The Church-Turing Thesis is a camouflage of Cthulhu

Csaba Henk csaba@redhat.com

Abstract

In this paper we make an attempt to find natural formal counterparts of basic concepts of computation in the first order theory of hereditarly finite sets. We also discuss what kind of representation hypotheses could or should we have in place of the Church-Turing Thesis. Finally we propose the investigation of the general set-theoretic representation problem, the Cthulhu Hypothesis.

Contents

An orthodox view of the Church-Turing Thesis

The story of the Church-Turing Thesis (in the sequel: *CTT*) is like the first season of a soap opera. While the story is duly full of quirks and exciting bends and unexpected fallouts, let us just provide here a quick summary of the plot

(BEWARE SPOILER!) that would enable you to chime in at the hypothetical second part.

> Math is the elderly rich guy who is coming over his spleen by becoming romantically attracted to the beautiful young girl from the country called Comp Sci. Comp Sci can't help falling for the suavity and gallantry of Math who (at least from her perspective) seems to be unlimited in power and resources, so a romance starts to blossom. They are just happy together without caring about the rest of the world. However, Comp Sci suddenly notices that she's expecting a baby from Math, and this situation forces them to make a decision. They think their love is strong and will prevail over any difficulty so they claim commitment to each other, make a come out about their liaison and decide on to bring up their child. When the time comes, the child is born and they baptize him to CTT.
>
> However, an awful thing happens. Time flies quick and CTT leaves behind his suckling age. As he's growing older, it becomes harder and harder to not notice the fact that CTT is a *mongrel*. At some point it becomes impossible. What CTT is and what CTT does, that's just uncomprehensively alien to them; it goes right against their expectations about having a child and out of their competence to handle (you would also like to have kids who pet the kitten instead of making experiments how long can it take to have them drowned, right?). Things just fall apart – Math and Comp Sci don't want to know about their mongrel son anymore so they dumped him gracefully, in the sense that they set up suitable funds and arrangements for his supply which can have them reassured that he won't die of starvation on the corner; they just pretend for themselves and the world as he'd be having studies far-far away and it's not known when would he be back (recall, it was a time when there wasn't yet commodity flights and email).
>
> Actually CTT was not as bad as to kill kittens; he was just too weird, but also lovable in a peculiar way. Thanks to such features, finally he was adopted by a jolly triumvirate of elderly sister spinsters, Philosophy, Physics, Philology, and their young niece, Programming. They related to the boy with hearty good will (pouring on him all their love that remained on stock in lack of having their own children) but they also had their quite solid views about the world, how things should be, how a boy should be, how he'd be taught and dressed. When CTT came to their harbor he was understandably love-hungry so he did not care much about the weird stuff he was put through. As time went on, he turned into a miniature caricature of the ladies (which was obvious for anyone but the ladies and CTT).
>
> However, time kept going on, and CTT become a young man; on the course of this, he had gone through some experiences that hinted him

> that he is not quite the same as that what he happens to be raised to. He is *different*. It is true that feeling to be strange and estranged in this way is what we can call normal at his age – so this phenomenon in itself is nothing special. But then something happens...

END OF FIRST SEASON.

Let's see what the plot means. Some parts are trivial to decipher, like the relationship of Math and Comp Sci.

But why is CTT a mongrel? Because the question addressed by CTT is out of the scope of the methodology of mathematics. CTT is not a mathematical question, it is *about* mathematics. It addresses the formalization of an intuitive idea. Mathematics (of computing) begins at the point where CTT ends – when we have a solid formal notion of an algorithm.

As of the dumping of CTT: that's how it happened historically. When the gold rush of the Foundation Age fell down, mathematics and theoretical computer science took a turn that targeted the particular and practical. Speaking about mathematics was not considered an organic part of doing mathematics anymore, mathematics and "mathesis" (using the word of László Surányi (Surányi (1997))) parted. CTT was still referred to and was taken granted (as in the plot, the parents did not disinherit him publicly), but not researched.

Not researched, as long mathematical research is concerned; but then comes the adoption part. The adoption of CTT by the P. ladies, who try to raise him into their own caricature. They continued research in various derivative directions that's was all stuffed under the umbrella of CTT. In Philosophy, it was continued into investigations about various epistemological aspects of various aspects of computing (Sieg (2007), Shapiro (2006)). In Physics, it was continued into investigations regarding whether current theoretical models allow the existence of a mechanism empowered with super-Turing capabilities (Németi and Dávid (2006), Syropoulos (2008), Yao (2003)). In Philology, investigations regressed into digging up and reinterpreting the original problematization, writings and discussions of Church and Turing and their contemporaries (Sieg (2006)). In Programming, investigations moved over to proposing and studying various theoretical models of computing which were deemed more expressive according to the expertise of the proposers or served as formal counterparts of certain computing scenarios like concurrent systems (Dershowitz and Gurevich (2008), Boker and Dershowitz (2008)).[1]

[1] It might be not clear why I use two separate characters with quite different roles for Comp Sci and Programming.

