Directions for Computability Theory Beyond Pure Mathematical

John Case [†]

University of Delaware
Newark, USA

This paper begins by briefly indicating the principal, non-standard motivations of the author for his decades of work in Computability Theory (CT), a.k.a. Recursive Function Theory.

Then it discusses its proposed, general directions beyond those from pure mathematics for CT. These directions are as follows.

(1) Apply CT to basic *sciences*, for example, biology, psychology, physics, chemistry, and economics.

[†] The author was supported in part by the National Science Foundation (NSF) grant CCR 0208616.

(2) Apply the resultant insights from (1) to philosophy and, more generally, apply CT to areas of philosophy *in addition to* the philosophy and foundations of mathematics.

(3) Apply CT for insights into engineering and other professional fields.

Lastly, this paper provides a progress report on the above non-pure mathematical directions for CT, including examples for biology, cognitive science and learning theory, philosophy of science, physics, applied machine learning, and computational complexity. Interweaved with the report are occasional remarks about the future.

1. Motivations

Ted Slaman [**159**] has nicely mentioned the central theme of his intellectual motivation (deriving from the influence of Sacks) for working in Computability Theory (CT), a.k.a. Recursive Function Theory, namely, *definability.* I like this very much; however, my own strongest intellectual motivations for devoting much of my research to CT have a very different nature and origin.

Since these latter motivations are directly or indirectly relevant to some of the directions I propose below, and are, I believe, fairly atypical among CT researchers, I describe them in some detail.

Before my undergraduate experiences, I was intellectually motivated by what I would now describe as philosophically oriented scientific curiosity. I naively hoped to discover the fundamental nature of the universe. It is likely I did not consider what that actually meant. I knew about Einstein, but not about Gödel and Turing. I considered studying physics, astronomy, or psychology. I remained flexible about fields in the future, expecting I should see what I liked, etc. I would now say I was a naive scientific reductionist and, so, chose physics (over psychology) for my

UG experience. My UG minors became mathematics and philosophy, and I took but one psychology course. Psychology seemed more easily learned on my own.

I considered physics, mathematics, or philosophy for graduate school. For various reasons, including generally better fit of cognitive style between myself and the field, I selected mathematics.

When I first learned CT the two major, intellectual ideas that captivated me were[1]

- ***that the definition of computable is an absolute,*** and
- ***my realization that, very likely, the universe, above some level at least, is computable***[2]***.***

In a sense, re the second bullet just above, with CT I felt that I was dealing with physics again in the form of a very abstract mechanics. Also, since biological organisms including humans are components of the physical universe, with CT I also had a very abstract handle on biology, psychology ... . Later I found [**121**] in which Myhill says on P. 149:

> ...in the author's view, the theorems of Church and Gödel are psychological laws. Mr. E. H. Galanter of the Department of Psychology, University of Pennsylvania, described them in conversation with the author as "the only known psychological laws comparable in exactitude with the laws of physics."[3]

[1] Many, for me, less major things also helped with the captivation, for example, the aesthetics of (some of) CT's tools, for example, recursion theorems [**23, 24, 143**].

[2] N.B. In a discrete, random universe but with *computable probability distributions* for its behaviors (for example, a discrete, quantum mechanical universe, perhaps, as I believe, ours is), the statistically *expected* behavior will still be computable [**60**] (and constructively so [**72, 73**]).

[3] However, when, many years later, I got to know Myhill and discussed with him the idea that people are essentially algorithmic mechanisms, he was, at that time, no longer in favor of the idea.

It is sometimes argued that Gödel's theorems imply people are not algorithmic. It is, I believe, never argued that there are people who can list or decide the set of truths of (first or second order)[4] arithmetic, or who solve the halting problem, or .... Anyhow, regarding the arguments that are presented: they suffer not only from the usual problems of confusing "**T** being consistent" with "**T** being known to be consistent," but also, *and more basically*, with confusion about productive sets [**148**]. Many times these arguments are essentially, "I know an algorithm witnessing the set of truths of some arithmetic is productive; i.e., I know an algorithm which provides counterexamples to alleged complete recursive axiomatizations [**113**]; therefore, I am not a machine." More simply, "I know an algorithm; therefore, I am not a machine." In this form, these arguments are seen as absurd. See also [**50, 51**].[5]

Beginning with the next paragraph I list my suggested directions. After I present the directions, I present examples of progress to date and suggest future work.

2. Directions

Direction 2.1. Apply CT to the basic sciences.[6]

This leads to a second

[4] I mean second order in the sense of [**148**].

[5] I plan to write a philosophical paper in which I present some new arguments in this area and which tend a bit in the opposite direction of supporting a mechanistic world, or at least one with mechanistic expected behavior.

[6] By basic sciences, I have in mind biology, psychology, physics, chemistry, economics, etc., sciences which, each to varying degrees of predictive success, apply scientific method.

I do not consider set theory to be one of the sciences in the above sense, and it is not clear platonists would consider it to be anyhow — even if they think sets are a component of the universe independent of human invention. I am not a platonist. For me, set theory requires deciding what is useful and interesting to mean by sets, and I personally expect that that will have no absolute answer.

Direction 2.2. Apply the understandings from successes re Direction 2.1 to philosophy *and, more generally*, apply CT to more areas of philosophy than at present.[7]

At least for my personally captivating intellectual motivations for CT bulleted above, I believe CT *deserves* to survive! It is not so clear that it will.[8] This *partly* motivates

Direction 2.3. Apply CT to engineering and other professional or applied fields more generally.

3. Progress So Far And How One Might Go From Here

In this section, I proceed approximately chronologically with respect to my own first associations with the general subject headings. I interweave with the progress report occasional remarks about the future. I also make occasional remarks about proof techniques employed thus far[9], and prove one sample theorem (Theorem 3.1 in Section 3.2.1 below). I intend the progress report material partly as evidence that progress is possible, not as

[7] I like very much the idea of people *continuing* to apply CT to foundations of *mathematics* and associated philosophy. Directions 2.1 and 2.2 are, I believe, a needed expansion to areas *outside* of mathematics itself, i.e., moves away from math-centricism.

[8] For U.S. mathematical science departments, NSF commissioned [**125**], and in Appendix 2 **Assessment of Subfields**, under **Foundations** we see, among other things, the following.

> **Recursion (or computability) theory** is quiescent, with a substantial body of completed work. Barring a major breakthrough, or the further exploitation of connections with computational mathematics and computer science, the next decade is not expected to be very active. England plays a leading role, with the United States as a contributor, but the aging research population is not being replenished.

[9] I have not seen a need for n-jump priority arguments yet. There may eventually need to be new, complex methods created to obtain results in my proposed directions.

how one must necessarily proceed next. I will not be proposing particular and well-defined problems. Instead *and in general* I propose the creation of interesting, *insightful, and interpretable* definitions, problems, theorems for the sciences, philosophy, and applied fields.

Standard computability-theoretic notation will be from [**148**]. For example, N will denote the set of natural numbers, $\{0,1,2,\ldots\}$. φ will denote a fixed acceptable programming system (or numbering) for the class of partially computable functions: $N \rightarrow N$, where φ_p is the partially computable function computed by program (or index) p in the system. For a partial function ψ, $\delta\psi$ and $\rho\psi$ denote the domain and range of ψ, respectively. We write W_p for the r.e. set accepted or enumerated by φ-program p, where, formally, $W_p \overset{\text{def}}{=} \delta\varphi_p$. We also write $\downarrow$ for converges or is defined and $\uparrow$ for diverges or is not defined.

3.1. Biology

Kleene [**95**, p. 59][10] was apparently the first to notice the connection between his second recursion theorem [**148**, p. 214] and Von Neumann's self-reproducing automata [**124, 17**]. I recall that Kleene told me (perhaps at the Kleene Symposium) that he had used his recursion theorem to understand Von Neumann's construction. This amazed me, since, by contrast, I had used von Neumann's construction to understand Kleene's proof of his recursion theorem [**21**].[11] Myhill's [**123**] seems to be the first *published* account featuring a connection between CT and self-reproducing automata although it does not employ a recursion theorem. This

[10] This paper is worth a read more generally.

[11] Wonderfully, the top level logic/refinement of Von Neumanns's construction is essentially identical to the top level logic/refinement of biological single-celled organisms' self-reproductive procedure. http://www.cis.udel.edu/~case/papers/krt-self-repo.pdf explicitly lays out the correspondence between Kleene's proof of his recursion theorem and (the top level of) a single-celled organism's self-reproductive procedure. This expands on the discussion of same in [**21**].

paper provided me with my first hint of how to connect CT to modeling the physical world, and it directly motivated my [**21**] (and the earlier [**20**]).

In [**123**], Myhill considers variants of machines which build strict copies of themselves, for example, machines which build distortions such as mirror images of themselves and machines which deterministically evolve with each generation a better automatic theorem prover.

Herein, in our brief discussion of [**21**], I omit most technical details. Suffice it to say one has a sequence of constructor machines $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \ldots$, and, besides their constructing capabilities, they have arbitrary effectively pre-assigned respective computing capabilities.

For these constructor machines, we write $\mathcal{M}_p \rightarrow \mathcal{M}_q$ to mean $\mathcal{M}_p$ constructs or begets $\mathcal{M}_q$.

One of the emphases in [**21**] was periodicity in generations of machines which construct offspring. For example, given any $n \in N$, we can obtain pairwise *distinct* constructor machines $\mathcal{M}_{e_0}, \mathcal{M}_{e_1}, \ldots, \mathcal{M}_{e_n}$ such that $\mathcal{M}_{e_0} \rightarrow \mathcal{M}_{e_1} \rightarrow \ldots \rightarrow \mathcal{M}_{e_n} \rightarrow \mathcal{M}_{e_0}$.[12] This sequence replicates with period $n+1$. Self-reproduction is the $n+1 = 1$ case. In nature we also see period two, the $n+1 = 2$ case: the metagenic cœlenterates *Aurelia* and *Obelia* [**84**, p. 246] alternate between an attached polyp generation and a free swimming medusa generation with each looking and behaving different from the other. We see period three, the $n+1 = 3$ case, in some parasites which occupy a succession of very different hosts. I could not find evidence of organisms reproducing with periodicity in generations greater than three. The obvious problem for biologists suggested by the existence of constructor machines which replicate every $n+1$ generations for arbitrary large n is, Why do we not see larger periods in nature?

[12] The proof can be done by a padded $n+1$-ary recursion theorem. To handle elegantly the cases of *a*periodicity in generations (with no sterile descendent), I invented my Operator Recursion Theorem (ORT) [**21**], an infinitary self-reference principle [**24**]. My ORT is called the Functional Recursion Theorem in [**127**].

Another example from [**21**], given any $n \in N$, we can constructively obtain *different* pairwise distinct constructor machines $\mathcal{M}_{e_0}, \mathcal{M}_{e_1}, \ldots, \mathcal{M}_{e_n}$ such that $\mathcal{M}_{e_0} \to \mathcal{M}_{e_1} \to \ldots \to \mathcal{M}_{e_n}$ *and* $\mathcal{M}_{e_n}$ is sterile, and, additionally, where each $\mathcal{M}_{e_i}$, for $i \leqslant n$, has the *same* arbitrary pre-assigned computing capabilities. The nth descendant of $\mathcal{M}_{e_0}$ exists and is sterile, where $\mathcal{M}_{e_0}$ is considered to be the zeroth descendant of itself. In drafting this chapter, I thought of an engineering application. Many times humans have introduced a new organism into an environment to control a pre-existing pest organism only to discover that the new organism is a pest itself. The application is to employ the trick for producing the $\mathcal{M}_{e_0}, \mathcal{M}_{e_1}, \ldots, \mathcal{M}_{e_n}$ of this paragraph to engineer genetically a proposed new organism to introduce for pest control so as to have its nth descendant exist and be sterile (for whatever n is desired), but to have its non-reproductive functions not be altered. Then, the altered new organism, if it turns out to be a pest itself, will die out anyway. If it does not seem to be a pest itself, and it is still needed to control the original pest, it can be reintroduced.

