CDM

Hypercomputation—A Rant

Klaus Sutner

Carnegie Mellon University

Fall 2023

Tired of Computation?

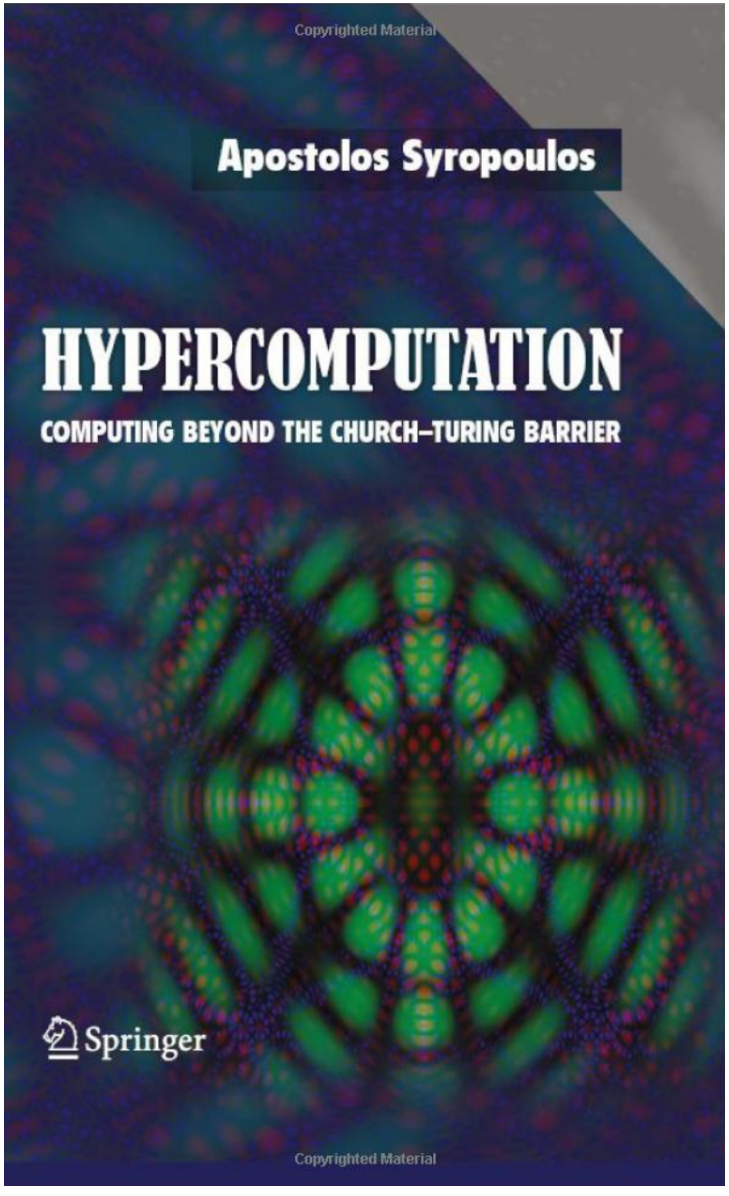
Tired of computability, complexity, algorithms, average case analysis, efficiency, non-computability, intractability?

No problem. There is now a small industry of people working in the brand-new and exciting field of

HYPERCOMPUTATION

Just wait a little bit longer, and your desktop machine will be replaced by a hypercomputer and will solve unsolvable problems on a routine basis. Merely intractable problems are handled instantaneously.

Wurzelbrunft has already placed a pre-order on Amazon.


Copyrighted Material
Apostolos Syropoulos
HYPERCOMPUTATION
COMPUTING BEYOND THE CHURCH–TURING BARRIER
Springer
Copyrighted Material

At present, the theory of computation falls into two major parts:

Classical Computability Turing/register machines, λ-calculus, decidability, semidecidability, arithmetical hierarchy, degrees of unsolvability, ...

Complexity Theory Time and space classes, deterministic and nondeterministic classes, circuits, $\mathbb{P}$ versus $\mathbb{NP}$, randomness, probabilistic classes, quantum computation ...

