simple example of corkscrews; if you subscribe to the dimensioned view, there is always a paraphrase of my claims in terms of the operations and parts of the corkscrew rather in terms of the operation of the corkscrew itself.

Let me return to the core of the problem with MR. My initial definition is vague, and we need to have strict standards of sameness and difference to show that MR actually occurs, and these standards cannot beg the question against reductionism. But other definitions currently espoused by defenders of MR do not seem to fare any better. For example:

> (Multiple Realization) A property *G* is multiply realized if and only if (i) under condition *$*, an individual *s* has an instance of property *G* in virtue of the powers contributed by instances of properties/relations $F_1$ -$F_n$ to *s*, or *s*'s constituents, but not vice versa; (ii) under condition *$**  (which may or may not be identical to *$*), an individual *s** (which may or may not be identical to *s*) has an instance of a property *G* in virtue of the powers contributed by instances of properties/relations $F^*_1$ -$F^*_m$ to *s** or *s**'s constituents, but not vice versa; (iii) $F_1$ -$F_n$ ≠ $F^*_1$-$F^*_m$ and (iv), under conditions *$* and *$**, $F_1$ -$F_n$ and $F^*_1$ -$F^*_m$ are at the same scientific level of properties (Aizawa & Gillett, 2009, p. 188).

According to (iii), we need to make sure that properties or relations are different in different realizers. But what are the standards, again? The ascription of function should be as fine-grained as the description of the function realization. However, the notion of function is inextricably linked with theoretical interests of the observer, thus MR is also heavily interest-dependent. (Note: even if there are proposals to make the function ascriptions as determinate as possible, they usually pertain only to the so-called notion of etiological notion of function, cf. (Price, 2001); for MR, however, the relevant notion is systemic or dispositional, as defined by Cummins (1975), and this notion is interest-dependent.) It does not mean that there are absolutely no facts of the matter in function ascriptions, as ascriptions may fail to be true; but they are still interest-driven (Craver, 2013).

So how could one decide when the functional capacity is the same, and when it is not? Is there any evidence one could use? For example, Shagrir noted that the evidence for the sameness of the psychological capacities in the case of neuroplasticity—or, more importantly, in case of realizing computational algorithms—is not really available for antireductionists, as all that they have is a set of "behavioral antecedents and consequences of subjects, and physical realizations of the mediating computational mechanisms" (Shagrir, 1998). This kind of evidence simply cannot settle the question at all. For this reason, an important topic in the debate has become the question of how to evaluate claims for MR (Polger, 2008; Shapiro, 2008).

So how can we evaluate the claim that a given computational algorithm or a computational system has multiple realizations? Here are possible answers. I will argue that in interesting cases—the ones usually held to be paradigmatic—there's no MR, contrary to appearances.

2. When are computations multiply realized?

Let me pick a particularly vivid example of two compatible IBM computers, 709 and 7090, one with tubes, and another with transistors. It seems to be ideally suited to test the claim whether one can fruitfully talk of multiple realization in the case of computational systems. In addition, some authors have earlier considered two IBMs to be realizations of the same computational system (Wimsatt, 2002), and I have argued earlier that it is not necessarily the case (Miłkowski, 2013). (IBM 7090 was also featured in Dr. Strangelove, which should be a good enough reason to consider it worthy of philosophical attention.)

IBM 709 and IBM 7090 share the same logical diagram (as witnessed by their technical documentation) but they use different electronics. Transistors work faster than tubes, so they differ also with their speed of operation. They have obviously different footprint, as transistors are smaller: the newer version is 50% smaller, and needs less ventilation because transistors need less cooling. Transistors also consume 70% less power, while tubes are 6 times slower. There's no denying that these machine differ; they also share a lot (for more technical specifications and photos, see ("IBM Archives: 709 Data Processing System," 2003, "IBM Archives: 7090 Data Processing System," 2003)). But is there MR in the strict meaning of the term?

