Abstractions and implementations: a computer science perspective on emergence, causality, and multi-level autonomy

Russ Abbott – Russ.Abbott@Gmail.com
Department of Computer Science, California State University, Los Angeles, Los Angeles, Ca. 90032 USA

Abstract. Fundamental to Computer Science is the distinction between abstractions and implementations. When that distinction is applied to various philosophical questions it yields the following conclusions.

- *Emergence*. It isn't as mysterious as it's made out to be; the possibility of strong emergence is not a threat to science.
- *Causality and interactions*. Physical interaction among higher-level entities is illusory. Abstract interactions are the source of emergence, new domains of knowledge, and complex systems.
- *The relationship between physics and the special sciences*. The new domains of knowledge derived from abstract interactions are the basis of the autonomy of the special sciences.
- *Downward causation*. It's a zombie idea that should have a stake put through its heart and be replaced by downward entailment.

Contents

A. Introduction

Question: what do emergence, downward causation, and the relationship between physics and the special sciences have in common?

Answer: the multi-level mystery.

Fodor (1997) famously put the special sciences version of the multi-level mystery like this.

> The very *existence* of the special sciences testifies to reliable macro-level regularities that are realized by mechanisms whose physical substance is quite typically heterogeneous. Does anybody really doubt that mountains are made of all sorts of stuff? Does anybody really think that, since they are, generalizations about mountains-as-such won't continue to serve geology in good stead? Damn near everything we know about the world suggests that unimaginably complicated to-ings and fro-ings of bits and pieces at the extreme *micro*-level manage somehow to converge on stable *macro*-level properties.

> On the other hand, the 'somehow' really is entirely mysterious. [Why should there be (how could there be)] macro-level regularities at all in a world where, by common consent, macro-level stabilities have to supervene on a buzzing, blooming confusion of micro-level interactions? …
>
> So, then, *why is there anything except physics*? … Well, I admit that I don't know why. I don't even know how to *think about* why. I expect to figure out why there is anything except physics the day before I figure out why there is anything at all. …
>
> The world, it seems, runs in parallel, at many levels of description. You may find that perplexing; you certainly aren't obliged to like it. But I do think we had all better learn to live with it.

I am not a professional philosopher. My PhD is in Computer Science, and I teach in a Department of Computer Science.[1] In computer science we don't find a world that runs on many level perplexing. In fact, we like it that way. In this paper I'll explain why—and why that might be of interest to philosophers.

1. The computational turn

The International Association for Computing and Philosophy (IACAP) defines its mission as promoting "scholarly dialogue on all aspects of the computational/informational turn and the use of computers in the service of philosophy."[2] What is the "computational turn?" My sense is that it has to do with how we computer scientists tend to think and the possible uses that style of thinking might have in philosophy.

Well, how do we think? In describing how we think I'll be using the term *abstraction* a lot. *Abstraction* is used in a number of ways in computer science. It's often said that abstraction is one of the core concepts of computer science. In this paper I'll be focusing on one particular sense of the word. I'll be using *abstraction* to mean an idea that has been fairly well thought out and pinned down but not necessarily specified in formal detail. An abstraction in this sense is something for which one has a fairly rigorous but not necessarily mathematically formal specification.[3]

The primary work of software development is (a) to identify the abstractions relevant to a domain of interest and then (b) to implement those abstractions as software. An example familiar to nearly everyone are the abstractions inherent in a word processing program such as Microsoft Word. Word processors offer users the ability to create and manipulate documents. They offer this capability by making available the use and manipulation of such abstractions as: word, paragraph, footnote, section, etc. We all have a fairly clear idea of what we mean by these words, but we do not have a mathematically formal specification for any of them.[4]

---

[1] I have had a long-standing interest in questions on the boundary of Computer Science and Philosophy and have presented and published papers in this area. See, for example (Abbott, 2008, 2009, 2010a, and 2010b).

[2] From the mission page of the IACAP website: http://www.iacap.org/about/the-mission-of-the-iacap/.

[3] The other primary uses of the term *abstraction* in computer science are (a) the noun form of the verb *to abstract*, which we as software developers tend to do a lot, and (b) what one produces by abstracting.

[4] One of the consequences of writing software that implements abstractions is that in the context of that software the abstractions are pinned down further. The details of what one means by *word*, *paragraph*, etc. are settled by the word processor. The decisions made by the software developer may not agree with every user's intuition, but there is no doubt what the software means by *word, paragraph*, etc.

One of the most important principles of modern software development is that the implementation of an abstraction should be invisible to users of the abstraction. People who use Microsoft Word do not know how it works internally—nor should they have to know. Modern software development does its best to build a conceptual firewall between the abstractions that software makes visible and the implementation of those abstractions.

2. Relationship to philosophical functionalism

As I said, this paper is built to a great extent on the two notions of abstraction and (conceptually quarantined) implementation. This may sound like an echo of functionalism. For example, (Putnam, 1975) includes the following.

> [Two] systems can have quite different constitutions [e.g. they might be made of *copper, cheese, or soul stuff*] and be functionally isomorphic.

To illustrate his point further Putnam asked how one should explain why a square peg can't fit into a round hole (of incompatible size). Should the explanation be given in terms of quantum mechanics or geometry? Putnam argued for geometry, pointing out that any explanation based on quantum mechanics can deal with only one specific peg, one specific hole, and one specific orientation of the peg with respect to the hole, etc. A geometrical explanation is much more general and hence more useful. Although he didn't say so explicitly, in arguing for the geometric explanation Putnam was making the case for abstractions: squares and circles. As long as the peg and hole implementations faithfully produce objects with the relevant square and circle properties, it makes more sense to discuss the insertion problem in terms of abstractions than in terms of any specific implementation.

Three important differences between the approach I am advocating and functionalism come to mind. (a) Functionalism was intended as a theory of mind; software, of course, has nothing to do with theories of mind. (b) Functionalism tended to be descriptive; software development tends to be constructive. By this I mean that in software development one is continually building new abstractions by making use of existing abstractions. The sense that one can build a continuingly growing edifice of abstractions was not among the concerns of functionalism. (c) Functionalism tended to describe a relatively static world; one of software's most important properties is that something actually happens when software is executed by a computer. Software has two aspects: what it looks like as text and what it does when executed. Functionalism was not concerned about this sort of dual identity. I will be discussing all three of these aspects of software later in this paper.

A word about *implementation* vs. *realization*. In the functionalism literature the term *realizes* is used in much the same way as *implements* is used in software. In both cases one is talking about a concrete embodiment of an abstraction. A difference, though, is that *to implement* implicitly incorporates a sense of intentionality whereas *to realize* doesn't. A software developer deliberately implements an abstraction as software. Nature realizes some (apparent) functionality with no intentionality implied. This doesn't make a naturally occurring realization less faithful to the abstraction it realizes than a programmer-produced implementation. Although nature is not intentional, a naturalized version of *function* is useful for understanding how evolution produces the complex (apparent) functionality it does. Either way, whether an implementation/realization is explicitly intentional or not the result is a concrete embodiment of an abstraction.

3. Abstractions, types, and kinds

The more formal term used in computer science for what I've been calling an abstraction is *type* (also *data type* or a*bstract data type*). Originally a type in software referred to a collection of possible values. Typical examples were the integers, Boolean values, characters, strings (of characters), etc. These types corresponded fairly closely to bit representations of data. As the discipline matured, we learned more about types as ways to organize our thinking about the information programs manipulate. An important step was the development of what were called enumerated types. These are types that are not associated with particular bit representations but that identify sets of possible symbolic values. Features were added to programming languages to allow programmers to declare their own enumerated types. A typical example might be the declaration of a type *traffic-light-color* consisting of the values *red, yellow, green*. Values in types of this sort were explicitly not linked to bit representations. They were understood simply as values, which were referred to by programmer-specified names. This was probably the first step toward type abstraction in software.

As computer science continued to mature, it became clear that the most important property of a type was the sorts of things one could do with instances of that type. Without associated operations, a type is simply a list of opaque values. Types with operations were originally called *abstract data types*. The standard example (Liskov, 1974) is the (last-in-first-out, i.e. LIFO) *stack*. The operations declared for stacks were: push a value onto a stack, pop a value off a stack, examine the top element of a stack, and ask whether a stack is empty. These operations were defined formally through equations. For example, one can write an equation that says that a push followed by a pop leaves the stack as it was before the push.

In parallel, much formal work was done on type systems. The most widely cited, the Hindley-Milner type system (Hindley, 1969) and (Milner, 1978), provides the theoretical underpinning for the Haskell programming language (Peyton Jones, 2003).

The upshot is that all modern programming languages take types seriously. Types and their importance are now taught as part of the standard computer science curriculum beginning in the first computer science course.

4. Abstractions, implementations, and the ship of Theseus

The paradox of the ship of Theseus offers a nice illustration of how thinking in terms of types or abstractions can be helpful. The question famously concerns the ship of Theseus, which was moored and maintained in the port of Athens. In the course of caring for the ship, each piece of the original was replaced—or at least that's how the story goes for the sake of the paradox. Is the resulting ship the same ship as the original?

The paradox arises because the ship as an abstraction is being confused with the ship as an implementation. For the purposes of honoring Theseus the Athenians cared about a ship of the overall conformation and appearance as that used by Theseus—not about a relic, i.e., a ship as particular pieces of wood. Were that not the case, then any modification of the ship would dishonor Theseus. But the Athenians did not consider maintenance dishonor and were not concerned about particular pieces of wood. But they wanted more than just the idea of the ship. They wanted an implementation, something

concrete that they could look at and touch. So, is the ship whose component parts have been completely swapped out the same as the original? Thinking in terms of types and abstractions lets us say that the abstraction is the same, but the implementation is different.

In software development the ship of Theseus would be known as a singleton. A singleton is an instance of a type that allows no more than one instance to be created. What would happen if, as the paradox is sometimes extended, the replaced pieces were put together as a second implementation. In a software system, this would violate the singleton rule of the Ship-of-Theseus type. The software would signal an exception condition—which means that the program would stop its normal course of operation and do what it can to handle the problem. I don't know what would have happened had this occurred in Athens.

In philosophy the term *kind* is often used for the same basic idea as *type* in computer science. (*Type* is used in philosophy as well.) For the remainder of this paper I will use the term *abstraction* for the common core of these ideas.

5. What this paper is about

Because our job as software developers is the creation and implementation of abstractions, we are typically skilled in understanding both (a) what it means to define new levels of abstraction[5] and (b) what it takes (and whether it is even possible) to implement those new abstractions in terms of existing abstractions. One of my goals in this paper is to apply our approach to creating and embodying abstractions to philosophical issues. In particular, I will demystify the multi-level mystery. I'll explore abstraction and implementation in emergence, in the special sciences, and in downward causation.[6] I'll also have a number of other things to say about these areas. In particular, I'll be dismissive of strong emergence—and eventually of emergence in general. I'll propose a way of settling the debate about whether the laws of physics entail the laws of the special sciences, and I'll suggest downward entailment as a better conceptual framework for understanding phenomena that are sometimes presented as examples of downward causation.

A disclaimer: although the issues I discuss are often associated with the theory of mind, this paper is not about mind or consciousness. I don't believe we know enough about consciousness to be able to say anything useful about it—other than that it's a puzzle we have hardly begun to unravel. Notwithstanding my reluctance to address the issue of consciousness directly, the subject arises in a few places in the paper.

---

[5] A level of abstraction is a collection of interrelated abstractions of sufficient completeness and usefulness to characterize a domain. The abstractions in Microsoft Word constitute a level of abstraction in this sense.

[6] The terms *downward causation* and *top-down causation* are often used synonymously in the literature. My preference is for *downward causation* because it does not imply that one is starting from a top-most point—as *top-down <anything>* would. However, one of the primary references for this paper uses *top-down causation* rather than *downward causation*. I will use the two terms interchangeably and intend no difference between them.

B. Emergence

Emergence has been on the philosophical stage for over a century and a half. McLaughlin (1992) traces it back to work by Mill in the mid-19th century. Emergence is one of a web of ideas associated with what has come to be known as complex systems.[7] O'Connor and Wong (2012) characterize emergence this way.

> [E]mergent entities (properties or substances) 'arise' out of more fundamental entities and yet are 'novel' or 'irreducible' with respect to them. … Each of the quoted terms is slippery in its own right, and their specifications yield the varied notions of emergence.

Emergence appears to be something of a free lunch. Put a bunch of familiar things together and something new (and surprising and interesting and perhaps even useful) may pop out. In the late 20th century emergence seemed to represent much of what was good about complex systems. The term became very popular. However, by the turn of the century the terms *complex system* and *emergence* had become so overused and faddish that they lost much of their meaning. I stopped using them—although I'll relent for this paper.

In the philosophical literature emergence has remained something of a mystery. As recently as 2008 the introduction to (Bedau and Humphreys, 2008) says that "the very idea of emergence seems opaque, and perhaps even incoherent." Perhaps because it seems so mysterious, philosophers have sliced and diced it to the point that I hardly recognize some of its uses. The primary divisions are weak and strong emergence. But there is also nominal emergence, pattern emergence, synchronic and diachronic emergence, static and dynamic emergence, epistemological and ontological emergence, probably others that I'm forgetting, still others of which I am not even aware, and various combinations of these. I would like to step back from all that and take another look.

I have always thought that what was most important about emergence was the idea that a new and conceptually intellectually autonomous domain can be created from elements that have nothing to do with that domain. To take one of the standard unity-of-science examples, biology "emerges" from chemistry, yet the study of biological phenomena is in many ways autonomous of chemistry. In particular, the most fundamental idea in biology—evolution through diversity and selection—has no counterpart in chemistry and cannot be expressed using concepts from chemistry.[8]

It also became clear to me that what software developers do for a living is to produce emergence. For example, through the efforts of team of programmers, Microsoft Word "emerged" from the collection of lower level operations—i.e., the "code"—in terms of which it is implemented. Because emergence in this sense is so well understood in the context of software I believe that it can be explained in relatively straightforward terms in other contexts as well.

---

[7] Other ideas related to complex systems include agent-based modeling, evolutionary processes, and various network topologies and effects.

[8] Of course we now know—although Darwin didn't—that evolution depends on DNA, which is understood chemically, but that's a matter of implementation, not conceptualization.

1. Emergence without the mystery

In software the term *conceptual model* refers to the collection of ideas that define how a software application operates. Consider the example of a word processor mentioned earlier. Its conceptual model, i.e., the level of abstraction it defines, is essentially what its *Help* system says about it. The *Help* system describes the program in terms of the abstractions that are relevant to how the user thinks about what the program can do.[9] These abstractions—e.g., words, documents, tabs, rulers, ribbons, styles, formats, tables of contents, footnotes, possible errors (indicated, perhaps, by red squiggly underscores), etc.—refer to types (e.g. paragraph), relations (e.g., orphan-line rules for how paragraphs should be formatted on pages), and operations (e.g., how one can drag a selected piece of text from one place to another, or correct a spelling error).

Standard (and best) practice in software development is to define a conceptual model independently of how it has been or will be implemented. Imagine writing a *Help* system before starting to write the program that it describes and then writing the program to be consistent with what the *Help* system says. In other words, a conceptual model is autonomous of and often precedes its implementation.[10]

The autonomy of some collection of entities, properties, etc. from the elements from which they "arise" is, of course, one of the distinguishing features of so-called emergent phenomena. In the case of software, the elements from which abstractions "arises" are the things one can write in a programming language. The conceptual model "arises" when the programmer puts those elements together in the right way.

The creation of something new from something old is in some ways magical. How is it possible to build a word processor by putting together lines of code in a programming language that has no word processor concepts? Yet we do this sort of thing all the time. Every creative act—the composition of music, the authoring of a work of fiction, the construction of an architecturally original building, the evolution of a species with a new way to survive in the world, etc.—illustrates that same magic. Human beings and nature both do it. It is this sort of constructive creativity that I think is important about emergence. Creativity may be magical and mysterious; implementation is not.[11]

Although we often speak as if emergence is about higher-level (macro) elements emerging from lower-level (micro) elements, that's not always the case. Not all emergent phenomena are implemented in terms of lower level or micro elements. The (emergent) lifecycles of parasites—and there are probably as many species of parasites as non-parasites—are built from elements that are macro to their micro. *Cordyceps* is a particularly interesting example. *Cordyceps* is a species of fungus that infects a species of ants. Infected ants (are somehow induced by the fungal infection to) climb up and "clamp onto a leaf vein about 25 centimeters off the ground—a spot where the humidity and other conditions [tends to] be ideal for a fungus to grow." (Zimmer 2011) Once the ant clamps onto a leaf, the fungus attacks the muscles that would allow the ant to release its grip—dooming it to stay clamped onto the leaf. The

[9] Or at least how the author of the *Help* system thinks the user thinks or wants the user to think.

[10] This is similar to, but not exactly the same as, what is called defining the requirements of a system.

[11] A longer discussion is available in Abbott (2010b).

fungus then produces a stalk, which releases fungal spores, which drift to the ground below and infect other ants.

Cordyceps, like every parasite, exploits its host's resources for its own needs. But *Cordyceps* exploits its host's *functionality*[12]—e.g., the ant's ability to climb a plant stem and to clamp onto a leaf. *Cordyceps*' lifecycle is emergent; it didn't exist before *Cordyceps* invented it. (Of course that's true for virtually any species.)

The description *Cordyceps*' life cycle involves both higher and lower level elements. The higher level elements are plants and ants and their properties and behaviors. The lower level elements include the biochemical reactions that result in the observed ant behavior. So *Cordyceps* implements its strange life-cycle by putting together the capabilities of both higher-level and lower-level elements.

I want to emphasize that *Cordyceps* relies on the macro-properties and macro-capabilities of ants and plants. Unlike many decomposer species *Cordyceps* doesn't use ants simply as biochemical fuel, i.e., for the chemical properties of the stuff from which they are made. *Cordyceps* relies on ants' ability, as ants, to climb plant stems and to clamp onto plant leaves. For this step in its life cycle the fungus needs a way to position itself about 25 centimeters above the ground. It uses the macro properties and capabilities of ants to do this. It also needs a stable platform while it grows its stalk. Again it uses the macro properties of ants (their mandibles and disabled muscles) and plants (their leaves and veins) to achieve this. Although *Cordyceps* is quite micro compared to ants and plants, it builds its (emergent but micro-level) lifecycle upon the macro-level properties of pre-existing macro-level entities.

We see a similar sort of emergence in our technology. Consider smart phones. Many of their components are pre-existing, but the phone's properties and functionalities are new. The phone is as emergent from its base of pre-existing elements as anything one can point to. Like *Cordyceps,* the "base" from which smart phones "arise" include elements that cannot all be said to be micro to the phone's macro. The telephone network and the GPS satellite system are both macro compared to a smart phone. Yet these are both pre-existing elements of the smart phone's "base."

Other widely cited examples of emergence include the hardness of diamonds, flocks of birds, traffic jams, a steel canoe (it floats even though its parts don't), and biological life. In other words, emergence is all around us. Emergence appears when existing things are arranged in new ways. My claim is that this is exactly all there is to it: the putting together of existing things and capabilities to get new things and capabilities. The creativity that produces emergence may amaze us. But when examined more closely emergence is simply a matter of implementing new abstractions by making (often) clever use of what

[12] Here and elsewhere, when I speak of functionality I intend it in a naturalized sense. One can describe ants in terms of how they behave and the functions that behavior achieves without implying that these functions are teleologically driven. A function in this sense is simply a behavior that produces a given result. There is no claim that either the behavior or the result has an intentional component. For example, one of the functional consequences of the earth spinning on its axis is the parade of alternating days and nights. Yet there is no claim that the earth is spinning in order to produce this result—or that the earth or its spinning could intend anything at all.

already exists. So what sorts of things are emergent? Everything except the fundamental particles of physics. If that trivializes the notion of emergence, so be it.

