

On The Object-Oriented Version of The Zombie Argument and Its Implications on The Mind-Body Problem

A Polemic essay

by Shay David, sd256@cornell.edu

Motivation and plan of work¹

Physicalism is an ideology that contends that, in the last instance, all the entities in the world are physical and that the things people usually consider as being non-physical or mental are in-effect not more than a lasting phantasmagoria. Copernicus taught us that Man's Earth was not the center of the universe; he showed that what was common sense and the conclusion of simple observation (that the sun rotates around the earth) was actually not the case. Copernicus showed that Man has a less significant role in space-time than he would like to have; he is pushed to an auxiliary orbit. Physicalism aims at a second Copernican revolution – it de-centers Man even further; it tries to teach us that cognitive human activity is not a special phenomenon: it is merely a specific chemical-biological arrangement of matter that to some misinformed people creates an illusion of a categorically different sphere which they call the 'mental'. A serious investigation should supposedly free us of this folly and show us that what we call 'mental' is simply either a higher-level description of the physical, an auxiliary effect of it, or that it is syntactic sugar for describing the physical itself.

Physicalism claims that you, I, and all the other people we ever met are all physical chunks of matter. So is our consciousness, our causal mental powers, our intentional mental states (memories, thoughts, concepts, feelings, beliefs, desires, etc...) our virtues and morals, our ability to learn new things and our bodily sensations. The memory of your first kiss, the aptitude to play

¹ I'm indebted to Professor Richard Boyd for useful comments on earlier drafts of this paper.

chess or speak several languages, the ability to understand what a prime number is, the feeling of despair that fills you when your favorite football team loses, the capacity to hate someone or love your children, the desire to be loved, the urge we feel to put a dung-beetle back on its feet when we find it stuck on its back, the faculty for feeling pain, the ability to move one's limbs at whim, the very notion that we are thinking beings – are all, argues the physicalist, simply and exclusively either physical in and of themselves or consequences of the physical. In other words, 'mental' things, events, and states simply don't exist independently of the physical.

Now, some may think this is not a problem. Epiphenomenalists like T.H. Huxley (1874) think that metaphorically speaking, mental events are not more than the steam-whistle that contributes nothing to the workings of the physical locomotive. But although most philosophers of mind are physicalists (Sturgeon 2001:202) most physicalists do not want to defend Epiphenomenalism but, rather, want to maintain some sort of place for the mental (Crane 2001:208). They just don't know how to do it and uphold that the world is totally physical at the same time. To try and salvage the mental or at list explain it, physicalists have tried to explain mental states, properties and events in one of four ways: (1) *reductive materialism* (property or type identity theory) whereby the mental is seen as identical to the physical; according to this view materialism is reductive but not eliminative: for example, C-fiber firings are real, they are identical to pains, and, thus, pains are thought of as perfectly real, causally efficacious, mental entities (that are identical to physical ones); (2) *Non-reductive Physicalism* that views mental states and properties as distinct from the physical but dependant upon it (Crane 2001:208); according to this view the mental is either realized by or supervenes on the physical; (3) *Emergentism*, whereby "mental properties 'emerge' from complex arrangements of matter in a way that is inexplicable from the perspective of the sciences of matter (ibid); (4) *Functionalism*, whereby mental states are defined causally, and can be explicated using input-output matrices.

Kim (2001) finds that the two most acute problems for the physicalist are strongly tied together: mental causation and consciousness keep physicalists and their critiques awake at night

as both phenomena seem to deny easy reconciliation with the physicalist world view. “The problem of determinism,” Kim asserts, “threatens human agency, and the challenge of skepticism threatens human knowledge. The stakes are even higher with mental causation, for this problem threatens to take away both agency and cognition.” (2001:273). Another ardent objector to Physicalism, Roger Penrose writes: “Consciousness is part of our universe, so any physical theory which makes no proper place for it falls fundamentally short of providing a genuine description of the world.” (1996:8).