3.2. Machine inductive inference and computability-theoretic learning

CT applied to these areas first appeared in [**141, 76**]. Associated textbook material appears in [**131, 86, 127**].

One of the endorsement sentences I composed for MIT Press regarding the then upcoming [**86**] reads as follows.[13]

> Just as a conservation assumption from physics provides boundaries on and insight into the physically possible so too the computability assumption on learning provides herein boundaries on and insight into the cognitively possible.

[13] Another sentence with very different content was actually used in my endorsement on the book jacket.

The material in this section features Philosophy of Science (Section 3.2.1), Cognitive Science and Language Learning (Section 3.2.2), and Applied Machine Learning (Section 3.2.3).

Some CT work applied to computational complexity aspects of learning appears in Section 3.4 below.

3.2.1. Philosophy of Science. On [**5**, p. 125] it says the following.

> 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, for example, a high-energy physics experiment, and y describes the results obtained, for example, the particles produced and their respective properties. The law he seeks is essentially an algorithm for computing the function $f(x) = y$.

Here is another example from [**35**]: x codes a particle diffraction experiment and $f(x)$ the resultant probable distribution (or fringe pattern) on the other side of the diffraction grating. Quantum theory provides *algorithmic* extraction of $f(x)$ from x. A program for f is, then, a *predictive* explanation or law for the set of such diffraction experiments.

If, in our universe, people, including scientists (and collections thereof, including over historical time), are essentially algorithmic (as I believe), one can use CT to get theorems in philosophy of science. This realization occurred to me in the mid 70s and, for me, it was extremely intellectually exciting.

Some computability-theoretic inductive inference publications with something to say for philosophy of science are [**141, 76, 4, 5, 93, 184, 46, 47, 37, 10, 35, 66, 105, 38, 1**].

Some philosophy of science publications influenced by computability-theoretic inductive inference are [**75, 91, 92, 101, 89, 102, 153, 154, 90**].

In the rest of this section, I present a few sample results from computability-theoretic inductive inference with corresponding indications of their philosophical meaning.

In the following we will model inductive inference machines **M** extensionally as partially computable functions which take for their inputs finite initial segments of functions $f : N \to N$ and which either go undefined or return programs in the φ-system. Intuitively, for a finite initial segment σ, if $\mathbf{M}(\sigma)$ is defined (written: $\downarrow$) $= p$, then p represents **M**'s conjecture or hypothesis as to a program for f based on the data points about f contained in σ. We write $\sigma \subset f$ to mean σ is a finite initial segment of f, i.e., σ is a finite initial segment of a function, and its graph is a proper subset of that of f. We write $\overset{\infty}{\forall}$ to mean for all but finitely many. $\mathcal{R}$ denotes the class of computable functions mapping N into N.

We next consider a criterion of successful inductive inference.

Definition 3.1. **M** **Ex**-*learns* $f \Leftrightarrow [(\forall \sigma \subset f)[\mathbf{M}(\sigma)\downarrow] \wedge (\exists p)(\overset{\infty}{\forall} \sigma \subset f)[\mathbf{M}(\sigma) = p \wedge \varphi_p = f]]$.

Intuitively, **M** **Ex**-learns f means that, **M** fed successively more data about f, outputs a corresponding succession of conjectures and *eventually* begins to output the same correct φ-program p for f over and over. For **Ex**-learning we think of **M** as eventually finding a predictive *ex*planation p for f [**47**].

For p satisfying the right hand side of Definition 3.1 just above, we write $\mathbf{M}(f)\downarrow = p$.

$\mathbf{Ex} \overset{\text{def}}{=} \{\mathcal{S} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathbf{M}\ \mathbf{Ex}\text{-learns each } f \in \mathcal{S}]\}$. For example, the class of one-argument primitive recursive functions *is* in **Ex**, but $\mathcal{R}$ is *not* [**76, 5**]. Hence while some single **M** is "clever" enough to **Ex**-learn every primitive recursive function, no single **M** is clever enough to **Ex**-learn each computable function.

Next we begin to consider alternative criteria of success.

Definition 3.2. **M** **Conf**-*learns* $f \Leftrightarrow [\mathbf{M}\ \mathbf{Ex}\text{-learns } f \wedge (\forall \sigma \subset f)[\mathbf{M}(\sigma)\downarrow \wedge (\varphi_{\mathbf{M}(\sigma)} \cup \sigma)$ *is single-valued*$]]$.

Let $f[n] \stackrel{\mathrm{def}}{=} f(0), \ldots, f(n-1)$.

We see that an $\mathbf{M}$ which **Conf**-learns a function f **Ex**-learns f *and* produces, on each input $f[n]$, a corresponding conjecture, $\mathbf{M}(f[n])$, based on the data in $f[n]$, *and* this conjecture does not explicitly output something *convergently contradicting* any data in $f[n]$. A program $\mathbf{M}(f[n])$ *may* go undefined on some inputs $< n$, but *on such inputs*, it, then, does *not converge* to anything different from what f does. This *seems* like a very reasonable, common sense restriction. In fact, the stronger looking restriction for the related **Cons**-*learning* criterion may seem reasonable too: each program $\mathbf{M}(f[n])$ must converge to f *on any inputs* $< n$. This restriction requires that each conjecture of $\mathbf{M}$ on f has to be correct on the data about f on which that conjecture is based.[14]

$\mathbf{Conf} \stackrel{\mathrm{def}}{=} \{\mathcal{S} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathbf{M}\ \mathbf{Conf}\text{-learns each } f \in \mathcal{S}]\}$. Also, $\mathbf{Cons} \stackrel{\mathrm{def}}{=} \{\mathcal{S} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathbf{M}\ \mathbf{Cons}\text{-learns each } f \in \mathcal{S}]\}$.

Surprisingly, these natural, common sense restrictions on inductive inference (for **Conf** and **Cons**) strictly limit learning or inductive inference power (as measured by **Ex**-learning).

Theorem 3.1 (Wiehagen [**184**]). $\mathbf{Cons} \subset \mathbf{Conf} \subset \mathbf{Ex}$.

For example, the second, more surprising non-containment in Theorem 3.1 just above entails that, in some cases for explanatory inductive inference, employing conjectures convergently contradicting known data gives one strictly greater inferring power than not contradicting known data! This result is, I believe, of great interest for philosophy of science.

Proof of Theorem 3.1. Of course $\mathbf{Cons} \subseteq \mathbf{Conf} \subseteq \mathbf{Ex}$. Since, offhand, I know of no easily available proof of the second, somewhat harder, more surprising non-containment, and to provide herein at least one *illustrative, short, and sweet* proof

[14] **Conf** is short for *Conformal*, and **Cons** is short for *Consistent.* I sometimes like referring to Conformal as *postdictively-consistent* (not explicitly contradicting known data points in one's conjectures based on them) and Consistent as *postdictively-complete* (not missing any known data points in one's conjectures based on them).

re inductive inference, I prove the more surprising of the non-containments.

Let

$$\mathcal{S} = \{f \in \mathcal{R} \mid \rho f \text{ is finite } \wedge \varphi_{\max(\rho f)} = f\}, \tag{3.1}$$

a self-referential class. We will show that

$$\mathcal{S} \in (\mathbf{Ex} - \mathbf{Conf}).^{15} \tag{3.2}$$

Trivially, $\mathcal{S} \in \mathbf{Ex}$ — as witnessed by a machine that, on any f, outputs the largest thing, if any[16], it has seen so far in the range of f.

Suppose for contradiction $\mathbf{M}$ witnesses that $\mathcal{S} \in \mathbf{Conf}$. Hence

$$(\forall f \in \mathcal{S})(\forall \sigma \subset f)[\mathbf{M}(\sigma)\downarrow \wedge (\varphi_{\mathbf{M}(\sigma)} \cup \sigma) \text{ is single-valued}]. \tag{3.3}$$

Claim 3.1.

$$(\forall \sigma)(\exists f \in \mathcal{S})[f \supset \sigma]. \tag{3.4}$$

Hence, by (3.4), (3.3)*,*

$$(\forall \sigma)[\mathbf{M}(\sigma)\downarrow \wedge (\varphi_{\mathbf{M}(\sigma)} \cup \sigma) \textit{ is single-valued}]. \tag{3.5}$$

PROOF. We let $\max(\emptyset) \overset{\text{def}}{=} 0$.

Suppose σ is given. By a padded version of Kleene's Recursion Theorem, there is program e such that $e > \max(\rho\sigma))$, *and*, on input x, e looks in a mirror to see which program it is[17], and, then,

$$\varphi_e(x) = \begin{cases} \sigma(x) & \text{if } x \in \delta\sigma, \\ e & \text{otherwise.} \end{cases} \tag{3.6}$$

Let $f = \varphi_e$. Then $f \supset \sigma$, and $f \in \mathcal{S}$. □

[15] This and other such examples for witnessing the rest of the theorem as well as for other, related results are stated, but not proven correct, in [**40**]. For $(\mathbf{Ex} - \mathbf{Cons}) \neq \emptyset$, see [**4, 5, 183**].

[16] If nothing has yet appeared in the range of f, the machine can output any program.

[17] See [**24, 143**] for more about this way of understanding recursion theorems and program or machine self-reference in terms of mirrors. It is also discussed in Section 3.3 below.

We continue with the proof of the theorem.

We write $\sigma \cdot i$ for the finite sequence consisting of σ followed by i.

By a padded version of Kleene's Recursion Theorem, there is a *different* program $e > 0, 1$ such that this e looks in a mirror to see which program it is, and, then, the rest of this e's behavior is described informally below.

```
begin Program $e$;
  set $\varphi_e(0) = e$;
  let $\varphi_e^s$ be the finite sequence of successive values of $\varphi_e$ defined before stage $s$ below[18];
  do stage $s$ for $s = 0$ to $\infty$;
    begin stage $s$;
      if (i) $\mathbf{M}(\varphi_e^s \cdot 0) = \mathbf{M}(\varphi_e^s \cdot 1)$ (* i.e., $\mathbf{M}$ is insensitive *)
        then
          set $\varphi_e^{s+1} = \varphi_e \cdot 0$ (* passive ploy *)
        else (ii) (* i.e., $\mathbf{M}$ is sensitive *)
          set $\varphi_e^{s+1} = \varphi_e \cdot \min(\{i \leqslant 1 \mid \mathbf{M}(\varphi_e^s) \neq \mathbf{M}(\varphi_e^s \cdot i)\})$ (* aggressive ploy[19] *)
      endif
    end stage $s$
  enddo
end Program $e$.
```

Clearly, by (3.5), φ_e is total and, then, is $\in \mathcal{S}$. Therefore, $(\exists p)[\mathbf{M}(\varphi_e)\!\downarrow = p \wedge \varphi_p = \varphi_e]$.

Hence $(\overset{\infty}{\forall} s)$[(i) holds at stage s] — since each stage in which (ii) holds forces $\mathbf{M}$ to make another mind change.

Pick s_0 so large that [(i) holds at stage $s_0 \wedge \mathbf{M}(\varphi_e^{s_0+1}) = \mathbf{M}(\varphi_e^{s_0}) = p$].

For each $i \leqslant 1$, let $\sigma_i = \varphi_e^{s_0} \cdot i$.

[18] φ_e^s is an initial segment of a function; hence, it has domain $\{0, \ldots, n-1\}$ for some $n \geqslant 0$.

[19] This strategy forces $\mathbf{M}$ to make a change of conjecture, a "mind" change, on φ_e.