To make MR claims about computational precise, one has to explain what it is to have a computational function. I can enumerate at least five ways one can understand the computational capacity of a given computer. I will explain these in turn but before I do so, I need to briefly introduce my mechanistic framework that allows talking about physical—or implemented—computation. One of the most widely endorsed views in the philosophy of special sciences is neo-mechanism (Bechtel, 2008; Craver, 2007; Machamer, Darden, & Craver, 2000). According to this view, to explain a phenomenon is to explain the underlying mechanism. Mechanistic explanation is causal explanation, and explaining a mechanism involves describing its causal structure. While mechanisms are defined in various ways by different authors, the core idea is that they are organized systems, comprising causally relevant component parts and operations (or activities) thereof. Components of the mechanism interact and their orchestrated operation contributes to the capacity of the mechanism.
A mechanism implements a computation just when the causal organization of the mechanism is such that the input and output information streams are causally linked and that this link, along with the specific structure of information processing, is completely described (the notion of information is not semantic in my account; for a similar treatment, see (Fresco, 2014)). Importantly, the link can be cyclical and as complex as one could wish. Mechanistic constitutive explanation, usually used to explain physical computation in cases where we deem physical implementation important, includes *at least* three levels of the mechanism: a *constitutive* (-1) level, which is the lowest level in the given analysis; an *isolated* (0) level, at which the parts of the mechanism are specified along with their interactions (activities or operations); and the *contextual* (+1) level, at which the function of the mechanism is seen in a broader context. These levels are not just levels of abstraction; they are levels of *composition* or organization.
The description of a *mechanistically adequate model* of computation comprises *two* parts: (1) an abstract specification of a computation, which should include all the variables causally relevant for the computation; (2) a complete blueprint of the mechanism on three levels of its organization. I call the first part the *formal model of the mechanism* and the second the *instantiation blueprint* of the mechanism (for a detailed study of how this framework is applied to physical and non-conventional computers, such as Physarum machines, see (Miłkowski, 2014)).

After this brief introduction of terminology, I can enumerate five possible ways of understanding the capacity of the computational mechanism. First, the computers could share the same causal structure associated with their formal model and differ in their instantiation blueprint (let's call it *formal-model* MR). Now, how much do they have to differ? Arguably, one could say that a certain level of tolerance for physical changes is required; otherwise, simple physical wear and tear or

replacement of worn parts would be enough to say that a computer is now a new realization of its older version (this is a new version of the ancient puzzle regarding the ship of Theseus). This is why only physical changes relevant to the execution of computation should matter; replacing a chip with a faster one would only make the speed of operation faster by a certain linear factor *k* (as is usually assumed in the theory of computational complexity), rather than mean that the computer has an altogether different substrate.

An emulation of the PowerPC computer on an Intel x86 architecture should qualify as formal-model MR, as they both share the formal model of the computation (in virtue of emulation on the second one) but the second machine has to include much more in its formal model, as the emulated machine is just virtual. So the Intel machine does not really have the *same* formal model; the formal model of PowerPC machine is rather a proper part of the complete formal model of the Intel x86 machine. In other words, these machines literally *share* the same formal model but they do not have exactly the same model.

Let me compare this case to what Bechtel and Mundale (1999) consider not to be an instance of MR. Usually, philosophers assumed that the functionality of the brain is multiply realized by various anatomies in different species. However, the human brain topographical map, created by Korbinian Brodman does rely on inter-species anatomical similarities, or homologies. In other words, anatomical similarity is used in function ascriptions. Actually, it seems that there are lower-level types that match higher-level types. The same consideration can be used in the case of my laptop: the kind of hardware it requires to function properly is specified with a certain level of tolerance. As long as memory chips, for example, match the specification of the motherboard, they can be used in my laptop without changing its functionality. But the similarity in question goes beyond the ability to replace parts; for example, one cannot replace the part of the human brain with a brain of a rodent and keep the same functionality. However, there are lower-level, anatomical types that still match the functionality of the brain in an orderly fashion, so that one can defend a kind of type identity between both. It seems therefore cogent that as long as there is sufficient similarity that allows classifying tokens of the realizing structures as belonging to the same type we do not have any kind of MR. Actually, the reason for similarity between brains is homology, or shared ancestry. In the case of IBM 7090, it is similar to IBM 709 because the latter is its ancestor, and we should expect a lot of hardware similarities, even if we cannot substitute tubes with transistors directly (owing to different voltages and so on). For the case of emulation of a PowerPC on an Intel x86 CPU, the similarity of functionality in question is not caused by common ancestry but by the software run, so there are no such hardware similarities we might expect between both IBMs.