Comp Sci represents the pre-computer / proto-computer age of computer science, when computer science was necessarily theoretical, and basically a branch of mathematics.

Programming represents the new generation of computer science, when there is already lot of experience, huge expertise with, and convenient access to computers. This gives a kind of understanding that was not possible earlier, and also pushes towards pragmatical goals.

The final part of the plot, when young CTT's eyes start to open up to who he is (or at least, who he is not) – well, that does not map to reality, it's just my pipe dream.

My point is, that I call the orthodox view, is as follows: giving due respect to all this mesh of research activity that runs branded with CTT, the original, I risk: primordial form of the Thesis is a distinguished one. Why? I come to that in the next section.

The primordial Church-Turing Thesis

Before continuing, let me point out a common misconception. It is true that people have a tendency to make originality a value factor, and when studying CTT, there is a tendency to go back to the original. In particular, philologists have a tendency to think they are making this movement when they study the original phrasings of CTT as it occurs in the works of the name givers of the Thesis and other significant contemporary scientists.

To mistake original phrasing for original meaning is a subtle but grave error. The first cut of a new idea is usually experimental and lacks the support of the terminology arsenal that's available with mature theories. However, if the idea gains significance and momentum, it starts to live its own life, terminology crystallizes around it while redundancies and stray ways are stripped off.

While I think philological research is exciting and important, we can be more confident than to seek support for understanding CTT in the original texts. As Koestler puts in Koestler (1986): Copernicus' *De revolutionibus orbium coelestium* was the scientific work of greatest impact that noone has ever bothered to read. It was an important manifesto but it's far from being the best source for understanding the heliocentric model of astronomy.

Having this pinned down, let's get back to originality as of my understanding.

The original CTT is distinguished from other epistemological / cultural / pragmatic attempts to capture computing-as-such (or computing-that-could-be) by being a *fundamental question*. "Fundamental" means that a whole scientific discipline – namely, computer science – bets its existence on CTT. To put it roughly: if we drop CTT, we cannot speak of computing in mathematical terms.

What CTT says actually sounds quite trivial; in most cases (considering other disciplines), such statements are not made because they are thought to be evident.

Yuri Gurevich seems to be an archetypical figure of the latter kind. He has his own view of computing in terms of the machines bearing his name. When I asked after his lecture how can his machines represent functional programming, he answered (something like that) functional programming is not important, computers are operated in an imperative manner. (I just recall it as an example for practice implied strong bias, not as criticism.)

Also note that I don't want to imply that the Comp Sci branch would be more scientific than the Programming branch.

It's like saying: "for to be able to walk, you need a pair of legs". It's saying: "the formalization of computing formalizes computing".

It's highly against intuition that we have to support ourselves by the crutch of making an explicit statement about it. Usually, formalizations are self-justifying; when the formal counterpart of some intuitive idea is in place, it's quite clear in retrospect how the formal version captures intuition. The simplest example for this is distance. In the formal spatial framework of vector spaces, distance of two points a, b given as vectors: $a = \langle a_1, a_2, a_3 \rangle, b = \langle b_1, b_2, b_3 \rangle$, their distance is defined as $\sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2 + (b_3 - a_3)^2}$. This definition is inspired by the Pythagorean theorem. Once you understand the Pythagorean theorem, you won't have doubts about this definition; it clicks and falls to place. That's how we expect things to be.

However, in case of computing, a strange dissonance creeps in between the proposed formal models of computing and our holistic perception of what computing is. It's like when you break a thermometer in the kitchen and a flux of mercury drops runs off on the floor to all corners and edges. You want to clean up; you do; you scrutinize the corners and edges and you are mildly sure that you've done a good job and you've collected and cleaned up all the mercury drops. Still, in the deep of your stomach, you feel anxious, *What if there are still some drops hiding somewhere and the kid will find a fancy shiny silvery ball? He's just at the age when his primal way for exploring the world is tasting...* In a few days' course, you mostly overcome your anguish as you do not find any rogue drop of mercury. Still, if you were asked, *Are you really sure there are no more mercury drops left there?*, you would not be able to answer "yes" in a completely convincing way. Indeed, what actually happens, is that you set up a mercury-free mental model of the kitchen and just give up worrying: you decide to live your life according to that model.