Then, $\sigma_0 = \varphi_e^{s_0+1} \neq \sigma_1 \wedge \delta\sigma_0 = \delta\sigma_1$.
Let $x^{s_0} = \max(\delta\sigma_0)$, which $= \max(\delta\sigma_1)$, and is > 0.
Since $\mathbf{M}$ is *in*sensitive at stage s_0, $\mathbf{M}(\sigma_0) = p = \mathbf{M}(\sigma_1)$.
By (3.5),

$$[(\varphi_{\mathbf{M}(\sigma_1)} \cup \sigma_1) \text{ is single-valued}]. \tag{3.7}$$

$x^{s_0} \in \delta\sigma_0 = \delta\sigma_1 \wedge \sigma_0(x^{s_0}) = 0 \neq 1 = \sigma_1(x^{s_0})$.
$\varphi_{\mathbf{M}(\sigma_0)} = \varphi_{\mathbf{M}(\sigma_1)} = \varphi_p = \varphi_e$.
$\delta\varphi_e = N$. Therefore, $x^{s_0} \in \delta\varphi_e$ too.
Since $\varphi_{\mathbf{M}(\sigma_0)} = \varphi_e$, $(x^{s_0}, 0) \in (\varphi_{\mathbf{M}(\sigma_0)} \cap \sigma_0)$.
Since $\varphi_{\mathbf{M}(\sigma_0)} = \varphi_{\mathbf{M}(\sigma_1)}$, $(x^{s_0}, 0) \in \varphi_{\mathbf{M}(\sigma_1)}$. Also, $(x^{s_0}, 1) \in \sigma_1$.
Hence

$$(x^{s_0}, 0), (x^{s_0}, 1) \in (\varphi_{\mathbf{M}(\sigma_1)} \cup \sigma_1), \tag{3.8}$$

a contradiction to (3.7) . □

Next is another theorem I like very much.

Theorem 3.2 (Bārzdiņš [**4**], Blum and Blum [**5**]). **Ex** *is not closed under union, i.e., there are classes* $\mathcal{S}_0, \mathcal{S}_1 \in$ **Ex** *such that* $(\mathcal{S}_0 \cup \mathcal{S}_1) \notin$ **Ex**.

Here are example such $\mathcal{S}_0, \mathcal{S}_1$ — similar to those from [**5**]. Let $\mathcal{S}_0 = \{f \in \mathcal{R} \mid \varphi_{f(0)} = f\}$ and $\mathcal{S}_1 = \{f \in \mathcal{R} | (\overset{\infty}{\forall} x)[f(x) = 0]\}$.

Theorem 3.2 essentially suggests that for success in inductive/scientific inference, one needs the diversity of incomparable "cognitive" styles of scientists: scientists $\mathbf{M}_0$ which **Ex**-learns $\mathcal{S}_0$ and $\mathbf{M}_1$ which **Ex**-learns $\mathcal{S}_1$ cannot be combined into a third scientist $\mathbf{M}_2$ which **Ex**-learns all that $\mathbf{M}_0$ does together with all that $\mathbf{M}_1$ does.

In physical optics there is a phenomenon called *anomalous dispersion*: the classical, quantitative explanation for different frequencies of light and, more generally, electromagnetic radiation being differentially bent according to frequency when passing through a prism does not work for X-rays.[20] Physicists used this model nonetheless until quantum mechanics provided a better model. In the mid 70s, I began to consider whether I could prove a theorem re machine inductive inference that would suggest a vindication of physicists' employing slightly faulty predictive explanations. For each $n \in N$, for partial functions η, θ, we write $\eta =^n \theta$ to mean that there are at most n counterexamples to $\eta = \theta$. We write $\eta =^* \theta$ to mean that there are at most finitely many counterexamples to $\eta = \theta$. For $a \in N \cup \{*\}$, we define a variant of **Ex**-learning, called $\mathbf{Ex}^a$-*learning*, in which the eventual final programs p are allowed to be mistaken on up to a inputs in computing the input function f, i.e., success requires only that $\varphi_p =^a f$. A theorem indicating an increase in inferring or learning power comes with tolerance of some few mistakes in one's predictive explanations follows.

Theorem 3.3 (Case and Smith [**46, 47**]). $\mathbf{Ex} = \mathbf{Ex}^0 \subset \mathbf{Ex}^1 \subset \mathbf{Ex}^2 \subset \ldots \mathbf{Ex}^*$.

Hence we see that tolerating up to just one single anomaly or mistake in one's final program provides a strict increase in inferring power, tolerating $n + 1$ anomalies provides an increase over tolerating no more than n, and tolerating a finite number provides an increase over tolerating a bounded number.[21] Of course, $\mathbf{Ex}^*$ is not so "practical" a criterion as $\mathbf{Ex}^n$, for small n, since, for the former, the finite number of anomalies in a final program p may include all the data points for which the predictive explanation p will ever be used. $\mathbf{Ex}^*$ is mathematically interesting though.

In [**47**], we pointed out that Popper's Refutability Principle, the principle that purported scientific explanations ought to be subject to refutation by suitable experiments, needs some revision. The anomalies providing the hierarchy of Theorem 3.3 above

[20] Of course it is the model or predictive explanation which is anomalous, not the physical phenomenon itself.

[21] [**5**] announced the case of $\mathbf{Ex} \subset \mathbf{Ex}^*$.

are *and must be* mistakes of *omission* [**47**], but this kind of mistake cannot be algorithmically detected in general. Explanations ought to be refutable when they make predictions, but may not be refutable when they fail to make a prediction at all — even if they should have.

Let $\text{minprogram}(f) \overset{\text{def}}{=} \min\{p \mid \varphi_p = f\}$.

Definition 3.3 (Freivalds [**68**]). For $\mathcal{S} \subseteq \mathcal{R}$, $\mathcal{S} \in \mathbf{Mex}$ as witnessed by $\mathbf{M} \Leftrightarrow (\exists \text{ computable } h)(\forall f \in \mathcal{S})[\mathbf{M}\ \mathbf{Ex}\text{-learns } f \wedge (\forall p)[\mathbf{M}(f)\downarrow = p \Rightarrow p \leqslant h(\text{minprogram}(f))]]$.

For $\mathbf{M}, \mathcal{S}, h$ as in the just above definition, $\mathbf{M}$'s final programs on $f \in \mathcal{S}$ are within "factor" h of minprogram(f). **Mex**-learning is intended as *a* model of inductive inference obeying a form of Occam's Razor. It is common in philosophy of science and in the applied part of artificial intelligence called *machine learning* to assume one's models for fitting and predicting data *should* obey some form of Occam's Razor. Yet we have the following theorem which shows that a simple, easily inferred subclass of $\mathcal{R}$ is *not* in **Mex**.

Theorem 3.4 (Kinber [**93**]). $\mathcal{S}_1 = \{f \in \mathcal{R} \mid f$ *is the characteristic function of a finite set*$\} \in (\mathbf{Ex} - \mathbf{Mex})$.

Hence, at least some forms of another common sense principle, Occam's Razor, *restrict* one's inferring or learning power! Theorem 3.4 just above can be proved by a recursion theorem argument together with a finitary cancellation (zero-injury priority) scheme.[22] For more on **Mex** and variants, see [**32, 33, 1**].[23]

There are costly criteria providing inferring or learning power beyond that of $\mathbf{Ex}^*$. $\mathbf{M}$ $\mathbf{Bc}$-*learns* $f \in \mathcal{R}$ means that $\mathbf{M}$ on f outputs an infinite sequence of programs $p_0, p_1, p_2, \ldots$, and $(\overset{\infty}{\forall} i)[\varphi_{p_i} =$

[22] Chen in [**32, 33**] showed that, by contrast, $\mathcal{S}_0 = \{f \in \mathcal{R} \mid \varphi_{f(0)} = f\} \in \mathbf{Mex}$, and yet, by a recursion theorem argument from [**5**], self-referential classes like $\mathcal{S}_0$ are so large they contain a finite variant of each element of $\mathcal{R}$.

[23] The latter features infinitary self-reference arguments employing my ORT Theorem [**21, 24**].

$f]$. **Bc** or *b*ehaviorally *c*orrect learning features semantic convergence to correct programs; whereas, **Ex**-learning features syntactic convergence to correct programs. $\mathbf{Bc} \overset{\mathrm{def}}{=} \{\mathcal{S} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathbf{M}$ **Bc**-learns each $f \in \mathcal{S}]\}$. Steel [**47**] showed $\mathbf{Ex}^* \subseteq \mathbf{Bc}$. Bārzdiņš [**4**] first studied **Bc** and showed that $(\mathbf{Bc} - \mathbf{Ex}) \neq \emptyset$. Harrington and I [**47**] showed that $(\mathbf{Bc} - \mathbf{Ex}^*) \neq \emptyset$.[24] Anyhow, the *cost* mentioned above of **Bc**-learning is that to realize its full power one has to contend with the final, correct programs being of *un*bounded size.[25]

In this section, we have quite plausibly been taking a predictive scientific explanation to be modeled as a φ-program for predicting the results of all experiments regarding a phenomenon to be explained.[26] Essentially, in terms of the arithmetical hierarchy [**148**], φ-programs are intercompilable with Σ_1^0-definitions of the corresponding (partial) functions. In [**37, 10, 48, 49**] the learning or inductive inference of Σ_2^0-definitions is also considered. Of course, from such definitions one may not be able to extract predictions about the outcomes of associated experiments, *but* in *some* cases, some higher order information may be extractable. Even if, from such a definition, one cannot calculate values for an f so defined, one may be able to extract data *refutable* global or shape information about the curve of f, for example, that f is monotone increasing.[27]

[24] Our proof was an infinitary recursion theorem argument based on my ORT [**21, 24**].

[25] Anomalous variants of **Bc**-learning, for example, $\mathbf{Bc}^a$ for allowing up to a anomalies in final programs, and corresponding hierarchies of learning/inferring power are studied in [**47**]. Harrington showed [**47**] that some machine witnesses $\mathcal{R} \in \mathbf{Bc}^*$, and [**27**] shows that such machines on infinitely many computable functions have their anomalies occurring in undesirable positions.

[26] Fulk [**70**] argues 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 from a finite alphabet and/or bounded-size, finite-precision images.

[27] The difference is somewhat analogous to the difference between predicting the location of planet at any time and predicting the shape of the planet's orbit [**37, 10**].

The *limiting-computable partial functions* are those computable by a total mind-changing algorithm, i.e., those that are the limit of a computable function [**157**]. With care one can intercompile between Σ_2^0-definitions and some form of programs for the limiting-computable partial functions.[28] For this reason we write **LimEx** for the inference criterion just like **Ex** *except* that the "programs" output and converged to are Σ_2^0-definitions (or a suitable form of limiting programs). N.B. We are, of course, interested in **LimEx** for classes of *computable* functions only. We take φ^2 to be an acceptable programming system (numbering) for the Σ_2^0 partial functions, where φ_p^2 is the partial function defined by suitable form of limiting program p.

For expressions E admitting translation into the language of first order arithmetic [**113**] we write $\ll \mathsf{E} \gg$ for a fixed, natural such translation.

$\eta =^\infty \theta \overset{\text{def}}{\Leftrightarrow} \|\{x \mid \eta(x) = \theta(x)\}\|$ is infinite.

