Algorithmic Scientific Inference
Within Our Computable Expected Reality

John Case

Computer and Information Sciences Department
University of Delaware
Newark, DE 19716 USA

Abstract
It is argued that, scientific laws, including quantum mechanical ones, can be considered algorithmic, that the expected behavior of the world, if not its exact behavior, is algorithmic, that, then, communities of human scientists over time have algorithmic expected behavior.

Some sample theorems about the boundaries of algorithmic scientific inference are then presented and interpreted. There is some discussion about (but there are not presentations of) succinct machine self-reference proofs of these theorems and whether non-artifactual self-referential examples may exist in the world.

There is also a brief discussion regarding the possibility that the expected behavior of reality may be infeasibly computable.

Keywords: machine inductive inference, physics, philosophy of science

Contents

1 Scientific Laws 2
1.1 Modeling Scientific Laws 2
1.2 A Quantum Mechanical Example 3

2 Data Types 3

Email address: case@cis.udel.edu (John Case)

Preprint submitted to Elsevier February 13, 2011
1. Scientific Laws

Below we describe in Section 1.1 how and why we model scientific laws in terms of algorithms, and, in Section 1.2, we provide important clarification with an example from quantum mechanics.

1.1. Modeling Scientific Laws

In the 1970s, I was motivated to work on the Theory of Machine Inductive Inference, Putnam and Gold [Put63, Gol67, Put75], thanks to the Blums’ assertion [BB75, Page 125] just below.

Consider the physicist who looks for a law to explain a growing body of physical data. His data consist of a set of pairs \((x, y)\), where \(x\) describes a particular experiment, e.g., a high-energy physics experiment, and \(y\) describes the results obtained, e.g., the particles produced and their respective properties. The law he seeks is essentially an algorithm for computing the function \(f(x) = y\).

Such an algorithm is a predictive explanation, Case & Smith [CS78, CS83]: if one has the good fortune to have such an algorithm, one can use it to predict the outcomes of the associated experiments.

Importantly, a predictive explanation must provide its predictions algorithmically! How else are we to get out the predictions — by magic? To be
sure, in, say, physics, the laws are typically not written down including how to extract algorithmically the predictions. That is implicit and may, in some cases, be difficult. The techniques are essentially covered by computably axiomatizable mathematics, algorithmic numerical techniques, etc. Of course physicists rarely resort to axiom systems directly, but, when mathematics is formulated axiomatically, one always sees a computably decidable set of axioms. How else could formal proofs be checked, e.g., when they cite an axiom, — by magic? Of course with a formal system having a computably decidable set of axioms, the set of corresponding theorems forms a computably enumerable set.

1.2. A Quantum Mechanical Example

Here is the promised example chosen on purpose to be from quantum mechanics. Essentially from Case, et al, [CJNM94, Cas07]:

\[ x \text{ codes a particle diffraction experiment} \& f(x) \text{ the resultant probable distribution or interference pattern on the other side of the diffraction grating. Quantum theory provides deterministic, algorithmic extraction of } f(x) \text{ from } x. \] A program for } f \text{ is, then, a predictive explanation or law for the set of such particle diffraction experiments.}

The program/law in this case does not tell us deterministically where the particles go. It tells us instead, deterministically, algorithmically, their statistically expected behavior! In the case an interference pattern is generated from an experiment } x \text{ where multiple particles are sent through a diffraction grating, it deterministically, algorithmically provides } f(x) \text{ which can be, then, a depiction of that interference pattern!}

Again, for the reasons spelled out at the end of Section 1.1 just above, a predictive explanation must provide its predictions algorithmically!

2. Data Types

In this section we indicate in detail how, without loss of generality, we can and will treat the functions } f \text{ such as those described above in Sections 1.1 and 1.2 just above.}

A countable set is (by definition) one in 1-1 correspondence with (some } \subseteq \text{) } \mathbb{N} = \{0, 1, 2, \ldots \}, \text{ the set of natural numbers.}
My former student, Mark Fulk, [Ful85] argued that the set of distinguishable experiments one can actually do and record on a phenomenon is countable: lab manuals can and do contain only finite notations, strings, and images from a finite alphabet of symbols, including gray and color pixel values.

