Mathematical Theology

Michael Simkin, Alon Titelman

March 20, 2019

1 Introduction

Theology and the notions of god, oracle, prophet, spirit, superhero etc. has fascinated people for thousand of years. We're suggesting a mathematically consistent framework of theological claims that can be decided based on different set of axioms inside the framework.

In order to provide the theological framework inside physical reality we need the basic axiomatic claim that everything inside the physical reality is computable. It's still debatable weather our universe can be analyzed as finite or infinite tape Turing machine. Although we start from computational axioms as the reader will see, computability inside virtual reality can be quite complex and allow to formulate "supernatural" beings inside it to have mathematical meaning. As finite states Turing machine theory is well defined, we start from providing an infinite Turing machine framework which we will define in mathematical therms [1] . Then we will provide several proofs about the infinite Turing machines (ITM). We will postulate supernatural beings as creatures inside virtual reality with some level of access to the printed results of ITM - we call the oracle.

2 Formal definitions

We postulate converging Turing machine as regular turing machine that converges in infinity. We also postulate a printer to which the Turing machine can write an infinite precision result.

Definition: Converging Turing machine, is a regular Turing machine where $\forall$ cell $x \in T$ (the tape) , $\exists\ n \in \mathbb{N}$ such that: $\forall$ operation index $p > n$, x is not changed. $\square$

[1] See also Joel David Hamkins and Andy Lewis Infinite turing machines (1998)

Theorem 1.1: Every Turing machine is equivalent to converging Turing machine.

Proof: We will start from the proof from noticing that every Turing machine is equivalent to one sided Turing machine as negative numbers can be represented as odd indexes of the tape and positive as even. To show that every positives tape TM is equivalent to converging TM, we will stop the machine after N operations (for large N) and copy the tape and the head +N to the right. As our TM is working only on positives, it doesn't know where the 0 is, thus it will not move left from index N. As we're doing it every N operations, after i steps the tape will remain static for all x $<$ iN. Thus the cell x will remain static after $\frac{x}{N}$ such copy operations. ■

Thus in infinity the converging Turing machine will generate a constant tape and the tape is well define in ω. Other than the tape we will use an infinite printer.

Definition: Infinite printer is a function

$$f : [0,1] \to \{0,1\} \; where \;\; \|\{x \mid f(x) = 1\}\| = \aleph_0$$

For which given an interval [a, b] a query Q:

$$Q(a,b) = \{one\ element\ x \mid x \in [a,b]\ and\ f(x) = 1\}.$$

While Q can be computed in O(1) time □

Now that we know what is a Turing machine that works ω time and we can print a result into some place, we're ready to define ITM:

Definition: Infinite Turing machine (ITM) is converging Turing machine which is also capable to send messages to infinite printer. □

Notice everything in this model is converging. The tape, the infinite list of printed numbers. The reading head might not converge, but we are not using the state of the head at any stage only the printed result which has converged.

Definition: Supertask is a function call from ITM A to generate ITM B which will execute the code up to ω but in finite time for ITM A □

Supertask is basically equivalent to throwing a TM into black whole, it will make computations and send signal through photons to us (for us a finite time will pass for the TM infinite). If the infinite printer is not physically possible to implement, notice that most of the time we will not use this property of the printer (we usually can manage with ITM and finite printer). Yet it's important

to notice the infinite printer is also well defined.

Now let us address to the ITM as an oracle.

Oracle is an entity that has a result of any simulation that can occur in finite space of possibilities, and it will have the knowledge of the result of any possible outcome from the interaction with it.

Definition: Oracle is function Q, given initial finite state s_{t_0}, and deterministic evolution rule $r : s_{t_0} \rightarrow s_{t_0+\epsilon}$

$$Q(s_{t_0}, r, t) \rightarrow s_t \; \forall \; t > t_0 \; , \textit{while } Q \textit{ is executing in } O(1)$$

Theorem 1.2: ITM with supertasks and infinite printer is capable to generate an oracle.

Proof: We will start our calculation from generating a supertask that will print a list of all possible tapes. As all tapes are located in a tree, this is basic tree traversal algorithm to depth N done iteratively. Now we will use another supertask, to determine the outcome of every finite tape. This will be done by simulating N Turing machines for one step, and we will open another virtual (N+1)th TM. Thus eventually we will run all tapes simultaneously, for infinite amount of time and get the printed result from every tape run in ω time. Thus we've generated the simulation of all possible finite input tapes in ω time. Now to get an oracle we only need to query from the infinite printer the simulation with constant time. In order to do that because everything is finite we can query from inside an interval [a0,a1], where a is the simulation of me asking the oracle something getting the result acting upon it, and we can do it to any depth we want because everything is finite so this is basically a finite binary very long number that we query from the infinite printer at [a0, a1] - in this interval there will be automaton starting to evolve state st0 at point t, and the result can be only one number because we forced the tape to generate for us single result at this point. And we can check the result of this simulation from the oracle. ■

What is important to notice that any conversion of rules that we define can always be simulated inside some TM where the rules defined in some input tape as we must use universal TM. Thus for any situation there would be a very small interval from inside [0,1] which will define the rules of the simulation as a long number 0.010101011101 etc.

