UCT

Hypercomputation

Klaus Sutner
Carnegie Mellon University
Spring 2022

1 Generalized Computation

2 Infinite Time Turing Machines

3 The Cult of Hypercomputation

4 Hypercomputation and Physics

Tired of Halting? 2

And other undecidability issues, like solving Diophantine equations?

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

HYPERCOMPUTATION

Just wait a little bit longer, and your desktop machine will solve unsolvable problems on a routine basis. Wurzelbrunft has already placed a pre-order.

Computability 3

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, ...

Could this be all? Nah ...

Going Hyper 4

Historical Hypercomputation has been studied extensively for almost a century, and is well understood.

Hysterical Hypercomputation is a fairly new idea, and relates to actual computability theory in about the way astrology relates to astronomy.

Generalized Computation? 5

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?

Classical Theory

The classical theory of computation is built around the structure

$$\mathfrak{N} = \langle \mathbb{N}, +, \cdot, 0, 1 \rangle$$

This is arguably the most natural environment, but there are many other structures where one would like to compute: the reals, the complexes, some rings, fields, set-theoretic universes and so on.

So a natural question is: is there any way to make sense out of the notion of computation in any of these structures?

Computation on Reals

Everyone would agree that computation is a key concern in analysis. Here one naturally operates in the structure

$$\mathfrak{R} = \langle \mathbb{R}, +, \cdot, 0, 1 \rangle$$

or various extensions thereof (e.g., we could add $\log$ and $\sin$, continuous functions, differentiable functions, and so on).

A large part of physics and engineering computes in this structure, sometimes in a slightly haphazard way (e.g. by using rational approximations without worrying too much about accuracy).

Several attempts have been made to define a rigorous theory of computation over $\mathfrak{R}$, but no single one has emerged as the clear "correct" description (computable analysis, domain theory, real RAMs, ...).

Aside

This is in stark contrast to computability on the naturals: all the standard models there agree. And, for Turing machines one can make a very tightly reasoned and sound argument for the following assertion:

> Turing machines capture exactly the intuitive notion of computability in discrete mathematics.

Of course, this is not a theorem, just a highly plausible assertion. Even without a complete axiomatization of physics, it seems to be well aligned with physically realizable computability.

Computation in continuous domains such as $\mathfrak{R}$ is much more problematic.

Generalized Recursion Theory

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.

1 Generalized Computation

2 **Infinite Time Turing Machines**

3 The Cult of Hypercomputation

4 Hypercomputation and Physics

Turing Machines with Infinite Computations

Here is a fairly tangible example of a theory of computation that strictly generalizes the classical theory.

The main idea is simple: find a way to explain what it means for a Turing machine to run an "infinite amount" of time (technically, measured in terms of ordinals). Of course, this is not a model that relates to anything realizable in physics, but a number of interesting results can be obtained this way.

We would expect to get strictly more compute power than from ordinary TMs; e.g., we should expect to be able to solve the Halting Problem on one of these devices[†].

[†] Note that these thingies are not machines in the classical sense, there is no conceivable way to implement them in actual physics.

Infinite Time Computation

It is clear what it means for a Turing machine to run a finite number of steps.

A non-halting Turing machine runs "infinitely many" steps. In a sense, this is the smallest sort of infinite computation.

We could imagine that we keep going after finishing all the finite levels.

Big Question: Can we give a useful and precise definition of an infinite time Turing machine (ITTM)?

The definition needs to be no less clear than the definition of ordinary Turing machines, no wishy-washy allowed.

ω (omega)

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.

. . . and Beyond . . .

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

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

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

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

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

. . . To Infinity

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.

Of course, we can keep going, and 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 minds. But rest assured, one can keep on going on, and on, and on $\ldots \mid$

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

All BS?

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?

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 operations (addition, multiplication, exponentiation, . . .).

Limit Stages

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 Joel Hamkins and his co-workers. We use a Turing machine with a one-way infinite tape that is subdivided into three tracks[†]:

input	u_0	u_1	. . .	u_n	. . .
scratch	x_0	x_1	. . .	x_n	. . .
output	v_0	v_1	. . .	v_n	. . .