One does not record measurements such as arbitrary infinite-precision real numbers of volts.

Beautiful continuous-mathematics (featuring uncountable sets such as that of the real numbers) is employed in physics many times to smooth out feasibly some much too complicated discrete reality, e.g., a giant cloud of electrons.

Interestingly, Maddy [Mad08] discusses the just prior paragraph, and provides a pointer, [ER06, Pages 290, 326], to cases where a continuous approximation to a discrete thermodynamic reality fails.

So, one of my working hypotheses is that reality is discrete. This is discussed further early in Section 3.1 below. Of course continuous mathematics is, in many cases, on a practical level, hard to replace.

In what follows, then, thanks to Gödel or code numbering an algorithmically circumscribed countably infinite set of experiments and outcomes for some phenomenon \( F \), e.g., some well circumscribed particle diffraction phenomenon: we imagine coding (algorithmically) the set of experiments associated with \( F \) onto \( \mathbb{N} \) and the possible outcomes into \( \mathbb{N} \), and we let the function \( f \) (associated with phenomenon \( F \)) map any experiment on phenomenon \( F \) with code \( \# x \), into the code \( \# y \) of the outcome of \( x \) on \( F \): \( f(x) = y \).

Hence, the type of our \( f \)s can and will be taken to be \( \mathbb{N} \rightarrow \mathbb{N} \).

Also, since we seek algorithmic explanations for \( F \), we can handle the cases only where \( f \) is also computable.

N.B. Our above discussion does not yet take into account error bounds on measurements, an important, crucial, practical consideration. For our approach, we can just consider that the code numbers of experiments and outcomes, include measurement error bounds.

3. Computability

In Section 3.1 just below is discussed my additional working hypothesis that the expected behavior of reality is algorithmic.
Then in Section 3.2 further below we explain what this has to do with human scientific endeavors.

Next, in Section 3.3, we consider objections based on apparent human creativity and free will.

3.1. Computability of Expected Reality

Researchers in the cellular automata approach to physics, e.g., [Fey82, Min82, FT82, Tof84, TM87, Mar84, SW86, Mac86, Tof77b, Tof77a, Vic84, Wol83, Svo86, FHP86, Has87], take seriously the idea that the universe, including space and time, may well be discrete. Here Feynman [Fey82] is crucial, and Minsky [Min82] lays out the ideas of Ed Fredkin on some of the different ways physical space could be discrete.\(^1\)

In a discrete, random universe but with computable probability distributions for its expected behaviors (e.g., a discrete, quantum mechanical universe with such distributions — as, I believe, ours is), the expected behavior will still be computable. It essentially follows from [dMSS56, San71, Gil72, Gil77] that one can compile any algorithm \(r\) having access to a random oracle, which oracle is subject to a computable distribution, into a deterministic algorithm \(d_r\) computing, in a sense, the expected outputs of \(r\).\(^2\)

Another working hypothesis of mine is, then, that the universe, besides being discrete, is algorithmic as to its expected behaviors.

N.B. We humans may be too finite ever to figure out completely how to compute the associated expected behaviors. But that’s just about human limitations.

3.2. Computable Expected Behavior of Science

We humans are components of the universe; hence, communities of scientists over time must also have computable expected behavior!

Herein, then, we’ll model scientists (and communities thereof over time) as algorithmic. Then we can have theorems about the boundaries of the (expected) behavior of science!

---

1The author of the present paper with others investigated lattice computers (a kind of cellular automata) in, for example, [CRS01], but he was also motivated therein by potential modeling thereby of spatial cognition [CRS94].