Even if (as I hope) the preceding has demystified emergence—perhaps only by trivializing it—a number of questions remain. (a) What mechanisms enable emergence to occur? That is, what makes it possible to put existing things together in a stable way? We know the answer for atomic physics and chemistry, but what about other sorts of entities? (b) Even though we may understand how emergent entities are constituted, why do they come to exist at all? and (c) Which emergent elements will persist? Appendix A proposes a framework for approaching these questions. Why am I putting it an appendix? Although the *how* of emergence is very important, the proposed framework is not inspired by the computer science notions of abstractions and their implementations, which underlie the main thread of this article.

2. Quasi-interactions and abstract interactions

Besides the three questions just raised regarding the mechanisms of emergence, there's another important question. How should we understand interactions among emergent entities? That is, when an ant (which is emergent) climbs the stem of a plant (which is emergent) and clamps onto one of its leaves (which is emergent), how should we describe that interaction. Should we describe it at the level at which I just expressed it, i.e., as emergent entities interacting in terms of their emergent properties and capabilities? Or should we describe it in terms of quantum physics, which in some sense is all there is. (This is similar to the question raised by Putnam (1975) about a square peg and a round hole.)

The answer to this question provides a framework for much of the rest of the discussion. In broad terms I see emergent entities interacting in two distinct ways: quasi (physical) interactions and abstract (theoretical) interactions.

Quasi-interactions

To illustrate quasi interactions, I'd like to start with an extract from Bedau (1997). The passage is about downward causation, which I discuss later. For our purposes here, I want to concentrate on the interactions.

> Ordinary macro causation consists of causal relations among ordinary macro objects. Examples are when a rock thrown at a window cracks it, or an ocean wave hits a sand castle and demolishes it. But macro-level causes can also have micro-level effects. This is termed "downward causation." Downward causation is a straightforward consequence of ordinary macro causation. To see this, choose some micro piece of the macro effect and note that the macro cause is also responsible for the consequent changes in the micro piece of the macro effect. For example, consider the first molecular bond that broke when the window cracked. The [macro] rock caused that molecular bond to break.

Like all macro objects, both rocks and windows are emergent. Did the impact of the rock cause the window to break? If so does this illustrate what Bedau calls "ordinary macro causation," i.e., causality (or interaction) at the level of emergent elements? More generally, does this illustrate interaction between macro (emergent) objects? If so, how should we understand such interactions?

My position is that as an emergent object like a rock is an implementation of an abstraction. The same is true of a window. By definition, abstractions are abstract; they are not physical. Consequently,

abstractions cannot be causes of physical effects. The rock abstraction did not and could not cause the window abstraction to break in any physical sense of the term *cause*.[13]

As Bedau explains—although for his own purposes—the breaking of a window is a matter of bonds being broken. The bonds are part of the implementation of the window abstraction. Breaking those bonds interferes with that implementation, a consequence of which is that the original window abstraction ceases to exist.

What broke the window abstraction's implementation? The rock abstraction's implementation. Were one to ask about the physical causes for bond breaking one would find oneself talking about the interactions among the electromagnetic forces in the rock and window implementations. To the extent that we want to talk about physical causation, it occurs only at the lowest level of physics.

Regularities in quasi-interactions

So what are we saying when we talk about interactions among higher level entities? Above, I said that *Cordyceps* relies on the ant-level capabilities of ants—their ability to climb plant stems and to clamp onto plant leaves. Ants, plant stems, ants climbing plant stems and clamping onto plant leaves all refer to macro-level entities and their interactions. Is it inconsistent to talk about these sorts of interactions and at the same time to say that abstractions can't interact physically? How is it possible for macro objects—which are implemented abstractions—to interact even though abstractions cannot participate in physical causal relationships.

To answer that question I want first to look at how higher-level interactions occur in software. Are there interactions between abstractions in Microsoft Word? One that comes to mind is the interaction between the word abstraction and the document margin abstraction—which is typically set and indicated on the ruler (abstraction). One way to think about this interaction is that the margins keep the word abstractions confined to certain areas of the page (abstraction). Anyone who remembers manual typewriters will recall that margins were set physically by putting barriers in the way of the typewriter carriage to prevent it moving outside the margins. How are margins set in Microsoft word?

The most straightforward answer is that the code that implements Microsoft Word is written so that the interaction between the margin abstraction and the word abstraction works as indicated. In saying this, I haven't really told you anything. Of course the code does it. Since it's the code that implements the abstractions, how else would it be done? It's the code that implements an abstraction that determines how that abstraction behaves.

Let's look at the the-code-made-them-do-it argument more closely. Imagine that Microsoft Word were developed through an evolutionary process. One has a population of incomplete or inaccurate versions of the system. A version whose code more accurately implements the desired abstractions is more likely to survive than one whose code is less accurate. It's the job of the software developers to hurry evolution along by the "more fit" selecting elements of this population, combining them via crossover and mutation, keeping the offspring that come closer to the desired abstractions, and discarding the

[13] See the discussion of physical causality below.

others.[14] Of course software developers don't think of their jobs in these terms. They think that when they combine multiple versions of a system, i.e., crossover, they do it by picking the good parts of each. And they think of modifying existing code and writing new code, i.e., mutation, as driven by insight into how the new code will behave. But the result is pretty much the same. The code evolves to become a more and more accurate implementation of the desired abstractions.[15] A part of the increasing accuracy of the implementation the individual abstractions interact in ways that are more faithful to the requirements.

Does this carry over to the physical world? I would say that it does. Consider the heart as an implementation of a pump abstraction. The closer a heart implementation comes to providing pump functionality, the more likely it is that the organism with that heart implementation will survive. The parallel is that for an organism to persist in the world, its components must work together in certain functional ways. That implicit design[16] sets implicit requirements for the components, which can evolve to realize those requirements more effectively.[17]

To return to the question of how abstractions interact, our answer is that their implementation are such that their interactions are what they are. To the extent that those interactions are stable and reliable, the abstractions themselves are more likely to persist.

Does this relatively uninformative answer help explain how interactions among nature's abstractions work? I would say that it does. Nature creates abstractions through what is typically a long process of trial and error. As just discussed part of the job of creating abstractions is to create their interactions. As is the case for all naturally occurring evolutionary processes, those that survive do so because the way they interact with each other and with their environment permits them to survive. So whatever their interactions happen to be, the implemented abstractions that survive long enough for us to experience them are the ones whose interactions we see.

---

[14] This is not as farfetched a description of the software development process as it may sound.

[15] Another difference between software development as an evolutionary process and evolution in nature is that in nature the criteria for survival are not fixed in advance. In software, the criteria for survival are based on the software requirements, i.e., the conceptual model, which is fixed in advance. (Recall our previous analogy: write software that makes this user manual true.) Of course as any software developer can confirm, requirements themselves are rarely fixed. They often change during development.

[16] All this talk of functionality and design is, of course, intended naturalistically. Even though nature is not teleological, elements that can be described functionally evolve. A heart can be described in terms of its effectiveness as a pump. Combinations of those elements can be seen as an implicit design that produces additional functionalities. Hearts, lungs, blood vessels, blood, and hemoglobin provide a way to distribute oxygen throughout an organism and to collect and dispose of carbon dioxide. Again, the naturalistic way of understanding this is that none of this is intentional, but this is, in fact, what happens. And all else being equal, the more effectively it happens the more likely the organism is to survive.

[17] It's fair to ask where that design came from. The answer is that it didn't come from anywhere; it evolved. Implicit (random) designs that (happen to) persist establish implicit component requirements to which their component elements can evolve. Of course those implicit designs can themselves evolve as variants that persist more successfully come into being.

One might still ask what makes anyone think that nature's abstractions will interact at all? And if they do, why should those interactions show any degree of regularity? Here's one way to look at it. Since implemented abstractions are ultimately physical it's reasonable to suppose that they will interact in some ways. Physics tells us that forces are among the fundamental properties of the universe and that forces are "felt" by other elements of the universe. Not all forces are felt by all elements. The strong force is felt only by quarks. But nature is such that there are forces, and forces are felt, which is the basis of interactions.

Why does that mean there will be regularities in the interactions among the (generally complex) implementations of higher level abstractions? Depending on how two implementations come together, the aggregation of their low level interactions may differ from one situation to another. In fact, that is what we find. It will be observed as a reliable regularity that when a rock (of at least a certain size and weight) hits (with at least a certain speed) a window (of a certain thicknesses and made of certain materials), the window breaks. But as the qualifications suggest it's not the case that every time a rock makes contact with a window, the window breaks. Because these apparent interactions are in reality interactions between abstraction implementations, it makes sense to refer to such interactions as *quasi-interactions*. One of the challenges faced by engineers is to keep track of all the special cases. (See Appendix E for a discussion of how engineering and computer science differ in this regard.)

To the extent that interactions among rocks and windows is of general enough interest, one may see the development of a special science (or engineering specialty) to study them. The abstractions themselves don't interact; but their implementations interact in sufficiently regular ways that it is useful to speak as if the abstractions themselves are interacting. Some quasi-interactions will be regular enough that it makes sense to speak of them as laws of a special science. Others will not be. Nonetheless, no matter what the interactions are and no matter how regular they turn out to be, interactions among abstraction implementations, i.e., quasi-interactions, depend ultimately on properties of the implementations.[18] To the extent that exceptions are observed (e.g., tempered glass, Gorilla Glass,™ HammerGlass,™ foam rocks used as stage and movie props), they can be added to the increasingly complex characterization of the regularity.

An interventionist view of causality (discussed below) will justify referring to such quasi-interactions as causal relationships. And even though there is no physical causality between abstractions, there is nothing wrong with taking advantage of regularities in quasi-interactions to think causally at the rock/window macro level. These are the kinds of things we deal with in our daily lives. Regularities at these levels are reliable enough for most purposes.

Abstract interactions

There is a second and perhaps more significant way that interactions occur between abstractions. As an example, consider evolution again, this time as an abstract process. Evolution depends on (a) the combination and transmission of genetic (design/structural) information from parent(s) to child and

[18] The famous case of the two "implementations" of jade are distinguishable by how their implementations interact chemically with the rest of the world.

(b) the possibility that mutation may occur during that transmission. These functional properties are themselves abstractions. Darwin thought about them without knowing how they are implemented.

Once one starts to think about evolution in terms of these abstractions larger theories—such as the evolution of altruism, evolutionary population dynamics, genetic drift, etc.—can be built. Since the evolutionary process does not depend on how its underlying properties—(a) and (b) above—are implemented, evolutionary theory is autonomous and stands on its own as an independent domain of knowledge. Consequently, as a form of interaction evolution differs from the sort of quasi-interactions described above—which depend entirely on interactions between implementations. Evolution depends on the properties of the hypothesized abstractions and *not* on how those properties are implemented.

A much simpler example—and one that we will encounter later—is Euclidean geometry. The theorems of Euclidean geometry derive from the axioms of Euclidean geometry. They have nothing to do with how a line, say, may be implemented physically. Truths based on the axioms of Euclidean geometry are abstract truths, which characterize how the abstractions of Euclidean geometry interact—which is why I have chosen to refer to these sorts of relationships as abstract interactions.

But the truths of Euclidean geometry are not completely disassociated from the physical world. To the extent that elements of the physical world implement Euclidean abstractions one can apply the truths derived about those abstractions to these implantations. This is not to say that implementations will always be perfect and will necessarily persistent over long stretches of time. But as long as some implementations conform to some given abstractions conclusions derived from the properties of the abstractions can usefully be applied to the implementations. (I discuss this sort of top-down reasoning in the section on downward causation.)

As yet another example of an interaction that depends on abstractions, consider the question of the optimal set of denominations of coins and currency—as discussed, for example, in (van Hove, 2001). Should one have denominations of, say, $1, $2, $4, $8, …, or would some other set of denominations, such as our $1, $5, $10, …, be better? Whatever the answer, the theory upon which the answer is based will be derived from the properties of numbers as abstractions and will be independent of the materials and mechanisms used to mint coins and print currency. The latter provides a means for implementing particular numerical values. But given any set of numerical values, theories about those values are built on the values as numbers. The theories have nothing to do with how those values are implemented.[19, 20]

Putnam's example of square pegs and round holes offers another nice illustration. The explanation based on geometry is an abstract interaction. To the extent that physical objects implement the abstractions of square peg and round hole it makes sense to apply geometrical results to the physical objects.—with the proviso that those results apply only so long as the physical objects remain faithful implementations of the geometrical abstractions.

[19] Nor is it the case that the social or intrinsic value of the implementing materials matter. Even if one mints coins of "precious metals," the question still remains as to the denominations to mint.

[20] In the section on downward entailment I will use the fact that it is possible to implement a Turing machine in the Game of Life to draw a conclusion about the Game of Life itself.

To a great extent the special sciences study three sorts of questions. (1) What abstractions appear to be useful in describing the phenomena in the domain of the science? (2) What theoretical consequences can be derived from those abstractions? (3) How do the phenomena being studied implement those abstractions, and how faithful are those implementations to the abstractions?

Economics provides some nice additional examples. Economics is the study of how economic entities interact within a context in which interactions are defined by the allowable economic actions. Macro-economic models are expressed in such terms as the money supply, the rate of inflation, the velocity of money, the unemployment rate, the interest rate on Treasury bills, etc. Clearly these are all abstractions.

Given such abstractions, one can develop theories about how they interact. For example, the economic principle that markets will always clear expresses a theoretical relationship that can be derived from the definition of prices, supply, demand, and how abstract participants in economic markets behave. One can rely on such principles to the extent that the world faithfully implements the economic abstractions on which it is based. But we know that markets don't always clear. If they did, there would never be unemployment. Labor would lower its prices to the point where demand met supply. But for various reasons that doesn't happen. For example, wages tend to be sticky because employers don't want to reduce their employees' wages. This reluctance is not economic; it's psychological. Lowering someone's wage is likely to cause ill will, etc. In other words the abstraction of *homo-economicus* upon which the law of supply and demand is based is not implemented perfectly in the actual world.

The incomplete or partial implementation of abstractions is common enough that theories can be developed to study its consequences. Two come immediately to mind. Fodor's example of Gresham's law is useful here. It is a statement about what happens when the abstraction of money is not reliably implemented. If units of money cannot be relied on to represent consistent units of value but are valued more for their implementations, the money abstraction will fail and the intended money implementations will be treated more in terms of their implementation than as the abstraction they were supposed to represent.

The example of *Cordyceps* and the ants offers another good illustration. *Cordyceps* was able to interact with ants on two levels. By interfering with the ants' biochemistry it interacted with ants on the ant-implementation level. In doing so, it didn't destroy the ant at an abstraction level; it modified how the abstraction worked—in ways that redounded to its own benefit.

The more general case is the phenomenon of biological arms races in which one specie interacts with another specie's implementation in ways that benefit the former. The victimized specie is often able to evolve defenses so that its implementation is less vulnerable. The aggressor specie may then evolve ways around the new defense, etc.

The same sort of phenomenon—vulnerable abstractions—drive much of our legal and regulatory system. Much of our police and judicial systems are intended to ensure—to the extent possible—that society operates according to abstractions we find useful. There is no such thing as private property, ownership, or contractual agreements in nature. But these abstract concepts are important enough that we spend significant effort to enable society to operate on the basis of those abstractions.

Of course once one starts down such a path, difficult cases will present themselves. What is fraud? What is competence with respect to the ability to enter into a contract? What are the appropriate consequences for not carrying out a contractual commitment? What are the various ways in which ownership may be transferred? How does your patent limit my right to do what I wish with things that I own? What are the liabilities of ownership? Etc. Many of these questions lead to ways in which the implementation of an abstraction may be compromised or subverted, which leads to more rules, which leads to more subversions, etc.

In general whenever a system relies on abstract interactions, i.e., interactions between abstractions, the system is vulnerable to attacks on the implementations of those abstractions. In an evolutionary environment, if those abstractions contribute to the survival of entities that possess them, mechanisms to protect and shore up the implementations frequently appear.

The development of complexity

James Burke built a career[21] identifying "connections"—which are typically prerequisite relationships—among technological developments. Virtually every technological creation is both enabled by and dependent upon pre-existing technologies. See, for example, (Ziman, 2000) and (Arthur, 2009) for discussions of technological innovation as an evolutionary process.[22]

The same enabling/dependency structure applies to other areas as well. In particular biological evolution builds new capabilities on top of previously evolved capabilities. I am not aware of any study that traces the history of biological technologies—although it would be a fascinating study—but (Lane, 2009) examines what he considers (by his own admission somewhat arbitrarily) the ten greatest inventions of evolution. In his discussion of each invention Lane describes how that invention works biochemically and what biological structures had to be in place before its development could occur. Among the inventions he discusses are photosynthesis, multi-cell organisms, sex, movement, and sight.

Not only does Lane look backwards, he also describes some of the developments that depend on these inventions. For example, about motion Lane writes that

> [the rise of motility transformed the world from one dominated by organisms that were anchored in place]—lampshells, sea lilies, and so on, filtering food for a meager low-energy living—to a new, more active world, dominated by animals that move around. ... The new lifestyles that came with motility gave animals a particular reason to be in a particular place at a particular time, and indeed a different place at a different time. That is to say, it gave them purpose—deliberate, goal-directed behavior. …
>
> Motility [also] brings with it a need to deal with rapidly changing environments, more interactions between plants and other animals, new lifestyles like predation, and more complex ecosystems. All these factors encouraged the development of better senses (better ways of 'sampling' the surrounding world) and a faster pace of evolution. … At the heart of all this innovation is a single invention, which made it all possible: muscle. While not perhaps engendering the same sense of perfection as organs like the eye, when viewed down the

[21] See his (2007, reprint), which was based on three TV mini-series produced by BBC and Time-Life. (See https://en.wikipedia.org/wiki/Connections_(TV_series) for details.) Other books and TV series followed. The current incarnation of Burke's perspective is the Knowledge Web (http://k-web.org/).

[22] which is the title of (Ziman 2000).

> microscope muscles are an awesomely purposeful-looking array of fibers acting in concert … *They are machines that convert chemical energy into mechanical force.* [Emphasis added.]

My field of computer science might be one of the purest examples of a domain in which new developments build on the old. In a later section I discuss the multiple levels of abstraction underlying most software products. Here I'd like to focus on a property possessed only by software. When a software developer writes code that implements a new abstraction, she has confidence that the abstractions she uses will work as advertised. Engineering and biological building blocks depend on physical implementations to operate properly. In contrast software is purely symbolic. Our lowest level element is the bit—a pure abstraction—and we build from there. Software development is one of the best examples of a domain that builds new abstractions by arranging existing abstractions in ways that will produce the new abstraction. Our focus is almost exclusively on thinking up new abstractions, which I believe is one reason that the range of areas to which software is applied and the breadth of abstractions that have been created has grown so explosively.

3. Strong emergence

In this section I'd like to comment on what has been called strong emergence. In particular, I'd like to respond to the implicit disparagement of science that typically accompanies discussions of strong emergence.