One could insist that vast physical differences between IBMs constitute different types. But there are also vast differences between how both IBMs operate, such as different speed of operation, which obviously makes difference for the use of computers, and by supposing that the functional capacity in question is what is specified just by the formal model of the mechanism, we decided to *ignore* such differences as irrelevant, which is an important and far-reaching assumption. Yet if we use low-grain distinctions for functional types, there is no *principled* reason to use fine-grain distinctions for structural types, beside the philosophical prejudice for MR. They are definitely software-compatible and could run the same FORTRAN programs.

A second way to understand the computational capacities of computers is to focus on the mathematical function that they compute, specified in terms of input and output values (this will be *mathematical function MR*). This is the level of computational equivalence usually presupposed in the computability theory: If one says that a computer C is Turing-machine-equivalent in terms of the set of C-computable functions, then one presupposes that they will produce the same output given the same input, whatever the way they compute it. Note however that this is not the level of grain usually presupposed in cognitive science when it speaks of computational algorithms realized by brains: in cognitive science, the way which a given output is produced is immensely important

(Fodor, 1968; Miłkowski, 2011; Shagrir, 1998). For example, reaction times are used as evidence to decide whether people use this or another algorithm (Meyer, Osman, Irwin, & Yantis, 1988; Posner, 2005), so algorithms are individuated with a finer grain in cognitive research.

Is there mathematical function MR for both IBMs? They definitely share the same mathematical functions (which follows immediately from the fact that they could use exactly the same software). Again, however, their hardware similarities make it possible to classify them as the same kind of machine—they aren't even realizing different computational architectures. The difference between tubes and transistors is just as irrelevant as the difference between four and five threads on my corkscrews.

So how would a cognitive scientist understand a computational capacity of a system? In this third version, the capacity would also include the features of the physical implementation, such as the speed of operation; this will be *instantiation blueprint MR*. Of course, there is again a certain level of tolerance, so that a mere replacement of one electronic part does not create another realization of the same type. In the case of two IBMs, we have different speeds, as tubes are simply slower. Of course, the bigger footprint of the tube-based IBM is not so important here (just like cognitive scientists usually ignore the fact whether subjects are overweight or not) but there are different patterns of breakdown, which are also important in neuropsychology, as pathologies are used to make inferences about function (Glymour, 1994). All in all, here even the capacity is different, so there is no chance for MR to occur.

Between the level of grain implied by mathematical input/output specifications and the one assumed in cognitive science, one can drive a wedge for another – fourth – specification. For example, any algorithm for sorting words alphabetically shares the same input/output relations but not the same sequence of steps, which is again specifiable in terms of input/output relations. So, one could understand a particular algorithm for sorting, such as QuickSort, as mathematical function that is decomposed into an ordered sequence of other more basic mathematical functions. The kind of MR in this case might be dubbed *decomposed mathematical function MR.* This is how, for example, Keeley seems to understand the algorithm underlying perceptual abilities of weakly electric fish (he cannot mean the whole formal model, as in the first meaning of the functional capacity above, because exact ways that the algorithm is realized are not known). Note however that we have reason to suppose that there is MR only when fish realize these steps using sufficiently different hardware that cannot be classified as instantiating the same kind of operations. Keeley (2000) thinks this is the case for weakly electric fish; but it does not seem to be the case for IBMs, as their hardware architectures are simply too similar to imply sufficient changes in lower-level types. They share the same sequence of mathematical steps but also the same type of hardware architecture, where tube/transistor difference is not essential to individuating types.