Could this be all? Nah ...

Historical Hypercomputation

This area is known as Generalized Recursion Theory (GRT) and has been studied extensively for almost a century, there are lots of results, everything is fairly well understood.

Hysterical Hypercomputation

A more recent idea, unencumbered by any sort of results. It relates to actual computability theory in about the way astrology relates to astronomy.

Recall my favorite quote from Stefan Banach:

> A mathematician is a person who can find analogies between theorems; a better mathematician is one who can see analogies between proofs and the best mathematician can notice analogies between theories. One can imagine that the ultimate mathematician is one who can see analogies between analogies.

Does this apply to computability?

Is there any reasonable way to generalize computability?

Is there any mileage one can get out of such generalizations?

A good number of "theories of computation" on structures other than the natural numbers have been developed: computation on ordinals, computation on sets, computation on algebraic structures, computation on higher types and so on.

There is even an axiomatization of computation:

> J. E. Fenstad
> General Recursion Theory: An Axiomatic Approach
> Springer 1979

Unfortunately, the axiomatization by Fenstad feels a bit awkward and overly technical (compared to, say, Hilbert's axiomatization of geometry or Zermelo-Fraenkel's axiomatization of set theory), but overall it captures the fundamental ideas behind computation fairly well.

Aside: Bad Slides Coming

All the experts tell you to

- keep text on slides to a minimum
- under no circumstances read your own slides
- use simple, stripped down graphics
- do **not** use multiple fonts, loud backgrounds, garish colors, useless pictures, cartoons, watermarks, anything distracting . . .

In general, I try to adhere to these principles[†] , but there will be too much text on the following slides, and I will read some of it aloud. Sorry.

[†] Except for item 1, I can't stand slides that are meaningless without a video.

There are some areas of intellectual discourse where the experts can engage in furious debates about the basic usefulness of certain theories, even beyond the question of whether the theories are correct. Psychology, philosophy, literature, history, economics and so forth come to mind.

However, in the hard sciences, these debates are quite rare and tend to focus on particular technical issues (is string theory falsifiable, is it still physics?).

Amazingly, we now have an example of an entirely meaningless "theory" in mathematics, or at least area close thereto.


OR

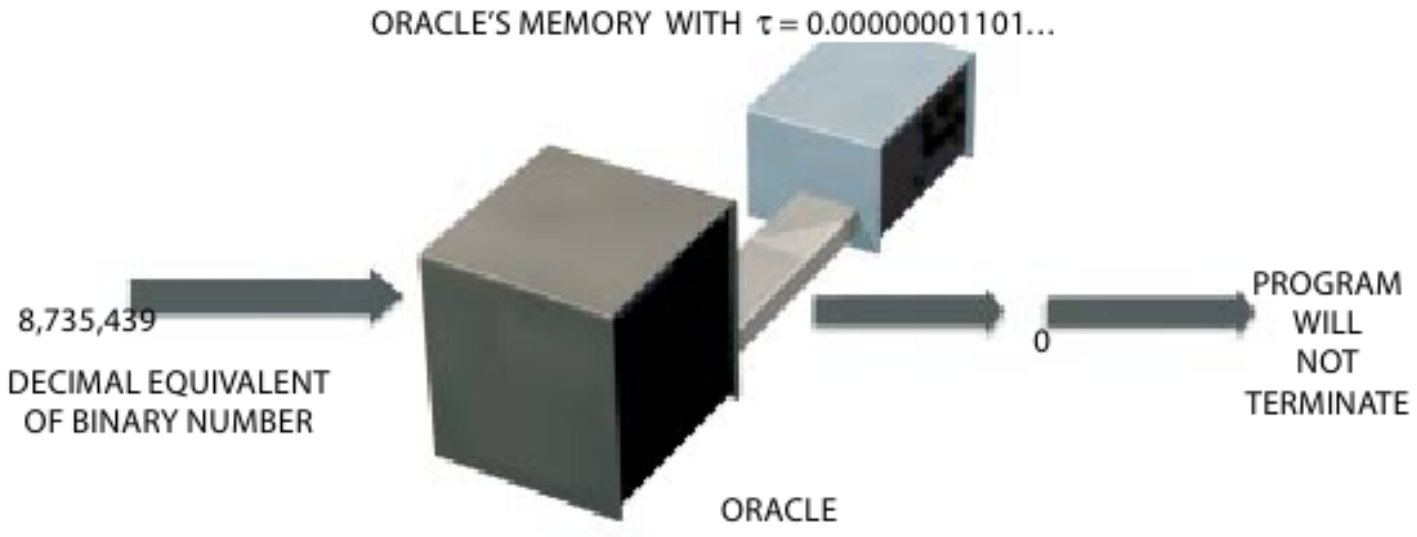
The term hypercomputation gained notoriety after a 1999 paper by Copeland and Proudfoot in the Scientific American titled