I want to focus on strong emergence as characterized by Bedau (2010). He declares that phenomena are to be considered strongly emergent when they have "irreducible causal powers." By this Bedau requires that

> strongly emergent causal powers … be brute (i.e., unexplainable) natural phenomena that could at best play a primitive role in science. We should accept brute natural phenomena only if we are convinced that they cannot be explained.

Earlier Bedau (2002) put it this way.

> Strong emergence starts where scientific explanation ends.

In both articles, Bedau defines strong emergence in order to dismiss it as "scientifically irrelevant." That is not my intent. I think that's too superficial a view. Rather than give up on the idea, I'd like to examine it more closely and see what its implications are for science. I'll use this version of strong emergence because it imposes the most stringent requirement—the appearance of a new physical force, i.e., one for which no explanation is available.

The literature—e.g., (Bedau 2010), (Chalmers 2006), (Kim 2007), (O'Connor 2013)—is fairly unanimous that strong emergence, were anything to exhibit it, would imply not only that physics is not causally closed but that its position as the ultimate authority for how the natural world works is under challenge. In other words, strong emergence, were it shown to exist, would be a spooky phenomenon that may lead to the overthrow of science. Chalmers (2006) puts it like this.

> Strong emergence, if it exists, can be used to reject the physicalist picture of the world as fundamentally incomplete.

I disagree. I don't see a possible violation of causal closure as a threat to science. I'd like to consider three phenomena that are (or were) good candidates for being labeled as strongly emergent. The first two are examples of brute, irreducible forces—forces for which prior to Einstein no scientific explanation existed. The third is still a mystery. All have strengthened science rather than weakened it.

Dark energy

Dark energy is reducible neither to the forces of the standard model of physics nor to gravity. It is a new force, brute and irreducible. It is not redundant. It is causally productive in that it causes the universe to expand in a direct physical way. In addition, dark energy satisfies Chalmers' (2006) condition that "truths concerning [strongly emergent phenomena] are not deducible even in principle from truths in the low-level domain" applies to dark energy. As the NASA dark energy web page[23] puts it, no one understands why it should be there at all.

But is dark energy emergent? As a property of space itself,[24] wasn't dark energy there from the beginning? Perhaps not. According to Smolin (2012, p 172), space—along with its inherent dark energy—may itself be emergent.

> I think it is likely that space will turn out to be an illusion of the sort that temperature and pressure are—a useful way to organize our impressions of things on a large scale but only a rough and emergent way to see the world as whole. …
>
> [The separation of time from space] leads to the revolutionary insight that space, at the quantum-mechanical level, is not fundamental at all but emergent from a deeper order.

Sean Carroll agrees.[25]

> Our best theories of reality are based on quantum mechanics, not on relativity, and I think that eventually we will understand space-time as an emergent semi-classical property, not at all fundamental. Time is very crucial in quantum mechanics, but space is only an approximate notion.

Yet even though dark energy violates what the known laws of physics were before dark energy was "discovered," it has not brought physics to its knees. Why not? Because physics has expanded to include it. There is now a place in physics for dark energy.

The same is true for the advance of the perihelion of Mercury. Prior to general relativity, no known force was able to explain that phenomenon. In some reasonable sense it was an example of strong emergence. Yet its existence didn't bring down science. On the contrary an advance in science was able to bring this formerly inexplicable phenomenon into its realm.

---

[23] http://science.nasa.gov/astrophysics/focus-areas/what-is-dark-energy/

[24] According to (Carroll 2013) dark energy is a property of space. The expansion of the universe is driven by the density of dark energy. As space expands, additional dark energy comes into existence along with the additional space. The result is a constant dark energy density and a constant rate of expansion, which produces an acceleration in the expansion itself.

[25] From an online interview: http://www.3ammagazine.com/3am/the-philosopher-physicist/.

Nuclear fusion

Should energy from nuclear fusion be considered an example of strong emergence? Nuclear fusion seems to satisfy the requirements for strong emergence: smaller things—the atomic nuclei of deuterium (a proton and a neutron) and tritium (a proton and two neutrons)—are brought together to create a larger thing—the atomic nucleus of helium (two protons and two neutrons) plus an unattached neutron. (The same number of protons and neutrons go into the reaction as come out.) Yet the fusion process also produces energy. In emergence terms, helium nuclei emerge from a base of deuterium and tritium nuclei. Along with the helium nuclei one gets an extraordinary amount of energy. Before 20th century science, such a result would have been brute and irreducible. It would have seemed mysterious and strongly emergent. Now that we know how it works, must it forego that label and the associated mystique?

Consciousness: using thought to control matter

Chalmers (2006) and Bedau (2010) offer consciousness as the strongest remaining possibility for strong emergence.

> [A]re there strongly emergent phenomena? My own view is that the answer to this question is yes. I think there is exactly one clear case of a strongly emergent phenomenon, and that is the phenomenon of consciousness. (Chalmers, 2006)

> The leading contender for genuine strong emergence is conscious mental states. Our inability to have found any plausible micro-level explanation for conscious mental states might reflect just our ignorance, but another possibility is that these phenomena really are strongly emergent. (Bedau, 2010)

Chalmers and Bedau focused primarily on the phenomenon of consciousness—what it's like to be something—rather than on its physical consequences. But suppose one could show that it is possible for thought to control matter. Would that qualify as strong emergence?

The control of matter by thought is not as far-fetched as it sounds. Brain Computer Interfaces (BCIs) is an active area of research (Sellers, 2013), and neural engineering has established itself as a recognized engineering discipline. One of the most important application areas of the latter is motor rehabilitation—to allow people with motor deficits to manipulate objects by thinking about how they want the objects to move. Techniques exist for monitoring brain activity and translating it into control signals for engineered devices. Work in this area has been ongoing for more than a quarter of a century.[26]

Does this show that consciousness has causal powers? It certainly looks like it does. Yet this work has not produced a revolution in either physics or philosophy. Why not? Perhaps because subjective experience itself is not being used to control external physical devices. Only its neural correlates are used. This work makes no progress in understanding what subjective experience is and how it comes about. I agree with Chalmers (1995) that subjective experience is (still) the hard problem of consciousness.

[26] See (Farina, Jensen, and Akay, 2013) for an up-to-date survey of the state of the art.

Yet by definition, subjective experience is subjective. How could something remain subjective and still have an observable objective effect? It would seem that the only way for subjective experience to have a physical effect is through its physical manifestation. Most naturalist philosophers agree that consciousness supervenes on brain activity.

If brain activity that is correlated with thought can be used to control external physical devices, and if subjectively perceived images can be "read" by monitoring brain waves,[27] why is this not causal consciousness? Researchers in Artificial intelligence (AI) complain that feats that would once have been considered demonstrations of "true" AI lose that status once we figure out how to do them. Perhaps consciousness researchers will soon voice a similar complaint.

Violation of causal closure is not a problem for science

Whether something violates the causal closure of the known laws of physics is not a problem for the scientific view of nature. All newly discovered forces—or equivalently the curvature of space-time and its consequences as described by general relativity—necessarily violate the causal closure of the existing forces and the known laws of physics. Otherwise they wouldn't be considered new forces. But once discovered, any new force will be (co-opted and) integrated into physics. Science will not be invalidated as our best approach for understanding nature. Being outside the causal closure of the known forces and laws of physics is the signature of a new force, not a sign of the death of science. If a phenomenon that looks like strong emergence were to be observed, the press would come up with a catchy name for it, and scientists would write grant applications and get to work. That's how science advances.

4. Emergence summary

Emergence is best understood as the appearance of new abstractions and functionalities in what may be a new (implicit or explicit) conceptual domain and their implementation in terms of what already exists. Other widely cited examples of emergence include the hardness of diamonds, flocks of birds, traffic jams, a steel canoe (it floats even though its parts don't), and even life.

The two most important features of emergence are (a) a new abstractly characterizable functionality or property and (b) an implementation of that abstraction in terms of the functionalities and properties of existing elements. The implementation may be complex: Microsoft Word involves millions of lines of code; living organisms involve complex, sophisticated, and coordinated chemistry in entities that range in size from a single cell to organisms with trillions of cells. Or it may be simple but clever: once one has perfected the technology for a watertight seal, a steel canoe is easy enough to build. But there is always a replicable implementation. In other words, emergence is a matter of implementing abstractions.

Strong emergence refers to phenomena we can't (yet) imagine how either to explain or to produce through known science and engineering. But there is no reason to believe that any phenomena will enjoy a permanent status as strongly emergent. Even if a new force has to be introduced, it will be integrated into science. Consequently I recommend that we retire the category of strong emergence.

[27] (Schoenmaker, et. al., 2013) recently demonstrated the ability to reconstruct subjectively perceived images by monitoring brain activity.

Perhaps the notion of emergence itself should be retired and replaced by the notion of the implementation of abstractions.

C. The special sciences

This section discusses the special sciences and their relation to physics. The first part draws an analogy between the special sciences and the kinds of conceptual models developed for computer applications. The second examines the debate about whether the laws of physics entail the laws of the special sciences.

1. The special sciences are to physics as software applications are to computer hardware

Recall the opening extract from Fodor in which he expressed amazement that "unimaginably complicated to-ings and fro-ings of bits and pieces at the extreme *micro*-level manage somehow to converge on [the geologically] stable *macro*-level properties [of mountains]." One might make a similar observation about Microsoft Word. Unimaginably complicated to-ings and fro-ings of bits/voltages in a computer manage somehow to converge on Microsoft Word.

Is this a fair comparison? To decide whether it is, let's consider the conceptual distance between the two ends of these two comparisons. Table 1 shows the conceptual steps in the two cases. In Fodor's example one might count: quantum physics to atomic physics, to chemistry to the special science of geology—since his immediate example was mountains. In the case of Microsoft Word one might count: physical machine to a virtual machine to a programming language abstract machine to Microsoft Word. I gave both sequence four steps. Could the steps be divided differently? Probably. Are the steps equally complex? It's difficult to say. Whether they are or not, the point I wish to make will not depend on the specific complexity of each step. It will depend primarily on the claim that each step defines an autonomous level of abstraction.

Table 1. Levels of abstraction (from low to high) in the special sciences and computer applications

Special sciences (e.g., geology)	Applications (e.g., Microsoft Word)
Chemistry	Programming language abstract machine
Atomic physics	Virtual machine
Quantum physics	Physical computer

I'll start by explaining the steps on the software side. Microsoft Word (and all other software applications) exist as lines of code written in a programming language.[28] There are a number of ways to define the semantics of a programming language. Operational Semantics (Plotkin, 2004), one of the most intuitive, defines the semantics of a programming language in terms of how execution of constructs in the language affect what is known as the language's abstract machine.

[28] The two most widely used language families are the C/C++ family and the Java family. Other widely used languages are C# for Microsoft applications and Objective C for Macintosh applications.

A language's abstract machine is a formally defined device that is capable of changing state, where a state is defined as the values of some of the parameters in terms of which the device is defined. A Turing machine is a simple example. A Turing machine consists of a finite automaton, a tape, and a read-write head that is positioned somewhere on the tape. The state of a Turing machine is the combination of the state of its finite automaton, the contents of its tape, and the position of its read-write head on the tape.

A program in a programming languages describes manipulations of the language's abstract machine. The program is written in such a way that the manipulations result in the implementation of some collection of abstractions—in this case of Microsoft Word. In other words, the Microsoft Word software implements the Microsoft Word abstractions by arranging manipulations of the abstract machine of the programming language in which the Microsoft Word program is written.

Programs in programming languages are processed by compilers. A compiler is a program that translates code in a programming language into code in a lower-level language. In practice most compilers translate what a programmer writes into instructions for what is known as a virtual machine. A virtual machine is a software implementation of an abstract machine. Perhaps confusingly, the target virtual machine for most compilers is generally not a software implementation of the programming language's abstract machine. It is a software implementation of an idealized physical computer for that abstract machine. The virtual machine for Microsoft programming languages is known as the Common Language Runtime.[29] The virtual machine for the Java programming language is known as the Java Virtual Machine.

As the table shows abstract machines are implemented by virtual machines. In reality, abstract machines are generally not realized at all. Programmers tend to think in terms of the abstract machines for the languages in which they write. But their code does not manipulate abstract machines. Their code is translated into instructions for the virtual machines that implement the abstract machine. Thus abstract machines, while present in the minds both of programmers and of virtual machine implementers is generally not realized directly.

The final step is the implementation of the virtual machine in terms of instructions of some physical computer. For each family of physical computers—e.g., the one or more families of computers built by a computer manufacturer—there is a separate implementation for each virtual machine, i.e., one for the Java virtual machine, one for Microsoft's Common Language Runtime, etc. Each of those implementations manipulates the abstractions of a particular physical computer to implement the capabilities of a virtual machine.

[29] The advantage of generating programs for a virtual machine is that when a program is run on a virtual machine the physical computer on which it is running is irrelevant. Java's motto is "Write once; run anywhere," which promises that a Java program will run the same way whether it is being run on a laptop manufactured by Asis or a desktop manufactured by Dell. So a compiler generates instructions to manipulate the abstractions of a virtual machine in a way that will produce the abstractions of the programming language's abstract machine.

Although software developers do not have to be concerned about it, there is yet an additional step. Each physical machine is implemented by logic gates and circuits. (See Appendix E for a discussion of the bit as the software primitive.)

Is all that comparable in complexity to the four steps from quantum physics to geology? I think a case could be made that it is.

Given the number of translations and implementations between Microsoft word and the physical computer on which it runs, it would be virtually impossible for anyone to look at the to-ings and fro-ings of bits in that physical computer—or voltages in the circuits that implements that computer—and understand how they correspond to the manipulation of Microsoft Word abstractions. Yet the convergence of those bit to-ings and fro-ings to Microsoft Word is successful.

Complex as it may seem—or as I have made it seem—applications like Microsoft Word are a relatively simple case. Consider web applications that run in a browser. Examples include Gmail, Facebook, the New York Times website, Twitter, Google maps, The Stanford Encyclopedia of Philosophy, and others. Running in a browser adds a few additional layers of abstraction.

The browser itself is an application (an "app"). Its connection to a physical computer is similar to the connection just described for Microsoft Word. One interacts with a browser *as an app* when one clicks the back button or enters a web address into the browser's address bar. A browser's relation to a physical computer is the same as that of Microsoft Word. But browsers are not web pages. Browsers provide a set of abstractions in terms of which web pages are implemented. These browser-level abstractions are defined in terms of such languages as HTML, CSS, and JavaScript. Implementation of a web page in a browser is similar in concept to implementation of an app in a programming language. By using a browser one is inserting additional levels of abstraction between a physical computer and what the user sees.

Still, we're not done. There is often a significant conceptual distance between a browser's abstractions and those of a web page. The abstractions that a browser offers are perhaps as close to those of a web page as the abstractions of a virtual machine are to an app. Bridging the gap between a browser and a web page requires something like a programming language. Because such a gap exists, software developers have created frameworks and libraries that provide abstractions to help span that gap. These are to web pages something like the way programming languages are to apps. So we've now inserted these frameworks and libraries as yet an additional level of abstraction.

It's also worth noting that there are a number of different browsers. The best known are Apple's Safari, Google's Chrome, Microsoft's Internet Explorer, and Mozilla's Firefox. These play roles similar to those played by the multiple hardware manufacturers. As you can see all this involves a great many implementations of many abstractions—which in the end are intended to be invisible to users.

Beyond all that, many organizations that offer browser-based web services also provide apps that run on mobile phones. Of the web services in the list above, all except the Stanford Encyclopedia of

Philosophy[30] have separate mobile phone apps. These apps run on the computer in mobile devices the same way that Microsoft Word runs on a laptop or desktop computer. Consider what that means. The abstractions that users associate with Gmail, Facebook, etc. are fairly consistent whether these services are accessed via a browser or via a mobile device application. Yet in almost all cases the apps that run on mobile devices offer significantly different access to those abstractions. If you have used any of these services on both a laptop (or desktop) and a mobile device you will have noticed the differences. So the abstractions are the same: Gmail deals with messages; the New York Times website deals with articles; etc. But users have somewhat different ways to interact with those abstractions depending on the platform.

Furthermore, the apps that run on mobile devices differ from each other depending on the mobile device family. Apps for Android-based mobile devices are written in Java; apps for Apple mobile devices are written in Objective C.

Why have I spent so much time enumerating this litany of software abstractions and their implementations? My point is that it's not at all unusual to find low level to-ings and fro-ings that converge—after many levels of abstraction—to high levels of abstraction. That to-ings and fro-ings in nature exhibit regularities that make it worthwhile for there to be special sciences to study them is no more remarkable than that to-ings and fro-ings of bits and voltages produce regularities that we as computer users understand and rely on.

One might object that in the case of software-implemented abstractions, the bit/voltage to-ings and fro-ings are all planned and coordinated in advance; so of course it's successful. Since nature doesn't plan, how can a similar convergence take place? The answer is that it doesn't always happen. When to-ings and fro-ings in nature don't converge to stable macro-level properties, macro-level properties don't appear. As Fodor says, mountains are made of all sorts of stuff. But mountains are also not made of all sorts of stuff. We don't have mountains of sand because sand isn't the sort of stuff whose underlying to-ings and fro-ings can converge to a stable mountain. If it were, perhaps we would.

A point made in a recent article by Dennett (2012) applies. Dennett cited evolution as an example of competence without comprehension. One can generalize this to nature in general. Nature implements many abstractions without comprehending how the implementations work, i.e., without planning and without coordination. Nature often achieves competence in implementing abstractions by replacing insight and planning with trial-and-error over extremely long periods of time. Is it conceivable that nature can do in hundreds of millions (perhaps billions) of years something similar in the complexity of its abstraction layers to what teams of programmers have done in (mere) decades?[31] I wouldn't bet against it.

[30] To get the Stanford Encyclopedia of Philosophy on a mobile device requires that one use the mobile device's browser, not an app.

[31] I'm granting the programmers decades to allow for both the implementation of the various intermediate levels of abstraction and the implementation of increasingly sophisticated versions of Microsoft Word.

How complex systems develop

It's important to note that abstractions are not built only as layers. As the example of *Cordyceps* illustrates, collections of abstractions can be built at all levels. Perhaps this is most easily seen in terms of technological developments. Technology is not a matter of layer, one upon another. It is more a matter of collections of abstractions that are then available for building other abstractions. Television is one collection of abstractions; tablet computers are another. Neither rests upon the other. The existence of both allows the development of what is now called second screen television[32] in which television viewers use tablets to interact in real time with web pages about the TV shows that they are watching.

It has become standard practice in software to develop and distribute libraries that implement various collections of abstractions.[33] Domains include artificial intelligence, various sciences, image processing, finance, data visualization, statistics, and many others, including a number of libraries for Bible studies. Most modern software systems depend on such libraries to provide initial sets of abstractions as a base upon which to build.

Considering how complex many modern software systems are, why don't they collapse under their own weight? One reason is that the complexity is divided into multiple encapsulated implementations of relatively autonomous collections of abstractions. Each collection of abstractions can be implemented and tested separately. Once a collection exists, it can be used in the implementation of other abstractions. The implementations that use it need not be concerned about how it works. All that matters is that it works as advertised.

Although this strategy of encapsulated implementations has not completely solved the problem, to a significant extent it has tamed the complexity demon. The effect is of a continuing growing catalog of collections of abstraction. Building a new capability does not require that one start from scratch and build the entire thing as a monolith. Building a new capability becomes more a matter of looking in the catalog, finding useful components, and figuring out how to put them together to achieve the desired result.