Now, there is yet another, fifth way to think of the computational capacity of computers, namely in terms of how they are connected to all peripheral devices; this will be *information-flow MR*. The pattern of connections is called a "schematic of data flow" in the original IBM documents, and it describes the connections of the central processing unit and magnetic core memory with (1) console lights and switches; (2) cathode ray tube output; (3) magnetic drum storage; (4) and three data syncronizer [sic!] units, each with connections to optional card readers, card punches, printers, tape control units, and other "external signal sources". This way of looking at the computer makes it basically a node in a network of connections, so two realizations would be of the same type of node if and only if they share all the connections. However, if they do, the exact electronic realization of the data flow is basically as irrelevant as in previous cases.

It seems therefore that different perspectives do not really help to see IBMs as multiply realizing the same computational kind. Either we end up with a different computational type for each IBM, as in the instantiation-blueprint MR, or with the same computational and lower-level types for both.

Note that the so-called dimensioned view of realization (Gillett, 2002; Wilson & Craver, 2007) cannot save the MR claim with regard to this example either, as my argument can be easily rephrased in terms of Cummins-style functional analysis. The argument is based on the fact that there has to be the same level of grain in individuating both high-level and lower-level types; in other words, we consider lower-level differences to constitute a different type not just in any situation but only when the differences are in a relevant way connected to the way they realize higher-level functionality. One could object that this way, there is no way for MR to occur, but his does not exclude that there are cases of genuine MR. Not at all; the emulator example serves as an instance of genuine MR. It's just that MR occurs only in a very limited number of cases of organizational similarity between computers, so one cannot say that it is in some way essentially linked to the notion of computation the way it was earlier presupposed. In other words, the mechanists should really be skeptical of the role assigned traditionally to MR.

## 3. Organizational invariance and substrate neutrality

At this point, the defenders of MR might still argue that the notion is useful. First, they might insist that there is yet another way of identifying the computational capacities of both IBMs. This is of course possible, and I have definitely not enumerated all possibilities. But I do think that my five options correspond to the usual ways of thinking about computational types. Still, there might be others; I'm not holding my breath to get to know them, however, as it's not so plausible that they might solve the main problem, which is how to have the same level of grain on both levels and still retain MR without begging the question against reductionism.

Another possible objection is to deny that only function-relevant lower-level properties count in identifying the cases of MR. One might bite the bullet and say that the number of threads and the color of the corkscrew do count as realization-relevant properties. The number of cases of MR would definitely increase, as any instance of a type would turn out to be another realization of the type. The question then becomes: Why introduce the notion of realization when we already have the notion of the token? It goes against theoretical parsimony.

Of course, one might point out that I didn't deny that MR is indeed possible and that there are cases of genuine MR. So a reply from a defender of MR might go like this: Maybe it is a bit counterintuitive that these IBM computers are not instances of MR but it's not so vital. We do have genuine MR, so reductionism is doomed anyway. However, I think such an answer would be too quick. I haven't dealt with the question whether MR really warrants antireductionism (and there are authors who stress it does not, cf. (Keeley, 2000)), so let me put this question to the side. The intuitive point about MR and computation, one that motivates the claim that MR is somehow essential to computation, is that MR is always logically possible for a given computational system. That may be still true, though achieving MR has a higher price than usually assumed; not only one cannot really make a complex computer out of Swiss cheese or toilet paper, which has been pointed out before (Shagrir, 1998), but also one needs to find relevant differences of the proper gain at the lower level. Still, logically this is possible, but no less than creating a system that exactly simulates the steps in another computational system; there are simply multiple ways one could express similarities and equivalences between computations.

As compared to other precise notions, such as bi-simulation (Malcolm, 1996), MR seems to be particularly vague, and there are no facts of the matter that would help decide whether MR occurs or not just because individuation the computational type is inextricably linked to the theoretical interest. This is why the notion needs to be made precise every time by specifying exactly what is meant by the computational type that is supposed to be multiply realized.