This debate has been going on for millennia, with renewed interest in the last thirty years, but without any side being able to make a conclusive argument. As Thomas Nagel puts it “The strange truth seems to be that certain complex, biologically generated physical systems, of which each of us is an example, have rich nonphysical properties. An integrated theory of reality must account for this, and I believe that if and when it arrives, probably not for centuries, it will alter our conception of the universe as radically as anything has to date” (Nagel 1986:51). I find that Physicalism is certainly not such an integrated theory of reality. Moreover, I find it evidently wrong; in my mind (no pun intended), Physicalism simply does not capture our very evident and basic cognizant nature that Descartes so arduously encapsulated in his *cogito*.

Following the sentiment of ideas developed by Chalmers, Kim, Nagle, Penrose, Searle and others, in this paper I attempt to add weight to the contention that Physicalism is indeed amiss and that the mental cannot be reduced to or be fully explicated by the physical except contingently. To substantiate this claim, I will look at a recent development in computing, the introduction of object-oriented programming (OOP), which presents a possibility to rebuild the controversial zombie argument in a new way that is arguably less prone to attacks by the standard criticism (namely, that when taken seriously functional zombies are metaphysically implausible). In the spirit of Turing and Searle, I want to accept, momentarily, the (untenable) functionalist premise that all human activity can indeed be recorded as a set of sensory-input/behavioral-output ‘functions’. Our computational zombie will encapsulate all these functions and try to imitate

human behavior without having any consciousness or causal mental powers. Our zombie is a purely physical entity, realized in silicon. What I plan to show using this thought-experiment of a computational, object-oriented zombie and the discussion that follows is that regardless of whether we can mistake the new zombie for a human or not, Physicalism reaches a dead-end. If I am right and this new kind of zombie is perceivable, then at least one of the following statements must be true (1) human mental ability cannot be reduced or otherwise be fully spelled out by physical entities alone, or (2) Physicalism implies Epiphenomenalism; mental events are caused by physical events in the brain, but have no effects upon any physical events. In either case Physicalism as most physicalists would like to defend it is shown to be wrong.

Some historical background

The rise of computational physicalism in the second half of the 20th century was stimulated, to a large degree, by developments in two fields: cybernetics and neurobiology. Newly constructed entities, Computers, so it seemed, were able to complete tasks that formerly required human cognitive skill. Computers that could play chess, translate text from one language to another, have semi-meaningful (text) conversations or understand stories—to name a few key areas, seemed within reach and these activities were suddenly and surprisingly not exclusive to sentient humans. In parallel, researchers were starting to understand the mechanisms by which the human brain interprets, stores, and uses information generally and the brain's neural mechanisms particularly. Taken together these two new paradigms brought to the forefront ideas that only a short period earlier, during the heydays of logical positivism, were considered to be unworthy *metaphysics*. The mind-body problem, everyone knew for years, was a Pandora's Box of philosophical conundrums that no serious philosopher should open—but due to advances in cybernetics and neurobiology, in the last thirty years, the old questions about the true nature of the human mind, its relationship to the body, and the meaning of consciousness re-emerged under the beam of a new investigative spotlight and demanded new attention.

This unholy marriage between computational conjectures and neurobiological speculations gestated two twin brainchildren which are the theoretical mirror images of each other: Strong AI (Artificial Intelligence) and a Computational Theory of Mind (CTM).