> Alan Turing's Forgotten Ideas in Computer Science

The article makes the most astounding claim that Turing "anticipated hypercomputation" in his technical work.

A clever PR trick, of course. Hiding behind one of the greats is usually a good idea (in particular if the person is dead and cannot complain).

To support this conclusion, the authors misinterpret Turing's concept of an oracle machine in patently absurd ways. There is a nice picture of an oracle machine in the paper:



The idea is that the big, ominous, gray box (the oracle) has access to a copy of the Halting set, living in the pleasant, smaller, blue box; here called $\tau \in \mathbf{2}^{\omega}$.

The authors comment:

> Obviously, without τ the oracle would be useless, and finding some physical variable in nature that takes this exact value might very well be impossible. So the search is on for some practicable way of implementing an oracle. If such a means were found, the impact on the field of computer science could be enormous.

No doubt, the impact on math would be enormous. For example, we could solve Diophantine equations and get answers to all kinds of open problems like the Riemann hypothesis.

Math departments would love such a machine.

But the impact would be much broader. Let $n = pq$ be the product of two large primes, so RSA depends on factoring n being hard.

Let's use a hypercomputer. For $2 \leq a \leq b \leq n$, construct a Turing machine $M_{a,b}$ that halts if there is some factor of n in the interval $[a, b]$.

Using the Copeland-Proudfoot machine, we could do a binary search[†] to find a factor.

Similar tricks would destroy just about all the cryptographic schemes relevant today (I suppose quantum-generated one-time pads would still work, if only you can get the pad safely to Bob).

So, the world financial system would implode immediately, followed by the whole world economy, and soon after we'd be back in the stone age.

[†]Not that it matters, we don't know how fast the machine is.

Alas, there is a small problem: storing an infinite amount of information in a finite physical system seems quite patently impossible.

And even if we somehow could store an infinite amount of information, we could not retrieve it—perhaps the authors are unaware of Heisenberg?

We would have to fire photons of arbitrarily high energy at our blue box to get at distant bits, which would destroy the device immediately.

Maybe Copeland did not really mean to have an infinite oracle, maybe he was only hoping to get the first, say, billion bits.

That would still have huge impact on math, depending on coding details we could immediately resolve open problems such as the Riemann hypothesis or the Goldbach conjecture, the consistency of Dedekind-Peano arithmetic, and so on. Cryptography might not be affected much.

Implementing a bit-string of length 10^9 would be entirely trivial, if only someone could provide the actual bits. Alas, that's exactly where things fall apart; any finite oracle exists abstractly, but we don't know how to construct it. If someone gave it to us, we would not recognize it.

Many objections could be raised to this proposal. The most relevant for us is that abstract mathematical entities are not the right kind of entity to implement a computation. Time and change are essential to implementing a computation: computation is a process that unfolds through time, during which the hardware undergoes a series of changes (flip-flops flip, neurons fire and go quiet, plastic counters appear and disappear on a Go board, and so on). Abstract mathematical objects exist timelessly and unchangingly. What plays the role of time and change for this hardware? How could these Platonic objects change over time to implement distinct computational steps? And how could one step "give rise" to the next if there is no time or change? Even granted abstract mathematical objects exist, they do not seem the right sort of things to implement a computation.