Theorem 3.5 (Case and Suraj [**48, 49**]). *Suppose* $\mathbf{T}$ *is a computably axiomatizable first order theory which extends Peano Arithmetic* (**PA**) *and in which one cannot prove sentences that are false in the standard model* ([**113**]). *Then there is a class of monotone computable functions* $\mathcal{C}$ *such that*

(1) $\mathcal{C} \notin \cup_{k\in N}\mathbf{Bc}^k$,

(2) $(\forall f \in \mathcal{C})(\forall p \mid \varphi_p =^\infty f)[\mathbf{T} \nvdash \ll \varphi_p \textit{ is monotone} \gg]$,[29] *and*

(3) *there exists a machine* $\mathbf{M}$ *which* **LimEx**-*learns every function in* $\mathcal{C}$ *and, for every* $f \in \mathcal{C}$, *for every* e *such that* $\mathbf{M}(f)\downarrow = e$,

  (a) $\mathbf{PA} \vdash \ll \varphi_e^2 \textit{ is monotone} \gg$,

  (b) $(\forall x, y)[\mathbf{T} \nvdash \ll \varphi_e^2(x) = y \gg]$[30], *and*

  (c) $\mathbf{PA} \vdash \ll \varphi_e^2 \textit{ is computable} \gg$.

[28] The trick is to express the limiting computable partial functions as the *uniform* limit of a single, suitable computable function [**143, 49**].

[29] Therefore, $(\forall f \in \mathcal{C})(\forall p \mid \varphi_p =^\infty f$ and φ_p is monotone $)[\mathbf{T} \nvdash \ll \varphi_p$ is monotone $\gg]$. This is, perhaps, surprisingly strong.

[30] Therefore, $(\forall x, y \mid \varphi_e^2(x) = y)[\mathbf{T} \nvdash \ll \varphi_e^2(x) = y \gg]$.

This theorem (Theorem 3.5), then, provides some strong tradeoffs between inferring Σ^0_1- vs. Σ^0_2-definitions. For $\mathcal{C}$, one can employ the latter, but *not* the former for successful inference *and* to prove monotonicity. But, by the important Clause 3b in Theorem 3.5 just above, one cannot predict, from the latter and $\mathbf{T}$, for the sake of Popper's Refutability Principle for science, any data points at all in the graphs of the φ^2_e's! The output Σ^0_2-definitions are not, in principle, refutable by incorrect data point predictions, but *in principle*, they admit of being refuted by non-monotonicity (in the input data itself). Hence this result exhibits, then, some new subtleties re the application of Popper's Refutability Principle: one cannot inductively infer $\mathcal{C} \subseteq \mathcal{R}$ without being forced to accept a weakened refutability principle.

I would like to see more CT learning theory theorems like those above with some insight and/or shock value for philosophy of science.

3.2.2. Cognitive science and language learning. In this section, we look at learning grammars for (formal) languages from positive information about them. The original paradigm was Gold's [**76**]. Thanks to code numbering and for mathematical convenience we can and will take our languages to be r.e. subsets of N. Grammars will be type 0 [**81**], and, hence, we can take a grammar g for an r.e. language L to be an r.e. index for L, i.e., such that $W_g = L$.

We say T is a *text* for $L \overset{\text{def}}{\Leftrightarrow} \{T(0), T(1), \ldots\} = L$.[31] We say, in this case, T is *for* L. In this section, $\mathbf{M}$s will computably map finite initial segments of texts into grammars/r.e. indices, and, without loss of generality for what we want to do, we take $\mathbf{M}$s to be total. Next are defined some criteria of successful language learning.

[31] In more formal expositions, we allow ρT to contain also #s, where a # models a *pause* and is not part of the language L. Then the text with successive values consisting only of #s is the only text for the empty language. Herein we need not be so careful about handling the empty language.

Definition 3.4 ([**42, 132, 25**]). Suppose $b \in (N^+ \cup \{*\})$, where $N^+ = \{1, 2, \ldots\}$ and $x \leqslant *$ means $x < \infty$.

(1) $\mathcal{L} \in \mathbf{TxtEx} \Leftrightarrow (\exists \mathbf{M})(\forall L \in \mathcal{L})(\forall T$ for $L)[\mathbf{M}$ on T outputs $g_0, g_1, g_2, \ldots \Rightarrow (\exists t)[g_t = g_{t+1} = \cdots \wedge W_{g_t} = L]]$.

(2) $\mathcal{L} \in \mathbf{TxtBc} \Leftrightarrow (\exists M)(\forall L \in \mathcal{L})(\forall T$ for $L)[\mathbf{M}$ on T outputs $g_0, g_1, g_2, \ldots \Rightarrow (\exists t)$ $[g_t, g_{t+1}, \ldots$ each generates/enumerates $L]]$.

(3) $\mathcal{L} \in \mathbf{TxtFex}_b \Leftrightarrow (\exists M)(\forall L \in \mathcal{L})(\forall T$ for $L)[\mathbf{M}$ on T outputs $g_0, g_1, g_2, \ldots \Rightarrow (\exists t)$ $[g_t, g_{t+1}, \ldots$ each generates/ enumerates $L \wedge \|\{g_t, g_{t+1}, \ldots\}\| \leqslant b]]$.

The class $\mathcal{F}$ of all finite languages $\in \mathbf{TxtEx}$, but the class of all regular languages (from automata theory [**81**]) is not [**76**]. $\mathcal{K} = \{K \cup \{x\} \mid x \in N\} \in (\mathbf{TxtBc} - \mathbf{TxtEx})$, where K is the diagonal halting problem [**148**]. $\mathbf{TxtFex}_b$ is like $\mathbf{TxtBc}$ except the set of final, correct programs has cardinality $\leqslant b$. $\mathbf{TxtFex}_1 = \mathbf{TxtEx}$ & $\mathcal{K} \notin \mathbf{TxtFex}_b$.

We have $\mathbf{TxtFex}_1 \subset \mathbf{TxtFex}_2 \subset \ldots \subset \mathbf{TxtFex}_* \subset \mathbf{TxtBc}$ (see [**25**]).[32] [33]

Some sample publications in computational learning theory re formal language learning are [**76, 3, 22, 131, 70, 71, 96, 8, 86, 25, 9, 29**].

The language learning model of the present section, although obviously limited as a model for human language learning, has, nonetheless, been influential in cognitive science and in contemporary theories of natural languages, for example, [**136, 181, 128, 182, 130, 11, 74, 94**].

Regarding this model, I gradually acquired the belief that, in spite of its limitations, there was the possibility for theorems with insights into cognitive science. In the rest of this section, I provide

[32] [**132**] showed $\mathbf{TxtFex}_1 \subset \mathbf{TxtFex}_* \subset \mathbf{TxtBc}$.

There are also anomaly hierarchies, but we will not go into them here. See [**25**].

[33] This hierarchy result contrasts with what happens with the criteria for learning programs in the limit for $f \in \mathcal{R}$ from Section 3.2.1 above: for **Ex** style learning, converging to finitely many correct programs in the limit offers no more learning power than converging to one. See [**47**].

an example.[34] The motivation comes from empirical observations from child cognitive development.

U-shaped learning behavior features the pattern of learning, unlearning, and relearning. It occurs in child development re, for example, verb regularization [**139, 119, 166**] and understanding of various (Piaget-like) conservation principles [**162**], for example, temperature and weight conservation and interaction between object tracking and object permanence. An example regarding irregular verbs in English follows. A child first uses *spoke*, the correct past tense of the *ir*regular verb *to speak*. Then the child overregularizes *in*correctly using *speaked*. Lastly, the child returns to using *spoke*. Our theoretical examples will involve the formal learning, unlearning, and relearning of type 0 grammars for whole formal languages L. The main "theoretical" concern of the empirically based cognitive science literature on U-shaped learning is with how to model U-shaped learning. For example, is U-shaped language learning done employing subconscious general rules vs. tables of exceptions [**14**]? That is a nice concern, but not at all my interest. My interest is in the following question. Is U-shaped learning an *un*necessary and harmless accident of human evolution *or* is U-shaped learning advantageous in that some *classes* of tasks *can* be learned in U-shaped way, but *not* otherwise? I.e., are some classes of tasks learnable only by returning to *abandoned* correct, learnable behaviors? Of course, as a question about humans, this is very difficult to answer. So, I sought some learning theory insights about what could *possibly* be true.

[34] The proof techniques for learning language grammars from positive data, unlike the results in the just previous section (Section 3.2.1) feature more than considerations of algorithmicity. They also feature finite extension arguments which, of course, can be conceptualized in terms of Baire Category Theory [**122, 148, 83, 129, 131**]. Some of the proofs in [**25**] employ such a mixture but resembling finite injury priority arguments. These are to obtain results for $\mathbf{TxtFex}_b$ having to do with without-loss-of-generality local and global *in*sensitivity to order of data presentation and whether the texts are restricted to being computable.

Next is the definition of language learning criteria which are restricted by *dis*allowing U-shaped learning behavior. We think of W_g as the [summary of the] *behavior* of g.

Definition 3.5. Suppose $\mathbf{C} \in \{\mathbf{TxtFex}_b, \mathbf{TxtBc}\}$. Then, $\mathcal{L} \in \mathbf{NonUC} \Leftrightarrow (\exists M)(\forall L \in \mathcal{L})(\forall T \text{ for } L)[\mathbf{M} \text{ on } T \text{ outputs } g_0, g_1, g_2, \ldots \Rightarrow (\forall i, j, k \mid i < j < k)[W_{g_i} = W_{g_k} = L \Rightarrow W_{g_j} = W_{g_i}]]$.

Non-U-shaped learners never abandon *correct behaviors* for learned $L \in \mathcal{L}$ and, then, return to those behaviors.

From [**29**], the transitive closure of the inclusions (denoted by $\longrightarrow$) in Fig. 1 holds *and* no other inclusions hold.

$$\begin{tikzcd}
\mathbf{TxtFex}_* \arrow[r] & \mathbf{TxtBc} \\
\mathbf{TxtFex}_3 \arrow[u, dotted] & \\
\mathbf{TxtFex}_2 \arrow[u] \arrow[r] & \mathbf{NonUTxtBc} \arrow[uu] \\
\begin{array}{l}\mathbf{NonUTxtEx}\\ =\mathbf{TxtEx}\\ =\mathbf{NonUTxtFex}_b\end{array} \arrow[u] \arrow[ur] &
\end{tikzcd}$$

FIGURE 1. Results on U-Shaped Learning

Hence U-shaped learning *is* needed for some class in **TxtBc**; is *not* for **TxtEx** learning, i.e., for learning *one* successful grammar in the limit; *is* needed for some class in $\mathbf{TxtFex}_2$ even if we allow finitely many grammars in the limit — but *not* if we allow infinitely many grammars in the limit; and *is* needed for some $\mathcal{L} \in \mathbf{TxtFex}_3$ even if we allow infinitely many grammars in the limit.

Now that we know some mathematical *possibilities*, a question for the cognitive scientists is: does the class of tasks humans must learn to be competitive in the genetic marketplace, like this latter $\mathcal{L}$, *necessitate* U-shaped learning?

I would like to see more CT learning theory results like the above which give cognitive science something new to think about.

3.2.3. Applied machine learning. In the context of dealing with the difficulties of actually applying learning in robotics, Drew McDermott [**110**] says, "Learning makes the most sense when it is thought of as filling in the details in an algorithm that is already nearly right." I suggested to colleagues that we get some corresponding learning theory results regarding learning programs for functions from approximately correct such programs (as well as from data on the functions). Martin Kummer came up with several nice ideas for such approximate programs for a computable 0-1 valued function — including decision programs for *bounded width* trees [**148**] containing or enveloping the function, and we produced [**41**]. A sample result from this paper implies that if the approximately correct programs are for enveloping trees of width $n > 0$, then some probabilistic machine (in the sense of [**137, 138**]) **Ex**-learns every 0-1 valued computable function with probability of success $\frac{1}{n}$. For **Bc** the probability is one.

In the late 90s, I started attending applied machine learning conferences and workshops. Early on I noticed practical interest in so-called *concept drift* and *context sensitive learning.*

A drifting concept to be learned is one which is a moving target. See, for example, [**6, 7, 61, 67, 79, 103, 186**]. I got some computability theory collaborators together to produce [**34**] in which we show, for various learnability criteria (including some, suggested by Frank Stephan, for learning Martingale betting strategies), bounds on the speed of the moving target that permit success at all.