Nature works in much the same way. Consider *Cordyceps* again. As a species it needs a way to transport itself to a spot where it could sprout a stalk which, when ripe, would rain down spores on potential future hosts. Let's anthropomorphize the design process. One approach would be to design a system to achieve that result by starting with basic physics. Since *Cordyceps* has no means for self-propulsion, doing so would seem virtually impossible. The alternative would be to look in nature's catalog of existing components and see whether anything there could be put to use. *Cordyceps'* solution—to use ants that can climb plants and latch onto leaf veins—was possible because plants and ants were already in the catalog. *Cordyceps* didn't have to design the ants or the plants. All it had to do was find a way to use their capabilities to satisfy its own needs. This is a much smaller and more feasible step than starting from scratch.

---

[32] For example see (Odijk, Meij, and de Rijke, 2013) or (Simon, Comunello, and von Wangenheim 2013).

[33] A Wikipedia page (http://en.wikipedia.org/wiki/List_of_free_and_open-source_software_packages) lists many open source libraries.

The lesson is that it makes perfectly good sense to understand nature as consisting of multiple collections of abstractions. It also makes good sense to establish a special science to study some of these collections as autonomous domains.

For both nature and software, though, it's important to remember that nothing would actually happen were the lowest level not to-ing and fro-ing. *Cordyceps* didn't have to design the lowest levels of ants and plants. But were those lowest levels not already to-ing and fro-ing, *Cordyceps* could not have been able to use ants and plants in its larger construction. This is essentially the same remark I made when discussing interaction among emergent entities. There are no higher-level forces or higher-level causal laws. It is only the lowest level of to-ings and fro-ings that make the world—both physical and software—go 'round.

Appendix B discusses the importance of the special sciences from the perspective of a modern-day Laplacian demon. Could such a demon foresee how the future will unfold if it knows only physics and not the special sciences? My answer will be that it can't.

Appendix C discusses platforms, a level of abstraction with additional richness and complexity.

2. Do the laws of physics entail all other laws?

This section examines whether it is possible to deduce the laws of the special sciences from the laws of physics. This is not a new question. Very highly-regarded scientists have taken both sides. I believe, however, that the two sides have been talking past each other and that they do not disagree as much as they seem to.

Two views of the special sciences: Einstein and Weinberg vs. Schrödinger and Anderson

Einstein (1918) thought that all the laws of the special sciences could be derived from the laws of physics. In an address to the Berlin Physical Society he put it this way.

> The painter, the poet, the speculative philosopher, and the natural scientist … each in his own fashion, tries to make for himself .. a simplified and intelligible picture of the world. What place does the theoretical physicist's picture of the world occupy among these? … In regard to his subject matter … the physicist … must content himself with describing the most simple events which can be brought within the domain of our experience … But what can be the attraction of getting to know such a tiny section of nature thoroughly, while one leaves everything subtler and more complex shyly and timidly alone? Does the product of such a modest effort deserve to be called by the proud name of a theory of the universe?
>
> In my belief the name is justified; for the general laws on which the structure of theoretical physics is based claim to be valid for any natural phenomenon whatsoever. With them, *it ought to be possible to arrive at … the theory of every natural process, including life, by means of pure deduction. … The supreme task of the physicist is to arrive at those elementary universal laws from which the cosmos can be built up by pure deduction.* [emphasis added]

Stephen Weinberg (1995) agrees. He calls this position grand reductionism.

> Grand reductionism is … the view that all of nature is the way it is (with certain qualifications about initial conditions and historical accidents) because of simple universal laws, to which all other scientific laws may in some sense be reduced.

In elaborating this point Weinberg takes the weather as an example.

> [T]he reductionist regards the general theories governing air and water and radiation as being at a deeper level than theories about cold fronts or thunderstorms, not in the sense that they are more useful, but only in the sense that the latter can in principle be understood as mathematical consequences of the former. The reductionist program of physics is the search for the common source of all explanations. …
>
> Reductionism … provides the necessary insight that there are no autonomous laws of weather that are logically independent of the principles of physics. … We don't know the final laws of nature, but we know that they are not expressed in terms of cold fronts or thunderstorms. … Every field of science operates by formulating and testing generalizations that are sometimes dignified by being called principles or laws. … But there are no principles of chemistry that simply stand on their own, without needing to be explained reductively from the properties of electrons and atomic nuclei, and in the same way there are no principles of psychology that are freestanding, in the sense that they do not need ultimately to be understood through the study of the human brain, which in turn must ultimately be understood on the basis of physics and chemistry.

It seems to me that Weinberg is conflating two things: reductionism and entailment. The reductionist claim is that for any given higher level phenomenon or regularity one will be able to explain it in terms of lower level phenomena and laws. To use the terms introduced earlier, reduction is about both quasi-interactions and how higher level abstractions are implemented.

The entailment claim is that one can start with the fundamental phenomena and their laws and from them prove all higher level laws as theorems. The entailment claim is that one can derive all abstract interactions by taking the fundamental laws of physics as a starting point.

These are significantly different claims. The entailment claim is much more difficult since it requires that one both invent and prove all the higher level laws. It's hard to see how that could be accomplished. How could one derive the law of supply and demand or the theory of the evolution of altruism from the fundamental laws of physics? In contrast all that is required of reductionism is that for an given higher level phenomenon or regularities one can find an explanation for it in lower-level terms. I explore this distinction further below.

In contrast to Einstein and Weinberg, in "What is Life" (1944), Erwin Schrödinger writes,

> Living matter, while not eluding the 'laws of physics' … is likely to involve 'other laws,' [which] will form just as integral a part of [its] science. …
>
> [F]rom all we have learnt about the structure of living matter, we must be prepared to find it working in a manner that cannot be reduced to the ordinary laws of physics. [N]ot on the ground that there is any 'new force' … directing the behaviour of the atoms within a living organism, but because the construction is different from anything we have yet tested in the physical laboratory.

Philip Anderson makes the same point in "More is Different" (1972). Like Schrödinger he first swears allegiance to scientific reductionism.

> Among the great majority of active scientists I think [the reductionist hypothesis] is accepted without question. The working of our minds and bodies, and of all the animate or inanimate matter of which we have any detailed knowledge, are assumed to be controlled by the same set of fundamental laws, which except under certain extreme conditions we feel we know pretty well.

But Anderson goes on as follows.

> [T]he reductionist hypothesis does not by any means imply a "constructionist"[34] one: The ability to reduce everything to simple fundamental laws does not imply the ability to start from those laws and reconstruct the universe. …
>
> The constructionist hypothesis breaks down when confronted with the twin difficulties of scale and complexity. The behavior of large and complex aggregation of elementary particles, it turns out, is not to be understood in terms of a simple extrapolation of the properties of a few particles. Instead, at each level of complexity entirely new properties appear, and the understanding of the new behaviors requires research which I think is as fundamental in its nature as any other. … At each stage entirely new laws, concepts, and generalizations are necessary.

Anderson, Einstein, Schrödinger, and Weinberg are among the best physicists of all times. Each has been awarded the Nobel prize. Yet they seem to disagree (two on each side) about whether higher level regularities, i.e., the special sciences, can be derived from the fundamental laws of physics.

Can these apparently contradictory positions be reconciled?

In a recent workshop (Carroll, 2012) Weinberg re-iterated his position. In doing so, he may have revealed why his position and Anderson's are not as far apart as they seem. Weinberg explained[35] what he means by entailment: that the fundamental laws of physics *entail* all other regularities. First he explained what he thinks is important: (a) which laws of nature are most fundamental and (b) whether they entail the other regularities.

> I am talking about what actually exists out there in the sense of entailment. Are the laws of thermodynamics, i.e., the gas laws, entailed by other laws like elementary particle physics? If so, one can talk about which laws are more fundamental. The fundamental levels are the levels from which the arrows of entailment go up. This gives a sense of order in nature.

Then Weinberg acknowledges that what we as humans tend to encounter in the world are generally far removed from the elements to which the fundamental laws apply.

> There's no question that thermodynamics is worth studying on its own terms. And in fact as has often been pointed out, if you could follow the motion of every particle in a glass of water you would still miss the idea of what the temperature of the water is. The things we as humans are interested in are often things at a higher level.

But what's important to him as a physicist is whether the things we encounter in our day-to-day lives are the way we find them because of more fundamental laws of nature.

> The question is whether or not principles, like the principles of thermodynamics, are what they are because of deeper principles. And if that kind of statement makes sense, as I think it does, then we have a sense of what is fundamental and what is not fundamental. It's a question of following the arrows of what explains what, of how nature works.

One of the other participants asked whether Weinberg really meant entailment or simply that higher level laws must be compatible with lower level laws. His response, "No, I mean entailment."

---

[34] Anderson uses *constructionist* to refer to the same process that Weinberg called *entailment*.

[35] Lightly edited from a video of the workshop.

This issue came up repeatedly. Most of the participants had a hard time believing that Weinberg really meant logical entailment, that one can formally derive all other regularities in nature from the fundamental laws of physics.

Another participant asked, "What justifies the belief that the lower levels entail everything above them?"

Weinberg's answer:

> Well, of course, we don't know; that's what we're trying to figure out. But that's the way history has worked. We find that there really are entailments in nature. The paradigm is that Newton shows that the inverse square law entails Kepler's laws of planetary motion; Boltzmann shows that the statistical mechanics of large numbers of molecules entails the laws of thermodynamics. This is our bread and butter.

But Weinberg understands why people have doubts about entailment.

> I think there is a reason why people are skeptical about this. It's the sort of thing Anderson wrote about in "More is Different." A lot of the features of higher level theories don't seem to depend very much on what is going on at the deeper levels. For example, in phase transitions there are certain numbers that describe how systems approach phase transitions. These are critical exponents, and you get the same critical exponents in a tremendous variety of entirely different contexts. It makes you think there are laws of phase transitions, which are somehow free-standing and do not require any deeper level to be entailed. They just are the way they are.

But even though Weinberg understands the case against entailment, he insists on it anyway.

> I don't think that's right. I think you have bodies of formalism, like the renormalization group theory of phase transitions, or thermodynamics, which apply in a great variety of circumstances. But *when they apply they apply because of the deeper level that says they apply* here and they don't always apply.
>
> For example, Boltzmann made a tremendous step forward when he showed that the atomic theory—that molecules are bopping around—explains, or entails the laws of thermodynamics. And then it was realized that those laws apply in entirely different contexts. As Hawkins showed, the surface of a black hole has a certain temperature. But in all cases where thermodynamics applies we can show why it applies in terms of a deeper level—and it doesn't always apply. A black hole only obeys the laws of thermodynamics when it's big, when it contains many Plank masses. And in the last instants before a black hole evaporates the laws of thermodynamics no longer apply to it.

It seems to me—and this is my interpretation of what Weinberg said—that the preceding begins to explain what Weinberg means by entailment. He is not saying that one can derive all of science—including such things as the renormalization group theory of phase transitions—from the fundamental laws of physics. He is saying that the fundamental laws of physics determine when those theories actually apply in nature. His next example seems to pin this down.

> To take an even more trivial example, the sum of the angles of a [physical] triangle [on earth] is 180 degrees. It doesn't matter if you make the triangle with pencil marks on a piece of paper or with sticks. In all those physical instances you get 180 degrees. You have to ask why that works in each case? The answer generally speaking is because the gravitational field on the surface of the earth is quite weak so it doesn't curve space very much. That's the reductionist answer to why the sum of the angles is 180 degrees.

This originally struck me as a very strange remark. Surely Weinberg wasn't claiming that the laws of physics entail the theorems of plane geometry. And, in fact, he didn't say they did. He was saying

something about triangles on the surface of the earth, not about triangles in the abstract. I think everyone at the workshop was confused by this example. The video shows that no one responded to it.

So there you have it. Weinberg is certainly not claiming that the theorem of plane geometry that says that the sum of the angles in a triangle comes to 180^{o} is entailed by the fundamental laws of physics. He is saying that the laws of physics determine when that theorem applies to the material world.

Weinberg goes on to explain his position further. It's important to keep in mind that Weinberg is a physicist; what's important to physicists is how the world actually works. All questions are fundamentally empirical: can we come up with usefully abstract and coherent descriptions of observed phenomena?

> There is a certain quality of "fundamentalness" that has some value in its own right. It's a question of following the arrows of determining what explains what when we describe how nature works. It is the goal that we have in physics. It's the goal that's been working out over the centuries. To deny the existence of more fundamental levels and less fundamental levels makes a lot of the history of physics pointless.
>
> If you discover a principle, like the principles of thermodynamics or the principle that sex ratios tend to be equal, male/female, you then ask why that principle is true. You might say that it's a fundamental law of life that organisms that have sexual reproduction tend to have equal numbers of males and females. But in fact we're not happy with that. We try to explain it in terms of a deeper level, which is evolutionary biology. But then we ask how evolution works? Well we explain that in terms of mutation and natural selection.
>
> Very often we can't do the whole job because historical accidents intervene, which by definition we can't explain. I'm not saying simply that quantum field theory "constrains" everything. I'm saying that *wherever you find at any level a principle that certain things are always true, you have to explain it. The explanation will always be in terms of something more fundamental.*
>
> If we agree on this I'm happy: until we get down to the fundamental laws of physics there are no free-standing principles at any level that do not have an ultimate explanation in terms of deeper principles.

I think the italicized sentences are the key. Two ideas are involved here. The first is the notion of recognizing what Weinberg calls a principle, a regularity in nature. Weinberg is not saying that every regularity will be characterized in a formalism that can be derived from the fundamental laws of physics. After all, there are no obvious *a priori* limits to the sorts regularities one might find or the formal languages in which they are best described. There is no reason to expect that they can all be derived from physics via logical deduction. The fundamental laws of physics entail neither the principles of evolution nor the theorems of plane geometry.[36]

The second idea is to answer for each regularity the question why that regularity is found in nature, i.e., in the material world. This is as I understand him where entailment comes in. Weinberg says that to answer a question about why a regularity appears in nature one must look to see how nature implements/realizes the regularity, what physical mechanisms create and support it. When one does that one might find other regularities. When one then looks to see how those regularities persist in the

[36] Of course, since Euclid's axioms entail the theorems of plane geometry, Euclid's axioms along with the fundamental laws of physics do also. But that's not the way one wants to look at this question.

material world, one may find still more regularities. Eventually this series of explanations leads to the fundamental laws of physics, which underlies them all.

It now seems clear to me that the primary difference between Anderson and Schrödinger on the one hand and Weinberg (and presumably Einstein) on the other is that Anderson and Schrödinger were talking about the regularities themselves, i.e., which regularities one might find in nature, whereas Weinberg was talking about how those regularities are realized in the physical world.

Schrödinger's claim that "Living matter … is likely to involve 'other laws'" and Anderson's statement that "at each level of complexity entirely new properties appear" are about the regularities one might find at levels of complexity above fundamental physics. These are what I referred to earlier as abstract interactions. Weinberg seems quite comfortable with the idea that new regularities—such as the laws of thermodynamics or phase transitions or plane geometry—will be found and that new principles will be required to characterize them. His claim is that whenever such new regularities are found the answer to how those regularities are realized in the physical world will depend eventually on the fundamental laws of physics. In my terms Weinberg is insisting that whenever one finds a new abstraction that holds in nature one must ask how that abstraction is implemented.

Anderson does not disagree. Anderson (1972) began by acknowledging that his position was fundamentally reductionist. "The working of our minds and bodies, and of all the animate or inanimate matter of which we have any detailed knowledge, are assumed to be controlled by the same set of fundamental laws." Schrödinger agrees: living matter does not elude the laws of physics.

The difference between the two camps is one of perspective: whether one thinks about the world analytically or synthetically. Anderson touched on this issue toward the end of "More is different" (1972).

> [In general], the relationship between [a] system and its parts is intellectually a one-way street. Synthesis is expected to be all but impossible; analysis, on the other hand, may be not only possible but fruitful in all kinds of ways.

Weinberg and Einstein take an analytical perspective. They insist that when one figures out how nature works, the answer will ultimately be based on the fundamental laws of physics. Anderson and Schrödinger take a synthetic perspective. They argue that as new levels of complexity come into existence, new abstractions will be found—and new laws and principles will be needed to describe them. When Weinberg says "there are no free-standing principles at any level that do not have an ultimate explanation in terms of deeper principles" he is not saying that there are no free standing principles. Indeed that the angles of any (abstract) plane triangle sum to 180 degrees has nothing to do with physics. Weinberg is saying that whenever one finds *in nature* a regularity that is characterized by a free standing principle, one is intellectually obligated to ask how nature realizes that regularity—in my terms how nature implements the abstractions that are used in the characterization of that principle—and in asking that one will be led inevitably to the fundamental laws of physics.

3. Summary position on the special sciences

The preceding sections addressed the issue of the autonomy of the special sciences. Many of the laws of the special sciences are autonomous. They exist at the level of abstract interactions. But if these laws

apply to nature they must stand on physically implemented phenomena—which must obey the laws of physics.

D. Review of causation

The goal of science is to understand nature and to describe how it operates. Science, or at least physics, describes nature by using concepts from general relativity, the standard model of elementary particles, and quantum field theory. None of these use the language of causation. Physics is happy to say: here's how one can describe[37] the world as it is, and here are some equations that can (in principle) be used to project such descriptions forward (and backward) in time. In saying that, however, there is no sense that anything is *causing* anything else. It is all descriptive, as if it were a 4-dimensional landscape. With this in mind, I'd like first to examine what I take to be the two most widely adopted philosophical approaches to thinking about causation.

1. Common sense causality formalized: interventionist causality

Although the laws of physics don't make use of the notion of causality, science does and scientists do. Science derives its results from experiments, deliberate or natural. Typically the goal of an experiment is to identify the effect of varying just one thing while keeping everything else fixed. Does exposure to cowpox produce an immunity to smallpox? What pattern appears on the other side of a barrier toward which a stream of photons is directed if the barrier has one vs. two slits? Would it matter if a photon sensor were placed at one of the two slits in the two-slit case? Does smoking cause cancer?

Computer scientist Judea Pearl (2009) and philosopher James Woodward (2003) both developed approaches to causality. Even though they worked separately their work comes to essentially the same conclusions: causality is involved when a change in one thing produces a change in another. Woodward (2011) characterizes what he calls the *Difference-Making* approach to causality.

> Suppose X and Y are variables. Then, in the simplest case, X is causally relevant (at the type level) to Y if and only if there is a possible intervention on X such that if such an intervention were to occur, the value of Y or the probability distribution of Y would change—in other words, some interventions on X make a difference for Y.

Pearl (2011) puts what is essentially the same idea somewhat more formally. First he points out that causality cannot be inferred from joint probability distributions. The old saw is true: correlation does not imply causality. What we really care about is how changes lead to other changes. He summarizes this as follows.

> [The task of causal analysis] is to infer not only the likelihood of events under static conditions [which is the domain of statistical analysis], but also the dynamics of events under *changing conditions*, for example, changes induced by treatments or external interventions.

[37] The description may be given in terms of complex numbers and probabilities, but it is still a well-defined description.

As part of his formalization Pearl invented what he refers to as a *do* operator, which sets the value of a probabilistic variable to a particular value. Performing a *do* operation is quite similar to Woodward's intervention.[38]

The two approaches to causality are enough alike that from here on I'll refer to them jointly as the Interventionist approach to causal explanation or Interventionist-style causality.

For Pearl and Woodward causality is captured in causal models—often represented by directed graphs or by equations that relate variable values or conditional probabilities. Pearl provides an algebra and a calculus for causal models that allows one (a) to solve for some causal relationships in terms of others and (b) to compute numeric answers to causality questions. He shows that once one has a causal model it is possible to answer the standard causality questions by simple calculations. Woodward make it clear that his approach handles what he calls *what-if-things-had-been-different questions* (or *w-questions*).