The method you apply here to come over your mercurial anxiety is called *partial induction*. It's not suitable for providing *evidence*; its proper use is *modelling*: when repeated experiments demonstrate common phenomena, we can craft a model that can be used for prediction based on the assumption that the phenomena are stable. Mathematical constructs need evidence; however, what has happened with respect to the formalization of computing, that was just modelling. To gloss over this gap, CTT was born: tentative models of computation were canonized on the premise of being able to represent any actual computation people came up with. In lack of evidence, this canonization had to be an explicit exception. The primordial CTT declares *the forced canonization of certain computational models.*

Opting out of the rite of passage

Most takes on the CTT warm up with defining a set of symbols, a universe of discourse, and various means to perform formal computations with these

artifacts. That is, they pick a computational model according to their preference set. Then they build up the standard traits of a computational model in order to justify their choice, including representation theorems that show that an already established model has the same computational power as the current one (in case of various CTT flavors "same" does not necessarily hold, but let's now just stick with the primordial CTT). Then they have the confidence to get into gourmandry, and discuss how their model excels in certain semantical or structural aspects.

This is the rite of passage in the tribe of Turing-capable computational models.

It is true that to be able to make arguments about CTT (at least, about its mathematical aspects) we need to pick a representation. However, by the model being the starting point, and justification coming after, these attempts are destined to stay within the existing confines of the CTT landscape.

Instead of further iterations of modelling, I feel it would be much more exciting to raise the bar and aim at evidence. Even conditional evidence, subject to some other hypotheses – it's much more exciting question, "What other (maybe more expressive or fundamental) hypotheses can / could / do we have that make CTT evident?" Because CTT as such is not a well designed hypothesis. Actually it's not designed or thought out at all, it's just a stop-gap measure born out of sheer necessity.

For such an attempt, the process should be the reverse. Instead of first presenting a model and then defending its virtues, we should start with thinking over what can we say about the right model, looking for evident (or conditionally evident) characteristics, then once we have a basic vision, refine, tweak and transform it to the point that we arrive to one of the friendly well established computational models.

One would start from the usual starting point, trying to delineate what an algorithm (or effective procedure) is.

OK, even before that, we have to introduce one concept, dubbed *finitary*. The word "finitary" sounds quite artificial, but it's a justified construction, as it has a stronger meaning than the related term "finite". Finitary means finite down to the bottom. The collection of circles of numbers one meets at school – natural, integer, rational, real – is a finite collection but not finitary, as the members of the collection are infinite. Being finitary sums up to being possible to draw a representation of the thing on a piece of paper in its entirety.

So, algorithms. They are usually introduced in a way similar to this:

> Given a universe of finitary objects, an algorithm is some finitarily coded set of instructions which can be unambiguously performed on any object of the universe, which either terminates by specifying another object as result, or runs ad infinitum.

First challenge: what universe of finitary objects? While the choice might be ad hoc, there is no problem with that. We can choose the finitary domain to be represented by numbers, strings, graphs, just as we want – the concept of finitarity is quite unambiguous, and seeing the equivalence of the various options by providing representations back and forth is easy.

So we have the arbitrary but fixed universe at hand, so far so good.

And then comes the lame stuttering. The circular explanation of the intuitive concepts we are to deal with in terms of each other. Well OK, I am a bit wicked here, because a decent exposé takes care of not being circular. The problem is: it just could as well be circular. These intuitive concepts are sort of "equi-intuitive", there is no clear advantage of tracing back one to the other; we remain at the same level of informality.

In contrary to a "scientific" explanation, what happens during an informal explanation is that we use descriptive terms bearing resemblance or being borrowed from natural language and we let ourselves be engaged in a language game, freely mixing intuitive terminology with semi-abstract terminology and examples; we let loose, letting circularity and self-referentiality flow free. We play this game up to the point that understanding glows. At the end of the day, you either know what an algorithm is or you don't know – if you have to ask, then it's too bad.

So, the bad news is that this circularity is inherent to the concept-set of algorithm. The good news is that circularity has the power to resolve itself. The bad news is that tapping into this power is hardly an option with our scientist hat on.

This is the psychological pivot point. Our despair is that of the vicar's in a swinger party. We see it's being done, we would like to do it, but we can't. Then at the deepest precipice of our despair, panicked, suddenly we make an audacious move of defense and we think we won. That's how we get tricked – this is the very moment that we lost the game.