Searle characterizes strong AI:

According to weak AI, the principal value of the computer in the study of the mind is that it gives us a very powerful tool. For example, it enables us to formulate and test hypotheses in a more rigorous and precise fashion. But according to strong AI, the computer is not merely a tool in the study of the mind; rather, the appropriately programmed computer really is a mind, in the sense that computers given the right programs can be literally said to understand and have other cognitive states. In strong AI, because the programmed computer has cognitive states, the programs are not mere tools that enable us to test psychological explanations; rather, the programs are themselves the explanations. (Searle, 1980:417)

Chalmers writes of CTM:

The theory of computation is often thought to underwrite the theory of mind. In cognitive science, it is widely believed that intelligent behavior is enabled by the fact that the mind or the brain implements some abstract automaton: perhaps a Turing machine, a program, an abstract neural network, or a finite-state automaton. ... it is hoped that computation will provide a powerful formalism for the replication and explanation of mentality. (Chalmers 1996:1)

While Searle wants to attack strong AI and indeed does so very successfully with his 'Chinese Room' argument, Chalmers wants to defend it and finds that the ambitions of artificial intelligence and the centrality of computation in cognitive science are justified. But what is important to us here is not any specific detail in the debate but rather the general form of the argument (or its mirror). Put simply, the combined cybernetic/neurobiological argument to uphold computational physicalism and strong AI goes something like this:

Premises

- (1) Computers are capable of accomplishing tasks that formerly were considered to require human mental activity
- (2) Being physical artifacts, it is evident that computers have no mental states, properties or activities which are not physical.

Therefore,

(3) Functionally, humans, can be viewed as not more than biological ‘super-computers’

And, more importantly,

(4) Humans don’t have non-physical mental states. Just like in the case of computers which are physically (electronically) realized, any seeming human mental activity can (and should) thus be understood by looking at the underlying neurobiological infrastructure and its functionality.

Chalmers summarizes:

First, underlying the belief in the possibility of artificial intelligence there is a thesis of *computational sufficiency*, stating that the right kind of computational structure suffices for the possession of a mind, and for the possession of a wide variety of mental properties. Second, facilitating the progress of cognitive science more generally there is a thesis of *computational explanation*, stating that computation provides a general framework for the explanation of cognitive processes and of behavior. (Chalmers 1993:2)

In this debate, I am much more sympathetic to Searle’s and Penrose’s views than to Chalmers, but to in order to build the zombie argument below, I have to accept the possibility of strong AI and a coherent CTM, if just for a while. Before we proceed, though, let’s look at what Object Oriented Programming is all about.

Some basic concepts of Object Oriented Design and their metaphysics

Object Oriented Programming (OOP) and Object-Oriented Design (OOD) are software development methodologies that model the characteristics of abstract or real objects using classes and objects. Although object-oriented conceptions of the world are not new (both Frege and Russell explored them in detail) their practical and metaphysical meanings are mostly unnoticed by philosophers. The assumptions that OOP and OOD employ presuppose that the world can be modeled by objects. This represents a departure from earlier, procedural computer programming design patterns at least in regard to the metaphysical assumptions about the world it tries to module. While earlier computer systems and programming languages (like COBOL, FORTRAN,

PASCCAL, and C) focused on the flow of information from one ‘function’ to another and maintained global variables to ‘remember’ the state of the world including everything in it, in modern OOP and OOD software languages (starting with LISP and SmallTalk and evolving into Java and C#) each object has its own *properties* and *methods*. The properties accumulate the object’s state, and its methods account for its functionality, the ways in which it interacts with other objects and operates on them and on itself. In OOP there is no external ‘world’ and therefore no state of this world, and no global functions beyond the accumulated properties and methods of the objects in the world. An object oriented model, is therefore both easier to describe in pure functionalist terms, and closer to the way most people think about the real world. One could argue, of course, that the departure from earlier systems is not that radical. The Church-Turing theory teaches us that a Turing machine is computationally equivalent and just as powerful as any modern-day computer, and that since Turing machines can store states, there is no impediment to building full object-oriented computational models using principle Turing machines, or any later variations of them. However, there are two points to note: (1) Object oriented concepts underwrite all the dominant forms of software programming and design today. Very few, if any, serious new software endeavors consider procedural programming as an option. If we want the philosophical-scientific debate to be up to date, and in this case we surely do, since as we saw the renewed interest in Strong AI and CTM was triggered by recent scientific discoveries, we should aspire to frame the computational discussion in modern terms. (2) since the criticisms of the zombie argument that will surface later all hinge on the plausibility or implausibility of zombies, modern OO concepts help us see that zombies are indeed plausible. Yes, we could build a computational zombie using a Turing machine, but an instruction like “imagine an infinitely long tape” is an easy target to implausibility attacks, while a description of nested objects is much less vulnerable.