The Interventionist approach has been quite successful. Pearl has won numerous awards and prizes for his work, and Woodward dominates the philosophical literature on causality. Baedke (2012) argues that Interventionist-style causality "qualifies as a useful unifying explanatory approach when it comes to cross-methodological research efforts. It can act as a guiding rationale (i) to link causal models in molecular biology with statistical models derived from observational data analysis and (ii) to identify test-criteria for reciprocal transparent studies in different fields of research, which is a shared issue across the sciences."

Pearl and Woodward agree on another point. A causal model does not explain how a causal relationship works, i.e., how a cause is physically connected to an effect. A causal model simply lets one record what one believes—on other grounds—the causal relationships to be and then provides tools for manipulating those relationships.

2. Physical causality: causal primitives and the exchange of conserved quantities

As the preceding discussion points out, knowing that *B* is causally dependent (from an interventionist perspective) on *A* doesn't explain how a change in *A* comes to affect *B*. Interventionist causal relationships are phenomenal. One can see them, and one can confirm the causal relationship, but that's about it.

It seems to me that in science, once the phenomenology of an area has been pinned down, the next step typically is to ask: what is the underlying mechanism? What is happening so that the phenomena and observed regularities come about? I like to think of this as the reverse engineering of nature: take it apart and figure out how it works. Broadbent (2011) suggests that an Interventionist causal relationship can be seen as claiming the existence of some underlying mechanism. As Baedke (2012) put it, "explanation [for an Interventionist causal relationship] is achieved when one shows how a phenomenon is produced by a mechanism."

[38] Since Woodward does not formalize his notion of an intervention in the same way as Pearl he adds conditions to prevent an intervention on a presumed cause from having any effect on the intended effect other than directly through the cause itself.

What is a mechanism in the sense just described? Typically it is a string of more detailed causal relationships that lead from the original cause to the original effect. But what if one wanted to know how each of the internal steps worked? One could investigate those as well. But these answers will involve additional causal relationship steps. A regress of this sort raises at least two questions. (a) Are there causal primitives for which no internal mechanism exists? (b) How should one account for the difference in time and space—assuming there is a difference—between cause and effect? That is, how can there be causality at a distance in time and space? Dowe's (2000) theory of physical causation helps to fill in what's missing.

Whereas Interventionist causality doesn't ask how a cause physically produces its effect, Dowe wants to know what's actually going on when one thing causes another. In seeking to understand what's really happening, Dowe finds himself looking for causation primitives—the most basic elements of causation. This is what he finds.

- A *world line* is a curve in space-time joining the positions of a particle throughout its existence.
- A *causal interaction* is an intersection of world lines that involves an exchange of a conserved quantity.

Attractive as this approach is—and I do find it attractive—it seems to me that it doesn't quite work at the quantum level. (We'll see in a few paragraphs, though, why that's a good thing.)

One problem is that there really aren't any particles. (See (Wilczek 2008), (Brooks, 2012), or (Hobson, 2013).) The best way to understand the goings on at the quantum level is to forget about particles and to think entirely in terms of fields. An electron, for example, has some value at all points in its field, i.e., for every point in the universe. But for all practical purposes an electron is localized to a small volume where its probability amplitudes are non-negligible. It is possible to trace something like the world line of an electron. But it is not a trajectory of points through space. The Schrödinger wave function characterizes the time-evolution of an electron's probability amplitude distribution throughout space.

This leads to problems with Dowe's definition. If particles are really all-pervasive fields, it doesn't make sense to talk about the intersection of the world lines of two particles. The fields of all particles are always intersecting since each one fills space. What matters is when and where they interact, and that depends on the values of their probability amplitudes.

Furthermore, the exchange of conserved quantities doesn't happen at a point in space or an instant in time. A photon is emitted by one field at one point in space and one instant in time and absorbed by another at a different point in space and a different instant in time. It is not even necessarily the case that the emitting event occurs before the absorption event. Feynman's classic (QED 1985) has an accessible discussion.

Although this sounds like it causes trouble for Dowe's view of causality, it seems to me that these apparent problems make his approach *more* successful as a causal primitive. Dowe's approach offers a concrete physical criterion for what counts as a causal interaction—the exchange of a conserved quantity. And given the preceding interpretation in terms of quantum fields, the approach also accounts for the passage of time. Although causal interactions are primitive in the sense of not being

decomposable into smaller steps, they are not instantaneous. They occupy a certain amount of space-time: they occur over a period of time, and they span some distance in space.

3. A computer science perspective on causality

Suppose you are sitting at your computer working on a paper. The cursor is positioned in the text. You press the "e" key. An "e" appears on the screen at the cursor position and (as you are later able to confirm) an "e" has been inserted into the document at that spot. Does the key-press *cause* the "e" to appear on the screen and to be inserted into the text?

This seems to fit the Interventionist template for a causal relation—change which key is pressed and the inserted letter will be different. Nevertheless my inclination would be to say that the key-press does not *cause* these results.

The way most interactive programs work is that they set up what are sometimes called *listeners*. A listener is a chunk of software that is notified, i.e. activated, whenever a listened-for event (in this case a key press) occurs. In response to such a notification, the listener examines the notification and, depending on its content (the notification indicates which key was pressed, whether the control or shift keys were down, etc.), it takes one or another series of actions—in this case to insert an "e" on the screen and into the document.

An analogy will help in thinking through the possible causal connection. Imagine you hear the phone ring. (The analogy is that you are a listener and are notified by the ringing that a call to your line has been made.) Suppose you pick up the phone and say "hello." Would you say that the ringing *caused* you to perform those actions? I doubt that many people would. In much the same way I would not say that pressing the "e" key *caused* the insertions of the "e" on the screen and into the document.

More generally, causation tends not to be a standard frame of reference in computer science. Computers and the software they run are deterministic.[39] It is totally predictable (for any input or supplied parameters) how they will behave. Such determinism notwithstanding, we don't talk about causality. Computer scientists and software developers do talk in terms of causal/mechanistic explanations. If you were to ask a software developer why the "e" appeared on the screen (i.e., what led to its appearance), she would say that it appeared on the screen because the user pressed the "e" key, which led to the program being notified of that user action, which led to the program taking certain actions, etc. The construction *led to <something> happening*[40] is understood to mean something like *set up a situation in which <something> happened*. This way of speaking does not necessarily attribute the

[39] There are programs that include randomization features. In some cases these are designed to depend on external signals to generate randomness. For example, the web site Random.org (http://www.random.org/) generates random numbers based on what it calls "atmospheric noise." There are other programs in which (a) activities are designed to occur in parallel, (b) it may not be possible to determine the order in which such activities complete their actions, and (c) the order in which the completions occurs makes a difference to how the program functions. But to make this discussion as straightforward as possible, I will ignore those cases.

[40] She may even use the term *cause* in place of *led to*, but I don't want that to confuse the situation. As explained in the text, the intuition is of greater disengagement than *cause* suggests.

<something> happening directly to whatever created the situation in which it happened. There is always a sense of disengagement of one thing from another.

Perhaps one reason for this sense of disengagement is that there is no physical causation—of the sort discussed in the previous section—in software. As I said about abstractions earlier, since software is not physical it cannot be party to a physical interaction.

I would also offer that this sort of disengagement carries over to our everyday experience. If you asked me whether flipping a light switch *caused* the light to go on, I would say that it didn't. Flipping the light switch enabled an electric current to flow through a circuit, which …, which led to the light going on. The flipping of the switch didn't itself have any direct connection to the light going on. That's clear from noting that if the electricity had been turned off, flipping the switch would have had no effect.

An interventionist might respond in three ways. As I said above, computer scientists do talk about causal (i.e., "because"-based) explanations. We are happy to tell stories about sequences of events and how one led to another. Interventionist causality is often understood to be a theory of causal explanation rather than a theory of causation. So in that sense it is compatible with practice in computer science. I realize that saying we are comfortable with causal explanations for software related phenomena but not with software causation may seem contradictory. I will simply leave it at that.

A second response might be that Interventionist causality is about probabilities rather than physical causation. Smoking does indeed raise the probability of cancer occurring. And pressing the "e"-key does indeed raise the probability of an "e" appearing on the screen and in the document. So interventionist causality is consistent with computer science practice in this sense as well.

Finally, and most interestingly, an interventionist might point out that if you follow the electrical circuits, pressing the "e"-key is in fact physically connected to the observed results. Pressing the "e"-key sends a signal from the keyboard to the computer, which triggers the software listener, which triggers the performance of the pre-programmed actions, which insert an "e" on the screen and into the document. At the hardware level, it is all driven in a forward chaining manner. One thing physically triggers another, which triggers another, etc. So from that perspective I would have a hard time arguing that pressing the "e" key does not cause an "e" to appear.

The thing is, as a software person, I don't think in terms of what the hardware does. I think in terms of how the software is programmed. A hardware explanation in this case is like a behavioral explanation. It would be like saying that the ringing of the telephone causes you on some physiological level to pick it up and say "hello." And in fact, one might be able to trace the physiological connection: the sound of the phone, triggered vibrations in the ear, which triggered nerve firing to the auditory center of the brain, which triggered, … which triggered your picking up the phone and saying "hello." But that's not how we think of our own behavior, and that's not how I think about software.

As it turns out this is a nice illustration of two levels of abstraction. At the level of abstraction represented by a programming language, there are listeners, which are notified of events, which then

decide[41] how to proceed. At the lower level of abstraction represented by hardware, events are forward chained: A triggers B, which triggers C, etc. The implementation of programming language abstractions in terms of hardware abstractions creates for the programmer (the illusion of?) the programming language's more disengaged level of abstraction. Blind to-ings and fro-ings at the hardware level become considered decision-making at the software level. Since this paper is not about mind or consciousness, I won't attempt to push this any further. But clearly there are parallels which might be worth further investigation.

4. Ongoing underlying machines and pushy explainers

A computer science view of causality is similar to how Hoefer (2010) characterized the generally held view of the laws of nature.

> In the physical sciences, the assumption that there are fundamental, exceptionless laws of nature, and that they have some strong sort of modal force, usually goes unquestioned. Indeed, talk of laws "governing" and so on is so commonplace that it takes an effort of will to see it as metaphorical. We can characterize the usual assumptions about laws in this way: the laws of nature are assumed to be *pushy explainers*. They *make things happen in certain ways* , and by having this power, their existence lets us *explain* why things happen in certain ways.

Much of the computer science view of the world is based on the notion of an underlying machine, which is always in operation doing something. These are the pushy explainers. Software provides a way to tap into that underlying activity and direct it in certain desired ways. A number of examples come to mind.

- A general purpose computer is a machine that continually executes a fetch/execute cycle. An instruction is fetched from storage and then executed after which another instruction is fetched, etc. This ongoing underlying fetch/execute cycle powers everything else.
- A database is a machine that stores information in tables. It is capable of creating new tables, modifying existing tables and the content, and responding to queries about the contents of tables. That ongoing underlying machine enables one to build and use databases. A database machine is typically implemented as software running on a general purpose computer. In the context of its use as a database, it's a database machine.
- A browser is a machine that displays what is called a Document Object Model (DOM). Typically DOM's are expressed in a language called HTML, which browsers read and convert into a DOM. As the DOM is changed, e.g., by software (often written in JavaScript) as it interacts with the user or another computer (the server), what is displayed also changes. It is this ongoing underlying activity of the browser as a DOM displayer that enables web pages to be made visible. A browser is typically implemented as software running on a general purpose computer. In the context of its use as a browser, it's a DOM display and manipulation machine.
- Earlier we discussed a keyboard connected to a computer. We portrayed this as two ongoing underlying machines. The keyboard itself is always available to respond to key presses and send signals regarding those key presses to a computer. Software listeners in the computer are

[41] in the software sense of *decide*. Software decides how to proceed by executing what are known as conditional statements: **`if`** `<condition>` **`then`** `<action 1>` **`else`** `<action 2>`.

> machines that respond to signals by taking various actions. So here we have two ongoing underlying machines—one mainly physical, the keyboard, and one mainly software, the listeners—that enable a user to interact through a keyboard with a document.

How are these pushy explainers? From the pushy explainer perspective the universe is a machine whose ongoing activity is characterized by the laws of physics. The universe as a physics machine is similar to a general purpose computer as a fetch/execute machine or a browser as a DOM display machine. In all cases one is relying on certain ongoing underlying processes that carry everything else along with them. In this view there is no causality at all—except to the extent that one wants to think of causality as the consequences of the activity of the underlying machine.

E. Downward ~~causation~~ entailment

I would now like to turn to downward (or top-down) causation.

1. Downward causation: a zombie idea

A zombie idea[42] is one that has been shown to be mistaken but refuses to die. As far as I'm concerned top-down causation is a zombie idea. Even though there is near universal agreement that the fundamental laws of physics[43] are sufficient to explain all observed phenomena, the Feb 2012 issue of *Interface Focus* (Ellis, Noble, and O'Connor 2012) was devoted to top-down causation, a phenomenon that on its face is not compatible with standard physics.

A standard interpretation of top-down causation is that (a) there are naturally occurring phenomena—i.e., phenomena that are part of the natural[44] world—that are not explainable by the fundamental laws of physics; (b) other, autonomous, laws govern those phenomena; (c) the entities those laws govern are higher level in the sense that they are composed of entities more elementary than themselves and are of a size, scale, or granularity significantly larger than that of the particles of which they are composed; and (d) the behavior of the entities those laws govern has an effect on lower level entities, including the fundamental elements of physics. In other words, top-down causation claims that phenomena occurring among macro entities and governed by their own autonomous laws are capable of affecting entities that are relatively micro with respect to them. A typical example—in fact one of the first to be used in this way—is a wheel rolling down a hill. The action of the wheel is governed by the wheel's geometry and the laws describing how a rolling circular object moves. Among the consequences of the wheel's motion is that the (cycloidal) trajectory followed by the particles that make up the wheel is determined by the wheel's shape and motion rather than by the fundamental laws of physics.

Item (b) above does not necessarily imply that these other laws are mysterious—the mathematics of a rolling circle is not especially complex—only that they are not derivable from the fundamental laws of physics.

---

[42] The notion of a zombie idea apparently first appeared in (Barer, Evans, Hertzman, and Johri, 1998). It's recently been popularized by Paul Krugman in his *Conscience of a Liberal* blog in the New York Times.

[43] The fundamental laws of physics are generally taken to refer to general relativity along with what is called the *standard model* of elementary particles and forces understood from the perspective of quantum field theory.

[44] I will follow (Ladyman and Ross 2006) in using *natural* rather than *material* or *physical*.

The laws of economics provide other examples. The "law" of supply and demand states that an excess of demand will result in an increase in price and that a deficit of demand will result in a reduction in price. It seems hopeless to attempt to reduce this law to or derive this law from the fundamental laws of physics. Yet economic laws have an effect on more elementary elements as money and goods change hands—and thereby move around in space.

The preceding examples notwithstanding, top-down causation seems to make no scientific sense. If the behavior of an element is determined (even probabilistically) by one set of laws (the fundamental laws of physics), that behavior can't also be determined by another set of laws—unless the two sets of laws demand the same behavior of the particle. But in that case, the second set of laws is redundant and can be discarded. This is Kim's (e.g., 2005) causal exclusion argument, which makes top-down causation seem incoherent.

Nonetheless the idea hasn't gone away. Top-down causation hasn't gone away because the (perhaps naïve) intuitive evidence in its favor seems so convincing. According to the introduction to (Ellis, et. al., 2012), "Many examples included in this issue provide good evidence for top-down causation." And indeed those examples, like the example of the wheel and the example of economic laws, seem, at least at first, quite convincing.

I would argue that the *Interface Focus* examples do not illustrate top-down causation. They reflect something akin to top-down causation, which I will call *downward entailment*. They are implications of abstract interactions for the elements that implement the abstractions.

2. The origin of downward causation: Sperry and Campbell

Although the term *downward causation* itself first appeared in Campbell (1974), Sperry (1970) used the wheel example in discussing the apparent phenomenon a few years earlier.

> I think of no simpler example to illustrate [downward causation] than the one of the wheel rolling downhill in which the displacement in time and space and the subsequent fate of the entire population of atoms, molecules, and other components within the system are determined very largely by the holistic properties of the whole wheel as a unit, like its shape, size, weight, etc.

This little example nicely illustrates the problem. A wheel, a macro object, apparently causes its (micro) atoms and molecules to follow a trajectory that depends on the size and shape of the wheel itself. But is this problem unique to a wheel? What about the molecules in the fender of a car riding along the freeway? Or those in the finger of a passenger in the car? The position of the molecules depends on the motion of the car. Yet we are not bothered by that? (Or are we?)

3. Bedau on downward causation

Bedau (2003) asserts that examples similar to Sperry's are not only unremarkable but "philosophically unproblematic."

> An ocean wave demolishes a sand castle, causing the violent dislocation of a grain of sand. A vortex in the draining bathtub causes a suspended dust speck to spin in a tight spiral. A traffic jam causes my car's motion to slow and become erratic. I take it as uncontroversial that such ordinary cases of downward causation are philosophically unproblematic. They violate no physical laws, they are not preempted by micro causes, and they are not viciously circular or incoherent.

The preceding statement surprises me since I thought that no form of downward causation was philosophically unproblematic. Given the allegedly unproblematic status of these instances of downward causation, it's not clear to me what else Bedau thinks need be said. Nevertheless, he continues by relating these examples to what he calls weak emergence.[45] Having labeled a wave, a vortex, and a traffic jam as weakly emergent Bedau then labels the examples of downward causation presented above as weak macro causes and weak downward causation. He then retracts the assertion that they are instances of true downward causation, asserting that they are nothing more than the aggregation of micro causes.

> [A] weak macro cause is nothing more than the aggregation of micro causes. [Therefore,] macro and micro causes are not two things that can compete with each other for causal influence. One constitutes the other. So, the micro causes cannot exclude [via causal exclusion] weak macro causes.

Bedau realizes that such a nothing-but position raises the issue of the autonomy of the macro causes. Bedau first explains the problem.

> The preceding discussion of downward causation emphasized that weak emergent phenomena are nothing more than the aggregation of the micro phenomena that constitute them. This prompts a final worry about whether the explanations of weak emergent phenomena are sufficiently autonomous. … If the underlying explanation of the macro phenomena is merely the aggregation of micro explanations, then all the real explanatory power resides at the micro level and the macro phenomena are merely an effect of what happens at the micro level. In this case, weak emergent phenomena have no real macro-level explanatory autonomy …

He responds with an example.

> Consider a transit strike that causes a massive traffic jam that makes everyone in the office late to work. Each person's car is blocked by cars with particular and idiosyncratic causal histories. The traffic jam's ability to make people late is constituted by the ability of individual cars to block other cars. Aggregating the individual causal histories of each blocking car explains why everyone was late. However, the aggregate micro explanation obscures the fact that everyone would still have been late if the micro causal histories had been different. The transit strike raised the traffic density above a critical level. So, even if different individual cars had been on the highway, the traffic still would have been jammed and everyone still would have been late. The critical traffic density provides a macro explanation that is autonomous from any particular aggregate micro explanation.