I grant that MR might also be defined in the way that such constraints on relevance of realization types and the pragmatic or perspective-bound character of the notion are gone but I do not really see the point of such an endeavor. The problem is that the account of MR should both avoid begging the question against reductionism and not trivialize the notion (by making it effectively as broad as the notion of the token). I think the prospects of such a project are quite gloomy.

So how can one express the intuitive idea that computation is not really linked to the physical substrate in the way many other properties are? IBM 709 and 7090 can indeed compute the same mathematical function and share the same abstract structure of their mechanisms, which is important in explaining and predicting their work. As *purely* computational systems, they are exactly the same, and physical differences are irrelevant, as they make no difference for realization of their capacity to compute the same set of mathematical functions in the same way (by running the same software and being compatible to the same set of peripheral devices, and so forth). For this reason, I suggest that computationalism should rather embrace the notion of *organizational invariance* (or *substrate neutrality*), as both systems share the same *relevant causal topology* on one level of their functioning. Note that under four abovementioned different understanding of computational types (the one related to the instantiation blueprint is an exception) there is organizational invariance between the type and both IBMs.

The notion of organizational invariance has been introduced by David Chalmers (2011) in his theory of physical computation. The notion is defined relative to the causal topology of a system, which is "the abstract causal organization of the system: that is, the pattern of interaction among parts of the system, abstracted away from the make-up of individual parts and from the way the causal connections are implemented." (Chalmers, 2011, p. 339) A property is organizationally invariant if it is invariant with respect to the causal topology. In other words, any change to the system that preserves the topology preserves the organizationally invariant property. There is a causal topology of two IBMs that both share, and this topology can be used to define organizationally invariant properties of both. These invariant properties may involve more than causal topology responsible for the computation, as they also have other organizational properties related to the way they are connected to peripheral devices or used in the mainframe lab, for example.

Daniel Dennett's idea of substrate neutrality is similar to organizational invariance and also related to what is intuitive about computation being independent (but not entirely!) from the physical instantiation (Dennett, 1995, p. 50). Physically realized algorithms, according to Dennett, are substrate-neutral in that their causal power is related to their logical structure rather than to particular features of their material instantiation. Now, obviously, without physical instantiation they would not have any causal powers but not all causal powers are relevant for their being computational. In other words, we may safely abstract away from certain physical properties as long as the logical causal powers are still retained, or, to express the same idea in the vocabulary of organizational invariance, as long as the logical causal topology of the computational system is retained.

The computational structure of two IBMs—considered in any of the five ways presented in the section 2—can be retained while we change the physical substrate (in the case of the instantiation blueprint the difference is that IBM 709 does not share the same computational structure as IBM 7090 but there might be, for example, clones of IBM 7090 that do). Of course, not any substrate will be suitable, and because the topology of the whole system has to be retained, one cannot replace one tube with a transistor, or vice versa. But one can create a hybrid IBM 709' that contains a tube-based part and a transistor-based subsystem, as long as one creates a special interface to retain the causal topology of the original IBM. It would still run FORTRAN but with more exotic mixture of hardware.

The idea of substrate neutrality or organizational invariance is that there is a subset of all causal factors in the system which is essential in the system's computational capacity. In the mechanistic framework, the organizationally invariant part is the part responsible for whatever we think is the computational capacity, and one is free to adopt a certain level of abstraction in specifying the whole mechanism (Levy & Bechtel, 2013). The paraphrase of the intuitive point about MR will be therefore as follows: The abstract organization of the computational system remains organizationally invariant, or substrate-neutral with respect to a certain level of physical changes. IBM 709 and 7090, considered as physical mechanisms, simply share the same abstract organization, whether it's conceived of as the formal model of computation, the set of computable functions, the sequence of some primitive computable functions, some part of the instantiation blueprint, or the internal and external connections of the computer.