This audacious move is: cutting through circularity by means of pinning down what those instructions or steps are; how they can combined and how they are coded. Pinning down precisely. Man, *precisely*. No ambiguity left, vagueness is down with. Hurray, we made a *model*!

Welcome back to the Turing tarpit.

Rewind. Forward, slow motion... stop! Again we are at the pivot point. Can we not think of any other option than regress to modelling?

Let's look for prior art. There is another core concept of mathematics that's considered quite tame these days, but on a closer look, it has a lot of common with algorithms. Look at the definition of continuity (no, it's not yet the referred concept, it's just an example):

> A function $f : \mathbb{R} \longrightarrow \mathbb{R}$ is *continuous* iff for each $x \in \mathbb{R}$ and each $\varepsilon > 0$ there is $\delta > 0$ so that f maps $(x - \delta, x + \delta)$ into $(f(x) - \varepsilon, f(x) + \varepsilon)$.

This is a very basic and innocent definition. However, having followed me so far, I ask the kind reader: don't you find anything suspicious, anything grotesque about it? Is this definition not *too smooth* by any chance?

The grotesque thing is that it speaks of *a function* with the same kind of uncontested banality as you would speak of the tea cup on the table. When we try to digest the above definition, most likely we have harder time with parsing the δ and the ε and untangling the quantifiers and recording their relation than with the mention of a function in passing. After all, a function is... just a function. Oh really? Does it not sound *circular*? Alarm rings. What is a function?

> A function is a mapping between two sets of objects that establishes a one-to-one correspondence.

Oh, no... this is a far worse case of lame stuttering than our above attempt to describe what an algorithm is. The analysis we've done for that description (up to the pivot point) could be repeated word-by-word for this case.

How can it be that we don't feel the same despair?

Well, that despair was there, historically. You can read up on the struggles of Euler, Cauchy, Dirichlet, Riemann and their contemporaries in Ferreirós (2007). It was a big problem for them as the earlier "a function is a curve" type of naive approach was not possible to maintain due to unexpected outcomes of limit constructs that become available with the advancement of mathematical analysis.

However, it's not a problem for us. Why not? What happened since then? Regarding the why: because as of now, the concept of function possesses a *formal definition*:

> A function is a set of ordered pairs so that there are no discint ordered pairs in the set with the same first components.

And that what happened since then is the Foundation of Mathematics, which based mathematics on set theory[2]. We have sets now, so we can just track back functions to sets in a snap.

Embracing the sets have other far reaching impacts on mathematical onthology. It made us easy to have abstract structures – these days, when we speak of groups or topological spaces, we can just nonchalantly say, "a group is a set endowed with some operation such that this and that holds", or "a topological space is set endowed with a family of subsets such that this and that holds". We don't have to be concerned about representational issues, about the identity of the

[2]The case of functions that we use here as an example is characteristic: Foundation happened largely as an attempt to remedy the grief said gentlemen had with the concept of function.

elements of the base set. This kind of separation of structure and representation was not possible in the pre-Foundation age, and it was also a major obstacle for the emergence of abstract algebra and topology.

Back to algorithms. So then, catching up with the trend, the plan would be to use the set theoretical Foundation to pull algorithms into the formal world.

As we noted, the finitary universe is quite a tractable and solid concept (although the representation might alter). However, what about *finitary mathematics* – all the mathematical statements we can make about finitary objects?

Set theory has its finitary fragment. The set theoretical formal counterpart of finitary is called *hereditarily finite*. So set theory, first of all, makes a choice with respect to the representation of the finitary universe, leaving us with the set of all hereditarily finite sets (that we denote by V_ω). The novel thing is that set theory is powerful enough to capture finitary mathematics as well: the formal counterpart of finitary mathematics is the first order theory of $\langle V_\omega, \in \rangle$ (read: "V_ω as a first order structure in the language of set theory" – in the sequel we'll use V_ω to refer to the set and to the first order structure on it interchangeably).

Thus, while we did not get a formal counterpart for algorithms, we got a formal counterpart for a broader range of intuitive mathematical constructs, that is, for the finitary ones. We are not yet quite there where we want to get, but the formalizations we established so far are evident.