So the traffic density—a property of the macro traffic jam—is the (downward) cause of people being late. But it seems to me that this is essentially the same as saying that the fact that water draining from a bathtub forms a vortex is the (downward) cause of the trajectory of the dust caught in the flow. It's also a restatement of Sperry's example. The material of which a wheel is made doesn't matter. Whatever the material, the shape of the wheel (downwardly) causes the elements that make it up to follow particular trajectories. It seems that we are now back where we started—Bedau's assertion that certain forms of downward causation are philosophically unproblematic. It's not clear to me what we got out of this round trip.

[45] Bedau included this discussion as part of his analysis of emergence. For our purposes here, though, the fact that Bedau labels a phenomenon as *weakly emergent* is not relevant to the issue of downward causation.

It seems to me that what's missing from Bedau's analysis are distinctions among quasi-interactions (Bedau's micro causes), abstract interactions (many of Bedau's macro causes), and abstract implementations (which should but don't appear in Bedau's discussion).

4. A better way to look at downward causation

I think there's a better way to look at downward causation.

O'Connor's definition of top-down causation

As preliminary steps to a definition for top-down causation, O'Connor (2012) offers two other definitions for causation: *causation as production* and *causation as counterfactual dependence*. These appear to characterize causation in Dowe's sense and in the Interventionist sense respectively. (I've reproduced the definitions in Appendix D along with a brief discussion.) O'Connor then introduces a third definition of *causation* specifically for top-down causation.

> **[Definition 3 of *Causation* from (O'Connor, 2012)]** *Causation, top-down:* higher level or systemic features that shape dynamical processes at lower levels.

This definition seems far too broad. Here is a list of higher level or systemic elements some of whose features would seem to "shape dynamical processes at lower levels": the plumbing in buildings, animal circulatory systems, streets and highways, most electronic communication systems, command-and-control systems (hierarchical or not), corporate charters, Robert's Rules of Order, and the syntactic structures of natural languages. As this list suggests, virtually any structure through which things move or any set of rules that specify or constrain how things operate or how they are organized would seem to have features that "shape dynamical processes at lower levels." But let's leave that issue and continue.

O'Connor goes on to identify what he calls two varieties of top-down causation.

> **[Continuation of Definition 3 of *Causation* from (O'Connor, 2012)]** One variety of top-down causation involves constraints that preclude many forms of physical possibility present in unorganized matter, possibilities that would lead to systemic dissolution. A stronger, more controversial variety would involve higher level features exerting an irreducible, productive causal influence upon lower level processes. (See emergence: productive causal.) [The "see emergence" reference is part of O'Connor's definition.]

I see the second variety as essentially the same as strong emergence, which I discussed earlier. It's the first variety of top-down causation that's most relevant here.

The focus on preclusion in O'Connor's first variety of top-down causation strikes me as backwards. It seems to me that what's important is not that constraints "preclude many forms of physical possibility" but that the constraints create a new structure. For example, consider a simple hinged door. Screws hold the hinges to the door; other screws hold the hinges to the door jamb. The two sets of screws constrain the door components in very specific ways. What's important about the constraints is that the ensemble of door, door jam, screws, and hinges becomes a hinged door. Of course many other possibilities are precluded. But that's not really the point. It's the binding together in specific ways that's important. Consequently I find the use of the term *constraints* somewhat misleading. This usage is a fairly common, however; for example, see Hooker (2012).

Structure and organization are central to this discussion. Constraints bind components together to create new entities. Whether a constraint is as unsophisticated as the screws in the preceding example, as elegant as the use of gravity and component rigidity to sustain the structure of an arch, or as arcane as the strong force which binds quarks together to form nucleons and atomic nuclei, constraints are the enablers of higher level structure and functionality. As O'Connor's definition indicates, were the constraints that hold a composite entity together to fail, the entity would—by definition(!)—dissolve into its components and (most likely) assume one of the other forms of physical possibility.

In my terms this is saying that when components are constrained to implement some higher level abstraction, then as long as the implementation persists properties of the abstraction can be applied to the components. From this broader perspective, it's not clear to me why O'Connor's definition doesn't imply that top-down causation is an inevitable consequence of the creation of any higher-level or composite entity, i.e., the implementation of virtually any new abstraction. Such a perspective would seem to strip the notion of top-down causality of much of its meaning.

Since the primary function of a constraint is to hold things together, one of the fundamental issues concerns the nature and variety of forces available for doing that. I discuss this in Appendix A. Here I will simply assume that there are higher-level or composite entities and that they have been created by binding some components together in some organized way.

The primary remaining question is whether there is anything worth saying about *how* higher-level entities shape the dynamical processes of some of their lower-level elements—i.e., what one can say about how properties of an abstraction apply to the components that implement the abstraction. It will be useful to look at Sperry's wheel again with this in mind.

Implications of having implemented an abstraction

Sperry's wheel is (let's assume) a collection of molecules that are bound together in such a way that their overall shape is that of a wheel. (Let's ignore the question of whether it's a wheel with spokes, a wheel rim with a tire mounted on it, a disk with a hole in the middle through which an axel is put and secured, etc. Let's just assume we have a wheel of some sort.) What is that really saying? It's saying that the forces that hold the molecules together to implement the wheel are stronger than the other forces those molecules are encountering, such as the buffeting of air molecules. Of course this doesn't mean that one couldn't break the wheel by applying still stronger forces, e.g., with a sledge hammer. But as long as the wheel is intact, that seems to be the situation.

What are the forces that hold a wheel's molecules together, and does it make sense to say that they are strong enough to do the job? The forces are the standard electromagnetic forces. These are the forces that hold any compound of atoms or molecules together. Depending on the materials being held together and the shape in which they are held, these can be very strong forces. The atoms of a diamond, for example, are held together by the electromagnetic force; the forces that makes glue sticky and that enable it to bind other things together is also the electromagnetic force; friction is a consequence of the electromagnetic force. In fact, in our daily lives, besides the kinetic force implicit in momentum and pressure, the only forces most people encounter are gravity and the electromagnetic force. So it certainly makes sense to say that the electromagnetic force is strong enough to hold some molecules

together in the shape of a wheel, i.e., to implement a wheel with such abstract properties as a diameter and circumference. These properties pertain to a physical wheel because the molecules that implement the wheel are put together in specific ways.

Once we have a wheel, what happens to it? The primary force to which the wheel itself is subject is gravity. Since gravity is a much weaker force than the forces that hold the wheel together the wheel remains in the shape of a wheel. If the wheel is on a hill, gravity will pull on it, and it will roll down the hill. Gravity is not strong enough to pull the molecules of the wheel apart, but it is strong enough to pull the wheel as a whole down the hill. As the wheel rolls down the hill, the molecules that implement it continue to be held together in a wheel shape—unless the wheel smashes into something that causes it to shatter. Consequently, the molecules that implement the wheel follow a cycloidal trajectory—unless the wheel has shattered, in which case they don't. With that we have Sperry's wheel example: the shape of the wheel—as long as it remains a wheel—determines the trajectories of the molecules that implement it.

Central to the preceding discussion is the shape of the wheel (round) and the fact that as gravity pulls on the wheel, it rolls down the hill the way round objects roll. Churchill is credited[46] with saying, "We shape our buildings; thereafter they shape us." That's a good way of looking at what happened to the components of the wheel. The wheel components were put together in such a way as to produce a round object. Once the wheel was built, the trajectory of its components was determined by the fact that they were part of the implementation of a round object.

If this is top-down causation, it's a promise that can't be kept

Is this top-down causation? According to the *Interface Focus* definition it is. The shape of a wheel, certainly helps to[47] shape the trajectories of those molecules. Furthermore, one can reasonably argue that this qualifies as Interventionist causality. The trajectories of the molecules depend on the shape of the wheel. Change the shape, and the trajectories will change.

Yet there are reasons not to refer to this as a causal relationship. Earlier we said that an Interventionist-style causal relationship promises (at least implicitly) that there is a physical mechanism that underlies that relationship. But that's not true in this case. There is no physical mechanism that relates the shape of the wheel—as a shape—to the wheel's molecules. A shape is an abstraction. As noted earlier, an abstraction cannot serve as a physical cause. A causal event—at least according to Dowe's definition—is an interaction between two physical elements. Since an abstraction is not physical, it can't be a party to a causal event. Therefore, it doesn't seem right to say that there is a physical causal relationship between the wheel's shape and the trajectories of the molecules that implement it.

Downward entailment

But there's certainly some connection between the wheel's shape and the trajectory of its implementing molecules. What is it? Rather than refer to the relationship between the shape of the wheel and the

[46] *Time*, September 12, 1960.

[47] I say "helps to" because gravity has something to do with it also.

trajectories of its molecule as causal, I suggest that we refer to it as one of entailment.[48] The shape of the wheel entails the trajectories of the molecules. From the shape and motion of the wheel as a wheel and from the position of any molecule in the wheel's implementation, we can compute the trajectory of that molecule. Since the entailment is from properties of the implemented abstraction (the wheel's shape and motion) to a property of a lower-level element (its trajectory), it makes sense to call this downward entailment. This is another way of saying that the wheel as an implemented abstraction interacts with the surface of the earth and with gravity as described by Newton's laws and the principles of geometry. The elements that implement the wheel as an abstraction come along for the ride. As long as the implementation is intact the way the abstraction behaves enables one to compute how its components behave.

This notion is not entirely new. What I'm calling downward entailment has been discussed elsewhere as mathematical causality.[49] In other contexts *mathematical causality* refers to the idea that if a mathematical relationship holds among variables and if one can solve for one of the variables in terms of the others, the value of the solved-for variable is causally dependent (in the sense of mathematical causality) on the others.[50, 51]

There is an important proviso with regard to downward entailment. As I noted above, were the forces that bind the atoms and molecules into the shape of a wheel to fail, i.e., were the implementation of the wheel to break down, the wheel would break apart as it bounces down the hill, and the molecules that formerly made up the wheel would no longer be an implementation of a round object. Consequently there would be nothing from which to deduce the trajectories of those molecules. The entailment would fail because the abstract shape ceased to exist.

Downward entailment applies in all cases in which an abstraction has been implemented. Once an instance of an abstraction exists, if that object behaves in the world and is operated upon in the world in terms of the abstraction it instantiates, then the implementing elements are carried along. Their fate is determined by the fate of the higher level element in its role as the higher level abstraction.

This may have been what Bedau had in mind when he called downward causation philosophically unproblematic and what Kim (2007) had in mind when he responded to Sperry's examples as follows.

---

[48] Although this is entailment in the same logical sense that Weinberg used the term—a consequence of a logical argument—the two uses of the term are not intended to echo each other. Weinberg's use of entailment suggested a relatively long and drawn out argument. As used here, entailment is expected to be a fairly direct logical consequence of a given situation.

[49] (Schmid, 2006) refers to the causality inherent in Aristotle's formal cause as *mathematical causality*: a physical statue is mathematically causally dependent on the abstract idea of the statue, it's Aristotelian formal cause. In modern dress, this is not far from what I am calling *downward entailment*.

[50] (Hofstadter and Sander, 2013) uses the term *mathematical causality* in this sense. (Birkhoff and Von Neumann, 1936) and (Plotnitsky, 2010) appear to be attempts to apply this sense of mathematical causality to quantum mechanics.

[51] Mathematical causality differs from what might be called statistical causality—e.g., the second law of thermodynamics.

> I fall from the ladder and break my arm. I walk to the kitchen for a drink of water and ten seconds later, all my limbs and organs have been displaced from my study to the kitchen. Sperry's bird flies into the blue yonder, and all of the bird's cells and molecules, too, have gone yonder. It doesn't seem to me that these cases present us with any special mysteries rooted in self-reflexivity, or that they show emergent causation to be something special and unique. For consider Sperry's bird: for simplicity, think of the bird's five constituent parts, its head, torso, two wings, and the tail. For the bird to move from point $\underline{p}_1$ to point $\underline{p}_2$ *is* for its five parts (together, undetached) to move from $\underline{p}_1$ to $\underline{p}_2$. The whole bird is at $\underline{p}_1$ at $\underline{t}_1$ and moving in a certain direction, and this causes, let us suppose, its tail to be at $\underline{p}_2$ at $\underline{t}_2$. There is nothing mysterious or incoherent about this. The cause—the bird's being at $\underline{p}_1$ at $\underline{t}_1$ and moving in a certain way—includes its tail's being at $\underline{p}_1$ at $\underline{t}_1$ and moving in a certain way. But that's all right: we expect an object's state at a given time to be an important causal factor for its state a short time later. And it is clear that Sperry's other examples, such as the water eddy and the rolling wheel, can be similarly accommodated.

If downward entailment provides what I take to be a clearer understanding of this class of phenomena, should we dismiss the notion of top-down (or downward) causation? I'd say we should—and finally kill off that zombie idea. The notion of downward entailment gives us pretty much everything we want from top-down causality (except the mystery), and we get it with an additional depth of understanding.

Downward entailment can impose real constraints: reducing one problem to another

> A mathematician and an engineer were at a conference. Returning from lunch they became concerned that they might be late for the afternoon session. Suddenly they spotted a dumpster on fire. The engineer noticed a garden hose hooked up to a spigot. He turned on the water and put out the fire. Then they hurried back to the conference, where they arrived just as the first presentation was about to begin.
>
> The next day they were again walking back from lunch, and again they were concerned about being late. As they passed the dumpster the mathematician struck a match and threw it in—thereby reducing today's problem to one that had already been solved.

In mathematics, computer science, and probably many other fields, one reduces an unsolved problem ***U*** to a solved one ***S*** by showing how a solution to the solved problem can be used to solve the unsolved problem. Induction proofs parley this technique indefinitely. To prove something for the *n+1* case one shows how a proof for the *n* case can be used to prove the *n+1* case.

Reduction can also be used to demonstrate a negative property such as undecidability. Suppose problem ***U*** is known to be Undecidable and suppose one wants to know whether problem ***S***, which is Suspected of being undecidable, is in fact undecidable. If one can reduce ***U*** to ***S*** then one knows that ***S*** is undecidable. Why? Because if a solution to ***S*** can be used to construct a solution to ***U*** then if there is a solution to ***S***, there is a solution to ***U***. But we know that ***U*** is undecidable. Therefore there must not be a solution to ***S***.

Reduction enables downward entailment to offer a bonus—namely a framework in which higher level elements can impose formally provable limits on lower level elements. I'll illustrate with an example

from the Game-of-Life.[52] It is known that one can implement an arbitrary Turing machine using the Game of Life as a platform. (Rendell, 2002) This shows that the Game of Life is Turing complete. Consequently it is (formally) undecidable whether a Game-of-Life grid started in an arbitrary initial position will ever reach a terminal, i.e., unchanging, state.

It's worth looking at this from the perspective of downward entailment. The undecidability of the Game of Life is entailed by the facts that (a) it is possible to implement a Turing machine as a higher level abstraction with the Game of Life as the lower level and (b) the halting problem for Turning machines is undecidable. In other words, a negative property of the Game of Life is entailed by a negative property of a higher level structure that can be built within the Game of Life.

This is a clear case in which we have downward entailment (something is true about a lower level because something is true about a higher level) rather than downward causation (the implemented Turing machine doesn't *cause* the Game-of-Life to be undecidable).

This example also offers another illustrates of how a special science can be autonomous and not a logical consequence of a lower level. We can think of Turing's theory of computability as the "science" of Turing machines. It holds the same relationship to the rules of the Game of Life as the special sciences hold to the laws of fundamental physics. It is an autonomous science that gives us information about Turing machines. It cannot in any useful sense be said to be a consequence of the Game of Life rules.[53]

Let's take this a step further with a fantasy thought experiment—you'll have to give me a lot of room for this fantasy. Suppose we are scientists. We are investigating the Game-of-Life world, but we know very little about it, not even the transition rules.

Let's assume that we run across a colony of Turing machines "in the wild" in the Game-of-Life world. (I warned that this would require a fairly generous suspension of disbelief.) We observe these creatures for a while and develop a science about them, namely the Theory of Computability. We capture some of them and show them along with the new science we developed to our friends the Game-of-Life physicists.

Fascinated, the physicists ask themselves: how do these creatures, which are described by such an interesting science, exist in a Game-of-Life world? That's a good question. In fact, the answer is quite complex, but the physicists are up to it. By examining the internal mechanisms of these Turing machine

[52] The Game of Life is a cellular automaton in which at each transition cells may switch states between live and dead, i.e., on or off, according to the following rule. A cell will be live after a transition if before the transition either (a) the cell and exactly two of its neighbors were live or (b) the cell (live or dead) had exactly three live neighbors. Otherwise the cell will be dead after the transition.

[53] One could define what might be called an extended Game-of-Life device: a Game-of-Life grid with an attached Turing machines. One could then prove the usual computability theorems about the Turing machine part of this compound device. One might claim that this device is derived from the Game of Life and that everything we can know about it is derived from the rules of the Game of Life. Is that at all a useful way of looking at it? A Theory of Computability developed on the basis of this device is not in any useful sense a consequence of the rules of the Game of Life. That the Game-of-Life rules are unused assumptions in this theory doesn't give them any special claim on the theory.

creatures, the physicists discover not only how they exist in a Game-of-Life world but also the underlying Game-of-Life rules themselves.

Everyone will have come out ahead. We have a new science of a new domain (Computability theory); we have discovered that this new domain can exist as higher level elements within a Game-of-Life world; as a result of examining how that can be, we have discovered more about how the Game-of-Life world itself works; and we have also discovered that some of the properties of the new domain entail some additional and otherwise unknown properties of the Game-of-Life.

F. What have we said?

As a computer scientist I have long believed that computational thinking can be of value when investigating philosophical questions. One of the fundamental principles of modern computer science is the distinction between a specified abstraction and the software that implements that abstraction. Adherence to that principle has enabled the software industry to develop enormously complex systems.

I distinguished between two sorts of interactions among implementations of abstractions: quasi-interactions that depends on how the implementations work and abstract interactions that depend on properties of the abstractions themselves.

I showed that a useful way to think about emergence is in terms of abstractions and their implementations. When looked at in this way, emergence loses its aura of mystery. I also pointed out that phenomena that at one point might have been candidates for strong emergence did not delegitimize science.

I argued that the conceptual distance between (a1) software abstractions as seen by users and (b1) bits to-ing and fro-ing at the level of logic gates is comparable to the conceptual distance between (a2) the phenomena studied by the special sciences and (b2) fundamental physics. In investigating how these conceptual distances are bridged, I argued for the autonomy of many of the levels and collections of abstractions developed along the way. This discussion also offered insights into how complex systems are able to develop and not crumble under the weight of their own complexity.

We visited the debate about whether the laws of physics entail the laws of the special sciences. In reviewing a video of a recent conference I suggested that the two sides were not as far apart as they seemed. One side was looking at the question of analysis; the other side was looking at the question of synthesis. Synthesis suggests that the special sciences can be autonomous; analysis replies that the important question for physicists is how entities that obey those laws can exist in a world governed by the fundamental laws of physics, i.e., how the abstractions that the laws describe are implemented.

I reviewed causation and its conceptual role in computer science—namely that we tend to use the language of causation when telling stories about how something might have happened, but we don't use that language when thinking about how software operates. As part of this exploration I discussed how the programming language abstraction of the conditional expression creates what may seem like an illusion of choice. I also discussed the similarity between a computer science view of causality based on ongoing processes and the pushy explainer view of how the laws of physics enables physical explanations.

