

[Extended Abstract]

Constructive emergence: a computer scientist looks at philosophy

Russ Abbott

California State University, Los Angeles
Los Angeles, CA 90032 USA

russ.abbott@gmail.com

Keywords

Computer science, philosophy, reduction, emergence, supervenience, conceptual grounding, thought externalization.

1. WHY DO PHILOSOPHERS BUT NOT COMPUTER SCIENTISTS FIND EMERGENCE MYSTERIOUS?

My paper last year [1] compared styles of thought in computer science and engineering. This year I compare computer science with philosophy. (As you might guess, I'm a computer scientist. I acknowledge from the start that this paper is a bit "in-your-face.") Since philosophy is such a broad discipline, I'm limiting myself to emergence, a topic about which I've recently written—for example, [2] and [3].

Emergence is the notion that higher level phenomena can be autonomous from the underlying phenomena on which they are based. I claim that a computer science approach to emergence resolves the fundamental issues. Yet philosophers still see emergence as mysterious. As recently as April 2008 the introduction to a collection of articles about emergence edited by Bedau and Humphreys [4] asserted that "the very idea of emergence seems opaque, and perhaps even incoherent."

In computer science, emergence is neither opaque nor incoherent. It is a fundamental software design principle—the intuition behind such ideas as software platforms, levels of abstraction, abstract data types, object oriented programming, the distinction between a specification and an implementation, APIs, etc.

The notion that higher level phenomena may be autonomous of a lower level base is also widely expressed in the philosophical literature. For more than three decades, functionalist philosophers such as Fodor [5] have argued for the autonomy of the special sciences—any science other than physics.

The very *existence* of the special sciences testifies to reliable macro-level regularities ... 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. ...

But Fodor continues (somewhat paraphrased):

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.

What I find mysterious is not the fact of emergence but Fodor's bafflement about it. But Fodor insists he doesn't know why.

So, then, *why is there anything except physics?* ... I expect to figure out why ... the day before I figure out why there is anything at all.

It's not just Fodor, Bedau, and Humphreys who are puzzled by emergence. It has provoked philosophical debate for years.

For computer science, emergence is our bread and butter. Every software application is an implementation of emergent phenomena. Microsoft Word, for example, implements such emergent abstractions as paragraphs, words, fonts, pages, documents, etc. These concepts and the rules that govern them are autonomous from the rules that govern the underlying levels. Depending on one's focus, the underlying level involves logic gates, or machine instructions, or programming language constructs, etc. None of these has anything to do with documents, paragraphs, words, characters, or fonts. Yet there is no mystery about how these autonomous higher level abstractions come to be. Computer science is largely about turning operations performed by logic gates into emergent Microsoft Word abstractions.

A useful example of emergence is a Turing machine implemented within the Game of Life. (See [2].) Turing machines compute functions, and Turing machines are bound by computability theory. But functions and computability theory were developed long before Conway invented the Game of Life. It seems reasonable therefore to say that they are autonomous of the Game of Life. Yet nothing happens on a Game of Life grid other than that cells go on and off as a consequence of the Game of Life rules. So a Game of Life Turing machine would seem to be an easily understandable and paradigmatic example of emergence: autonomous higher level phenomena resulting from lower level activities. (This also seems to me to illustrate why multiple realizability is not relevant: autonomy exists without it.) But Fodor [5] dismisses emergence of this sort.

2. DIFFERENCES BETWEEN PHILOSOPHY AND COMPUTER SCIENCE

It is not news to philosophers that Turing machines can be implemented within the Game of Life. Dennett [6] noted this in a widely cited paper more than a decade and a half ago. Yet emergence is still seen as a philosophical puzzle. In attempting to understand why, I've examined some of the relevant philosophical literature. I've noticed two important differences between computer science and philosophy.

One is conceptual and terminological. A number of concepts are used similarly but not identically in philosophy and computer science. Pairings include: emergence/abstraction, reduction/implementation, autonomy/specification-implementation, kind/type, causality/execution, individual/object, and supervenience/functional dependency.

A second difference concerns the degree to which philosophers and computer scientists feel themselves obliged to ground their

thinking. Computer science is grounded by the requirement that ideas must (usually) execute as software. There is rarely a question about what a term means: at worst it means what the software that implements it does. Computer science is a constructive discipline. We build new concepts on top of existing concepts, which themselves are grounded by executable software. As a result, computer science discussions are almost always framed in terms of well defined and well understood concepts.

To my computer science eyes, many philosophical discussions don't seem to be similarly grounded. They often seem to be about theories expressed using terms that themselves are not well understood. (That's my guess about why philosophers so often claim to find mistakes in each others' papers: "Y is wrong about Z" or "X is wrong about Y being wrong about Z.")

