The reductionist blind spot

Russ Abbott

Department of Computer Science, California State University, Los Angeles, California
Russ.Abbott@GMail.com

Abstract. Can there be higher level laws of nature even though everything is reducible to the fundamental laws of physics? The computer science notion of level of abstraction explains how there can be.

1. Introduction

When a male Emperor penguin stands for two frigid months balancing an egg on its feet to keep it from freezing, are we to understand that behavior in terms of quarks and other fundamental particles? It seems unreasonable, but that's the reductionist position. Here's how Albert Einstein put it. [5]

> 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]

The italicized portion expresses what Philip Anderson [4] calls the *constructionist hypothesis*: the idea that one can start with physics and reconstruct the universe.

More recently Steven Weinberg [10] restated Einstein's position as follows.

> Grand reductionism is … the view that all of nature is the way it is … because of simple universal laws, to which all other scientific laws may in some sense be reduced. …

> [For example,] the reductionist regards the general theories governing air and water and radiation as being at a deeper level than theories about cold fronts or thunderstorms … [T]he latter can in principle be understood as mathematical consequences of the former. … [T]here are no autonomous laws of weather that are logically independent of the principles of physics. …
>
> 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 … there are no principles of psychology that are free-standing, 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.

Not all physicists agree with Einstein and Weinberg. As Erwin Schrödinger [8] wrote,

> [L]iving 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.

In arguing against the constructionist hypothesis Anderson extended Schrödinger's thought.

> [T]he ability to reduce everything to simple fundamental laws … [does not imply] the ability to start from those laws and reconstruct the universe. …
>
> At each level of complexity entirely new properties appear. … [O]ne may array the sciences roughly linearly in [a] hierarchy [in which] the elementary entities of [the science at level n+1] obey the laws of [the science at level n]: elementary particle physics, solid state (or many body) physics, chemistry, molecular biology, cell biology, …, psychology, social sciences. But this hierarchy does not imply that science [n+1] is 'just applied [science n].' At each [level] entirely new laws, concepts, and generalization are necessary.

Notwithstanding their disagreements, all four physicists (and many others) agree that everything can be reduced to the fundamental laws of physics. Here's how Anderson put it.

> [The] workings of all the animate and inanimate matter of which we have any detailed knowledge are … controlled by the … fundamental laws [of physics]. … [W]e must all start with reductionism, which I fully accept.

Einstein and Weinberg argue that that's the end of the story. Starting with the laws of physics and with sufficiently powerful deductive machinery one should be able to reconstruct the universe. Schrödinger and Anderson disagree. They say that there's more to nature than the laws of physics—but they were unable to say what that might be.

Before going on, you may want to answer the question for yourself. Do you agree with Einstein and Weinberg or with Schrödinger and Anderson? Is there more than physics—and if so, what is it?

2. *Preview*

The title of this paper gives away my position. I agree with Schrödinger and Anderson.

The computer science notion of level of abstraction explains how there can be higher level laws of nature—even though everything is reducible to the fundamental laws of physics. The basic idea is that a level of abstraction has both a specification and an implementation. The implementation is a reduction of the specification to lower level functionality. But the specification is independent of the implementation. So even though a level of abstraction depends on lower level phenomena for its realization it cannot be reduced to that implementation without losing something important, namely its specification.

This demonstrates, as Wing [11] pointed out, that conceptual tools developed by computer scientists can apply more broadly.

3. *Levels of abstraction*

A level of abstraction (see [6]) is (a) a collection of types—which for the most part means categories—and (b) operations that may be applied to entities of those types. A standard example is the stack, which is defined by the following operations.

```
push(stack: s, element: e)    — Push an element e into a stack s and return the stack.
pop(stack: s)                 — Pop the top element off the stack s and return the stack.
top(stack: s)                 — Return (but don't pop) the top element of a stack s.
```

Although the intuitive descriptions are important for us as readers, all we have done so far is to declare a number of operations. How are their meanings defined? Axiomatically.

```
top(push(stack: s, element: e)) = e.
                              — The top element of s after e id pushed onto it is e.
pop(push(stack: s, element: e) = s.
                              — After pushing e onto s and then popping it off, s remains.
```

Together, these declarations and axioms define a stack as anything to which the operations can be applied while satisfying the axioms.