2We don’t know which, any, of the many provably disparate theoretical models of randomness (see, for example, [LV08, BST10]) corresponds to the randomness of, say, alpha-decay — and we’d like to know.
Just as a conservation assumption from physics provides boundaries on and insight into the physically possible, so too the computable expected behavior assumption on scientific inference provides boundaries on and insight into what’s possible with scientific inference.

In [Cas07] I discuss related language learning examples for cognitive science (not treated herein).

I invite physicists to explore the consequences for physics of our universe having computable expected behaviors. I’d really like to see something come out of that.

3.3. Creativity and Free Will

First we discuss creativity.

In a world with only computable expected behaviors, what about human creativity? How does my somewhat mechanistic working hypothesis account for the [Cas92] unbidden images which occur to people and which lead to solutions of difficult problems and/or works of great beauty and significance for the human condition?

I argue [Cas92] that humans are mostly not consciously aware of the brain processes that invoke such insights; hence, we have the illusion they aren’t algorithmically produced. Our conscious thoughts are the mere tip of an iceberg.

Post [Pos44] described as creative cases where algorithmic processes are algorithmically transcended. His examples generalize a bit the algorithmic process of Gödel [Göd86] essentially for transforming an algorithm for deciding a set of “consistent” axioms for an arithmetic into a corresponding Gödel sentence. Adding (trivially algorithmically) that sentence to the axiom set provides the transcendence.3

Next we discuss free will.

Libet, et al, [LGWP83] found that particular, experimentally detectable unconscious cerebral activity always strictly precedes conscious human experiences of willing to do something. This is somewhat suspicious methinks re the existence of human free will.

Conway and Kochen interpret their (Strong) Free Will Theorem [CK06, CK09] to mean, if some human has free will (about setting the details of some quantum mechanics experiment), then so do some particles.

3[Cas07] briefly refutes the argument that Gödel’s process falsifies mechanism.
They want to retain human free will, so they ascribe it to some particles too. Of course, at least in the case of particles, they mean by it only non-determinism.

I’m inclined to see conscious free will as another one of many human illusions. We may have some non-determinism, but our expected behavior does not.\footnote{We might have (in some sense) less non-determinism than mere particles since, for example, our DNA has some error-correcting capabilities.}

4. Machine Inductive Inference

Next we begin to describe a model of scientific inference.

\[
(0, f(0)), \ldots, (t - 1, f(t - 1)) \xrightarrow{\text{in}} M \xrightarrow{\text{out}} p_t
\]

Above \( M \) is an algorithmic device receiving \( f \)'s data points \((t, f(t))\), for \( t = 0, 1, \ldots \). N.B. For simplicity herein we’ll restrict the order of presentation of data from \( f \) to be in \textit{this} order (this matters in \textit{some} cases).

\( M \)'s output above, having seen the data sequence

\[
f[t] \overset{\text{def}}{=} (0, f(0)), \ldots, (t - 1, f(t - 1)),
\]

is \( p_t \), \textit{where} \( p_t \) is a program in some fixed, general programming system.\footnote{When \( t = 0, f[t] \) is the empty sequence.} We write \( M(f[t]) = p_t \). N.B. For simplicity herein we’ll restrict ourselves to the case where \( M \) on \( f[t] \) does not go into an infinite loop never producing \( p_t \) (this matters in \textit{some} cases).

\textit{Perhaps}, if \( M \) is “clever” enough and \( f \) is associated with a phenomenon \( F \) that is not too hard to figure out, eventually, i.e., for suitably large \( t \)s, the \( p \)s may come usefully close to computing \( f \). More on this topic, in Section 4.1 just below where we begin to discuss in more detail what can be meant by \textit{successful} scientific inference.

Then, in Section 4.2, we provide with interpretations some sample theorems about scientific inference.\footnote{[Cas07] provides additional examples.} Near the end of Section 4.2, we segue into Section 4.3 which discusses machine self-reference techniques which can, many times, be used to provide \textit{very succinct} proofs, relevantly herein, of results regarding scientific inference.
Lastly, in Section 4.4, is discussed, whether the self-referential examples employed might actually correspond to (non-artifactual) examples in the real world.