This paper was originally motivated by the striking persistence of the idea of top-down causality. It's fairly clear from a naturalistic perspective that there can't be anything like top-down causality. Yet the notion persists. I asked why. I pointed out that downward causation in the form reviewed is much too broad a concept. I then argued that phenomena that look like they illustrate downward causation are better understood as instances of downward entailment. I pointed out that downward entailment can be used both positively—to compute properties of lower-level elements on the basis of properties of higher-level abstractions—and negatively—to show that limitations of higher-level abstractions entail limits on the lower-level.

References

Abbott, Russ (2008) "If a tree casts a shadow is it telling the time?" *International Journal of Unconventional Computing*, 4/3, pp 195-222. Best presentation award at the *Conference on Unconventional Computing*, York, UK, 2007.

Abbott, Russ (2009) "The reductionist blind spot," *Complexity*, 14/5 (May/Jun 2009) pp 10-22.

Abbott, Russ (2010a) "Bits don't have error bars: upward conceptualization and downward approximation." in Ibo van de Poel and David E. Goldberg (eds.) *Philosophy and Engineering: An Emerging Agenda*, Springer.

Abbott, Russ (2010b) "Abstract data types and constructive emergence," *Newsletter on Philosophy and Computers* of the American Philosophical Society, v9-n2, Spring 2010, pp 48-56.

Anderson, P.W. (1972) More is Different. *Science* 177 (4047): 393–396.

Arthur, Brian (2009) *The Nature of Technology: What it is and How it Evolves*. Free Press.

Baedke, Jan (2012) Causal explanation beyond the gene: manipulation and causality in epigenetics. *Theoria* 74 (2012): 153-174

Barer, Morris L., R. G. Evans, C. Hertzman and M. Johri (1998). *Lies, Damned Lies, and Health Care Zombies: Discredited Ideas That Will Not Die.* Vancouver: Centre for Health Services and Policy Research.

Barkan, Leonard (2013) *Mute Poetry, Speaking Pictures*, Princeton University Press.

Bedau, Mark (1997) Weak Emergence. In J. Thomberlin, (ed.) *Philosophical Perspectives: Mind, Causation, and World*, Vol 11 Blackwell, pp 375-399.

Bedau, Mark (2002) Downward Causation and the Autonomy of Weak Emergence. *Principia*, 6(1) 5-50, Epistemology and Log Research group, Federal University of Santa Catarina (UFSC), Brazil.

Bedau, Mark (2010) Weak Emergence and Context-Sensitive Reduction. In Corradini, Antonella and Timothy O'Connor (eds.) (2010 and paperback 2013) *Emergence in Science and Philosophy*, Routledge (Studies in the Philosophy of Science).

Bedau, Mark and Paul Humphreys (2008) *Emergence*, MIT Press.

Birkhoff, Garrett and John Von Newmann (1936) The Logic of Quantum Mechanics. *The annals of Mathematics*, 2nd Ser., 37/4, Oct 1936, 823-843.

Bobro, Marc (2013) Leibniz on Causation. *The Stanford Encyclopedia of Philosophy* (Summer 2013 Edition), Edward N. Zalta (ed.) http://plato.stanford.edu/archives/sum2013/entries/leibniz-causation/.

Broadbent, Alex (2011) Inferring causation in epidemiology: Mechanisms, black boxes, and contrasts. In Phyllis McKay Illari, Frederica Russo, Jon Williamson (eds.) *Causality in the sciences*, Oxford University Press.

Brooks, Rodney A, (2012) *Fields of Color*. Rodney A. Brooks.

Burke, James (2012) *Connections* (reprint edition), Simon and Schuster.

Campbell D.T. (1974) Downward causation in hierarchically organised biological systems. In Francisco Jose Ayala and Theodosius Dobzhansky (eds.), *Studies in the philosophy of biology: Reduction and related problems*, pp. 179–186. London/Bastingstoke: Macmillan, p. 179-186.

Carroll, Sean (2010) *From Eternity to Here: The Quest for the Ultimate Theory of Time*, Dutton.

Carroll, Sean (2012) *Moving Naturalism Forward,* Organized by Carroll and sponsored by the Division of Physics, Mathematics, and Astronomy and the Moore Center for Theoretical Cosmology and Physics, California Institute of Technology, http://preposterousuniverse.com/naturalism2012/.

Carroll, Sean (2013) Why does dark energy make the universe accelerate? *The preposterous universe*, Nov 16, 2013: http://www.preposterousuniverse.com/blog/2013/11/16/why-does-dark-energy-make-the-universe-accelerate/.

Chalmers, David John (1995) Facing Up to the Problem of Consciousness. *Journal of Consciousness Studies*, 2(3):200-19.

Chalmers, David John (2006) Strong and weak emergence. *The Re-emergence of Emergence*, Oxford University Press.

Chalmers, David John, David Manley, Ryan Wasserman (2009) *Metametaphysics: new essays on the foundations of ontology*, Oxford University Press.

Cohen, S. Marc (2012) Aristotle's Metaphysics. *The Stanford Encyclopedia of Philosophy* (Summer 2012 Edition), Edward N. Zalta (ed.) http://plato.stanford.edu/archives/sum2012/entries/aristotle-metaphysics/.

Corradini, Antonella and Timothy O'Connor (eds.) (2010 and paperback 2013) *Emergence in Science and Philosophy*, Routledge (Studies in the Philosophy of Science).

Dennett, Daniel C. (2012) What Darwin's theory of evolution reveals about artificial intelligence. *The Atlantic*, The Atlantic Monthly Group, June 22, 2012. *A*dapted from S. Barry Cooper and Jan van Leeuwen (eds), *Alan Turing: His Work and Impact*, Elsevier, 2013.

Dowe, Phil (2000) *Physical Causation*, Cambridge University Press.

Dowe, Phil (2012) The causal-process model theory of mechanisms. In Illari, Phyllis McKay, Russo Fererica, and John Williamson, *Causality in the Sciences*, Oxford University Press.

Einstein, Albert (1918) Principles of Research (address). Physical Society, Berlin, reprinted in Ideas and *Opinions*, Crown, 1954. http://www.cs.ucla.edu/~slu/on_research/einstein_essay2.html.

Ellis, George F. R., Denis Noble and Timothy O'Connor (2012) Special issue on top-down causation. *Interface Focus*, Royal Society Publishing, 2012 2.

Ellis, George F. R., Denis Noble and Timothy O'Connor, (2012) Introduction: Top-down causation: an integrating theme within and across the sciences? *Interface Focus*, Royal Society Publishing, Vol 2, No 1, pp 1-3.

Evans, David, Andrei Hagiu, and Richard Schmalensee (2006) Invisible Engines : How Software Platforms Drive Innovation and Transform Industries. MIT Press.

Evans, David S. (2013) The Consensus Among Economists on Multisided Platforms and its Implications for Excluding Evidence that Ignores It. Presented at the American Bar Association Section of Antitrust Law Spring Meeting, Washington, D.C.

Falcon, Andrea (2012) Aristotle on Causality. *The Stanford Encyclopedia of Philosophy* (Winter 2012 Edition), Edward N. Zalta (ed.) http://plato.stanford.edu/archives/win2012/entries/aristotle-causality/.

Farina, Dario, Winnie Jensen, and Metin Akay (2013) *Introduction to Neural Engineering for Motor Rehabilitation*, Copyright Institute for Electrical and Electronic Engineers (IEEE), John Wiley and Sons.

Feynman, Richard (1985) *QED: The Strange Theory of Light and Matter*, Princeton University Press.

Filistrucchi, Lapo, Geradin, Damien and van Damme, Eric (2012) Identifying Two-Sided Markets. TILEC Discussion Paper No. 2012-008.

Fodor, Jerry (1974) Special Sciences (or: the disunity of the sciences as a working hypothesis). *Synthese*, 28 (2):97-115.

Fodor, Jerry (1997) Special Sciences: Still Autonomous After All these Years. *Noûs*, Vol. 31, Supplement: *Philosophical Perspectives, 11, Mind, Causation, and World*, (1997), pp. 149-163 Blackwell Publishing.

Hall, Ned (2004) Two Concepts of Causation. In J. Collins, N. Hall, and L. A. Paul, eds., *Causation and Counterfactuals*. MIT Press, pp. 181-204.

Hindley, Roger (1969) The Principal Type-Scheme of an Object in Combinatory Logic. *Transactions of the American Mathematical Society*, Vol. 146, pp. 29–60.

Hobson, Art (2013) "There are no particles, there are only fields," *American Journal of Physics*, Volume 81, Issue 3, 211-223, March 2013.

Hoefer, Carl, "Causal Determinism", *The Stanford Encyclopedia of Philosophy* (Spring 2010 Edition), Edward N. Zalta (ed.). http://plato.stanford.edu/archives/spr2010/entries/determinism-causal/.

Hofstadter, Douglas and Emmanuel Sander (2013) *Surfaces and Essences: Analogy as the Fuel and Fire of Thinking*, Basic Books 2013.

Hooker, Cliff (2012) On the import of constraints in complex dynamical systems. *Foundations of Science*, October 2012, Elsevier.

Humphreys, Paul (1997) Emergence, Not Supervenience. *Philosophy of Science* 64, pp. S337-S345.

Kim, Jaegwan (2007), *Physicalism, or Something Near Enough*, Princeton University Press.

Lally, Jim (2005) Accelerometer Selection Considerations: Charge and ICP Integrated Circuit Piezoelectric. PCB Piezotronics, Inc. http://www.pcb.com/techsupport/docs/vib/TN_17_VIB-0805.pdf.

Lane, Nick (2009) *Life Ascending: The Ten Great Inventions of Evolution*. W. W. Norton.

Liskov, Barbara (1974) Programming with abstract data types. *Proc. ACM Conf. on Very High Level Languages*. SIGPLAN Notices, Vol. 9, April 1974, pp. 50-59.

Loewer (2008) Why There *Is* Anything except Physics. In Jakob Hohwy and Jesper Kallestrup, *Being Reduced: New Essays on Reduction, Explanation, and Causation* pp 149-163.

Loewer (2009) Why Is There Is Anything except Physics. *Synthese* 170(2):217-233.

McLaughlin, Brian (1992) The rise and fall of British emergentism. In Ansgar Beckermann, Hans Flohr, Jaegwon Kim (eds) *Emergence Or Reduction?: Essays on the Prospects of Nonreductive Physicalism*, Walter de Gruyter, pp 49-93.

Milner, Robin (1978) A Theory of Type Polymorphism in Programming. *Journal of Computer and System Science (JCSS)* 17, pp. 348–374.

O'Connor, Timothy (2012) Glossary from the Introduction to Ellis, George F. R., Denis Noble and Timothy O'Connor, "Special Issue on Top-Down Causation," *Interface Focus*, Royal Society Publishing, 2012 2.

O'Connor, Timothy, Wong, Hong Yu (2012) Emergent Properties. The Stanford Encyclopedia of Philosophy (Spring 2012 Edition), Edward N. Zalta (ed.) http://plato.stanford.edu/archives/win2006/entries/properties-emergent/.

Odijk, Daan, Edgar Meij, and Maarten de Rijke (2013) Feeding the second screen: semantic linking based on subtitles. *Proceedings of the 10th Conference on Open Research Areas in Information Retrieval. pp 9-16.*

Pearl, Judea (2000 and 2009) Causality: Models, Reasoning, and Inference, Cambridge University Press.

Pearl, Judea (2011) The Structural theory of causation. In Illari, Phyllis McKay, Russo Fererica, and John Williamson, *Causality in the Sciences*, Oxford University Press.

Pearl, Judea (2012) Causal Inference in Statistics: A Gentle Introduction. Tutorial presented at the *Joint Statistical Meetings* (JSM-12), American Statistical Association, San Diego, CA, July 29, 2012, http://bayes.cs.ucla.edu/jsm-july2012.pps.

Pearl, Judea (2013) The Mathematics of Causal Inference: with Reflections on Machine Learning. *Microsoft Research Machine Learning Summit 2013*, April 23, 2013, http://research.microsoft.com/apps/video/default.aspx?id=191888.

Peyton Jones, Simon, ed. (2003). *Haskell 98 Language and Libraries: The Revised Report*. Cambridge University Press.

Plotkin, Gordon (2004) A Structural Approach to Operational Semantics. *Journal of Logic and Algebraic Programming*, 60-61, 17-139.

Plotnitsky, Arkady (2010) On physical and mathematical causality in quantum mechanics. *Physica E*, 42/3, 279-286, Elsevier.

Putnam, Hilary (1975) Philosophy and our Mental Life. In *Mind, Language, and Reality*. Cambridge University Press, pp 291-303.

Rendell, P. (2002) Turing universality of the game of life. In A. Adamatzky (ed.) *Collision-Based Computing*, Springer, 513-540.

Rochet, Jean-Charles, and Jean Tirole (2003) Platform competition in two-sided markets. *Journal of the European Economic Association* 1.4 (2003): 990-1029.

Schmid, Alexandre (2006) What Does Emergence in Computer Simulations? Simulation between Epistemological and Ontological Emergence. In Flaminio Squazzoni (Ed.) *Epistemological Aspects of Computer Simulation in the Social Sciences*, Second International Workshop, EPOS 2006, Brescia, Italy, October 5-6, 2006, Revised Selected and Invited Papers, Springer.

Schrödinger, Erwin (1944) *What is Life?* Lectures delivered at Trinity College Dublin. Available in *What is Life*, Cambridge University press, reprint edition (2012).

Schoenmaker, Sanne, Markus Barth, Tom Heskes, and Marcel van Gerven (2013) Linear reconstruction of perceive images from human brain activity. *NeuroImage*, July 22, 2013, DOI: http://dx.doi.org/10.1016/j.neuroimage.2013.07.043, Elsevier.

Sellers, Eric W. (2013) New horizons in brain-computer interface research. *Clinical Neurophysiology*. 2013 January; 124(1): 2–4.

Simon, Heloisa, Eros Comunello, and Aldo von Wangenheim (2013) Enrichment of Interactive Digital TV using Second Screen. *International Journal of Computer Applications (0975 – 8887) Volume 64– No.22, February 2013*

Smolin, Lee (2012) A Real Ensemble Interpretation of Quantum Mechanics. *Foundations of Physics*, 42/10 pp 1239-1261, Springer.

Sperry, Roger (1970) An Objective Approach to Subjective Experience. *Psychological Review*, Vol. 77.

van Hove, L. (2001) Optimal Denominations for Coins and bank notes: in Defense of the Principle of Least Effort. *Journal of Money, Credit, and Banking* 33, 1015-21.

Weinberg, S. (1995) Reductionism Redux. *The New York Review of Books*, October 5, 1995. Reprinted in Weinberg, S., *Facing Up*, Harvard University Press, 2001.

Wilczek, Frank (2008) *The Lightness of Being*. Basic Books.

Woodward, James (2003) *Making things happen*, Oxford University Press.

Woodward, James (2011) Mechanisms revisited. *Synthese*, December 2011, Volume 183, Issue 3, pp 409-427.

Ziman, John (2000), *Technological Innovation as an Evolutionary Process*. Cambridge University Press.

Zimmer, Carl (2011) More eldritch ant horror! *The Loom, Discover Magazine* May 9, 2011: http://blogs.discovermagazine.com/loom/2011/05/09/4514/#.UOn2c288CSo.

A. Appendix A. Why is there anything except physics?[54]

1. How are higher level entities held together?

Following is a key sentence in O'Connor's definition of *Causation, top-down*.

> One variety of top-down causation involves constraints that preclude many forms of physical possibility present in unorganized matter, possibilities that would lead to systemic dissolution.

[54] The titles of this appendix and the next were selected as a salute to Loewer's identically titled articles (2007) and (2008). I have found Loewer's work in this area among the most insightful and informative of any that I have read. The appendices don't speak directly to his articles, however.

I noted that this sentence seemed to have been written backwards, that a constraint in this context holds something together. Indeed, a constraint limits something's freedom of movement. But the constraint is important in this context not because it enforces a limit but because it keeps the components of a larger structure from moving out of place. In other words, the reason we are interested in constraints is so that we can understand how it's possible to build entities at a higher level from those at a lower level. In this section I will offer a brief summary of my views about how things are held together—about how it's possible for there to be higher level entities. I discussed this issue in a number of earlier papers. The most recent is Abbott (2010b).

As I see it, there are two broad classes of higher level entities: static and dynamic. The static entities are held together by virtue of their being in an energy well. Examples include atoms and molecules at the micro level, things at our level that are held together by screws, nails, etc., and solar systems and galaxies at the highest levels of granularity. In each case, energy is released when the entity is formed. A consequence of this is that static entities have less mass than the mass of their components taken separately[55]—which gives them a strong claim to their own ontological status. Because energy was released when static entities are created, to pull a static entity apart requires the addition of energy. Once created, a static entity needs no further energy to remain an entity—although static entities tend to deteriorate as a result of being buffeting by things around them.

Dynamic entities are held together by ongoing expenditures of energy. Examples are living organisms and groupings of living organisms. A consequence of this is that dynamic entities have more mass than the mass of their components taken separately—which gives them also a strong claim to their own ontological status. Examples of dynamic entities include biological organisms as well as colonies, clubs, corporations, and countries. In all cases, were the members of the dynamic entity to stop expending energy to help hold the entity together, it would cease to exist as an entity. Dynamic entities typically include some static components: biological organisms include physical molecules, and corporations include corporate headquarters. One of the ways in which dynamic entities expend energy is in the maintenance of their static parts.

There is a third but more artificial class of higher level entities. These are symbolic entities within computer systems. I have called them subsidized entities. They persist because there is no deterioration within the symbolic framework of computer systems. Of course computers themselves are subject to deterioration. But the energy used to maintain them doesn't count against the symbolic entities within. Symbolic entities thus live in a charmed world of pristine symbols—you can't break a bit—and pure structure. They are free from any worry about energy required to maintain them. For further discussion see Abbott (2010b).

2. Which higher level entities will come to exist?

Related to the question of how higher-level entities are built is the question of which higher level entities will be built—and will persist. It seems to me that there is a separate answer for static and dynamic entities. The static entities that will come to exist are those that are at a local energy minimum.

[55] Humphreys (1997) made a similar suggestion.

Since higher level static entities exist in energy wells, to the extent possible, nature will flow downhill in terms of energetics. This doesn't mean that there aren't local minima that are not global minima. Water behind a dam and combustible materials that aren't burning are two examples of local energy minima.

Dynamic entities develop and persist through a process of evolution. Dynamic entities require energy to persist. Those that are able to supply themselves with energy will persist; those that can't won't. Over the long term, dynamic entities that are more effective at securing the necessary energy resources are the ones that will survive. Of course evolution depends on a mechanism for recording and reusing designs, and fitness depends on the environment. But my overall sense is that the world of dynamic entities can be understood in standard evolutionary terms.

B. Appendix B. Why there is anything except physics

Loewer (2008) explains what he takes to be Fodor's view of the relationship between the special sciences and physics as follows.

> [E]ach special science taxonomizes nature into natural kinds in terms of its own proprietary vocabulary. What makes a special science a *science* is that it contains lawful regularities stated in its proprietary vocabulary that ground explanations and counterfactuals. He is clear that what makes a special science regularity *lawful* is a fact that is irreducible to the laws and facts of fundamental physics (and other special sciences). That is, the lawfulness of special science regularities is a fact about the world as basic as and independent of the lawfulness of the laws of fundamental physics. Fodor's view can be illustrated with the help of a souped up version of Laplace's demon. The demon knows all the physical facts obtaining at all times and all the fundamental dynamical laws of physics, has perfect computational powers and also a "translation" manual connecting special science and physical vocabularies. The demon is thus able to tell which micro physical situations correspond to, for example, a philosophy conference and is able to determine which generalizations about philosophy conferences are true and which are false. It can do the same for all the special science. It will also be able to tell which special science regularities hold under counterfactual initial conditions and so which hold in all physically possible worlds (i.e. all the worlds at which the fundamental laws of physics obtain). But on Fodor's view the demon *will not* be able to discern which regularities are laws. Because of this "blindness" the demon will be missing those counterfactuals and explanations that are underwritten by special science laws and so will not have an understanding of special science phenomena. Although the demon will be able to predict and explain the motions of elementary particles (or whatever entities are physically fundamental) from the state of the universe at any time and so could have predicted the stock market crash of 1929 it will not understand why it crashed. To do that it would need to know economics.