This is similar to how mathematics is axiomatized. Consider the non-negative integers as specified by Peano's axioms.[1]

1. *Zero* is a *number*.
2. If *A* is a *number*, the *successor* of *A* is a *number*.
3. *Zero* is not the *successor* of a *number*.
4. Two *numbers* of which the *successors* are equal are themselves equal.

[1] As given in Wolfram's MathWorld: http://mathworld.wolfram.com/PeanosAxioms.html.

5. (Induction axiom) If a set S of *numbers* contains *zero* and also the *successor* of every *number* in S, then every *number* is in S.

These axioms specify the terms *zero, number,* and *successor*. Here *number* is a type, *Zero* is an entity of that type, and *successor* is an operation on *numbers*. These terms stand on their own and mean (formally) no more or less than the definitions say they mean.

Notice that in neither of these definitions were the new terms defined in terms of pre-existing terms. Neither a *number* nor a *stack* is defined as a special kind of something else. Both Peano's axioms and the stack definition define terms by establishing relationships among them. The terms themselves, *stack* and a *number*, are defined *ab initio* and solely in terms of operations and relationships among those operations.

This is characteristic of levels of abstraction. When specifying a level of abstraction the types, objects, operations, and relationships at that level stand on their own. They are not defined in terms of lower level types, objects, operations, and relationships.

4. Unsolvability and the Game of Life

The Game of Life is a 2-dimensional cellular automaton in which cells are either alive (on) or dead (off). Cells turn on or off synchronously in discrete time steps according to the following rules.

- Any cell with exactly three live neighbors will stay alive or become alive.
- Any live cell with exactly two live neighbors will stay alive.
- All other cells die.

In a Game of Life universe the preceding rules are analogous to the fundamental laws of physics. They determine everything that happens on a Game of Life grid.

Certain on-off cell configurations create patterns—or really sequences of patterns. The glider is the best known. When a glider is entered onto an empty grid and the rules applied, a series of patterns propagates across the grid. Since nothing actually moves in the Game of Life—the concept of motion doesn't even exist—how should we understand this?

Gliders exist on a different level of abstraction from that of the Game of Life. At the Game of Life level there is nothing but grid cells—in fixed positions. But at the glider level not only do gliders move, one can even write equations for the number of time steps it will take a glider to move from one location to another and when a glider will "turn on" a particular cell. What is the status of such equations?

Before answering that question, recall that it's possible to implement Turing machines by arranging gliders and other Game of Life patterns. Just as gliders are subject to the laws of glider equations, Turing machines too are subject to their own laws—in particular, computability theory.

Game of Life gliders and Turing machines exemplify the situation described by Schrödinger and Anderson. While not eluding the Game of Life rules, in both cases autonomous new laws apply to phenomena that (a) appear on a Game of Life grid but (b) are defined on a different and independent level of abstraction. These additional laws are not expressible in Game of Life terms. There is no such thing as a glider or a Turing machine at the Game of Life level. The Game of Life is nothing but a grid of cells along with rules that determine when cells go on and off.

But higher level laws are not disconnected from Game of Life elements. Because the halting problem is unsolvable, it is unsolvable whether an arbitrary Game of Life configuration will ever reach a stable state. Similarly, glider equations let us predict when a glider will "turn on" a particular cell.

When autonomous higher level laws apparently effect lower level phenomena the result has been called [3] downward causation. But downward causation doesn't make scientific sense. It is always the lower level phenomena that determine the higher level: the Game of Life rules are the only things that determine whether cells go on or off. So how can computability theory and glider equations let us draw conclusions about whether cells will go on or off? In [1] I call this *downward entailment*. Autonomous laws that apply at a higher level of abstractions can have implications for elements at a lower level when the lower level implements the higher level.