How evident? Not as evident as the case of distance, because unlike distance, we rely on things that specific to the set theoretic Foundation (which can be disputed). However, we have the same kind of evidence as in the case of functions. This might be taken with doubt by someone who has gone through a set theory primer but misses a hands-on experience with set theoretic constructs, as the formalization of functions in terms of ordered pairs is strikingly evident, while the mechanics of formalization of arbitrary mathematical constructs to the first-order language of set theory is less so. Still, striking or not, the degree of evidence is the same in the two cases.

To wrap it up: the concept of algorithm is still a mystery, but at least we know that each particular algorithm (being a finitary construct) has a formal counterpart which is a first-order formula of V_ω.

Some readers might have the impression that what we are in an attempt to carry out a fraud that boils down to the following scheme:

- dump good old CTT as a standalone hypothesis
- bring in a fishy new hypothesis that's similar in power, but uses more hip terms like "finitary"
- establish CTT relying on the new hypothesis
- have the face to call all of this *evident*

The problem with this evaluation is that there is no *new* hypothesis. One can argue about the merits and trade-offs of selling out, but once being sold out, this becomes pointless; then, what it makes sense is to harvest the merits of the sellout and find optimal ways to live together with its trade-offs. Modern mathematics is sold out to set theory for long. I'm not making new hypotheses, I'm just trying to point this out.

If someone has still doubts, I'm open to discussion; all I ask that before giving sound to criticism, please think it over what's your proposal for capturing the notion of a function without relying on sets; and how you plan to supply algebraists and topologists with a suitably abstract ontology without relying on sets. Because if we choose to rely on sets, my "new hypothesis" *is* evidence.

We'll get back to this, but before that, let me complete my cunning scheme – the "establishing CTT" part is still to be done.

The sermon of the decision procedures

The first trick we do, a very old one, is to trace back computability to decidability. An algorithm is a *decision procedure* if on any input, it halts with a result that is either `True` or `False`. A property of finitary objects is *decidable* if there is a decision procedure such that on objects that have property its result is `True`, on objects that don't have the property its result is `False`. We can see easily that a function is computable if and only if its graph (the set of $\langle x, f(x)\rangle$ pairs) is decidable: the non-trivial part of it is showing that decidability of the graph implies computability of the function. For that, we define an algorithm to compute f: given a finitary object x, we enumerate all the finitary pairs of the form $\langle x, y\rangle$ (y being an arbitrary finitary object), and for each of them, we perform the graph's decision procedure. If the decision procedure evaluates to `True`, we give back the respective y as result.

I emphasize this was an intuitive reasoning. Our intuition regarding finitary and algorithms is good enough to be able to carry out such reasonings – we can do simple math without a formal version of algorithms. All we used here is that the finitary universe can be enumerated and that we are able to define algorithms in terms of others. ("Simple math" ends when we want to make statements that involves the entirety of algorithms, most prominently to make negative statements regarding computability or decidability.)

So a formalization of decidability would do the job: in the set theoretic context, if we have a formal definition of decidability, we can just dumbly say, "*Definition*: a function $f : V_\omega \longrightarrow V_\omega$ is computable iff its decidable" (appealing to the feature of set theory of identifying the function with its graph).

We'll use the following terminology: given a property P, an *instance* of P is the question "Does x have P?" for some object x. For example, "Is 6 a prime?" is an instance of primality.

We can say that a decision procedure for P is *a systematic reduction of instances of P to a query about a finite constellation of elementary facts.*

Let's decipher this.

By an *elementary fact* we mean that basic kind of relation or structural adjunction of objects that gives their identity; the kind of relation that we check as a step of an algorithm. In graphs, having an edge between to vertices is an elementary fact. In ordered sets, the order of two elements is an elementary fact. In arithmetics, the fact that the value of arithmetic operation on two numbers is a certain third one, like $2 * 3 = 6$, is an elementary fact.

A *finite constellation of elementary facts* is a listing of some objects and elementary facts about them. In arithmetics, a $k \times k$ multiplication table is such an example.

A *query about a finite constellation* is a question about the objects involved that can be answered simply by checking the elementary facts of the constellation. Like in the case of the $k \times k$ multiplication table, we can ask, "Does n occur in the table?" To answer this, we don't need to know the general definition of multiplication, we directly check it from the table.

So if P would be compositeness (of integers), the instance "Is 6 composite?" can be *reduced* to: "Does 6 occur in the 5×5 multiplication table?"

By *systematic reduction* we mean that we have an uniform approach to deal with the certain instances. In case of compositeness, we can perform the reduction of the instance "Is n composite?" to "Does n occur in the $(n-1) \times (n-1)$ multiplication table?", for *any* value of n.