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

[†]In complexity theory we would use three tapes, but this scenario is slightly easier to deal with

Limit Stages Defined

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

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

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

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

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

Preserving Information

Note that the machine limit state q_{limit} is the same for each transfinite limit stage 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.

The only way to preserve information across limit stages is via tape cells; state and head position are always the same when we wake up on the other side.

In essence, we can have the subcell x_0 on the work tape hold a 1 at a limit stage iff something happened over and over arbitrarily close to the limit stage.

An ordinary Turing machine simply cannot do this, it dies at level ω.

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.

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_{limit}, takes one more step, and halts and rejects at time $\omega + 1$.

Example: INF

Halting is at 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 even harder than plain Halting.

More precisely, one can show that INF is Π_2-complete in the arithmetical hierarchy. Pretty hopeless ...

No Problem

Lemma

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

Deciding Halting is bad enough, but deciding INF is really way overboard. This model of computation does not correspond to anything one might call natural, intuitive computability.

Still, it's pretty neat and requires far fewer technicalities than other models in generalized computability. Far, far fewer.

Proof

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;

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.

Quoi? 24

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 ns.

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

Are We Done? 25

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

Arithmetic truth can be decided by a ITTM.

We'll stop here, but ITTMs are even substantially more powerful than this.

How Long Does It Take? 26

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: 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.

Well-Orders 27

We can code a well-order of the natural numbers as a binary ω-word W:

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

Note that this requires infinite input to the ITTM, but that's just fine: we can write it on the top track of the tape.

Theorem

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

For those familiar with the analytical (not arithmetical) hierarchy: checking a well-order is Π^1_1-complete. Note the superscript 1: this requires universal quantification over sets.

1 Generalized Computation

2 Infinite Time Turing Machines

3 **The Cult of Hypercomputation**

4 Hypercomputation and Physics

Aside: Bad Slides Coming 29

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, 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.

The Experts

Edward Tufte

Jean-Luc Doumont

Patrick Winston

Fashionable Nonsense

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.

Hypercomputation

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

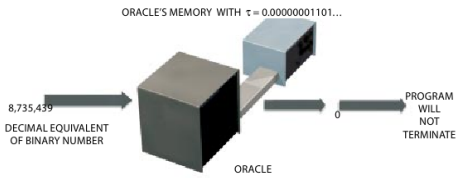
> Alan Turing's Forgotten Ideas in Computer Science

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).

Oracles

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 $\tau \in \mathbf{2}^{\omega}$.

Copeland and Proudfoot

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.

Exact value? Really? Does the name Heisenberg ring a bell?

As to the impact, no doubt it would be enormous. Alas, there is a small problem: storing an infinite amount of information in a finite physical system seems quite patently impossible.

A Misunderstanding?

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.

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; the oracle exists abstractly, but we don't know how to construct it.

Copeland versus Math

There have been suggestions that the whole universe might be some sort of gigantic computation, maybe a simulation by some über-beings.

> 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.

Come Again?

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.

Is Hypercomputation Real?

A search for "hypercomputation" generates 22,300 hits on Google, but "Kardashians" produces a healthy 184,000,000 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, ITTMs are hypercomputers in this sense. As are some of the models considered in generalized recursion theory in the last 80 years. So what?

Toby Ord

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 Argument from Hyper

> 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 generalized recursion theory was created for. What are the new results or insights obtained from this approach?

Ord Hedges

> 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, GRT. Absolutely nothing new here.

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

Ord Halts

> 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).

1 Generalized Computation

2 Infinite Time Turing Machines

3 The Cult of Hypercomputation

4 **Hypercomputation and Physics**

Wolfgang Pauli

Das ist nicht einmal falsch.

This is not even wrong.

Hypercomputation and Physics

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.

Theory of Everything

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

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

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.

Not Fair

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? 48

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.

The Resistance: Andrew Hodges 49

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.

Hodges on Copeland/Proudfoot

There is also pdf of an article at this site that is quite interesting.

The Resistance: Martin Davis 50

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.

The Message 51

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. The flip-side is `viXra`, an inexhaustible source of insanity.

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