What, then, are the metaphysical assumptions behind the practice of OOP? The discussion below will try to understand these metaphysics and the ways they inform

programmatic concepts such as class, object, construction, destruction, realization, instantiation, inheritance, encapsulation, reflection, and polymorphism, which are every day terms used by people that practice OOP. Since Java is the most ubiquitous object oriented language today, I take the official definitions of these concepts from The Java tutorial by Sun Microsystems (2003).

Classes and Objects are the basic entities in an OO model. A class is a blueprint or prototype that defines the variables and the methods common to all objects of a certain kind. An object is a software bundle of related variables and methods. Software objects are often used to model real-world objects you find in everyday life. So anyone who chooses to program using classes and objects espouses a theory of natural kinds, and has a clear idea of what they entail:

...Real-world objects share two characteristics: They all have state and behavior. For example, dogs have state (name, color, breed, and hunger) and behavior (barking, fetching, and wagging tail). Bicycles have state (current gear, current pedal cadence, two wheels, and number of gears) and behavior (braking, accelerating, slowing down, and changing gears). Software objects are modeled after real-world objects in that they too have state and behavior. A software object maintains its state in one or more variables. A variable is an item of data named by an identifier. A software object implements its behavior with methods. A method is a function (subroutine) associated with an object. You can represent real-world objects by using software objects. You can also use software objects to model abstract concepts. For example, an event is a common object (Sun Microsystems, 2003)

Instantiation is the process whereby a specific object is generated by the blueprint of a class. An *instance* is a specific object of a particular class.

Using object-oriented terminology, we say that your bicycle object is an instance of the class of objects known as bicycles. Bicycles have some state (current gear, current cadence, two wheels) and behavior (change gears, brake) in common. However, each bicycle's state is independent of and can be different from that of other bicycles...In object-oriented software, it's also possible to have many objects of the same kind that share characteristics: rectangles, employee records, video clips, and so on. Like the bicycle manufacturers, you can take advantage of the fact that objects of the same kind are similar and you can create a blueprint for those objects. A software blueprint for objects is called a class (Ibid)

Metaphysically, therefore, classes are in essence a modern version of the Platonic forms. On Plato's view particular entities *aspire to participate* in certain universal, ideal entities which he

called the forms. From his “one over many” principle he concluded that a plurality of things may have a single ideal Form apart from the particulars themselves. In OO we find that classes are universals, objects are particulars, and just like things in the real world are particulars that participate in the universal forms so in OOP an object instantiates a class (or a form) to participate in it. In the argument between Plato and Aristotle about what comes first: the particular or the universal, OO designers are all on Plato’s side. There couldn’t be a particular without the universal template first.

How are objects created and discarded? **Constructors and Destructors** are pseudo-methods that create or destroy an object. The thing to note here is that constructors and destructors embody an assumption about the temporal existence of objects. An object doesn’t live forever; it has a clear time of creation (the time of the call to its constructor) and a limited lifespan (until the call to its destructor).

So far we have seen the basic entities of OOD which are sufficient to design and write a simple object oriented program but in order to capture the complexity of the real world OOD introduces a few more concepts. Classes can have hierarchical relationships to each other via the concept of **Inheritance**.