Let's see a more complex example. Graph planarity: the graph G is planar if its vertices can mapped to the plane so that the line segments between the mapped vertices which corresponds to edges of G do not intersect.

While the plane as such is out of the finitary realm, it's easy to rephrase planarity in terms of finitary constructs: it suffices to consider only maps to points of integer coordinates, and intersection can be expressed by arithmetical equations. Such grid-planar maps could serve as finite constellations for positive instances; however, there seems no evident way to capture non-planarity with a finite constellation.

There is though the Kuratowski theorem, according to which a graph G is planar if and only if neither the five-node complete graph (K_5), nor the 3+3 node complete bipartite graph ($K_{3,3}$) is a topological subgraph of G. So a decision procedure for planarity can be obtained by the enumeration of all the topological subgraphs of G and checking whether they happen to be K_5 or $K_{3,3}$.

How is it a reduction to a query on a finite constellations? If G is non-planar, we could say the finite constellation is the K_5 / $K_{3,3}$ subgraph itself, and the query would assert the isomorphism to either of these. However, now planar graphs became a problem, as this reduction does not apply to them.

Resolution: let the finite constellation be *the run of the decision procedure* on the given instance; in this case, the iteration of the K_5 / $K_{3,3}$-checker routine over the topological subgraphs of G. This iteration itself can be objectified, and turned into a finite constellation with sufficient amount of support objects getting drafted.

We can always represent the run of a decision procedure on an instance as some finite constellation, and make a query regarding the result – that is modelling.

So:

- A decision procedure for P boils down to finitary verification / refutation for instances of P.
- The above statement is established by a reference to modelling.
- We speak of modelling, but we are happy to do it in abstract terms. We do not make any assumption on the form of modelling, in order to preserve genericity.
- The way I expressed these things can be deemed too vague (for some meaning of vague); however, the phenomenon that decision procedures have a "finitary trace" is essential – no matter how we tweak it, an algorithm can do only finitely many things up to ending with a result –, and that's my main point.

The formal counterpart

The kind of things we bet our luck on – finite constellations of elementary facts and the queries on them – turn out to hit the jackpot: they have an evident formal counterpart.

To see this, first thing to understand is that set theory is *founded*. A set's identity is determined by its elements, the identity of which in turn are determined by their elements, etc. This kind of recursive resolution of identity will eventually terminate (in any system of set theory worth to this moniker), winding down to the empty set[3]. To put it succinctly, in set theory the identity of objects is established by looking down; with the terminology of formal language theory, sets are context-free; what a set is that does not depend where we put it.[4] For example, whether a set is a function (a set of ordered pairs with well-defined second components), that's intrinsic: it depends only on its elements, ie. on its identity.

[3]Or in some approaches, "urelements", a distinguished variety of empty sets. That does not make an essential difference, in this discussion we stick with strict extensionality.

[4]The use of sets basically is representation. In the context of a specific representation, that what a given set represents might depend on the context. However, that's an externally added layer; here we speak of intrinsic features of sets.

Sets which carry the full identity of their members should be closed with respect to the aforementioned identity resolution, that is, if x is in such a set U, and $y \in x$, $y \in U$ should also hold. Sets that meet with this criteria are called *transitive sets*.

Transitive sets are also complete encyclopediae of the elementary facts (in set theory, occurrences of membership) that hold among their elements. So finite transitive sets are the suitable set-theoretic counterpart of finite constellations.

An "instance of a property" is also offers itself to formalization, it just becomes "a valuation of a formula".

So, we would consider a formula $\varphi(x)$ to be one that describe a decision procedure if for each valuation $v : x \mapsto V_\omega$ there is a finite transitive set U, such that... such that what? What should the *query* on the finite constellation be morphed into?

What we want to express is that the calculation the truth value of $\varphi[x/v]$ is in some sense local to U; on the course of doing this calculation, we rely only on elementary facts within U.

This can be captured by means of context invariance. That only U matters means that whatever context we add to U, whatever alternative universe it is placed into, the truth value of $\varphi[x/v]$ will be the same.

These placements should of course keep U's identity intact (otherwise, it's not U anymore that we are placing). So we have to consider embeddings of U into other structures so that no rouge element sneaks in below U. Such embeddings are called *end extensions*.[5]

For the invariance of truth value within a class of structures there is also standard terminology: *absoluteness*.

Thus we arrive to the following formula / set property as a formal counterpart:

> **Definition** (Henk (2009)). A formula $\varphi(x)$ of the language of set theory is called *finitely determined* if for each valuation $v : x \mapsto V_\omega$ there is a finite transitive set U that includes $v(x)$ and $\varphi[x/v]$ is absolute over end extensions of U.
>
> A subset of V_ω is finitely determined if it can be defined by a finitely determined formula.