Context sensitive learning involves trying to learn Y by first [**178, 179, 167, 56, 57, 58, 62, 174, 169**] *or* simultaneously [**18, 19, 116, 12, 59, 111, 140, 161**] trying to learn also X — even in cases where there may be no inherent interest in learning X. There is, in many cases, an empirical advantage in doing this for some X, Y. It can happen that Y is not learnable by itself, but is learnable if one learns X first or simultaneously. For example, to teach a robot to drive a car, it is useful to train it also to predict the center of the road markings (see, for example, [**15, 19**]). I realized there was already a CT learning theory paper that I liked very much, [**2**], which showed mathematically these context sensitivity phenomenon *must* happen for *some* tasks X, Y. Later we produced [**36**] providing a kind of strengthening for the case one learns X, Y simultaneously.[35]

These results regarding context sensitive learning provide mathematical support for the corresponding empirical phenomena suggesting the *possibility* that these empirical phenomena are not just accidental or illusory.

I would like to see more of these kinds of CT learning theory papers.

Next is an interesting four part story.

Part I of the four part story. In my visits to the School of Computer Science and Engineering, University of New South Wales, Sydney, Australia, I have learned about the machine learning projects of Claude Sammut there. I became particularly interested in the *behavioral cloning* approach to machine learning of reactive process-control. This is surveyed in [**16**] and involves using data from the (non-verbal, performance) behavior of master or expert human controllers in order to make machine learning

[35] Of course machine learning is an engineering endeavor. However, philosophers of science as well as practitioners in scientific disciplines should, I believe, be considering their relevance to their endeavors.

of complex control feasible/possible. For example, it has been used successfully to teach an autopilot to fly an aircraft simulator [**16, 158, 120, 151, 152**] and to teach a machine to operate efficiently a (simulated) free-swinging shipyard crane [**16, 175**].

One of the difficulties Claude made me aware of in the learning-to-fly project was that attempts to make use of the behavioral data from more than one human expert at a time had failed miserably. Different pilots had very different strategies, and it was not clear how to mix them.

Part II of the four part story. In a visit to Martin Kummer he put me onto his *theoretical* work on learning, from programs for game trees, etc., winning strategies for infinite reactive process-control games called *closed computable games* [**98**]. I would not provide here the details but will give the computability-theoretical flavor of these games with two contrasting examples.[36]

Example 3.1. Fix $n_0 \in N$. Player I is a digital thermostat, Player II is the temperature (which is subject to a discrete unseen physical disturbance); *winning* for Player I is: past time (= move) n_0 keeping the temperature within some pre-assigned integer bounds.

Example 3.1 *is* a closed computable game. Importantly, Player I *can* algorithmically detect if he/she/it has, at any point, lost.

Example 3.2. Player I is a digital thermostat, Player II is the temperature (which is subject to a discrete, unseen physical disturbance); *winning* for Player I is: past *some* time (= move) n keeping the temperature within some pre-assigned integer bounds.

Example 3.2 is *not* a closed computable game. Importantly, Player I can *not* algorithmically detect if he/she/it has, at any point, lost.

[36] For more on these games, also see [**45, 118, 168**].

Of course the behavioral cloning games in Part I are not infinite, but there is otherwise some suggestive similarity with the closed computable games.

Kummer's co-author, Matthias Ott, had some ideas already for adding the *behavior* of masters playing winning strategies as additional information for the learning of closed computable games. This looked like behavioral cloning from Part I of the story! We produced [**44**], and one of the theorems there said there *existed* cases where cloning $n + 1$ disparate masters enable learning to win more games than merely cloning n. This was theoretical support, then, for the *possibility* that, in the behavioral cloning experiments, there could be a way to clone behaviorally multiple masters or experts — and with some performance advantage over merely cloning one master.

Part III of the four part story. I went to an applied machine learning workshop, and told participants who cared about behavioral cloning about the just above result that there are cases for which cloning more experts is better than cloning fewer. I am not sure if I expected them to say, in effect, Oh, good, I will go home and figure out how to apply that to my behavioral cloning problems. Instead they asked me how to do it for practical problems. Our existence theorem had not provided me just how to do it. I did try after that to get Sammut's group to see what we could do, but I was never around them long enough to get much work done on it.

Part IV of the four part story. Some time later I found out, from Mike Bain in Sammut's group, about Dorian Šuc's wonderful doctoral dissertation in Ljubljana, Slovenia, [**177**]. He had found a way to clone behaviorally more than one human expert simultaneously for the free-swinging shipyard crane problem — by having more than one level of feedback control, *and* he got enhanced performance from cloning the multiple experts! Dorian had not known anything about our suggestive theoretical result, he just solved the problem.

What I would like to see: get more CT learning results which *should* inform machine learning practitioners.

3.3. Machine self-reflection

This paragraph is based mostly on [**24**]. Kleene's (Second) Recursion Theorem can be conceptualized as follows. Given any pre-assigned algorithmic task, there is a φ-program e which first looks in a mirror[37] to see in detail and exactitude its own code script, flow chart, or wiring diagram, and, then, e uses this image in the mirror as a datum (and its external input as another datum) for input to the pre-assigned algorithmic task — which task it then carries out with these two inputs. Essentially, then, e has a perfect self-model (a copy of itself) and employs it according to the algorithmic pre-assigned task which describes how to use it (together with its external input). No infinite regress is required since e's copy is projected *external to e*. Such e are self-reflecting/self-knowing programs.[38] [39]

In the late 70s, I realized that the constructive form of Kleene's Recursion Theorem (I will call it **KRT**) could be conceptualized as a kind of *non*-denotational program control structure [**163**]. Typical *denotational* control structures are **if–then–else** and **while–loop**. I believed it would be possible to develop a general theory of control structures in the context of CT-style programming systems (numberings). It was. I supervised the doctoral dissertations

[37] We can suppose the mirror is a corner mirror so the image in it does not appear left-right reversed.

[38] Examples of using self-knowledge in a simple way were presented in the proof of Theorem 3.1 in Section 3.2.1 above. Examples of using self-knowledge in more complex ways are in [**21**].

[39] I intend to write the paper version of [**26**] in which I describe what, I believe, Kleene's Recursion Theorem has to do with the self-reflection component of consciousness. N.B. I will not provide an elucidation of what Dave Chalmers in his very influential book [**31**] describes as the hard problem of consciousness, for example, the problem of qualia. I will provide some ideas on the problems of why we are not unconscious zombies [**31**] and how we can be machines and, yet, differ in kind from Searle's famous Chinese Room [**155**].

[**144, 149**] to help work this out.[40] In the context of programming systems (numberings) for the class of partially computable functions where each system has a universal program inside the system, I showed that the acceptable programming systems [**147, 148**], are characterized as those in which each possible control structure has an implementation [**144, 149**]. One of my principal goals in all this was to try to characterize **KRT** insightfully — in the interest of understanding the utility of self-knowledge. The ancient Greeks thought self-knowledge was important, and, perhaps, one could obtain some mathematical insight into its utility. Characterizations have been elusive, but we have had better luck at insight into what epitomizes the "complement" of **KRT**. Here is one of my favorite theorems of Jim Royer on this latter subject. Again, the programming systems (numberings) considered are for the class of partially computable functions where each has a universal program inside the system.

Theorem 3.6 (Royer [**149**]). ***KRT*** *and* ***if–then–else*** *are complementary in the sense that*:

(1) *For each there is a programming system with an implementation of that one but with no implementation of the other one; and*
(2) *If a programming system has an implementation of both, it is acceptable; i.e., has an implementation of all control structures.*

Hence decision branching and self-reflection are complementary.

I noticed, from the proofs of this theorem (Theorem 3.6) and related ones in [**149**], that one of the crucial elements was the constructivity component of **KRT**, but I wanted to understand the self-knowledge component, period. Let **krt** be the not necessarily constructive Kleene Recursion Theorem. With a new Ph.D. student, Sam Moelius, we have begun to find epitomizers of the

[40] For definitions, etc., see [**144, 145, 149**]. For more on this CT approach to control structures, see [**146, 108, 85, 39**].

complement of **krt**. This is work not yet completed. We will see how it goes.

I would like to see more CT work on mathematically understanding machine self-knowledge.

3.4. CT for computational complexity

In this section, we explore a tiny fraction of the available and somewhat recent literature. I like very much, though, the early results of abstract complexity theory such as the surprising Blum Speed-Up Theorem [**13, 187**], its strengthening [**114**], and [**117**].[41] Many more recent results in complexity theory involve limiting some CT techniques to severely time or space bounded realms. See, for example, [**143**] and its bibliography. Actually, in co-creating [**143**] I had in mind bringing CT techniques far down into the subrecursive realm, for example, all the way down to linear time computable. Of course, extremely complicated priority arguments or even finite injury priority arguments with no computable bound on the injuries do not seem to fit well this realm.[42] Priority constructions with bounded finite injury *can* sometimes be used to get complexity theory results, e.g, in [**97**] at the cost of exponential time. Impressively, [**30**] applies carefully bounded priorities toward feasible learnability. Employing CT tricks to provide the practitioner with feasible algorithms, while very difficult, would be highly desirable for the future.

[**28**] presents learnability applications of CT to prove results about the *quality* of the final learned programs. Below is a special case of one of the results. Suppose $k > 0$. Run times are measured

[41] [**127**] surveys much of this work.

Proofs of such results by complicated Kleene Recursion Theorem arguments can be conceptually simplified by employing instead my ORT. See [**109, 160**] for examples of how I do this.

[42] Possibly, these kinds of arguments could be introduced into this realm by employing Hybrid recursion theorems from [**143**]. These permit, for *example*, self-other reference between low level subrecursive programming systems and systems for functions partial computable in K.

with respect to multi-tape Turing machines, and we suppose φ^{TM} is an acceptable system based on them — with Φ_p^{TM} the run time (partial) function of φ^{TM}-program p [**13**]. Let $\mathcal{P}^k \overset{\mathrm{def}}{=}$ the set of characteristic functions of sets decidable in k-degree polynomial time (in the length of inputs). Pick an inverse α to Ackermann's function computable in linear time — of course α is very slow growing [**43**]. Let $\mathcal{Q}^k \overset{\mathrm{def}}{=}$ the set of characteristic functions of sets decidable in time a k-degree polynomial of n times $\log(n)$ times $\alpha(n)$, where n is the length of the input. $\mathcal{P}^k \subset \mathcal{Q}^k$ [**80, 81**], and this is a tightest known separation. Since each of $\mathcal{P}^k$ and $\mathcal{Q}^k$ are r.e. classes of computable functions, by the enumeration technique in [**5**], they are **Ex**-learnable. For example, then, $\mathcal{P}^k$ is so learnable by a machine all of whose output conjectures run in k-degree polynomial time.

Theorem 3.7 (Case, Chen, Jain, Merkle, and Royer [**28**]). *Suppose* **M Ex**-*identifies* $\mathcal{Q}^k$, *where* $k \geqslant 1$. *Then there is an "easy" f, the characteristic function of some finite set, such that* $(\forall a)(\overset{\infty}{\forall} x)[\Phi^{\mathrm{TM}}_{\mathbf{M}(f)}(x) > a \cdot (|x|+1)^k]$.

Hence, to learn $\mathcal{Q}^k$, a little bigger class than $\mathcal{P}^k$, we have severe complexity deficiencies in the final programs on very easy functions f.

Theorem 3.7 just above is proved by delayed diagonalization (or slowed simulation) [**104, 143**] with cancellation [**13**] (or zero injury), complexity-bounded self-reference [**143**], and very careful subrecursive programming [**143**].