4.1. Criteria of Success

Definition 1 (Success Criteria $\text{Ex}^a$). Suppose $a \in (\mathbb{N} \cup \{\ast\})$. Suppose $S$ is a class of computable functions $f$. ‘Ex’ stands for ‘Explanatory.’ $a$ stands for anomaly.

$S \in \text{Ex}^a$ iff there is a suitably clever $M$ so that, for every $f \in S$, for some associated $t$, $M(f[t]) = M(f[t+1]) = \cdots$ and $M(f[t])$ computes $f$ — except at up to $a$ data points. Here, up to $\ast$ points means up to finitely many.

Informally, $M$ witnesses that $S \in \text{Ex}^a$ means, on any $f \in S$, $M$’s output programs on $f$, eventually settle down syntactically to a single program “for” $f$ which program has at most $a$ anomalous predictions re values of $f$.

In science, we don’t know when (if ever) we begin to have predictive explanations that are pretty good; we don’t know $t$’s value in the above Definition.

Definition 2 (Success Criteria $\text{Bc}^a$). ‘Bc’ stands for ‘Behaviorally correct.’

$S \in \text{Bc}^a$ iff, for some $M$, for every $f \in S$, for some associated $t$, programs $M(f[t]), M(f[t+1]), \cdots$ each computes $f$ — each except at up to $a$ data points.

For these $\text{Bc}^a$ criteria, the programs $M(f[t]), M(f[t+1]), \cdots$ can be (syntactically) quite different from one another.

For the criteria $\text{Ex}^a$ and $\text{Bc}^a$, my original motivation for the importance of small values of $a$, i.e., a few anomalies being tolerated in final predictive explanations, came from anomalous dispersion: the classical explanation for the degree of bending of “light” passing through a prism, fails for the X-ray case, an anomalous case.

4.2. Sample Theorems

Theorem 3 (Gold & Blum [Gol67, BB75]). The class of polynomial time computable functions $\in \text{Ex}^0$.

Theorem 4 (See [CS78, CS83]). $\text{Ex}^0 \subset \text{Ex}^1 \subset \cdots \subset \text{Ex}^* \subset \text{Bc}^0 \subset \text{Bc}^1 \subset \cdots \subset \text{Bc}^*$, where $\subset$ is proper subset.

Hence, tolerating anomalies strictly increases inferring power as does relaxing the restriction of (syntactic) convergence to single programs.
Physicists’ use of slightly faulty explanations is vindicated!

The anomalies that must be exploited to prove the $\text{Ex}^a$-hierarchy above are anomalies of omission or incompleteness: the predictive explanations’ errors are where they loop infinitely with no prediction [CS78, CS83].

Hence, thanks to the unsolvability of the Halting Problem [Rog67], Popper’s Refutability Principle [Pop68] is violated in a way Popper didn’t consider [CS78, CS83]!

We next present some very interesting restricted versions of $\text{Ex}^0$.

**Definition 5** (Postdictive Completeness [Bär74, BB75, Wie76, Wie78]). $S \in \text{PdCompEx}$ iff, some $M$ witnesses that $S \in \text{Ex}^0$ and, for every $f \in S$, for every $t$, for each $s < t$, the I/O behavior of program $M(f[t])$ on input $s$ must agree with $f$ on input $s$.

$\text{PdCompEx}$ provides a strong common sense constraint on $\text{Ex}^0$: a scientist should always hypothesize a program which at least postdicts his known data.

**Definition 6** (Postdictive Consistency [Wie76, Wie78, Cas07]). $S \in \text{PdConsEx}$ iff, some $M$ witnesses that $S \in \text{Ex}^0$ and, for every $f \in S$, for every $t$, for each $s < t$, either the I/O behavior of program $M(f[t])$ on input $s$ must agree with $f$ on input $s$ or program $M(f[t])$ on input $s$ loops infinitely.