First, let's ignore the absurdly simplistic Platonic model imputed by these lines. At the very least, all logicians and philosophers of mathematics would laugh at this.

Second, the authors seem to be unaware that the many models of computation developed over the last century are all perfectly capable of being expressed precisely in terms of mathematical objects. In fact, that is their very purpose.

Without meaning to beat a dead horse, modeling temporal change is one of the central ideas in calculus. Have differential equations not made it all the way down to New Zealand? Are they upside-down? Just asking.

A search for "hypercomputation" generates 80,500 hits on Google, but "Kardashians" produces a healthy 557,000,000 hits. Insufficient evidence for a real thing.

> **Wikipedia**
> Hypercomputation or super-Turing computation refers to models of computation that can provide outputs that are not Turing computable. For example, a machine that could solve the halting problem would be a hypercomputer; so too would one that can correctly evaluate every statement in Peano arithmetic.

Sure, many of the models considered in generalized recursion theory in the last 80 years are hypercomputers in this sense. So what?

The next quote is taken from a 2006 article titled "The many forms of hypercomputation" by Toby Ord, an Oxford educated thinker and one of Copeland's acolytes.

In the interest of fairness, Ord is really a moral philosopher and quite impressive as such, see Giving What We Can.

> The new machines go beyond Turing's attempts to formalize the rote calculations of a human clerk and instead involve operations which may not even be physically possible. This difference in flavor is reflected in the terminology: they are hypermachines and perform hypercomputation. Can they be really said to "compute" in a way that accords with our pre-theoretic conception? It is not clear, but that is no problem: they hypercompute. Hypercomputation is thus a species of a more general notion of computation which differs from classical Turing computation in a manner that is difficult to specify precisely, yet often easy to see in practice.

Perhaps, but what is the point? This is exactly what GRT was created for. What are the new results or insights obtained from this approach?

> Let us suppose, however, that hypercomputation does turn out to be physically impossible–what then? Would this make the study of hypercomputation irrelevant? No. Just as non-Euclidean geometry would have mathematical relevance even if physical space was Euclidean, so too for hypercomputation. Perhaps, we will find certain theorems regarding the special case of classical computation easier to prove as corollaries to more general results in hypercomputation. Perhaps our comprehension of the more general computation will show us patterns that will guide us in conjectures about the classical case.

Again, that's old hat, GRT has done exactly that. Absolutely nothing new here.

Also note the evolution on the issue of implementability, apparently the oracles are still at large.

> Thus the claims that such problems are "undecidable" or "unsolvable" are misleading. As far as we know, in 100 years time these problems might be routinely solved using hypermachines. Mathematicians may type arbitrary Diophantine equations into their computers and have them solved. Programmers may have the termination properties of their programs checked by some special software. We cannot rule out such possibilities with mathematical reasoning alone. Indeed, even the truth or otherwise of the Church-Turing Thesis has no bearing on these possibilities. The solvability of such problems is a matter for physics and not mathematics.

So now implementability is critical again, we actually want to build and run our hypermachines.

And undecidability is a matter of physics, not math. In Ord's defense, others have come up with similar claims (Landauer, Deutsch).

Halting Turing machines perform finite computations and produce finite output, no problem. But remember the characterization of semidecidable sets A as being exactly the recursively enumerable sets: there is a computable function with range A.

A nice way of thinking about this is to set up a Turing machine with a special output tape. The machine runs an "infinite amount of time" and writes the characteristic function of A on the output tape (alternatively, it could write the elements of A on the tape). For example, the Halting set could be produced this way (dovetailing).

This is perfectly fine intuitively, we just have to wait a bit longer for the result. Of course, ultimately we need to figure out a precise way of defining "infinite amount of time."

Example: Zeno Machines

Here is one pseudo-physical approach to organize an infinite computation: find a way to speed up the computation of a Turing machine. The first step takes one second, the second $1/2$ a second, the nth takes 2^{n-1} seconds.