It turns out that it is equivalent to a CTT-blessed formal version of computablity:

> **Theorem** (Henk (2009)). A subset of V_ω is finitely determined if and only if it's Δ_1.

[5] The definition of end extension is as follows: take a first order language which has a binary relation symbol $\lhd$. If $\mathfrak{A}$ is a structure of this language and $\mathfrak{B}$ is a substructure of it, we say $\mathfrak{A}$ is an end extension of $\mathfrak{B}$ iff $a \lhd b \in \mathfrak{B}$ implies $a \in \mathfrak{B}$.

Proof. The proof basically follows the thought behind the description of a decision procedure as a "systematic reduction of instances of to a query about a finite constellation of elementary facts", only in the formal context of finitary set theory. For details, see Henk (2009).[6]

The Cthulhu of all sets

Let's do a summary (revisiting some sidekicks).

It turned out that the characteristics of computation can be easily represented in the first order theory of hereditarily finitary sets.[7]

[6] This theorem, as far as I know, is original work of mine.

However, the arguments and strategy for finding a formal counterpart can also be expressed with respect to enumerability (where an enumeration of P could be described informally as an "existential statement regarding a finite constellation of elementary facts"). For that, the formal counterpart would be the formula / set property called *positively finitely semi-determined*, that differs from finite determination by asserting the existence of the transitive U only for positive evaluations. The equivalence of being positively finitely semi-determined and Σ_1 is a well known fact of recursion theory.

[7] Before drawing any conclusion, let's stop here for a short intermezzo on the "how".

On the course of establishing the representation, we did two things that are quite unusual:

- We've done some explicitly informal mathematics (when tracing back computability to decidability). We can see the very same train of thought, with similar lingual informality at several places (courses, textbooks); however, at those places there is always a tacit assumption that we work in standard, safe, formalized maths, we just use informal constructs as abbreviation or added color; if there is any ambiguity, we can go back to a more formal level to resolve. Even those who refuse ZFC-based foundation (intuitionists, constructivists) have their safety belts on by disallowing certain linguistic-logical constructs. Since the Russel paradox, unsafe, unbound informality is frowned upon.

  Regardless, I just declared informality and went on with my reasoning. This was partly a constraint, as we are forced to make arguments about the informal when discussing CTT; but also it was an interesting experiment: despite the reputedly elusive nature of informal computability, we successfully made arguments about it! Probably it would be interesting to discuss what it mean epistemologically, what type of evidence can we infer; personally I tend to have an anything goes approach, according to which if it's not the case that it obviously does not make any sense, then it's OK.

- We went deep into the discussion of a formalization attempt. Usually such a thing is not done, because either the formalization is evident, or if not evident right away, the there is an expectation that newcomers will be indoctrinated on the go, or the informal side is simply just ignored due to a heavy bias toward the formal.

  Again, this was a constraint, given CTT *is* a plea for a formalization attempt. This cannot be measured in the same way as math is measured (even informal math); this is already half poetry where not strictly rational delivery of ideas is OK (also see my comments on circularity wrt. "definitions" of algorithm). Here I stop saying anything goes (because that's not true in poetry!); I'm completely clueless and I think it is even more interesting epistemologically how would it be possible to evaluate a discussion of this type. I just point out once more it's not pure math so turning down my approach by blindly applying mathematical criteria is not a beneficial approach.

Does it mean we verified the CTT? Not in its purity, because to see any use of this fact, we need to believe that the first order theory V_ω captures finitary mathematics.

One possibility is to make it a hypothesis (call it now V_ω hypothesis) and interchange CTT to it. While I called the V_ω hypothesis "fishy", and this possibility a fraud, that was just for giving emphasis to my upcoming thought (to which we'll return shortly); indeed, this interchange seems to be quite favorable. As noted (and not only for the drama), I think CTT is not a well chosen hypothesis. Its scope is narrow and its theme is quite specific; it was not made with the purpose of cleaning up our tacit assumptions regarding Foundation, it was just a patch to mend up methodological inadequacy upon the urging need of doing math on algorithms. It's like treating your washing machine with a fair kick when it gets stuck.