In [**28**], we have other results of this ilk. For example, *if* the classes polynomial time and non-deterministic polynomial time *do* separate, then **Ex**-learning the latter with output conjectures non-deterministic polynomial time bounded Turing machines will force there to exist some easy functions f (characteristic functions of finite sets) whose final learned programs will have some otherwise unnecessary and undesirable non-determinism. Also obtained is a similar result comparing quantum polyomial time and polynomial time (again, *if* they separate), where, in learning the then larger class, the complexity deficiency in final programs for some easy functions is otherwise unnecessary quantum parallelism. Standard diagonalizations are too rough to be used in these realms where we are not even sure currently if there are separations. We resort instead to lifts of arguments about more delicate Σ_2^0-*in*separability of certain subrecursive index sets [**23, 143**].

In [**28**], there are additionally results about cases where final programs *are* asymptotically optimal, but they are *informationally deficient*: one cannot prove about them even suboptimal run time bounds.

The lesson for the practioner of such results from [**28**] is: do not try to learn too much (if you do not have to); else, you may get undesirable learned programs.

I would like to see more CT results in complexity theory with even more remarkable advice to the practitioner.

Jim Royer has been working for some time on a program to bridge between European theoretical computer scientists who seek to understand higher types in programming languages, but who generally ignore even issues of algorithmicity and U.S. theoretical computer scientists interested only in feasible algorithms. For example, [**150**] presents an analog of the Kreisel-Lacombe-Shoenfield Theorem [**148**] for *feasible* type-2 functionals [**112, 87, 88, 156, 82**].

I would like to see more of this level attempt to provide some *eventual* advice to the practitioner, for example, to the designer of elegant, new programming languages.

3.5. Physics and all the rest

Kreisel has written about the problem of whether the physical world permits calculations beyond the Turing-computable, for example, [**99, 100**]. See [**126**] for nice discussion of the issues. *Hypercomputation* involves allowing infinitely many computation steps in finite time.[43] The problem is whether in our universe such computations are executable. Norman Margolus at MIT whose background includes both physics and computer science explained to me a few years ago that such computations would require an unlimited supply of energy. See also [**52, 53, 54, 55, 64**] for further arguments that this sort of computation is not available in the real world.

Along different lines, we see, though, the impressive and surprising work of Pour-El and Richards [**133, 134, 135**]. In [**134**] they provide a (higher type) *un*computable solution to the wave equation with a (higher type) *computable* boundary condition![44]

On the other hand: when I first studied Maxwell's Equations as an undergraduate I noticed that they were applied beautifully and elegantly to clouds of electrons. Problem: the clouds are discrete, yet the mathematics is essentially continuous. Of course, it is too hard for practical purposes to model a large cloud of electrons discretely, and the continuous mathematics nicely smooths out the discretness and provides good enough experimental predictions. My reaction, though, was disillusionment. I naively expected physicists to seek absolute knowledge and at least to apologize for not providing it. Of course, they do not care about such matters. As you may note from some of the things I wrote about above, for example, in Section 3.2.2, I no longer expect absolute knowledge.[45]

So, then, *is* at least *some* of physical reality *absolutely* modeled by continuous mathematics involving real numbers? Perhaps all but physical *space* is discrete? [**78**, pp. 164–165] argues that there must exist as a universal constant in nature a smallest length. It may be that the universe, *including space*, is discrete. Researchers in the cellular automata approach to physics

---

[43] A recursive iteration of the idea would lead to Kreisel's $\aleph_0$-mind computability (characterizing the Π^1_1-computable partial functions) [**148**].

[44] For a different perspective on this work, see [**180**].

[45] But, anyway, let me at least apologize for not providing it.

(see [**63, 115, 69, 173, 170, 107, 165, 106, 172, 171, 176, 185, 164, 65, 77**])[46] take this idea seriously.

So regarding the work referenced above by Pour-El and Richards, while I admire this work very much, I believe one has to be careful about work on physics *equations* which may be only wonderfully convenient continuous approximations to various discrete realities. The resultant work will not really be about *physics*.

So, I am left with not so many examples of prior applications of CT to physics I would like to see more of in the future. There was at least the quantum computing example in Section 3.4 above.

Anyhow, I would like to see future applications of CT with insights and/or advice to physics (and all the science and engineering disciplines for which I have provided no example prior applications of CT).

References

1. A. Ambainis, J. Case, S. Jain, and M. Surajm, *Parsimony hierarchies for inductive inference*, J. Symb. Log. **69** (2004), 287–328.
2. D. Angluin, W. Gasarch, and C. Smith, *Training sequences*, Theor. Comput. Sci. **66** (1989), no. 3, 255–272.
3. D. Angluin, *Inductive inference of formal languages from positive data*, Inf. Control **45** (1980), 117–135.
4. J. Bārzdiņš, *Two theorems on the limiting synthesis of functions* (in Russian), Theory of Algorithms and Programs, Riga, Latvian State Univ. **210** (1974), 82–88.
5. L. Blum and M. Blum, *Toward a mathematical theory of inductive inference*, Inf. Control **28** (1975), 125–155.
6. P. Bartlett, S. Ben-David, and S. Kulkarni, *Learning changing concepts by exploiting the structure of change*, In: Proceedings of the Ninth Annual Conference on Computational Learning Theory, ACM Press, 1996, pp. 131–139.

[46] Here [**63**] is crucial, and [**115**] lays out the ideas of Ed Fredkin on some of the ways physical space might be discrete.

7. A. Blum and P. Chalasani, *Learning switching concepts*, In: Proceedings of the Fifth Annual Conference on Computational Learning Theory, ACM Press, 1992, pp. 231–242,

8. G. Baliga, J. Case, and S. Jain, *Language learning with some negative information*, J. Comput. Syst. Sci. **51** (1995), 273–285.

9. G. Baliga, J. Case, and S. Jain, *The synthesis of language learners*, Inf. Comput. **152** (1999), no. 1, 16–43.

10. G. Baliga, J. Case, S. Jain, and M. Suraj, *Machine learning of higher order programs*, J. Symb. Log. **59** (1994), no. 2, 486–500.

11. R. Berwick, *The Acquisition of Syntactic Knowledge*, The MIT Press, 1985.

12. K. Bartlmae, S. Gutjahr, and G. Nakhaeizadeh, *Incorporating prior knowledge about financial markets through neural multitask learning*, In: Proceedings of the Fifth International Conference on Neural Networks in the Capital Markets, 1997.

13. M. Blum, *A machine independent theory of the complexity of recursive functions*, J. Assoc. Comput. Mach. **14** (1967), 322–336.

14. M. Bowerman, *Starting to talk worse: Clues to language acquisition from children's late speech errors*, In: U-Shaped Behavioral Growth, S. Strauss and R. Stavy (Eds.), Academic Press, 1982.

15. S. Baluja and D. Pomerleau, *Using the representation in a neural network's hidden layer for task-specific focus of attention*, Technical Report CMU-CS-95-143, School of Computer Science, CMU, May 1995. [To appear in Proceedings of the 1995 IJCAI]

16. M. Bain and C. Sammut, *A framework for behavioural cloning*, In: Machine Intelligence 15, Intelligent Agents, K. Furakawa S. Muggleton, and D. Michie (Eds.), Oxford Univ. Press, 1999, pp. 103–129.

17. A. W. Burks (Ed.), *Essays on Cellular Automata*, Univ. Illinois Press, 1970.

18. R. A. Caruana, *Multitask connectionist learning*, In: Proceedings of the 1993 Connectionist Models Summer School, pp. 372–379.

19. R. A. Caruana, *Algorithms and applications for multitask learning*, In: Proceedings of the 13th International Conference on Machine Learning, 1996, pp. 87–95.

20. J. Case, *A note on the degrees of self-describing Turing machines*, J. Assoc. Comput. Mach. **18** (1971), 329–338.

21. J. Case, *Periodicity in generations of automata*, Math. Syst. Theory **8** (1974), 15–32.
22. J. Case, *Learning machines*, In: Language Learning and Concept Acquisition, W. Demopoulos and A. Marras (Eds.), Ablex Publishing Company, 1986.
23. J. Case, *Effectivizing inseparability*, Z. Math. Logik Grundlagen Math. **37** (1991), no. 2, 97–111. [http://www.cis.udel.edu/∼case/papers/mkdelta.pdf corrects missing set complement signs in definitions in the journal version]
24. J. Case, *Infinitary self-reference in learning theory*, J. Exp. Theor. Artif. Intell. **6** (1994), no. 1, 3–16.
25. J. Case, *The power of vacillation in language learning*, SIAM J. Comput. **28** (1999), no. 6, 1941–1969.
26. J. Case, *Machine self-reference and consciousness*, In: Proceedings and Abstracts of the Third Annual Meeting of the Association for the Scientific Study of Consciousness, London, Ontario, 1999. [http://www.cis.udel.edu/∼case/slides/krt-consc-slides.pdf]
27. J. Case, K. Chen, and S. Jain, *Costs of general purpose learning*, Theor. Comput. Sci. **259** (2001), no. 1-2, 455–473.
28. J. Case, K. Chen, S. Jain, W. Merkle, and J. Royer, *Generality's price: Inescapable deficiencies in machine-learned programs*, Ann. Pure Appl. Logic **139** (2006), no. 1-3, 303–326.
29. L. Carlucci, J. Case, S. Jain, and F. Stephan, *Non U-shaped vacillatory and team learning*, In: Algorithmic Learning Theory: 16th International Conference, ALT 2005, Singapore, October 8-11, 2005. Proceedings, S. Jain, H. U. Simon, and E. Tomita (Eds.), Lect. Notes Comput. Sci. **3734**, Springer, 2005,
30. Z. Chen and S. Homer, *The bounded injury priority method and the learnability of unions of rectangles*, Ann. Pure Appl. Logic **77** (1996), no. 2, 143–168.
31. D. Chalmers, *The Conscious Mind: In Search of a Fundamental Theory*, Oxford, Oxford University Press, 1996.
32. K. Chen, *Tradeoffs in Machine Inductive Inference*, PhD Thesis, Computer Science Department, SUNY at Buffalo, 1981.
33. K. Chen, *Tradeoffs in the inductive inference of nearly minimal size programs*, Inf. Control **52** (1982), 68–86.

34. J. Case, S. Jain, S. Kaufmann, A. Sharma, and F. Stephan, *Predictive learning models for concept drift*, Theor. Comput. Sci. **268** (2001), no. 2, 323–349.

35. J. Case, S. Jain, and S. Ngo Manguelle, *Refinements of inductive inference by Popperian and reliable machines*, Kybernetika **30** (1994), no. 1, 23–52.

36. J. Case, S. Jain, M. Ott, A. Sharma, and F. Stephan, *Robust learning aided by context*, J. Comput. Syst. Sci. **60** (2000), 234–257.

37. J. Case, S. Jain, and A. Sharma, *On learning limiting programs*, Int. J. Found. Comput. Sci. **3** (1992), no. 1, 93–115.

38. J. Case, S. Jain, and A. Sharma, *Machine induction without revolutionary changes in hypothesis size*, Inf. Comput. **128** (1996), no. 2, 73–86.

39. J. Case, S. Jain, and M. Suraj, *Control structures in hypothesis spaces: The influence on learning*, Theor. Comput. Sci. **270** (2002), no. 1-2, 287–308.

40. J. Case, S. Jain, F. Stephan, and R. Wiehagen, *Robust learning – rich and poor*, J. Comput. Syst. Sci. **69** (2004), 123–165.

41. J. Case, S. Kaufmann, E. Kinber, and M. Kummer, *Learning recursive functions from approximations*, J. Comput. Syst. Sci. **55** (1997), 183–196.

42. J. Case and C. Lynes, *Machine inductive inference and language identification*, In: Automata, Languages and Programming: Ninth Colloquium Aarhus, Denmark, July 12-16, 1982, M. Nielsen and E. M. Schmidt (Eds.), Lect. Notes Comput. Sci. **140** Springer, 1982, pp. 107–115.