5. Evolution is also a property of a level of abstraction

The Game of Life is an abstraction. Evolution, a real-world phenomenon, is also a property of a level of abstraction.

Evolution occurs in the context of a *population* of *entities*. The entities exist in an *environment* within which they may *survive* and *reproduce*. The entities have *properties* that affect how they *interact* with their environment. Those interactions help determine *whether* the entities will survive and reproduce. When an entity *reproduces*, it *produces offspring* which *inherit* its properties, possibly along with some *random variations,* which may *result in new properties*. In some cases, pairs of entities *reproduce jointly*, in which case the offspring *inherit* some *combination* of their parent's properties—also with the possibility of *random variations*.

The more likely an entity is to survive and reproduce, the more likely it is that the properties that enabled it to survive and reproduce will be passed on to its offspring. If certain properties—or random variations of those properties, or the random creation of new properties—enable their possessors to survive and reproduce more effectively, those properties will propagate. We call the generation and propagation of successful properties *evolution*. By helping to determine which entities are more likely to survive and reproduce, the environment effectively selects

the properties to be propagated—hence evolution by environmental (i.e., natural) selection.

The preceding description introduced a number of terms (in italics). If elaborated, the terms could be defined more formally. As in the case of stacks and Peano numbers, the new terms are defined *ab initio* at the evolution level of abstraction.

The great contribution of Darwin and Wallace was to describe the evolution level of abstraction. They did so even though they knew nothing about DNA. But since their model required some mechanism for recording and transmitting properties, a prediction of their model is that any implementation of the evolution level of abstraction must provide such a mechanism. We now know that for biological organisms DNA is that mechanism. Prediction confirmed—at least in this case.

6. The reductionist blind spot

Physics recognizes four fundamental forces. Evolution is not one of them. Similarly there is no "computational functionality" in a Game of Life universe. Neither evolution nor computation have any causal power; they are both epiphenomenal. Do we need them? In some sense we don't.

Game of Life Turing machines don't *do* anything. It is only the Game of Life rules that make cells go on and off. Reductionism has not been overthrown. One could trace the sequence of Game of Life rule applications that transform an initial Game of Life configuration (which could be described as a Turing machine with input *x*) into a final configuration (which could be described as a Turing machine with output *y*). One could do this with no mention of Turing machines, tapes, symbols, etc.

Similarly one could presumably—albeit with great difficulty—trace the sequence of chemical and physical reactions and interactions that produce a particular chemical configuration (which could be described as the DNA that enables its possessor to thrive in its environment). One could do this with no mention of genes, codons, proteins, etc.

One can always reduce away macro-level terminology and phenomena and replace them with the underlying micro-level terminology and phenomena. It is still the elementary mechanisms—and nothing but those mechanisms—that turn the causal crank. So why not reduce away epiphenomenal levels of abstraction?

Reducing away a level of abstraction produces a reductionist blind spot. Computations performed by Game of Life Turing machines cannot be described as computations when one is limited to the vocabulary available at the Game of Life level of abstraction. Nor can one explain why the Game of Life halting problem is unsolvable. These concepts exist only at the Turing machine level of abstraction. Similarly, biological evolution cannot be explicated at the physics and chemistry

level of abstraction. The evolutionary process exists only at the evolution level of abstraction.

Furthermore, reducing away a level of abstraction throws away elements of nature that have objective existence. At each level of abstraction there are entities, such as Turing machines and biological organisms, that instantiate types at that level. These entities are simultaneously causally reducible and ontologically real—a phrase coined by Searle [8] in another context. Entities on a level of abstraction that are implemented by a lower level of abstraction are causally reducible because the implementation provides the forces and mechanisms that drive them. But such entities are ontologically real because their specifications characterize what they do.