$\text{PdConsEx}$ provides a weaker common sense constraint on $\text{Ex}^0$: a scientist should never conjecture an hypothesis which makes an explicit prediction contradicting his known data.

**Theorem 7** ([Bär74, BB75, Wie76, Wie78, CJSW04, Cas07]).

$\text{PdCompEx} \subset \text{PdConsEx} \subset \text{Ex}^0$!

Hence, surprisingly, for example, judiciously employing hypotheses explicitly contradicting known data can strictly enhance inferring power!

For example, it can be shown by a machine self-reference argument [Rog67, Kleene’s Recursion Theorem, Page 214] that the class of all computable $f$ with finite range and where max(range($f$)) codes a program for $f$ is $\in (\text{Ex}^0 - \text{PdConsEx})$ [Wie76, Wie78, CJSW04, Cas07].
To show this self-referential class $\in \text{Ex}^0$ is straightforward: have $M$ always output the program coded by the largest number it’s seen so far in the range of $f$. This makes the proof of the positive half extremely short.

To show this class $\notin \text{PdConsEx}$ succinctly employs machine self-reference mixed with so-called diagonalization [Rog67].

4.3. Self-Reference Techniques

The robot above has a transparent front through which its complete program (flowchart, wiring diagram, ...) can be seen. It stands in front of a mirror and a writing board, so it can copy its complete program on the board for use as data in its computations.

A simplest case, then, of machine self-reference involves a program (like the robot’s above) which makes a copy of itself to use as data. It, then, has usable, perfect self-knowledge!

The robot shown uses a mirror to make its self-copy. Simple self-replication works in the general case [Cas74, Cas94].

Machine self-reference can involve many programs, including infinitely

---

7 [Cas07] contains a proof of a related result which proof also essentially works for this result.

8 It can be correctly argued that the No Cloning Theorem of Quantum Mechanics [WZ82, Die82] implies that self-copies perfect down to quantum state can not be made. Fortunately, self-copies at usefully higher levels than the quantum states level can be created so as to persist long enough for human endeavors. Crucial example: computer bit settings can be copied with the copies’ bits stably persisting for very long time intervals and with corresponding negligible probability of degradation during those very long intervals.
many programs each making a self-copy for data-use by all of them [Cas74, Cas94]!

Consider the class $\mathcal{SB}_c$ of all computable $f$ such that all but finitely many numbers in the sequence of $f$’s successive values, $f(0), f(1), f(2), \ldots$, code programs for $f$.

It is straightforward to see that $\mathcal{SB}_c \in \mathbb{B}_c^0$: have $M$ successively output the programs coded by the succession of values of $f$, $f(0), f(1), f(2), \ldots$.

When I was co-creating [CS78], I had the intuition that $\mathcal{SB}_c$ captured the essence of $\mathbb{B}_c^0$.

In particular I thought that if any class would be in $(\mathbb{B}_c^0 - \mathbb{Ex}^*)$, $\mathcal{SB}_c$ would be. I showed with Harrington [CS78, CS83], by an infinitary machine self-reference argument, that, in fact, $\mathcal{SB}_c \notin \mathbb{Ex}^*$.

We can now formally define in strong senses what it means for a class to capture the essence of a success criterion and can prove for $\mathcal{SB}_c$ that, in one sense, it does (although, newer, related, so-called self-learning classes, capture in a stronger sense). See Case and Kötzing [CK11a] (see also [CK08], and, for related work re language learning, see [CK10b, CK10a, CK11b]).

We can now formally define, in a strong sense, what it means for a class to capture the essence of a success criterion and can prove for self-referential classes like $\mathcal{SB}_c$ it does. See Case and Kötzing [CK10b] for preliminary work.

4.4. Self-Reference in Reality

Generally, machine self-reference proofs for theorems like the above are more succinct than alternative proof techniques. I like them.