43. T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, The MIT Press, 2001.

44. J. Case, M. Ott, A. Sharma, and F. Stephan, *Learning to win process-control games watching game-masters*, Inf. Comput. **174** (2002), no. 1, 1–19.

45. D. Cenzer and J. Remmel, *Recursively presented games and strategies*, Math. Soc. Sci. **24** (1992), no. 2-3, 117–139.

46. J. Case and C. Smith, *Anomaly hierarchies of mechanized inductive inference*, In: Conference Record of the Tenth Annual ACM Symposium on Theory of Computing, San Diego, California, 1-3 May 1978, pp. 314–319.

47. J. Case and C. Smith, *Comparison of identification criteria for machine inductive inference*, Theor. Comput. Sci. **25** (1983), 193–220.
48. J. Case and M. Suraj, *Inductive inference of* Σ_1^0*-vs.* Σ_2^0*-definitions for computable functions*, In: Proceedings of the International Conference on Mathematical Logic, Novosibirsk, Russia, 1999.
49. J. Case and M. Suraj, *Weakened refutability for machine learning of higher order definitions* 2006. [Working paper for eventual journal submission]
50. M. Davis, *Is mathematical insight algorithmic?* Behav. Brain. Sci. **3** (1990), 659–660.
51. M. Davis, *How subtle is Gödel's theorem? More on Roger Penrose*m Behav. Brain. Sci. **16** (1993), 611–612.
52. M. Davis, *The myth of hypercomputation* In: Alan Turing: Life and Legacy of a Great Thinker, C. Teuscher (Ed.), Springer, 2004, pp. 195–212.
53. M. Davis, *Computability, computation and the real world*, In: Imagination and Rigor: Essays on Eduardo R. Caieniello's Scientific Heritage, S. Termini (Ed.), Springer, 2005, pp. 63–70.
54. M. Davis, *Why there is no such subject as hypercomputation*, Appl. Math. Comput., 2006. [To appear]
55. M. Davis, *The Church-Turing thesis: Consensus and opposition*, In: Proceedings cCiE 2006, Springer Notes on Computer Science, Swansee, July 2006.
56. H. de Garis, *Genetic programming: Building nanobrains with genetically programmed neural network modules*, In: IJCNN: International Joint Conference on Neural Networks, Vol. 3, IEEE Service Center, Piscataway, New Jersey, June 17–21, 1990, pp. 511–516.
57. H. de Garis, *Genetic programming: Modular neural evolution for Darwin machines*, In: International Joint Conference on Neural Networks, Vol. 1, M. Caudill (Ed.), Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey, January 1990. pp. 194–197.
58. H. de Garis, *Genetic programming: Building artificial nervous systems with genetically programmed neural network modules*, In: Neural and Intelligenct Systems Integeration: Fifth and Sixth Generation Integerated Reasoning Information Systems, B. Souček and The IRIS Group (Eds.), John Wiley and Sons, 1991, Chapt. 8, pp. 207–234.

59. T. G. Dietterich, H. Hild, and G. Bakiri, *A comparison of ID3 and backpropogation for English text-to-speech mapping*, Mach. Learn. **18** (1995), no. 1, 51–80.

60. K. deLeeuw, E. Moore, C. Shannon, and N. Shapiro, *Computability by probabilistic machines*, Automata Studies, Ann. Math. Studies **34** (1956), 183–212.

61. M. Devaney and A. Ram, *Dynamically adjusting concepts to accommodate changing contexts*, In: Proceedings of the ICML-96 Pre-Conference Workshop on Learning in Context-Sensitive Domains, Bari, Italy, M. Kubat and G. Widmer (Eds.), 1994. [Journal submission]

62. S. Fahlman, *The recurrent cascade-correlation architecture*, In: Advances in Neural Information Processing Systems 3, R. Lippmann, J. Moody, and D. Touretzky (Eds.), Morgan Kaufmann, 1991, pp. 190–196.

63. R. Feynman, *Simulating physics with computers*, Int. J. Theor. Phys. **21** (1982), no. 6/7.

64. R. Feynman, *Feynman Lectures on Computation*, A. Hey and R. Allen (Eds.), Perseus Books, 2000.

65. U. Frisch, B. Hasslacher, and Y. Pomeau, *Lattice-gas automata for the Navier Stokes equation*, Phys. Rev. Letters **56** (1986), no. 14, 1505–1508.

66. M. Fulk and S. Jain, *Approximate inference and scientific method*, Inf. Comput. **114** (1994), no. 2, 179–191.

67. Y. Freund and Y. Mansour, *Learning under persistent drift*, In: Proceedings of the Third European Conference on Computational Learning Theory (EuroCOLT'97), S. Ben-David (Ed.), Lect. Notes Artif. Intell. **1208**, Springer, 1997, pp. 94–108.

68. R. Freivalds, *Minimal Gödel numbers and their identification in the limit*, In: Mathematical Foundations of Computer Science 1975 4th Symposium, Marianske Lazne, September 1-5, 1975, J. Becvar (Ed.), Lect. Notes Comput. Sci. **32**, Springer, 1975, pp. 219–225.

69. E, Fredkin and T. Toffoli, *Conservative logic*, Int. J. Theor. Phys. **21** (1982), no. 3/4.

70. M. Fulk, *A Study of Inductive Inference Machines*, PhD Thesis, SUNY at Buffalo, 1985.

71. M. Fulk, *Prudence and other conditions on formal language learning*, Inf. Comput. **85** (1990), no. 1, 1–11.

72. J. Gill, *Probabilistic Turing Machines and Complexity of Computation*, PhD Thesis, University of California, Berkeley, 1972.

73. J. Gill, *Computational complexity of probabilistic Turing machines*, SIAM J. Comput. **6** (1977), 675–695.

74. L. Gleitman, *Biological dispositions to learn language*, In: Language Learning and Concept Acquisition, W. Demopoulos and A. Marras (Eds.), Ablex Publ. Co., 1986.

75. C. Glymour, *Inductive inference in the limit*, Erkenntnis, **22** (1985), 23–31.

76. E. Gold, *Language identification in the limit*, Inf. Control **10** (1967), 447–474.

77. B. Hasslacher, *Discrete fluids*, Los Alamos Sci. **15)** (1987), 175–217.

78. W. Heisenberg, *Physics and Philosophy*, Harper and Brothers Publishers, 1958.

79. D. Helmbold and P. Long, *Tracking drifting concepts by minimizing disagreements*, Mach. Learn. **14** (1994), no. 1, 27–45.

80. J. Hartmanis and R. Stearns, *On the computational complexity of algorithms*, Trans. Am. Math. Soc. **117** (1965), 285–306.

81. J. Hopcroft and J. Ullman, *Introduction to Automata Theory Languages and Computation*, Addison-Wesley, 1979.

82. R. Irwin, B. Kapron, and J. Royer, *On characterizations of the basic feasible functional (Part I)*, J. Funct. Program. **11** (2001), no. 1, 117–153.

83. T. Jech, *Set Theory*, Academic Press, 1978.

84. N. Jessop, *Biosphere: A Study of Life*, Prentice-Hall, 1989.

85. S. Jain and J. Nessel, *Some independence results for control structures in complete numberings*, J. Symb. Log. **66** (2001), no. 1, 357–382.

86. S. Jain, D. Osherson, J. Royer, and A. Sharma, *Systems that Learn: An Introduction to Learning Theory*, The MIT Press, 1999.

87. B. Kapron and S. Cook, *A new characterization of Mehlhorn's polynomial time functionals*, In: Proceedings of the 32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991. IEEE Computer Society 1991, pp. 342–347.

88. B. Kapron and S. Cook, *A new characterization of type-2 feasibility*, SIAM J. Comput. **25** (1996), no. 1, 117–132.

89. K. Kelly, *The Logic of Reliable Inquiry*, Oxford Univ. Press, 1996.

90. K. Kelly, *The logic of success*, Br. J. Philos. Sci. **51** (2001), 639–666.

91. K. Kelly and C. Glymour, *Convergence to the truth and nothing but the truth*, Philos. Sci. **56** (1989), 185–220.

92. K. Kelly and C. Glymour, *Theory discovery from data with mixed quantifiers*, J. Philos. Logic **19** (1990), no. 1, 1–33.

93. E. Kinber, *On a theory of inductive inference*, In: Fundamentals of Computation Theory: Proceedings of the 1977 International FCT-Conference, Poznan-Kornik, Poland September 19-23, 1977, M. Karpinski (Ed.), Lect. Notes Comput. Sci. **56**, Springer, 1977, pp. 435–440.

94. D. Kirsh, *PDP learnability and innate knowledge of language*, In: Connectionis: Theory and Practice, S. Davis (Ed.), Oxford Univ. Press, 1992, pp. 297–322.

95. S. Kleene, *Origins of recursive function theory*, Ann. Hist. Comput. **3** (1981), no. 1, 52–67.

96. S. Kapur, B. Lust, W. Harbert, and G. Martohardjono, *Universal grammar and learnability theory: The case of binding domains and the 'subset principle'*, In: Knowledge and Language, Vol. I, E. Reuland and W. Abraham (Eds.), Kluwer, 1993, pp. 185–216.

97. S. Kurtz, S. Mahaney, and J. Royer, *The structure of complete degrees*, In: Complexity Theory Retrospective, A. Selman (Ed.), Springer, 1990, pp. 108–146.

98. M. Kummer and M. Ott, *Learning branches and learning to win closed games*, In: Proceedings of the Ninth Annual Conference on Computational Learning Theory, ACM Press, 1996, pp. 280–291.

99. G. Kreisel. *Mathematical logic*, In: Lectures in Modern Mathematics III, T. L. Saaty (Ed.), J. Wiley and Sons, 1965, pp. 95–195.

100. G. Kreisel, *A notion of mechanistic theory*, Int. J. Theor. Phys. **29** (1974), 11–26.

101. K. Kelly and O. Schulte, *The computable testability of theories with uncomputable predictions* Erkenntnis, **43** (1995), 29–66.

102. K. Kelly, O. Schulte, and C. Juhl, *Learning theory and philosophy of science*, Philos. Sci. **64** (1997), 245–267.

103. M. Kubat, *A machine learning based approach to load balancing in computer networks*, Cybernet. Syst. **23** (1992), 389–400.

104. R. Ladner, *On the structure of polynomial time reducibility*, J. Assoc. Comput. Mach. **22** (1975), 155–171.

105. S. Lange and P. Watson, *Machine discovery in the presence of incomplete or ambiguous data*, In: Algorithmic Learning Theory, K. Jantke and S. Arikawa (Eds.), Lect. Notes Artif. Intell. **872** , Springer, 1994, pp. 438–452.

106. Thinking Machines. Introduction to data level parallelism. Technical Report 86.14, Thinking Machines, April 1986.

107. N. Margolus, *Physics–like models of computation*, Physica 10D, (1984), 81–95.

108. Y. Marcoux, *Composition is almost (but not quite) as good as s-1-1*, Theor. Comput. Sci. **120** (1993), no. 2, 169–195.

109. D. Moore and J. Case, *The complexity of total order structures*, J. Comput. Syst. Sci. **17** (1978), 253–269.

110. D. McDermott, *Robot planning*, AI Magazine, **13** (1992), no. 2, 55–79.

111. T. Mitchell, R. Caruana, D. Freitag, J. McDermott, and D. Zabowski, *Experience with a learning, personal assistant*, Commun. ACM **37** (1994), no. 7, 81–91.