[OO] allow classes to be defined in terms of other classes. For example, mountain bikes, racing bikes, and tandems are all kinds of bicycles. In object-oriented terminology, mountain bikes, racing bikes, and tandems are all subclasses of the bicycle class. Similarly, the bicycle class is the superclass of mountain bikes, racing bikes, and tandems...Each subclass inherits state (in the form of variable declarations) from the superclass. Mountain bikes, racing bikes, and tandems share some states: cadence, speed, and the like. Also, each subclass inherits methods from the superclass. Mountain bikes, racing bikes, and tandems share some behaviors: braking and changing pedaling speed, for example. However, subclasses are not limited to the state and behaviors provided to them by their superclass ... You are not limited to just one layer of inheritance. (Ibid)

Inheritance is important to our discussion for two reasons. First, inheritance complicates the concepts of programmatic definitions which (unlike explanatory definitions) are based on explicating a concept’s causal role (functional definitions are examples of such programmatic definitions). An object which is an instance of an inherited class, instantiates at one and the same

time more than once class. As we will see later such programmatic definitions are the heart of a zombie's plausibility. Second, inheritance (together with Polymorphism that we'll discuss shortly) opens the way to build a rich model of the world that captures the variation among tokens and types but maintain their family lineage at the same time. But sometimes as we now, classifying items as participating in one hierarchical inheritance chain is problematic. A mushroom, for instance, could be considered as being both a plant and an animal. For these border cases OO introduces the concept of **Interfaces**. Unlike full inheritance, an Interface is a contract in the form of a collection of methods and declarations. When a class implements an interface, it promises to implement all of the methods declared in that interface, but it does not commit itself to the interface lineage. An object-oriented mushroom, in this case, could, therefore inherit from the class of 'living things' and implement the interfaces of both 'plants' and 'animals' in parallel. The assumption here, of course, is that an object can participate in more than one natural kind. OO Variation is accounted for not only at the natural kind level, but also at the functional level. **Polymorphism** allows one class to define multiple methods that realize a single functionality in different ways. This involves the ability of new representations to be defined as variations of existing representations, where new implementations are introduced but specifications remain the same such that a specification has many implementations (Alhir 1998). Metaphysically, what polymorphism entails is the idea that functional descriptions may not be complete without a description of the interaction an object has with other objects. An object's functionality might be polymorphic and can only be fully determined when all the environmental conditions are set.

The last three key concepts we need to discuss are **Abstraction**, **Encapsulation** and **Reflection**.

Abstraction involves the formulation of representations by focusing on similarities and differences among a set of entities to extract intrinsic essential characteristics (relevant common features) and avoid extrinsic incidental characteristics (irrelevant distinguishing features) in order to define a single representation having those characteristics that are relevant to defining every element in the set.

Encapsulation involves the packaging of representations by focusing on the hiding of details to facilitate abstraction, where specifications are used to describe what an entity is and what an entity does and implementations are used to describe how an entity is realized. (Ibid)

Reflection is perhaps one of the most intriguing concepts of OOP, and perhaps one of the driving forces for the creation of OO as a practice in the first place. “Reflection has long been studied in philosophy and formalized to some extent in logic. It arised [sic] naturally in artificial intelligence, where it is intimately linked to the end goal itself: reflection is viewed as the emergent property responsible, at least in part, for what is considered an ‘intelligent behavior’.” (Demers and Malenfant 1995). Computational reflection is the ability of a computational model to ‘reflect’ on itself, and to become aware of its state. In OOP, reflection means that the object-system becomes a meta-systems that is causally connected and is able to reason and act upon itself. Reflective objects can know what class (natural kind) they instantiate, which interfaces they implement, and what are the methods at their disposal. OO Reflection, then, is a key concept that illuminates our discussion of ‘intelligent’ objects.

We are now ready to construct our object-oriented zombie.

The Zombie argument and a template for a Zoombie

The principle structure of the basic zombie argument is this:

Premise:

- (1) Assume that Physicalism is right, and that some functionalist account of human behavior, including all its mental aspects holds.
- (2) Suppose the existence of a human H and a zombie Z who function in exactly the same way and are functionally the same as each other in every possible world. Z is a functionalist copy of H.
- (3) If (2) is right than an outsider cannot tell the difference between H and Z

Now note that

- (4) Zombies by definition are bodies without minds and are, therefore, by definition not conscious (and have no other mental states, properties, or events)
- (5) We assumed (2), so therefore, H and Z are indiscernible.