Interesting work exists on whether separation results from Section 4.2 above hold if one “destroys” the self-reference tricks [Zeu86, KS89, Ful90, CJO+00, CJSW04, JSW01, Jai99, OS02] We’ll not pursue this further herein.

Instead, our interest herein is whether self-referential examples entail the existence of (non-artifactual) real world witnessing examples.\(^9\)

Case [Cas86] notes that in some views of the world it is a network with parts reflecting on the whole. That resembles multiple machine self-reference. Human social cognition is an imperfect such network.

Case [Cas99] argues that a machine self-reference argument is such a simple reason for a truth, the “space” of reasons for its truth may be broad enough to admit natural examples.

\(^9\)One could, in principle, build artifactual black box devices which work (and could be inductively inferred from their behavior) like the members of $\mathcal{SB}_c$ above.
Also noted therein is that, empirically, while Gödel [Göd86] proved his famous first incompleteness theorem by a (linguistic) self-reference argument\(^{10}\), later researchers [PH77, Sim85, Sim87] found quite natural examples of incompleteness.

I think machine self-reference proofs for the existence of situations are harbingers of natural examples witnessing the same situations.

5. Infeasibly Computable Expected Reality

In modern computer science there is a sensible, practical focus on a special case of computable, namely, feasibly computable.

Originally this meant polynomial-time computable (in the length of input on multi-tape Turing machines) [Cob64], but recently it can mean BQP-computability, a quantum-parallelism version of polynomial-time computability [BV97].

Does the universe have some non-artifactual expected behaviors which, while computable, are infeasibly so? An artifactual computer with access to increasing memory resources and which employs any algorithm for deciding Presburger arithmetic would, in principle, have infeasible expected behavior [FR74].

Fix a meaning for feasibly computable. Humans (including human scientists) don’t currently know how to compute feasibly the infeasibly computable. I conjecture that our massively parallel processing brains are sometimes good for handling largish constants in front of whatever run-time bounds.\(^{11}\)

\(^{10}\)It can also be proved by a machine self-reference argument.

\(^{11}\)So-called hypercomputation typically involves allowing infinitely many (say, Turing machine) computation steps in finite time (see, for example, [Kre65, Kre74, Rog67, Dav04, Dav06a, Dav06b, Dav06c, Dow10, AMN+10]. A recursive iteration of the idea leads to Kreisel’s \(\aleph_0\)-mind computability (characterizing the \(\Pi^1_1\)-computable partial functions); \(\aleph_0\)-mind computability permits deciding first order arithmetic, but not second order arithmetic (see Rogers [Rog67] for definitions, results, and very nice discussion).

The big question is whether, in our physical universe (which happens to currently include human brains), such computations are actually executable. I, of course, believe they are not — although I wish they were. Margolus [Mar95, Cas07] pointed out to me that such computations would require infinitely much energy, which is ostensibly not available to apply anywhere locally in our universe. Wonderful tricks possibly to achieve real world hypercomputations which exploit temporal differences inside and outside black holes (see, for example, brief mention and references in [Dow10]) won’t work either — since, thanks
Feynman [Fey82] was the first to remark that some of our (humanly produced, predictive) explanations for quantum phenomena and run on (deterministic) parallel processing computers produce answers significantly more slowly than nature itself produces those answers! This led to the beginning of thinking of possible speed-ups with quantum parallelism.\(^{12}\)

I expect any humanly-produced and employed acceptable programming systems\(^{13}\) are quite unlike the (unknown) acceptable system of non-artifactual physical reality.

### References


---

\(^{12}\)Kemeny \[Ken59\] provides a wonderful, informal (self-reference) argument for cases where one cannot predict reality before that reality happens.

\(^{13}\)The acceptable systems are essentially the natural, general purpose systems for computing all the partial-computable functions Mathematically, they are those in which any other system can be interpreted \[\text{Rog58, Rog67, MY78, Ric80, Ric81, Roy87}\]. An example humanly producible and employable one would be C++ implemented on some deterministic parallel processing computer, which computer permits dynamic memory extensions.