112. K. Mehlhorn, *Polynomial and abstract subrecursive classes*, J. Comput. Syst. Sci. **12** (1976), 147–178.

113. E. Mendelson, *Introduction to Mathematical Logic.* Chapman and Hall, London, 1997.

114. A. Meyer and P. Fischer, *Computational speed-up by effective operators*, J. Symb. Log. **37** (1972), 48–68.

115. M. Minsky, *Cellular vacuum*, Int. J. Theor. Phys. **21** (1982), no. 6/8, 537–551.

116. S. Matwin and M. Kubat, *The role of context in concept learning*, In: Proceedings of the ICML-96 Pre-Conference Workshop on Learning in Context-Sensitive Domains, Bari, Italy, 1996, M. Kubat and G. Widmer (Eds.), pp. 1–5.

117. E. McCreight and A. Meyer, *Classes of computable functions defined by bounds on computation*, In: Proceedings of the First Annual ACM Symposium on Theory of Computing, 1969, pp. 79–88.

118. O. Maler, A. Pnueli, and J. Sifakis, *On the synthesis of discrete controllers for timed systems*, In: STACS 95: 12th Annual Symposium on Theoretical Aspects of Computer Science Munich, Germany, March 2-4, 1995 Proceedings, E. W. Mayr and C. Puech (Eds.), Lect. Notes Comput. Sci. **900**, Springer, 1995, pp. 229–242.

119. G. Marcus, S. Pinker, M. Ullman, M. Hollander, T. J. Rosen, and F. Xu, *Overregularization in Language Acquisition*, Univ. Chicago Press, 1992. [Includes commentary by H. Clahsen]

120. D. Michie and C. Sammut, *Machine learning from real-time input-output behavior*, In: Proceedings of the International Conference on Design to Manufacture in Modern Industry, 1993, pp. 363–369.

121. J. Myhill, *Some philosophical implications of mathematical logic: I. three classes of ideas*, Rev. Metaphysics **6** (1952), no. 2.

122. J. Myhill, *A note on the degrees of partial functions*, Proc. Am. Math. Soc. **12** (1961), 519–521.

123. J. Myhill, *Abstract theory of self-reproduction*, In: Views on General Systems Theory, M. D. Mesarović (Ed.), J. Wiley and Sons, 1964, pp. 106–118.

124. J. Von Neumann, *Theory of Self–Reproducing Automata*, Univ. Illinois Press, 1966. [Edited and completed by A. W. Burks]

125. *Report* of the assessment panel for the international assessment of the U.S. math sciences, Technical Report NSF9895, National Science Foundation, March 1998. [http://www.nsf.gov/publications/pub_summ.jsp?ods_key=nsf9895]

126. P. Odifreddi, *Classical Recursion Theory*, North-Holland, 1989.

127. P. Odifreddi, *Classical Recursion Theory. Vol. II*, Elsivier, 1999.

128. D. Osherson, M. Stob, and S. Weinstein, *Ideal learning machines*, Cognitive Sci. **6** (1982), 277–290.

129. D. Osherson, M. Stob, and S. Weinstein, *Note on a central lemma of learning theory*, J. Math. Psychol. **27** (1983), 86–92.

130. D. Osherson, M. Stob, and S. Weinstein, *Learning theory and natural language*, Cognition **17** (1984), no. 1, 1–28.

131. D. Osherson, M. Stob, and S. Weinstein, *Systems that Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists*, The MIT Press, 1986.

132. D. Osherson and S. Weinstein, *Criteria of language learning*, Inf. Control **52** (1982), 123–138.

133. M. Pour-El and M. B. Richards, *A computable ordinary differential equation which possesses no computable solution* Ann. Math. Logic **17** (1979), 61–90.

134. M. Pour-El and M. B. Richards, *The wave equation with computable initial data such that its unique solution is not computable*, Adv. Math. **39** (1981), 215–239.

135. M. Pour-El and M. B. Richards, *Computability in Analysis and Physics*, Springer, 1989.

136. S. Pinker, *Formal models of language learning*, Cognition **7** (1979), no. 3, 217–283.

137. L. Pitt, *A Characterization of Probabilistic Inference*, PhD Thesis, Yale University, 1984.

138. L. Pitt, *Probabilistic inductive inference*, J. Assoc. Comput. Mach. **36** (1989), 383–433.

139. K. Plunkett and V. Marchman, *U-shaped learning and frequency effects in a multi-layered perceptron: Implications for child language acquisition*, Cognition **38** (1991), no. 1, 43–102.

140. L. Pratt, J. Mostow, and C. Kamm, *Direct transfer of learned information among neural networks*, In: Proceedings of the 9th National Conference on Artificial Intelligence (AAAI-91), 1991.

141. H. Putnam, *Probability and confirmation*, In: Voice of America, Forum on Philosophy of Science, Vol. 10, 1963. [Reprinted as [**142**]]

142. H. Putnam, *Probability and confirmation*, In: *Mathematics, Matter, and Method*, Cambridge Univ. Press, 1975.

143. J. Royer and J. Case, *Subrecursive Programming Systems: Complexity and Succinctness*, Birkhäuser, 1994.

144. G. Riccardi, *The Independence of Control Structures in Abstract Programming Systems*, PhD Thesis, SUNY Buffalo, 1980.

145. G. Riccardi, *The independence of control structures in abstract programming systems*, J. Comput. Syst. Sci. **22** (1981), 107–143.

146. G. Riccardi, *The independence of control structures in programmable numberings of the partial recursive functions*, Z. Math. Logik Grundlagen Math. **48** (1982), 285–296.

147. H. Rogers, *Gödel numberings of partial recursive functions*, J. Symb. Log. **23** (1958), 331–341.

148. H. Rogers, *Theory of Recursive Functions and Effective Computability*, McGraw Hill, 1967. [Reprinted: The MIT Press, 1987]

149. J. Royer, *A Connotational Theory of Program Structure*, Lect. Notes Comput. Sci. **273**, Springer, 1987.

150. J. Royer, *Semantics versus syntax versus computations: Machine models for type-2 polynomial-time bounded functionals*, J. Comput. Syst. Sci. **54** (1997), 424–436.

151. C. Sammut, *Acquiring expert knowledge by learning from recorded behaviors*, In: Japanese Knowledge Acquisition Workshop, 1992.

152. C. Sammut, *Automatic construction of reactive control systems using symbolic machine learning*, Knowledge Engineering Rev. **11** (1996), no. 1, 27–42.

153. O. Schulte, *Means-ends epistemology*, Br. J. Philos. Sci. **50** (1999), 1–31.

154. O. Schulte, *Inferring conservation principles in particle physics: A case study in the problem of induction*, Br. J. Philos. Sci. **51** (2000), 771–806.

155. J. Searle, *Minds, brains, and programs*, Behav. Brain. Sci. **3** (91980), 417–424.

156. A. Seth, *Complexity Theory of Higher Type Functionals*, PhD Thesis, University of Bombay, 1994.

157. N. Shapiro, *Review of "Limiting recursion" by E.M. Gold and "Trial and error predicates and the solution to a problem of Mostowski" by H.Putnam*, J. Symb. Log. **36** (1971), 342.

158. C. Sammut, S. Hurst, D. Kedzier, and D. Michie. *Learning to fly*, In: Proceedings of the Ninth International Conference on Machine Learning, D. Sleeman and P. Edwards (Eds.), Morgan Kaufmann, 1992, pp. 385–393.

159. T. Slaman, *Long range goals*, COMP-THY Archives, #13, April 1998. [http://listserv.nd.edu/archives/comp-thy.html]

160. C. Smith, *A Recursive Introduction to the Theory of Computation*, Springer, 1994.

161. T. J. Sejnowski and Ch. Rosenberg, *NETtalk: A parallel network that learns to read aloud*, Technical Report JHU-EECS-86-01, Johns Hopkins University, 1986.

162. S. Strauss and R. Stavy (Eds.), *U-Shaped Behavioral Growth*, Academic Press, 1982.

163. J. Stoy, *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*, The MIT Press, 1977.

164. K. Svozil, *Are quantum fields cellular automata?* Physics Letters A, **119** (1986), no. 4, 153–156.

165. J. B. Salem and S. Wolfram, *Thermodynamics and hydrodynamics with cellular automata*, In: Theory and Applications of Cellular Automata, S. Wolfram (Ed.), World Scientific, 1986.

166. N. A. Taatgen and J. R. Anderson, *Why do children learn to say "Broke"? A model of learning the past tense without feedback*, Cognition, **86** (2002), no. 2, 123–155.

167. F. Tsung and G. Cottrell, *A sequential adder using recurrent networks*, In: IJCNN-89-WASHINGTON D.C.: International Joint Conference on Neural Networks. Vol. 2, IEEE Service Center, Piscataway, New Jersey, June 18–22, 1989, pp. 133–139.

168. W. Thomas, *On the synthesis of strategies in infinite games*, In: STACS 95: 12th Annual Symposium on Theoretical Aspects of Computer Science Munich, Germany, March 2-4, 1995 Proceedings, E. W. Mayr and C. Puech (Eds.), Lect. Notes Comput. Sci. **900**, Springer, 1995, pp. 1–13.

169. S. Thrun, *Is learning the n-th thing any easier than learning the first*, In: Advances in Neural Information Processing Systems, 8, Morgan Kaufmann, 1996.

170. T. Toffoli and N. Margolus, *Cellular Automata Machines*, The MIT Press, 1987.

171. T. Toffoli, *Cellular automata machines*, Technical Report 208, Comp. Comm. Sci. Dept., University of Michigan, 1977.

172. T. Toffoli, *Computation and construction universality of reversible cellular automata*, J. Comput. Syst. Sci. **15** (1997), 213–231.

173. T. Toffoli, *CAM: A high–performance cellular–automaton machine*, Physica 10D, (1984), 195–204.

174. S. Thrun and J. Sullivan, *Discovering structure in multiple learning tasks: The TC algorithm*, In: Proceedings of the Thirteenth International Conference on Machine Learning (ICML-96), Morgan Kaufmann, 1996, pp. 489–497.

175. T. Urbančič and I. Bratko, *Reconstructing human skill with machine learning*, In: Proceedings of the Eleventh European Conference on Artificial Intelligence, A. Cohn (Ed.), John Wiley and Sons, 1994.

176. G. Y. Vichniac, *Simulating physics with cellular automata*, Physica 10D, (1984), 96–116.

177. D. Šuc, *Machine reconstruction of human control strategies*, In: Frontiers in Artificial Intelligence and Applications. Vol. 9, IOS Press, 2003.

178. A. Waibel *Connectionist glue: Modular design of neural speech systems*, In: Proceedings of the 1988 Connectionist Models Summer School, D. Touretzky, G. Hinton, and T. Sejnowski (Eds.), Morgan Kaufmann, 1989. pp. 417–425.

179. A. Waibel, *Consonant recognition by modular construction of large phonemic time-delay neural networks*, In: Advances in Neural Information Processing Systems I, D. S. Touretzky (Ed.), Morgan Kaufmann, 1989, pp. 215–223.

180. K. Weihrauch and N. Zhong, *Is wave propagation computable or can wave computers beat the Turing machine*, Proc. London Math. Soc. **85** (2002), 312–332.

181. K. Wexler and P. Culicover, *Formal Principles of Language Acquisition*, The MIT Press, 1980.

182. K. Wexler, *On extensional learnability*, Cognition, **11** (1982), no. 1, 89–95.

183. R. Wiehagen, *Limes-Erkennung rekursiver Funktionen durch spezielle Strategien*, Electron. Inform.-verarb. Kybernetik **12** (1976), 93–99.

184. R. Wiehagen, *Zur Theorie der Algorithmischen Erkennung*, PhD Thesis, Humboldt University of Berlin, 1978.

185. S. Wolfram, *Statistical mechanics of cellular automata*, Rev. Modern Phys. **55** (1983), no. 33, 601–644.

186. S. Wrobel, *Concept Formation and Knowledge Revision*, Kluwer, 1994.

187. P. Young, *Easy constructions in complexity theory: Gap and speed-up theorems*, Proc. Am. Math. Soc. **37** (1973), 555–563.