Weinberg argues that higher level entities like cold fronts and thunderstorms are just conceptual conveniences. Are they? Robert Laughlin [7] argues that higher level entities are objectively real. He talks about what he calls "protectorates," including both the solid state of matter and Newtonian physics. Like computation and evolution, protectorates exhibit properties that simply don't have a meaning at lower levels. For example, the solid state of matter includes concepts—such as hardness, shear strength, torque, tensile strength, load bearing ability, etc.—that are meaningless at the level of elementary particle physics. Newtonian mechanics similarly has properties of its own. Laughlin emphasizes that protectorates are not approximations of lower level phenomena—Newtonian physics is not an approximation of quantum mechanics. They are new conceptual domains that obey new laws.

Higher level entities are objectively real in two additional ways. They have reduced entropy, and they have either more mass or less mass than their components considered separately. In [1] and [2] I discuss the two types of higher level entities: static entities and dynamic entities. Static entities persist in a reduced entropy state because they are in an energy well. Atoms, molecules, solar systems, etc. are examples. Because they are in an energy well, static entities have less mass than the aggregation of their components considered separately.

Dynamic entities persist in a reduced entropy state by extracting energy from the environment. Biological and social entities are examples. (Hurricanes are the only naturally occurring non-biological and non-social dynamic entity of which I am aware.) Because dynamic entities have energy flowing through them, they have more mass than the aggregation of their components considered separately.

These considerations convince me that one is justified in considering these higher level entities as objectively real and more than just conceptual conveniences.

The goal of science is to understand nature. Reducing away levels of abstraction discards both real scientific explanations—such as the evolutionary mechanism—and objectively real entities—such as biological organisms. Reducing away levels of abstraction is bad science.

7. Constructionism and the principle of ontological emergence

Levels of abstraction illustrate Schrödinger's perception that although higher level phenomena don't elude the laws of physics they are governed by new laws. Because higher level laws are not derived from the laws governing the implementing level, knowledge of the lower level laws does not enable one to generate a specification and implementation of the higher level. In other words, constructionism fails. No matter how much deductive power one has available, one would not expect to be able to generate, for example, the specification and implementation of a web browser given the specification of logic gates—or the specification and implementation of biological organisms given fundamental physics. Besides, why a web browser rather than, say, Linux? Why not both, or all three?

But in another sense constructionism succeeds. It has taken billions of years, but nature *has* implemented biological organisms—and web browsers and Linux. And it did so starting from quantum mechanics. If one considers nature a mechanism that generates and implements possible levels of abstraction, then nature embodies constructionism—although as the random enumeration of possibilities and not in the deductive/explanatory sense suggested by Einstein and Weinberg.

Nature generates multiple interrelated levels of abstractions. Which levels of abstraction persist? It depends on the environment. Molecules persist only in environments with low enough temperatures; biological organisms persist only in environments that provide nourishment; and hurricanes persist only in environments with a supply of warm water. This can be summarized as the principle of ontological emergence: extant levels of abstraction are those whose implementations have materialized and whose environments enable—or at least do not prevent—their persistence.

8. Summary

The need to understand and describe complex systems led computer scientists to develop concepts that clarify issues beyond computer science. In particular, the notion of the level of abstraction explains how higher level laws of nature help govern a reductionist universe.

Note. Two "sidebars" follow the references.

References

[1] Abbott, Russ, "If a tree casts a shadow is it telling the time?" *International Journal of Unconventional Computation*., (4, 3), 195-222, 2008. Preprint: http://cs.calstatela.edu/wiki/images/6/66/If_a_tree_casts_a_shadow_is_it_telling_the_time.pdf.

[2] Abbott, Russ, "Emergence explained," *Complexity*, Sep/Oct, 2006, (12, 1) 13-26. Preprint: http://cs.calstatela.edu/wiki/images/9/95/Emergence_Explained-_Abstractions.pdf.