Therefore:

- (6) One of the following must be true
- a. By perceiving a zombie we have conceptually shown that human activity can be reduced to a mindless, consciousnesses-absent account. If Physicalism is true it implies Epiphenomenalism holds for humans as well as zombies.
 - b. Since we evidently know that we humans are conscious, and we know zombies are not, than if Physicalism (Functionalism) is true consciousness is shown to be contingent.

But either part of (6) leads to the inescapable conclusion:

- (7) Functionalism (or any other physicalistic account, for that matter) is false – the functional account of humans can account for consciousness only contingently.

As the argument presumably shows, we can easily imagine a zombie that is not identical to a human. We know that humans have mental abilities, so if the mental and the physical are identical how can we imagine having something physical that isn't mental? Physicalism must be wrong.

Rejections of this argument hinge on a putative circularity of statement (4) which assumes that zombies have no mental states. This seems to assume what we are trying to prove without showing that the existence of zombies is indeed a real possibility. Following Shoemaker Chalmers argues (against Searle and Block) that the idea of a Functional Zombie is “implausible at the very least.” (1993:7) Trying to eat the cake and leave it whole, he goes on to say that

Shoemaker was arguing against the possibility of “Absent Qualia”, or Functional Zombies. These are supposed to be arguments against functionalism, demonstrating that we might duplicate any given functional process perfectly without any accompanying subjective experience. Shoemaker argues that we *know* about qualia; our knowledge is caused by the existence of these qualia; therefore qualia play some causal role. The

conclusion is that Absent Qualia are impossible, as removing qualia would mean changing the causal structure. This is generally construed as an argument for the third-person approach (for functionalism, in particular). Nevertheless, I am sympathetic with this argument and with functionalism, and believe that Absent Qualia are impossible. ...

Both functionalism and epiphenomenalism have plausible aspects, but neither gives a complete account of the problems of consciousness. I will suggest that a correct theory of consciousness shares aspects of both functionalism and epiphenomenalism (1993:11)

But in a footnote Chalmers gives us a hint as to what might be a way out of this circularity, and what can be a beginning of an argument for absent qualia “It should be noted that the implausibility of Functional Zombies does not necessarily imply the impossibility of Computational Zombies.” (1993:11) Acting on this hint in order to escape the lethal circularity critique let us perform a thought-experiment in which we imagine a different kind of zombies. We will construct a blueprint for computational, object-oriented zombies, that we will call **Zoombies**².

Zoombies, unlike functional zombies are a very plausible possibility. In fact Turing’s original “Turing Test” framework, and Searle’s Chinese room example are the proposals for a computational zombie thought-experiment. What is not clear in Turing’s or Searle’s respective frameworks is how such systems should be designed in practice. Neither Turing, nor Searle give design principles beyond saying the systems would employ symbolic manipulation (Searle does consider specific AI programs developed at several universities as counter-examples). In Turing’s and Searle’s spirit in our experiment too we will interact with our zoombie indirectly but by exchange of texts (it doesn’t matter if this exchange is digital or analog). The addition in this new thought-experiment is the object-oriented nature of such zoombies. As we saw before, unlike

² The reader that disproves of the concept of computational zombies to begin with because he believes that a computational process cannot represent a body without a mind in the first place, should think of object-oriented ghosts, mirror images of zoombies, instead. Zombies, bodies-without-minds, and ghosts, minds-without-bodies present the same challenge to the physicalist. We can construct the whole argument in terms of ghosts and not zombies, with similar results.

procedural computation, OOD gives us powerful tools to model the world as we experience it, with nested levels of complexity. This should alleviate implausibility critiques.

So how should a zombie be designed to make it plausible? Remember, we are working under the functionalist assumption that human mental states, events, and properties are functionally exhaustible (I take functionalizing mental abilities to be in the sense described by Kim, 2001:279). We will use the concepts of OOD and OOP that we saw in the previous section to design our zombie template.