I suspect that many defenders of MR will retort that what I mean by substrate-neutrality is exactly what they mean by MR. That well may be, but this is a conceptual confusion on their part. Substrate-neutrality does *not* imply that there is an additional relevant difference between high- and low-level types that somehow makes cross-level type identification impossible. There is a level of substrate-neutrality for corkscrews, as long material used to build them is sufficiently stiff and so forth. Substrate-neutrality will co-occur with MR in some cases but it has less constraints. Brodman maps do not imply MR but they do imply some substrate-neutrality, as the latter implies that we abstract from some but not all physical detail in describing the relevant causal topology. And we do when we compare anatomical structures for the purposes of topographical mapping.

The acknowledgment of the same abstract organization is not threatened by adopting a reductive stance; irrespective of whether one treats the organization as irreducible or not, it will remain causally relevant, and hence, indispensable in attaining models of computational mechanisms that are both appropriately general and include necessary specific details. There is simply no reason for refining definitions of MR while we can express the point in a simpler manner, and without begging the question against reductionism. Substrate-neutrality is non-committal, as it does not exclude type-identity of computational and physical kinds. Granted, if one is interested in defeating type-identity and reductionism, then it is not attractive, but it was not my aim to show that computation is not reducible to the physical. To show that it is (nor not) is a task for another paper, and possibly for another author.

Acknowledgements

The work on this paper was financed by National Science Centre under the program OPUS, grant no. 2011/03/B/HS1/04563. The author wishes to thank Aaron Sloman for an extended discussion of his idea, to the audience at PT-AT 13, and to the anonymous referee of the previous version of the paper.

References

Aizawa, K., & Gillett, C. (2009). The (Multiple) Realization of Psychological and other Properties in the Sciences. *Mind & Language*, *24*(2), 181–208. doi:10.1111/j.1468-0017.2008.01359.x

Bechtel, W. (2008). *Mental Mechanisms*. New York: Routledge (Taylor & Francis Group).

Bechtel, W., & Mundale, J. (1999). Multiple realizability revisited: Linking cognitive and neural states. *Philosophy of Science*, *66*(2), 175–207.

Block, N. (1990). Can the mind change the world? In G. Boolos (Ed.), *Meaning and Method: Essays in Honor of Hilary Putnam* (pp. 137–170). Cambridge: Cambridge University Press.

Chalmers, D. J. (2011). A Computational Foundation for the Study of Cognition. *Journal of Cognitive Science*, (12), 325–359.

Craver, C. F. (2007). *Explaining the Brain. Mechanisms and the mosaic unity of neuroscience*. Oxford: Oxford University Press.

Craver, C. F. (2013). Functions and Mechanisms: A Perspectivalist View. In P. Hunemann (Ed.), *Functions: selection and mechanisms* (pp. 133–158). Dordrecht: Springer.

Cummins, R. (1975). Functional Analysis. *The Journal of Philosophy*, *72*(20), 741–765.

Davies, P. S. (2001). *Norms of nature: naturalism and the nature of functions*. Cambridge: MIT press.

Dennett, D. C. (1995). *Darwin's dangerous idea: Evolution and the meanings of life*. New York: Simon & Schuster.

Fodor, J. A. (1968). The appeal to tacit knowledge in psychological explanation. *The Journal of Philosophy*, *65*(20), 627–640.

Fodor, J. A. (1974). Special sciences (or: The disunity of science as a working hypothesis). *Synthese*, *28*(2), 97–115. doi:10.1007/BF00485230

Fresco, N. (2014). *Physical Computation and Cognitive Science*. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-41375-9

Gillett, C. (2002). The dimensions of realization: a critique of the Standard view. *Analysis*, *62*(4), 316–323.

Gillett, C. (2011). Multiply realizing scientific properties and their instances. *Philosophical Psychology*, *24*(6), 1–12. doi:10.1080/09515089.2011.559625

Glymour, C. (1994). On the Methods of Cognitive Neuropsychology. *The British Journal for the Philosophy of Science*, *45*(3), 815–835. doi:10.1093/bjps/45.3.815

Haimovici, S. (2013). A Problem for the Mechanistic Account of Computation. *Journal of Cognitive Science*, *14*(2), 151–181.