Loewer then offers what seems to me to be an ingenious proposal for how the special science laws can be understood as inherent in the fundamental nature of the universe. He requires only the laws of physics and the assumption that the entropy of the universe was very low at its beginning. I will not attempt to analyze or comment on Loewer's proposal. In this section I want simply to look at the hypothesized Laplacian demon and see how it might work.

To start I want to strip the souped-up demon of its translation manual. As I understand it, the demon does not use its translation manual for computing. The primary use the demon makes of its translation

manual is to communicate a computed state of the world to those of us who want to see the world in terms of special science types.

In making that claim I am bothered by the statement that the demon "is able to determine which generalizations about philosophy conferences are true and which are false." Since the demon doesn't know the laws of the special science of philosophy conferences, I don't understand how it can determine which generalizations are true and which are false. As I understand it for any putative regularity about philosophy conferences the demon could check to see if that regularity holds for a particular philosophy conference. But I don't see how it can verify the regularity in general. If it had the ability to do that, it would be able to check any purported special science law about philosophy conferences, which we said it was unable to do. Perhaps I'm missing something, but from here on I will limit my demon to working solely with that to which the fundamental dynamic laws of physics apply. My focus will be on the plausibility of such a more limited demon.

My demon will be a quantum physics demon. Its job will be to take a given wave function of the universe and project it forward. Since quantum physics is deterministic we already know that such a job is possible in principle. Such a projection is not possible in this universe because the amount of computation required exceeds any conceivable computational power the known universe would be able to provide. But we are not worrying about that. We are assuming that the computational power is available. One might imagine this as if one had a quantum computer to which we gave the wave function of the universe and set it computing. And to the best of our knowledge, the universe itself is such a quantum computer, and it is doing that computation all the time.

What does the wave function of the universe look like? It is a collection of hypothetical snapshots of the universe, where each snapshot shows the elements of the universe in one of its possible states. The collection of snapshots contains one snapshot for each possible combination of states. In other words, every possible state of the universe is in that collection.[56]

Each of these snapshots is associated with a probability amplitude. A probability amplitude is a complex number. The probability of the universe being in the configuration described by a snapshot is the snapshot's probability amplitude times its complex conjugate.[57] In short, the wave function of the universe is a collection of all the possible states of the universe with something like a probability associated with each one. If we assume we are starting at a given state, the wave function will have 1 associated with that states' probability and 0 with all the others.

---

[56] Not that it matters for our purposes, but that's a lot of snapshots. If, for example, there are 10^{80} atoms in the universe, and if each atom were limited to only two possible states, we are talking about $2^{(10\text{^}80)}$ snapshots, where 10^80 means 10^{80}. Of course atoms are not elementary, and each elementary piece, whatever it is, will have more than two possible states. The point is, we are talking about lots of snapshots. But as I said, that doesn't really matter for our purposes. It would matter if we were concerned about how much computing power it would take to do this computation. But we aren't.

[57] The complex conjugate of a complex number is the complex number with the same real part and the negation of the imaginary part. The product of a complex number with its complex conjugate is always a real number: $(a+ib)*(a-ib)=a^2+b^2$.

In giving the "wave function of the universe" I am assuming that the entire universe is entangled and that it propagates as a single wave function.[58] Quantum mechanics tells us we can propagate the wave function forward. The problem is that at the next step there will be multiple states with non-zero probability amplitudes. At the step after that, there will most likely be even more states with non-zero probability amplitudes. Etc. At the end of the computation, however far in future we ask the demon to compute, there will almost certainly be many states with non-zero probability amplitudes.

So, even though the demon can compute the probabilities of all the possible configurations of the universe for this final step, when it's done computing we would like it to give us one answer. You might think of this as asking the demon to do the equivalent of collapsing the wave function at the end of the computation. How should the demon perform this collapse operation? That's where I think the interesting problems arise.

One reasonable strategy would be for the demon to take all its snapshots at the end of the computation and group them into clusters of similar snapshots. Then weight each cluster by the probabilities of the snapshots it contains. Finally, as its collapsed answer the demon will select one snapshot at random from the cluster with the highest weight. In saying this, I haven't said either what we want *similar* to mean or how our clustering algorithm is supposed to work. For example, do we want to specify in advance the number of clusters we want to end up with? What is our criterion—if we want to specify it in advance—for how similar two snapshots must be to be included in the same cluster? I don't want to attempt answers to those questions, but for the rest of this section I want to look at what we might mean by *similar*.

To make the discussion easier, I'm going to think of our snapshots in other terms. Instead of thinking of a snapshot as something like a list of all the elements in the universe with a state associated with each, I'd like to think of a snapshot as something more like an image composed of pixels. Imagine the universe as a 3–dimensional array of pixels. Let that be a snapshot. Since the wave function covers the entire universe, it seems just as reasonable to organize it as a three-dimensional array as a list of objects. Each pixel has, as usual, some color bits. It may also have other primitive properties such as charge, mass, etc. depending on where the actual objects are. When transformed in this way it may be easier to think of a reasonable way to assess similarity.

To sharpen the problem a bit further, let's consider how we would assess similarity between two small regions of two snapshots. Can we compare snapshots of a single small region and determine how similar they are?

We currently have software that can do pattern matching on images and compute a measure of how similar two images are. For example, one can give Google an image and ask it to find similar images. The algorithm works by looking for patterns such as large patches of sandy-colored areas (which may indicate a beach) or a great many horizontal and vertical lines (which may indicate buildings). Google's

[58] What I am describing is essentially the many worlds version of quantum mechanics. See Carroll (2011), chapter 11 for an accessible discussion.

algorithm is much more sophisticated than that, but at its heart it's a pattern recognizer. Let's suppose that a similar algorithm could be constructed for our snapshots.

Would that do the job? Suppose we started, say, with Schrödinger's cat and wanted to know whether the cat will be dead or alive after a certain amount of time. The problem, of course, is constructed so that there is a non-negligible probability of either. So we shouldn't be surprised if we don't expect to get a definitive answer. But restating the problem in terms of a cat raises additional problems.

One serious problem is that our pattern recognition algorithm may not be able to distinguish a snapshot of a dead cat from that of a live cat that happened to be sleeping in the same position. (I'm assuming that the dead cat died just recently and hadn't yet started to decompose.) After all, the algorithm is doing relatively gross pattern matching. At that level of comparison a snapshot of a recently deceased cat would be virtually indistinguishable, even at the physiological level, from that of a sleeping live cat. Certainly someone knowledgeable about cats would know where to look and would know how to distinguish one from the other. But our similarity algorithm doesn't know anything about cats. All it has is snapshots, and all it can do is to compare them to each other in relatively simple ways.

This illustrates one of the important properties of higher levels of abstraction, namely that certain differences matter much more than others. Relatively small differences in physiology matter much more than possible large differences in the shape, say, of food bowls. Our similarity algorithm would rate two scenes that are essentially identical except that one had a recently deceased cat and the other a live sleeping cat in the same position as quite similar. Yet it would rate two scenes with two cats sleeping in the same position but with differently sized, shaped, and positioned food bowl as very different. This would happen because the scenes with the different food bowls would have more differences from each other at the pixel level than the scenes with the dead and live cats.

It's easy to construct other examples along these lines. Imagine two scenes that are identical except for the bank balances of two people. When examined at the pixel level the two scenes would appear virtually identical—the physical differences corresponding to the two bank balances are minimal. Yet in reality the differences could be vast.

Could we modify our similarity measuring algorithm to be aware of which differences are important? One approach would be to build into the similarity measuring algorithm knowledge about the higher level structures about which differences matter. This seems like cheating since we are now building into our physics-level propagation system knowledge of higher level structures and functionalities. The original point was to show that all one needs is physics. But let's assume we took that route. How would it work?

First, we would have to provide the similarity algorithm a way to parse a snapshot so that it could associate individual pixels with higher levels of organization. A pixel makes a difference in a structure only if it occurs in an important place in the structure. So the similarity algorithm would have to do far more than gross pattern matching. It would have to be able to do something like a syntactic analysis of the snapshot it is given. What role does each pixel play at each level of the given scene? Recall that we stripped our demon of the ability to do essentially this job. If we re-instated that capability, the parsing task would be taken care of. It's worth noting, though, that this parsing job is not at all trivial.

Once the snapshots are parsed the similarity algorithm would need to determine which differences in the higher level structures might make a difference. In other words, the similarity algorithm would need semantic information about the higher levels. In particular it would have to have information about what might be referred to as the control systems for these higher level structures. Nerve firings and other physiological details are part of the control system for biological entities. Bank balances are data used by the control systems for social entities. If two scenes differed in these areas the algorithm would then need a way to determine whether the differences mattered. The difference between, say, $10,000.00 and $10,000.01 won't make much of a difference. The difference between $110,000.00 and $11,000.00 is likely to make a much greater difference. But if you look just at the symbols, they differ in just one or two symbols. So knowledge about how to interpret the differences in snapshots would have to be built into the system. If we are dealing with people such an analysis would have to extend to the semantic analysis of language—both spoken and written. Perhaps the only difference between two scenes are the words someone is reading or hearing. The similarity algorithm would have to determine whether what one is seeing or hearing are words, and whether those words matter.

In other words, at the semantic level in order to determine which differences make a difference, the similarity checker would have to know the laws of the relevant special sciences, which is exactly what we were trying to do without.

Besides all that, the similarity checker would require constant updating to keep up with technological developments. For example, if the similarity checker were originally written before cell phones, it would have to be updated to recognize and understand cell phones once they become widespread. The same for other technologies and other newly evolved biology—such as bacterial resistance to anti-bacterial treatments, which might make a life-or-death difference.

Now that we've examined it, this more difficult approach has two obvious problems. It would be so difficult to write that it's hardly imaginable that it could be coded. More importantly: even if it could be written—and continually updated—doing so would demonstrate the opposite of what was originally intended. It would show that to make a similarity judgment about two snapshots one would have to take into account all the laws of the special sciences that are relevant to the various differences.

The laws of the special sciences describe real regularities. Consequently I conclude that it would make no sense to attempt to characterize all observed phenomena in terms of the fundamental laws of physics. Any such characterization will require directly or indirectly the reconstruction of the special sciences.

C. Appendix C. Platforms

A platform is a generalization of a level of abstraction. Most platforms enable interaction among users. Facebook is a good example. Facebook users interact by posting items to their Facebook pages and by looking at and commenting on items of the pages of other users. The repertoire of possible user actions (i.e., the abstractions) that Facebook makes available enables these interactions.

A platform often gives rise to what is known as an ecosystem of participants. In the case of Facebook, the ecosystem includes companies that advertise on Facebook, companies that develop games and other apps for Facebook users, and companies that develop tools to help companies that develop Facebook apps. Most platforms are multi-sided, which means that users can be grouped into multiple categories. The three categories just listed are three sides of the Facebook platform. Each group uses Facebook for a different purpose.

Platforms are often subject to what is known as a network effect. The more regular users there are, the more valuable the platform is to each group. It's more valuable to other users because you don't want to be on a social network with no one else. It's more valuable to game and app developers because more users means more potential customers for your games. It's more valuable to advertisers because more users means more eyes on your ad. Similarly the more game and app developers there are, the more valuable the platform is to the regular users because they will have a wider variety of games and apps from which to choose. Because user-group size is so important there is intense competition among platform developers to become the preferred platform in a given area. Success for a platform in terms of a large user base means that the platform can support more niches. For example if there are enough users it is likely that there will be enough chess players among them for it to make sense for a game developer to develop an app that allows users to play chess with each other. In other words, as a platform grows in popularity it is likely to become increasingly rich and diverse.

Multi-sided platforms are not limited to computer systems. Many businesses are also platform based. Among the first cited examples (Rochet and Tirole, 2003) was the credit card service, a platform that facilitates a certain form of interaction among buyers and sellers. It serves buyers by making it possible make purchases without carrying money or establishing credit-worthiness. It serves sellers by making it easier to make sales to unknown customers and then to collect payment.

Other multi-sided platforms include shopping malls (customers and stores), markets (such as farmer's markets and stock markets) where buyers and sellers meet to transact business, communication systems like the telephone and the postal systems (there is a single group of users who want to communicate among themselves), and advertiser-supported media like television (the sides are viewers, show producers, and advertisers). For more on multi-sided platforms see (Evans, Hagiu, and Schmalensee, 2006), (Evans, 2013), and (Filistrucchi, Geradin, and van Damme, (2012).

Multi-sided platforms offer a model for how natural ecosystems can develop. As a platform develops it becomes richer and more diverse. But it must start with a sufficiently broad and rich collection of abstractions to attract an initial set of users. That initial user base must be large enough to produce a cycle of self-reinforcing growth and development.

D. Appendix D: The first two definitions of causality in Ellis (2012)

The first *Interface Focus* definition of causation seems to correspond most closely to Dowe's (2000) what-actually-happens version of causality. I found some confusing points, though. Here is the definition.

> **[causation, definition 1 from (O'Connor 2012).]** *Causation as production:* A productive cause of an effect is the set of factors that produce, bring about or make happen the effect. It is controversial whether causation in this basic, metaphysical sense ever obtains outside the realm of fundamental physics. (Causal closure-denying emergentism maintains that it does so obtain.) And on some philosophical views, there are no productive causes in this sense, only patterns of counterfactual dependence among types of events.

It's not clear to me how to read this definition. The first sentence sounds like Interventionist causation since the Interventionist approach to causation is intended to identify factors that if triggered would bring the effect about. I originally suspected that the term *productive* has a technical sense in the causality literature and that its inclusion is intended to limit the bringing about to something that is directly physical—and thereby to exclude the Interventionist interpretation. But that doesn't seem to be the case. For one things, neither Dowe (2000) nor Woodward (2003) includes either *productive* or *production* in its index.

Secondly, Hall (2004) seems to have coined this usage when he wrote that in contrast to dependent causation productive causation refers to the generation, bringing about, or production of an event. But Hall does not limit a productive cause to one that is necessarily either direct and physical. A productive cause need not be simultaneous with its effect and may be productive through transitivity. In fact, the focus of Hall (2004) is on situations involving omission, prevention, failed prevention (e.g., Billy tripped and was unable to prevent Suzy from throwing a stone at the window), double prevention (e.g., Billy prevents Valentino from sweet talking Suzy and thereby distracting her from throwing a stone at the window), over-determination (e.g., Billy and Suzy both throw stones at the window), and related causal complications. It is not a search for anything like a physical causal primitive.[59]

Yet Bobro (2013) says that for Leibnitz *efficient cause* and *productive cause* are synonyms. And according to *The Free Dictionary,*[60] "*efficient cause* is frequently used interchangeably with proximate cause—the immediate act in the production of a particular effect."[61]

Based on the second sentence in the definition, and since the second *Interface Focus* definition seems to refer to Interventionist causation, my best guess is that this first definition is intended to refer to causation in the what-actually-happens sense, i.e., Dowe-style causation. A further source of confusion, though, is that Dowe, the primary expert on what-actually-happens causation, does not limit that form

---

[59] For what it's worth, my sense is that for Hall, given a set of entities and a set of possible events, a cause is productive if there is no possible configuration of events that includes the effect but not the cause. In effect, this says that a cause is productive for an effect if the effect does not occur without it. Pearl (2000, p 310) says essentially the same thing. "[Hall's] notions of dependence and production closely parallel those of necessity and sufficient respectively." Pearl goes on to provide a formalization of *productive* in this sense. The formalization does not attempt to define a *productive cause* as one that's physically primitive.

[60] http://legal-dictionary.thefreedictionary.com/Efficient+Cause

[61] But the waters are muddied for me by Cohen (2012), who says that Aristotle uses *efficient* and *productive* as synonymous but not necessarily to mean directly physical. An *efficient* cause, is "the primary source of change or rest" (194b30). According to Aristotle, an adviser is the cause of an action, a father is the cause of his child, and in general the producer is the cause of the product. Also, according to Falcon (2012), for Aristotle both the artisan and the art of bronze-casting are efficient causes for a statue.

of causation to fundamental physics. In response to a complaint by Craver that conserved-quantity causation seems to drive all causation down to fundamental physics Dowe notes (2012, p 873) that a steel ball possesses the conserved quantities mass and charge as much as an electron does.

The second *Interface Focus* definition of causation refers straightforwardly to Interventionist-style causation.

> **[causation, definition 2 from (O'Connor 2012).]** *Causation as counterfactual dependence:* B counterfactually depends on A just in case B would not have occurred had A not occurred. Patterns of covariant dependence clearly obtain at many levels of nature, not just in physics. Plausibly, it is this minimalist notion of causation that guides methods for drawing causal conclusions from statistical data.

E. Appendix E. Software developers build on firmer foundations than do engineers

Much of what I said about creating and implementing abstractions applies to engineers as well as to software developers. Engineers, too, create products that embody new abstractions and functionalities. And they do it by putting together components that offer existing abstractions. "Smart" phones provide many examples. A nice one is the smart phone's ability to re-orient its display (portrait or landscape) depending on how the phone is held. How do engineers get phones to do that? They use a pre-existing device, the small-scale accelerometer, which had been developed for and has a wide range of uses in industrial and commercial products. One of its more familiar uses is in game controllers such as the Nintendo Wii. An accelerometer in a game controller can tell the software in the controller about the user's motion. When the user swings her arm as if holding a tennis racket, the controller can display a picture of someone swinging a racket in a way that parallels the user's movements. In a smart phone, the accelerometer is used much more simply—to let the software in the phone know how the phone was moved and how it is oriented. The point is that a device whose functionality had been used for one purpose is now used for another.

Software developers have an important advantage over engineers.[62] We need not be concerned about the reliability of our primitives. The lowest level software element is the bit. It is an atomic ideal. It cannot be decomposed into smaller elements, and it cannot wear out. As far as software developers are concerned, bits are forever; you can't break a bit. Engineers have no such solid foundation. When building something they must always be concerned about whether the components are appropriate for the application of the device. This is not a trivial issue; there is no *a priori* limit to the depth of analysis required to provide the assurance that the components will work when the device is put to use. For example, there are many different types of accelerometers. Some are quite rugged; others fail when not in a laboratory environment. In addition, there is trade-off between accelerometer size and sensitivity.

[62] Confusingly software developers are often referred to as software engineers.

Larger accelerometers are better at sensing smaller movements, but heavier mobile phones don't sell as well. See (Lally, 2005). Issues of these sorts require engineers always to be looking down toward the physical foundations of the devices they use as well as up toward the new abstractions they are building. Only software developers (and God[63]) can be secure in the certainty of their knowledge of the fundamental elements of their universes. (See (Abbott, 2010a) for additional discussion.)

[63] This analogy was suggested by Debora Shuger, private correspondence. She cited Barkan (2013, p 13) for a discussion of Plato's transmutation of the "public craftsman," i.e., engineer, into an equivalent for God.