[3] Andersen,Peter Bøgh, Claus Emmeche, Niels Ole Finnemann and Peder Voetmann Christiansen, eds. (2000): *Downward Causation. Minds, Bodies and Matter*. Århus: Aarhus University Press.
[4] Anderson, Philip W., "More Is Different," *Science,* 4 Aug. 1972, (177, 4047), 393-396. 1972
[5] Einstein, Albert, "Principles of Research," (address) Physical Society, Berlin, 1918, reprinted in *Ideas and Opinions*, Crown, 1954. http://www.cs.ucla.edu/~slu/on_research/einstein_essay2.html.
[6] Guttag, John, "Abstract data types and the development of data structures," *Communications of the ACM*, (20, 6) 396-404, June 1977. http://rockfish.cs.unc.edu/204/guttagADT77.pdf.
[7] Laughlin, Robert B., *A Different Universe: Reinventing Physics from the Bottom Down*, Basic Books, 2005.
[8] Schrödinger, Erwin, *What is Life?*, Cambridge University Press, 1944. http://home.att.net/~p.caimi/Life.doc.
[9] Searle, John, *Mind: a brief introduction*, Oxford University Press, 2004.
[10] Weinberg, Steven., "Reductionism Redux," *The New York Review of Books*, October 5, 1995. Reprinted in Weinberg, Steven., *Facing Up*, Harvard University Press, 2001. http://www.idt.mdh.se/kurser/ct3340/ht02/Reductionism_Redux.pdf.
[11] Wing, Jennifer, "Computational Thinking," *Communications of the ACM*, (49, 3), 33-35, March 2000. http://www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf.

All Internet accesses are as of April 3, 2008.

(Sidebar) *Mathematics, science, and engineering (including computer science)*

The notion of the level of abstraction suggests characterization of the similarities and differences among mathematics, science, engineering, and computer science.

Mathematics is the study of the entities and operations defined on various levels of abstraction—whether or not those levels of abstraction are implemented. Mathematicians devise formal (or at least "rigorous") specifications of levels of abstraction—such as Peano's non-negative integers. They then study the consequences of those specifications—which in the case of Peano's axioms is number theory.

Science is (a) the characterization of observed natural phenomena as levels of abstraction, i.e., the framing of observed phenomena as patterns, followed by (b) a determination of how those levels of abstraction are implemented by lower level mechanisms.

Engineering (including computer science) is the imagination and implementation of new levels of abstraction. The levels of abstraction that engineers and computer scientists implement are almost always defined informally—most real-world systems are too complex to specify formally. They are characterized as *requirements*, natural language descriptions of required functional and performance

properties. Engineers and computer scientists implement systems that meet requirements.

Whereas engineers and computer scientists imagine and implement new levels of abstraction, scientists identify existing levels of abstraction and discover the mechanisms nature uses to implement them. In other words, science is the reverse engineering of nature.

Why did computer science rather than engineering develop the notion of level of abstraction? Computer scientists start from a well defined base level of abstraction—the bit and the logical operations defined on it—and build new levels of abstraction upwards from that base. Engineers work with physical objects implemented at multiple and arbitrary levels of abstraction. Since there is no engineering base level of abstraction, engineers construct mathematical models that approximate nature downwards as far as necessary to ensure that the systems they build have reliable physical foundations. (See [1].)

Reference

[1] Abbott, Russ, "Bits don't have error bars," Workshop on Engineering and Philosophy, October 2007.
http://cs.calstatela.edu/wiki/images/1/1c/Bits_don't_have_error_bars.doc.

(Sidebar) *Constraints, software, broken symmetry, and conceptual models*

The definitions of the stack and Peano numbers are given in terms of types, operations, and constraints. To implement a level of abstraction is to impose constraints on the implementation level—similar to the constraints imposed on a computer by a software application. By guiding a computer's operation, the application constrains the range of states and state trajectories the computer may assume—a constructive form of symmetry breaking. These constraints—i.e., the application's conceptual model—determine the allowable states, their relationships, and the transitions among relationship, i.e., the higher level laws, that may occur within the conceptual model.