The V_ω hypothesis is much nicer. It's generic (limited by size, not by topic). It's phrased in context of set theory, which makes it scale (as there are bigger universes along the ordinal scale, there is space for strengthening). It sanctifies a representation which is already a comfortable and familiar working environment. Its two sides – the informal and formal – seem to be unambiguously chosen and map well to each other (unlike CTT, where a dozen frameworks and minor dialects compete for the role of the formal counterpart). It's also not as big as to go out of control – it's the last grade on the Jacob's ladder of set theory from where we still can see the ground. V_ω seems to be the all-over over-all Mr. Nice Guy of representation hypotheses.

Only if it had any use. Postulating the V_ω hypo is like reacting to the fire alarm by turning it off.

The real question is representation of mathematics in set theory. It's not a far-fetched philosophical abstraction – as it was pointed out with respect to functions and abstract structures, it interweaves our working days. And then we didn't even mention the Axiom of Choice. So if something deserves a representation hypothesis, then it does.

But it doesn't even has a name or an accepted formulation. Why is it the case?

Partly, I think, it's a historical accident. CTT just was there in the right place and right time to reap the good press. The need for it was manifest. Also it is a good toy: the basic problem is easily worded, it's easy to understand and new approaches are cheap to create. It took all the limelight. CTT is the PHP of the philosophy of mathematics. Anyone can get a gig.

Meanwhile, under the surface, unspeakable horror dwells.

We gambled all our mathematics on set theory but we don't even have words to what happened. If we try to speak about it, people will think we've gone nuts: "Set theory captures mathematics" *is* nuts, because set theory is mathematics, therefore set theory captures set theory, so we end up with things like the set of all sets. So then, set theory captures mathematics, except for... what? There

is no obvious exception. The furthest point where people go is to speak of "Set theory captures mathematics" in quotes, or smilies appended – the You-Know-Which statement of which we know that there is something along those lines, but we don't say because people will think we've gone nuts.

We gambled all our mathematics on set theory but we don't know to what extent. But I stop making metaphors here, as the perfect one is already made. The Cthulhu mythos (Lovecraft (1963)) tells it all – Lovecraft captures "set theory captures mathematics".

That's good news, because then we can at least give it a name. It will be much easier to start a discussion once we know how to call it. Let's call it the Cthulhu Hypothesis.[8]

If we give an acceptable formulation even to a fragment of the Cthulhu Hypothesis, we can laugh at the CTT.

References

Boker, Udi, and Nachum Dershowitz. 2008. "The church-turing thesis over arbitrary domains." In , ed. Arnon Avron, Nachum Dershowitz, and Alexander Rabinovich, 199–229. Berlin, Heidelberg: Springer-Verlag. http://dl.acm.org/citation.cfm?id=1805839.1805851.

Dershowitz, Nachum, and Yuri Gurevich. 2008. "A natural axiomatization of computability and proof of Church's Thesis." *Bull. Symbolic Logic* 14: 299–350.

Ferreirós, José. 2007. *Labyrinth of thought*. Basel: Birkhäuser Verlag.

Henk, Csaba. 2009. "A model theoretic analysis of the Church-Turing Thesis." http://goya.ceu.hu/record=b1138232.

Koestler, Arthur. 1986. *The Sleepwalkers*. Harmondsworth Eng.: Penguin.

Lovecraft, Lovecraft. 1963. *The Dunwich Horror and Others*. Sauk City: Arkham House.

Németi, István, and Gyula Dávid. 2006. "Relativistic computers and the Turing barrier." *Appl. Math. Comput.* 178: 118–142.

Shapiro, Stewart. 2006. . In *Church's thesis after 70 years*, ed. Jan Wolenski Adam Olszewski and Robert Janusz, 420–55. {Ontos Verlang}title.

Sieg, Wilfried. 2006. "Gödel's Conflicting Approaches to Effective Calculability." In *CiE*, ed. Arnold Beckmann, Ulrich Berger, Benedikt Löwe, and John V. Tucker, 3988:536–537. Springer.

[8] In Henk (2009) I make attempts to give sensible formulation of the Cthulhu hypothesis, see the Short Manifesto and the Long Manifesto, but discussing those is out of the scope of this writing.

———. 2007. "On mind & Turing's machines." *Natural Computing* 6: 187–205.

Surányi, László. 1997. *Metaaxiomatikai problémák*. Budapest: Typotex.

Syropoulos, Apostolos. 2008. *Hypercomputation*. New York: Springer.

Yao, Andrew Chi-Chih. 2003. "Classical physics and the Church–Turing Thesis." *J. ACM* 50 (jan): 100–105. doi:10.1145/602382.602411. http://doi.acm.org/10.1145/602382.602411.