The whole infinite computation takes a mere 2 seconds.

Of course, this is physical nonsense, but everyone would agree that, given a better universe, we could produce the Halting set in just 2 seconds.

Technically, we need two main ingredients to formalize our idea:

- The use of transfinite ordinals rather than just natural numbers to count steps in a computation.
- A mechanism to preserve information when a computation reaches a limit stage.

Of course, we wind up with a model of computation that is eminently unrealizable, we cannot build such a device within the framework of physics (in stark contrast to Turing machines). Still, it turns out that it fits in nicely with other ideas from generalized recursion theory, and a number of interesting results can be obtained this way.

It is easy to add another step to any computation, the real problem is to deal with limit stages when infinitely steps have already been performed. Here is the basic idea.

We can associate the steps of an ordinary Turing machine with the natural numbers:

$$0, 1, 2, \ldots, n, n{+}1, \ldots \mid$$

The notation $\ldots \mid$ is meant to indicate infinitely many steps, as opposed to the first $\ldots$, which stands for finitely many steps[†]. We write ω for all these infinitely many steps.

The idea is that a machine, after running through all the finite steps n, arrives at the (first) infinite level ω. The machine goes into a trance along the way, and wakes up at level ω.

[†] The ellipsis is one of the most consistently abused symbols in all of math.

So suppose our machine has reached level ω. Nothing can stop us from taking another step, leading to level $\omega + 1$. And then to $\omega + 2$, . . . , $\omega + n$, and so on. If we keep going, we finally wind up at

$$0, 1, 2, \ldots, n, \ldots \mid \omega, \omega + 1, \omega + 2, \ldots, \omega + n, \ldots \mid$$

which we express as $\omega + \omega = \omega \cdot 2$. So these are two infinite blocks, one after the other.

You guessed it, we can also get $\omega + \omega + \omega = \omega \cdot 3$. In fact, we can get

$$\omega, \omega \cdot 2, \omega \cdot 3, \ldots, \omega \cdot n, \ldots \mid$$

We denote this level by $\omega \cdot \omega = \omega^2$: ω many blocks of size ω each.

In a similar way we can get to higher powers ω^k and ultimately to ω^ω.

Moving right along, we get ω^{ω^ω} and so on.

Nothing can stop us now, we get to level

$$\varepsilon_0 = \omega^{\omega^{\omega^{\cdot^{\cdot^{\cdot}}}}}$$

So this is a stack of ω many ωs all exponentiated somehow.

We'll stop here to avoid injury to malleable young minds. But rest assured, one can keep on going on, and on, and on . . . |

Aside: Induction up to ε_0 is needed to prove the consistency of Dedekind-Peano arithmetic, your favorite system from 15-151.

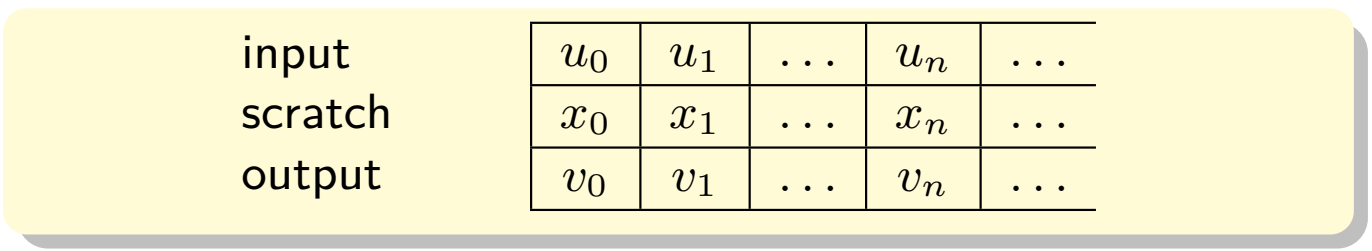
This may all sound very alluring, but does it actually hold water?

In particular, is there are way to formalize these ordinals in, say, Zermelo-Fraenkel set theory so that we can actually prove theorems about them?