First and foremost, we note that a zombie is an object. Before we can instantiate a zombie, we need to design its form, its class. The zombie class *inherits* from the Object class, the basic class of all. We thus free ourselves of any requirements of what a zombie might be. But there is one twist. The zombie is not constructed as a zombie. It is constructed as a human. So in essence we are not designing the class ‘zombie’, but rather class ‘human’. Any functionality the zombie will acquire, will be *encapsulated* in the design of its class. The zombie will have other objects as properties, and its functional tendencies will be realized by its methods. How will we make sure, then, that the zombie is ‘function-for-function’ the same as a human? By implementing *interfaces*. We shall now turn to the huge table of functions the functionalists prepared for us, and for each function define an interface. By way of example: our zombie class will implement the interface FeelsPain. FeelsPain defines several methods: IsInPain?, ExhibitsPain, RespondsToPain, etc. The details of implementation that we will have to define depend on the same functional matrix, only now, when writing the implementation code, we will look at the columns that define the usual human behavioral output. So, for instance, to implement the method RespondsToPain, we will look at the functional matrix, and see that the human response to pain is to avoid the cause of pain, and we will program the zombie to do the same. In much the same way we will define interfaces such as PlaysChess, SpeaksChinese, WantsToBeLoved, KnowsGoodAndEvil, etc. Depending on our functionalist world view (and the humans we use as our model), we might even go as far as to define and implement interfaces such

as BelievesInGod or IsRepublican. In cases where our functionalist account is vague, we could implement the interface in a *polymorphous* way, and we can delay implementation details to execution time wherein more data will be available.

As said earlier, the zoombie class we'll have properties in the form of other encapsulated objects. In this way lower level functionality can be *abstracted* and we can develop almost infinitely complex nesting. Encapsulated objects could, at our will be designed to inherit functionality from other classes. We thus have the ability to represent the richness of natural kinds we find in the world. An example of an encapsulated object like this is Body. Body is a complex class itself, which encapsulates objects like Head and Limb and implements interfaces such as AgesWithTime and NeedsFoodAndDrink. To understand the relationship between zoombie and Body, the concept of reflection would come in handy. Since zoombie is reflexive, it will have an easy time handling its own complex properties. When asked 'what are you' the zoombie will reflect, and answer 'I am a human' since it would investigate its own properties and methods and find them to be of the type human.

Last but not least, we need to implement the two special pseudo-methods human-Constructor and human-Destructor. The constructor will assign values to the zoombie's properties when it will be instantiated. The constructor will decide, for instance, what value to assign to the primitive properties Age and Gender, and to the complex property LearnsNewLanguagesEasily. The destructor will do clean-up operations and clear our zoombie from the host computer's memory.

All we have to do now in order to finish our thought-experiment, is imagine one of Searle's rooms, or one of Turing's test complexes, imagine a sufficiently powerful computer in it, and give the computer the instruction to instantiate one object of the 'human' (zoombie) class. If we followed the instructions above carefully enough, we should have a zoombie that would pass a Turing test, and speak Chinese. Or so the functionalist must admit.

Conclusion

As the last section showed, zombies are a plausible possibility that demonstrate an imaginable silicon-and-plastic entity which is qualia-absent in any sense physicalists are likely to defend. And this means the functionalist is in trouble, since he cannot escape the absurdity of this conclusion. He must admit that either the ‘mental’ has no real role in our world, or that functionalizing the mental is not a real possibility to begin with. I, of course, believe in the latter.