This is not to say that philosophers aren't careful. Philosophers have been so careful that supervenience, for example, has probably a dozen definitions. The related computer science concept is functional dependency: a set of database attributes is functionally dependent on another set if the latter determine the former. The issue for computer scientists is not what functional dependency means (or should mean) but when to declare a functional dependency. A database that includes attributes for both molecular descriptions and appraised value would simply not declare the latter functionally dependent on the former.

Another example of a theory built (in my view) on uncertain foundations is the near universal agreement that "the mental supervenes on the physical." Not only does supervenience seem not to be solidly defined, but (it seems to me that) we don't understand the properties of either the mental or the physical well enough to talk about whether a functional dependency exists between their properties. Presumably the claim isn't really about supervenience but something to the effect that dualism is wrong. Supervenience may not even be the best way to establish that since all it guarantees is co-variation.

The notion of scientific reduction also illustrates the differences between the two disciplines. Reduction has an enormous and growing philosophical literature. As far as I can tell there is no philosophical consensus about how to define it. In contrast, the computer science notion of implementation, which is something like our version of reduction, is well defined. This is not to say that computer science's implementation resolves all the philosophical issues associated with reduction. In fact the philosophical literature distinguishes between what is called realization—which is also like our implementation—and reduction. Functionalists claim that realizations may exist when (or when multiple realizations exist as evidence that) reduction is not possible. For people like Fodor [7] the difference seems to be that reduction requires realization along with a simple mapping from natural low level kinds to natural high level kinds. (The philosophical notion of a natural kind is also quite unsettled. In Computer Science we call kinds "types," a fairly well defined concept. We don't talk about natural types.)

A current movement in philosophical reduction shows an interest in models and mechanisms, for example [8] and [9]. Mechanisms and models are quite compatible with the computer science understanding of reduction as implementation. (See [2] and [10].)

3. SUMMARY

When computer scientists talk about functional dependency, implementation, and types, we know what we mean. When

philosophers talk about supervenience, reduction, and kinds, things seem far less settled. Perhaps it isn't so surprising after all that a concept like emergence is well understood in computer science but problematic in philosophy. (My other paper in this workshop offers a concrete example of this contrast.)

Much of computer science is about developing tools to express (i.e., externalize) one's ideas in a concrete operational form. (See [3].) Much of philosophy is about finding and expressing ideas that seem right. Neither discipline can capture conscious meaning: ideas once expressed are just symbols. But the discipline of having to operationalize those symbols as software often clarifies the ideas and helps one decide whether a particular way of expressing them is really what one wants to say.

4. REFERENCES

- [1] Abbott, Russ (2007) "Bits don't have error bars," *Workshop on Engineering and Philosophy*, October 2007. To be included in the selected papers from the conference. http://cs.calstatela.edu/wiki/images/1/1c/Bits_don't_have_error_bars.doc
- [2] Abbott, Russ (2006) "Emergence explained," *Complexity*, Sep/Oct, 2006, (12, 1) 13-26. Preprint: http://cs.calstatela.edu/wiki/images/9/95/Emergence_Explained-Abstractions.pdf
- [3] Abbott, Russ (2008) "If a tree casts a shadow is it telling the time?" *International Journal of Unconventional Computation.*, (4, 3), 195-222. Preprint: http://cs.calstatela.edu/wiki/images/6/66/If_a_tree_casts_a_shadow_is_it_telling_the_time.pdf.
- [4] Bedau, Mark and Paul Humphreys (2008) *Emergence*, MIT Press. Introduction available: <http://mitpress.mit.edu/catalog/item/default.asp?tttype=2&tid=11341>.
- [5] Fodor, Jerry A. (1997) "Special Sciences; Still Autonomous after All These Years," *Philosophical Perspectives*, 11, *Mind, Causation, and World*, pp 149-163.
- [6] Dennett, Daniel C. (1991) "Real Patterns," *The Journal of Philosophy*, (88, 1), pp 27-51.
- [7] Fodor, Jerry (1974): "Special sciences and the disunity of science as a working hypothesis", *Synthese*, 28, pp 77-115.
- [8] Bechtel, W. (2007). "Reducing psychology while maintaining its autonomy via mechanistic explanation." In M. Schouten and H. Looren de Jong (Eds.). *The Matter of the Mind*. Oxford: Basil Blackwell. <http://mechanism.ucsd.edu/~bill/research/reducingpsychologywhilemaintainingautonomy.withfigures.pdf>.
- [9] Howard, Don, (2007) "Reduction and Emergence in the Physical Sciences." In *Evolution and Emergence: Systems, Organisms, Persons*. Nancy Murphy and William R. Stoeger, S.J., eds. Oxford University Press. <http://www.nd.edu/~7Edhoward1/Reduction%20and%20Emergence.pdf>
- [10] Abbott, Russ (2008) "The reductionist blind spot," *North American Computers and Philosophy Conference*. http://cs.calstatela.edu/wiki/images/c/ce/The_reductionist_blind_spot.pdf

Internet accesses are as of August 31, 2008.