No problem, this is one of the many accomplishments of von Neumann. He gave a very concise and clean definition of ordinals as sets. And he explained how one can use transfinite recursion to define all the required arithmetic operations (addition, multiplication, exponentiation, . . .). Plus, they can be used to define cardinal numbers—as opposed to the customary wishy-washy about injections and bijections.

Let's just take for granted that this transfinite level scheme can actually be formalized. The key questions now is: how can we make sense of the computation of a Turing machine at stage ω? Or any other limit stage for that matter?

Here is a useful model developed by J. Kidder, J. Hamkins and others. We use a Turing machine with a one-way infinite tape that is subdivided into three tracks:



We can safely assume that the alphabet is $\mathbf{2}^3$, so each track contains a word in $\mathbf{2}^\omega$. We have a finite state control and a read/write head, as usual.

Now suppose the Turing machine has already performed all steps $n < \omega$. We define the configuration at time ω as follows:

- The machine is in a special state $q_{\lim}$.
- The head is in position 0.
- The content of a subcell such as x_i is the limsup of its contents at times $n < \omega$.

The definition for an arbitrary limit level λ is exactly the same. Thus, the entry in a subcell is 1 at time λ iff

$$\forall\, \beta < \lambda\, \exists\, \alpha \left(\beta < \alpha < \lambda \ \wedge \ \text{symbol at time } \alpha \text{ is } 1 \right)$$

Think of a light blinking infinitely (actually, cofinally) often before time λ.

Note that the special limit state $q_{\lim}$ is the same for each transfinite limit time λ; there is no special q_ω, $q_{\omega+\omega}$, $q_{\omega\cdot\omega}$ and so on. So the state set of a ITTM is still finite.

Thus, the only way to preserve information at a limit time is via tape subcells; state and head position are always exactly the same when we wake up on the other side.

For example, we can have the subcell x_0 on the work tape hold a 1 at a limit time λ iff something happened over and over arbitrarily close to λ.

An ordinary Turing machine simply cannot do this, it dies at level ω and never wakes up again.

Example: Halting

Consider the Halting problem for ordinary Turing machines: given an index e we want to know if $\{e\}$ halts on the empty tape.

This can be handled by an ITTM $\mathcal{M}$: e is written on the input track. $\mathcal{M}$ then simulates the ordinary computation of $\{e\}$ on empty tape using the scratch track but not x_0.

If $\{e\}()$ halts after finitely many steps, $\mathcal{M}$ also halts and accepts.

Otherwise we reach the limit stage ω. $\mathcal{M}$ wakes up in state $q_{\lim}$, takes one more step, and halts and rejects at time $\omega + 1$.

We can do better than this: we can dovetail all computations $\{e\}()$, $e \in \mathbb{N}$, and write a 1 in position e of the output track whenever the corresponding computation converges.

At time ω, the (characteristic function of the) Halting set will appear on the output track.

More generally, we can use ITTMs to compute functions

$$f : \mathbf{2}^\omega \longrightarrow \mathbf{2}^\omega$$

essentially functions on the reals.

Halting is near the bottom level of the arithmetical hierarchy, but we can also handle higher levels via ITTMs.

For example, recall INF, the collection of all Turing machines that halt on infinitely many inputs:

$$\text{INF} = \{\, e \in \mathbb{N} \mid \{e\} \text{ converges on infinitely many inputs} \,\}$$

Intuitively, this is clearly harder than plain Halting.

In fact, we have seen INF is Π_2-complete in the arithmetical hierarchy. Pretty hopelessly hopeless ...

Lemma

We can decide membership in INF *by an ITTM.*

Again, ITTMs are not physically realizable. There is nothing wrong with the fact that they can decide problems that are highly undecidable by perfectly realizable ordinary Turing machines.

Plus, they are a rather neat model and require far fewer technicalities than other models in generalized computability. Far, far fewer.

As before, e is written on the input track.