IBM Archives: 709 Data Processing System. (2003, January 23). Retrieved January 11, 2014, from http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_PP709.html

IBM Archives: 7090 Data Processing System. (2003, January 23). Retrieved January 11, 2014, from http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_PP7090.html

Keeley, B. L. (2000). Shocking Lessons from Electric Fish: The Theory and Practice of Multiple Realization. *Philosophy of Science*, *67*(3), 444–465.

Levy, A., & Bechtel, W. (2013). Abstraction and the Organization of Mechanisms. *Philosophy of Science*, *80*(2), 241–261. doi:10.1086/670300

Machamer, P., Darden, L., & Craver, C. F. (2000). Thinking about Mechanisms. *Philosophy of Science*, *67*(1), 1–25.

Malcolm, G. (1996). Behavioural equivalence, bisimulation, and minimal realisation. In *Recent Trends in Data Type Specification* (pp. 359–378). Berlin, Heidelberg: Springer. doi:10.1007/3-540-61629-2_53

Meyer, D. E., Osman, A. M., Irwin, D. E., & Yantis, S. (1988). Modern mental chronometry. *Biological Psychology*, *26*(1-3), 3–67. doi:10.1016/0301-0511(88)90013-0

Miłkowski, M. (2011). Beyond Formal Structure: A Mechanistic Perspective on Computation and Implementation. *Journal of Cognitive Science*, *12*(4), 359–379.

Miłkowski, M. (2013). *Explaining the Computational Mind*. Cambridge, Mass.: MIT Press.

Miłkowski, M. (2014). Computational Mechanisms and Models of Computation. *Philosophia Scientiæ*, *18*(3).

Piccinini, G. (2007). Computing Mechanisms. *Philosophy of Science*, *74*(4), 501–526. doi:10.1086/522851

Piccinini, G. (2010). Computation in Physical Systems. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Fall 2010.).

Polger, T. W. (2004). *Natural Minds*. Cambridge, Mass.: MIT Press.

Polger, T. W. (2008). Evaluating the evidence for multiple realization. *Synthese*, *167*(3), 457–472. doi:10.1007/s11229-008-9386-7

Polger, T. W., & Shapiro, L. A. (2008). Understanding the dimensions of realization. *Journal of Philosophy*, *105*, 213–222.

Posner, M. I. (2005). Timing the brain: mental chronometry as a tool in neuroscience. *PLoS biology*, *3*(2), e51. doi:10.1371/journal.pbio.0030051

Price, C. (2001). *Functions in mind: a theory of intentional content*. Oxford / New York: Clarendon Press.

Putnam, H. (1975). Philosophy and our mental life. In *Mind, Language and Reality: Philosophical Papers* (Vol. 1, pp. 291–304).

Shagrir, O. (1998). Multiple realization, computation and the taxonomy of psychological states. *Synthese*, *114*(3), 445–461. doi:10.1023/A:1005072701509

Shapiro, L. A. (2000). Multiple realizations. *The Journal of Philosophy*, *97*(12), 635–654.

Shapiro, L. A. (2004). *The Mind Incarnate*. Cambridge, Mass.: MIT Press.

Shapiro, L. A. (2008). How to Test for Multiple Realization. *Philosophy of Science*, *75*(5), 514–525. doi:10.1086/594503

Sober, E. (1999). The multiple realizability argument against reductionism. *Philosophy of Science*, *66*(4), 542–564.

Wilson, R. A., & Craver, C. F. (2007). Realization: Metaphysical and Scientific Perspectives. In P. Thagard (Ed.), *Philosophy of Psychology and Cognitive Science* (pp. 81–104). North Holland. doi:10.1016/B978-044451540-7/50020-7

Wimsatt, W. C. (2002). Functional organization, analogy, and inference. In A. Ariew, R. Cummins, & and M. Perlman (Eds.), *Functions: New essays in the philosophy of psychology and biology* (pp. 173–221). Oxford: Oxford University Press.