As Searle and Penrose write:

No purely formal model will ever be sufficient by itself for intentionality because the formal properties are not by themselves constitutive of intentionality, and they have by themselves no causal powers except the power, when instantiated, to produce the next stage of the formalism when the machine is running. And any other causal properties that particular realizations of the formal model have, are irrelevant to the formal model because we can always put the same formal model in a different realization where those causal properties are obviously absent. (Searle 1980)

The cognitive capabilities of human beings are not based on any knowably sound algorithm or algorithms and thus the essential features of the human mind will never be explained computationally. (Penrose 1994)

So where does this leave us? I think Kim was right to quote Schopenhauer who famously called the mind-body problem a ‘Weltknoten’ – a ‘World-Knot’ (2001:271). As Kim shows the strings of this knot are so tightly tied as to suggest there is no meaningful solution to the problem, at least in physicalist terms. Perhaps what we need is simply to once again cut the Gordian knot. Indeed, when Descartes tried it almost four centuries ago, his sharp sword turned out to be double edged; the Cartesian legacy bled to death in the battles to defend dualism. But at the same time physicalism leads us nowhere. Perhaps it is time to once again free the human mind from the body—at least until science can better explain the mind-body connection.

References

- Chalmers, D., "Consciousness and Cognition" Unpublished essay. 1993.
Available on line at < www.u.arizona.edu/~chalmers/papers/c-and-c.html >. Last visited December 1st 2003.
- "Does a Rock Implement Every Finite-State Automaton?", *Syntheses* 108:309-33, 1996.
Available online at < <http://www.u.arizona.edu/~chalmers/papers/rock.html> >. Last visited December 1st 2003.
- "A Computational Foundation for the Study of Cognition" Unpublished. 1993.
Available on line at < <http://www.u.arizona.edu/~chalmers/papers/computation.html> >. Last visited December 1st 2003.
- Crane T. "The significance of emergence," in Carl Gillett and Barry Loewer (eds.) *Physicalism and Its Discontents*, Cambridge: Cambridge University Press, 2001.
- Demers, F. and J. Malenfant, "Reflection in logic, functional and object-oriented programming: a Short Comparative Study" In IJCAI '95 Workshop on Reflection and Metalevel Architectures and their Applications in AI, pages 29--38, August 1995. Available online at < <http://citeseer.nj.nec.com/cache/papers/cs/3200/http:zSzzSzwww.iro.umontreal.ca/Sz~demers/Szjcai95.pdf/reflection-in-logic-functional.pdf> >. Last visited December 1st 2003.
- Huxley, T. H. "On the Hypothesis that Animals are Automata, and its History", *The Fortnightly Review*, n.s.16:555-580, 1874. Reprinted in *Method and Results: Essays by Thomas H. Huxley* New York: D. Appleton and Company, 1898.
- Kim, J. "Mental Causation and Consciousness: The Two Mind-Body Problems for the Physicalist," in Carl Gillett and Barry Loewer (eds.) *Physicalism and Its Discontents*, Cambridge: Cambridge University Press, 2001.
- Nagel, T. *The View From Nowhere*. Oxford: Oxford University Press, 1986.
- Penrose, R. *Shadows of the Mind: A Search for the Missing Science of Consciousness*. Oxford: Oxford University Press, 1996.
- Alhir S. "The Object-Oriented Paradigm," Unpublished essay. 1998. Available online at < <http://home.earthlink.net/~salhir/TheObjectOrientedParadigm.PDF> >. Last visited December 1st 2003.
- Searle, J.R.. "Minds, Brains and Programs," *Behavioral and Brain Sciences*, 3: 417-424, 1980.
Available online at < <http://members.aol.com/NeoNoetics/MindsBrainsPrograms.html> >. Last visited December 1st 2003.
- Shoemaker, S. "Functionalism and Qualia," *Philosophical Studies*, 27: 271-315. 1975.
- Sturgeon S. "The roots of reductionism," in Carl Gillett and Barry Loewer (eds.) *Physicalism and Its Discontents*, Cambridge: Cambridge University Press, 2001.

Sun Microsystems. "The Java Tutorial: Object-Oriented Programming Concepts" Available online at < <http://java.sun.com/docs/books/tutorial/java/concepts/> >. Last visited December 1st 2003.