Then ITTM $\mathcal{M}$ runs the following program:

foreach $n \in \mathbb{N}$ **do**
    if $\{e\}(n)\downarrow$
    then flash x_0; // turn on and then off right away

if $x_0 = 1$ // we are at a limit stage
then accept
else reject

So $\mathcal{M}$ flashes a light whenever it finds a convergence. Infinitely many flashes means we are in INF.

If $\mathcal{M}$ finds that $\{e\}$ converges on n, it turns bit x_0 on, and then off again at the next step. Of course, $\mathcal{M}$ will not use subcell x_0 for the simulation, just for messaging.

If the machine $\{e\}$ diverges on n, we just spend ω many steps finding out, without ever flashing x_0.

At the limit stage corresponding to completion of the main loop, subcell x_0 will hold a 1 iff there were infinitely many good arguments n.

The check for each n may require up to ω steps, so the total running time is between ω and ω^2.

Not at all, we can similarly show that we can climb up the arithmetical hierarchy.

Lemma

For any n, all problems in Σ_n can be decided by a ITTM.

In fact, we can even handle all levels in the arithmetical hierarchy at once:

Lemma

Arithmetical truth can be decided by a ITTM.

Actually, ITTMs are even more powerful than this.

Recall that the arithmetical hierarchy is obtained by placing alternating blocks of quantifiers ranging over $\mathbb{N}$ in front of a decidable relation.

Analogously, we can construct the analytical hierarchy by using alternating blocks of quantifiers that range over subsets of $\mathbb{N}$. Without going into technical details, it is not terribly surprising that this should produce far more complicated sets than our old arithmetical hierarchy.

Here is an example: suppose we have an order relation $\prec$. If we want to assert that $\prec$ is a well-order we have to say something along the lines of

$$\forall\, X \left(\exists\, x\, (x \in X) \Rightarrow \exists\, u\, (u \in X \wedge \forall\, v\, (v \in X \Rightarrow \neg(v \prec u)))\right)$$

Less formal but easier to read:

$$\forall\, X \neq \emptyset\, \exists\, u \in X\, \forall\, v \in X\, (v \not\prec u)$$

In terms of the analytical hierarchy, this shows that being a well-order is a property expressed at level Π^1_1 (note the superscript 1).

In fact, it is well-known that checking whether a given order on $\mathbb{N}$ is well-order is Π^1_1-complete. Incidentally, the whole arithmetical hierarchy is properly contained in Π^1_1. In fact, it fits into $\Delta^1_1 = \Pi^1_1 \cap \Sigma^1_1$, the place in the analytical hierarchy where arithmetic truth lives.

One might wonder whether this sort of complexity class makes any sense whatsoever. The arithmetical hierarchy is bad enough, why bother with these monsters? One reason is that quantification over sets is essential for analysis, we cannot formalize the reals, continuity, differentiability, integrability and so without such high-powered tools.

We can code a well-order $\prec$ on $\mathbb{N}$ as a word $W \in \mathbf{2}^\omega$, really just an infinite bit-vector:

$$x \prec y \iff W(\langle x, y\rangle) = 1$$

where $\langle ., . \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is a standard coding function. We can write W on the input tape of a ITTM; since the machine can have transfinite running time, there is no problem in reading all this information.

Theorem

It is ITTM-decidable whether $W \in \mathbf{2}^\omega$ codes a well-order.

As it turns out, any ITTM-decidable set is in Δ^1_2, so this is close to the limit of the computational power of ITTMs.

How Long Does It Take?

Here is a nice example of a theorem for ITTMs that exhibits a new type of behavior. This is the kind of result that makes GRT worthwhile.

An ordinary Turing machine can halt, enter a loop or diverge (run through an ω-sequence of non-repeating configurations). By contrast, an ITTM either halts or enters a loop of sorts: even if it "diverges" for a while, it will ultimately end up in a limit cycle. How far do we have to go before the computation ends in this sense?

Theorem

Every computation of a ITTM either halts or enters a loop after countably many steps.

So we do not have to run the machine $\aleph_1$ or $\aleph_{17}$ many steps, some $\alpha < \aleph_1$ will do. Small consolation, but at least we are not totally out to lunch.

Das ist nicht einmal falsch.

This is not even wrong.

It seems clear that hypercomputationists need to cling to physics for dear life: without implementability they are adrift in generalized recursion theory, a field that they apparently are unaware of or do not understand.

To get any traction, one has to ask whether the physics of our actual universe somehow supports hypercomputation. Of course, this is an exceedingly difficult question: Hilbert's problem $\#6$, when expanded to all of physics, is still unanswered: no one knows how to axiomatize physics in its entirety.

> The investigations on the foundations of geometry suggest the problem: To treat in the same manner, by means of axioms, those physical sciences in which already today mathematics plays an important part; in the first rank are the theory of probabilities and mechanics.

Ideally we could build an axiomatization Γ of physics, in some sufficiently powerful logic. So the real world would be a structure that models Γ (and perhaps is not uniquely determined). Γ would be a Theory of Everything.

Then, with a bit of effort, one might be able to show that

$$\Gamma \vdash \exists\, M \,\big(\text{device } M \text{ solves Halting}\big)$$

Or, we might be able to prove the negation. Maybe Γ could be loop quantum gravity, or string theory, or some such.

Of course, there is also the vexing problem of validity: is our ToE actually correct? Obviously there is no such thing as a correctness proof in the strict mathematical sense, just empirical observations.

OK, so playing with physical theories, even partial or inaccurate ones, is not all useless.

It is an excellent exercise to fix some particular theory Γ of physics (not a ToE) and try to show that in Γ it is possible to "construct a device" that solves the Halting problem.

For example, assume Newtonian physics: gravitating point masses, no relativity theory, no quantum theory. It's a nice exercise; alas, it has no bearing on implementability, none whatsoever.

Exercise

Concoct a hypercomputer in your favorite fragment of physics.

So What's The Difference?

Copeland's oracle computer can "solve" the Halting problem as can ITTMs. Why should one care about one but not the other?

Because ITTMs produce an interesting theory of computation that has close connections to other areas of generalized recursion theory, along the lines discussed above. The proof techniques are interesting and quite complicated.

Copeland's machines, on the other hand, are utterly useless, just a shallow PR stunt that is of no importance anywhere.

Hodges has written the definitive biography of Turing (incidentally, very well worth reading). He found it necessary to comment on the Copeland/Proudfoot article, an unusual step in the genteel world of math.

> I was appalled that this hare-brained idea should be associated with Alan Turing as his 'lost brainstorm.' Scientific American said that this 'hypercomputation' is a 'hot idea' which Alan Turing had 'anticipated in detail.' I suspect many people with a physical or engineering background took it, on reading this nonsense, that Turing had come up with this ridiculous idea because he was an impractical logician without understanding of reality. What a slur on his reputation!

Here is the link: Hodges on Copeland/Proudfoot

In 2006, Martin Davis could not stand it any longer, and published a paper

The Myth of Hypercomputation

In the paper, he very sweetly demolishes, annihilates and eviscerates the idea of "hypercomputation." Again, this kind of direct and scathing criticism is highly unusual; bad work is typically ignored, not openly criticized.

Needless to say, the Cult of Hypercomputation simply ignores Davis's paper, as well as all other criticism.

At this point, several areas of what used to be science have been invaded by fashionable nonsense.

There is some amusement value in this, and it even may be welcome as a sign of a more open, less dogmatic and hierarchical world. For example, `arXiv` is a big step forward from the traditional publishing machine and some blogs are absolutely great. The flip-side is `viXra`, an inexhaustible source of garbage, and blogs that are simply lousy.

Democratizing science is great. But, it's also dangerous, without any control mechanism we get "alternative facts" instead of science.

> This drifting of figures and geometric figuring, this irruption of dimensions and transcendental mathematics, leads to the promised surrealist peaks of scientific theory, peaks that culminate in Gödel's theorem: the existential proof, a method that mathematically proves the existence of an object without producing the object.
>
> Paul Virilio

Virilio is described as a "cultural theorist, urbanist, and aesthetic philosopher."