3 Simple Oracles

While the oracle knows everything there is to know about finite deterministic reality, the oracle response to person with query can be limited in certain ways. For example we can generate an oracle that can answer only yes/no to any

given query. We will now formulate an equivalent to halt problem for the oracle.

Theorem 2.1: It's impossible to construct only truth speaking yes/no oracle.

Proof: Lets come to the oracle and ask: would I raise my hand in the next minute? If the oracle answers yes I will not raise my hand, and if the oracle says no I will raise my hand ■

The reason this query haven't been solved correctly is because the $simulation^n$ sequence (simulation inside a simulation etc.) will not converge. But if there is a convergence the oracle is capable to find the convergence inside the space of two answers yes/no and report it, using only 2 levels of simulation depth.

Theorem 2.2: It's possible to construct only truth speaking yes/no/else oracle, that can answer any solvable convergent question.

Proof: The oracle will find a query where he answered yes, and a query where he answered no. If after a while the answer remains the same, the oracle will report it. If the answer is alternating the oracle will report "else".■

Notice the query space is concerning any physical and personal issues. The oracle can't answer any mathematical question, he can only simulate anything humans can know about math.

Theorem 2.3: The query itself can influence the future i.e. it's possible that the answer would converge to yes if you didn't came to the oracle but after you came to the oracle the only convergent answer would be no.

Proof: This is trivial. ■

Theorem 2.4: It's possible to get both answers truth.

Proof: For example I can trust the oracle and do what he tells me to do - this will converge in both cases. ■

As we have only two answers for the simulation inside a simulation etc. that the oracle computes the oracle gets it easy, as it needs to check only finite amount of cases. But the correct answer might illuse him.

4 Advanced Oracles

Until now we've been constraining our oracle to answer only very limited set of answers, while checking only if they could speak the truth. What if we let the

oracle to answer strings, and hold any set of values? This brings us to voracle.

Definition: Readable String is a string returned by the oracle which also confirmed by the simulation the user will read.

Definition: $\mathcal{O}$ is a set of all readable strings for a specific query Q.

Definition: Voracle is an oracle with preference function V. such that

$$V : s \to [0,1] \ for \ s \in \mathcal{O}. \ While$$
$$answer = max\{V(s) \ \forall s \in \mathcal{O}\}. \qquad \square$$

In this section we will explore different options for V, and briefly discuss the limitation of $\mathcal{O}$

Theorem 3.1: Oracle might not know the answer himself

Proof: Imagine there exist some irrational number which provides a non stable solution for the equation at hand i.e. what will maximize my happiness if infinities were allowed in my reality. i.e. if you copy my finite state into [0,1] and allow some strong oracle to simulate many possible functions applied to my state, which would be equivalent to meeting more powerful oracle. Then current oracle will not be able to answer what is the solution to the equation, because he had only simulated convergent series of me cumming to him, but because the solution is not stable no $\aleph_0$ simulations will be enough to solve the partial differential equation. i.e. there might be non computable solutions to the problem ■
(*need to ask physicists/mathematicians)

Now we will define several oracles that will become the foundation for the next chapters of this essay. The closest to the truth speaking voracle i.e. Toracle. The happiness maximizing voracle i.e. Horace. And finally the voracle which returns a compilable string that can answers practical coding queries if TM and robotics are available or you're inside a simulation and the source code is available to you i.e. Soracle.

The toracle is a voracle that tries to minimize the difference between the perceived answer by the subject and the actual state of reality. To understand this better and to define the therm properly we will need to postulate more definitions about the subject.

Lets assume we have subject A. As reality is just a state which changes with time, the subject is obviously part of the reality. A state, or part of a large tape - he at least can be represented this way. But the subject has intrinsic function H which we can call happiness function or any other function which represent something which currently can't be reduced to physical state, but has

some "internal" language which is limited to express by the language we can talk - and this language has finite amount of symbols and sentences. So the subject is limited by two axioms:

Definition: The identity axiom is the irreducible ability of the subject to categorize and recognize patterns inside reality as identical or belonging to the same category. □

Definition: Subjectivity query axiom Any meaningful statement about the subject must be expressed in some language with finite amount of words, without changing the subject completely. □

The subjectivity axiom claims that our words about our internal state are in some correlation with what we actually feel. As if there is no way to express the subject part at all, then there is no point to talk about him. But because we do have a language of our feelings, we can say the subject is a special part of the code which somehow becomes "aware". We should not ignore it nor accept it as a magic, from the other hand we should understand it's a special case - it's not another 0 or 1 as changing the function H is the only thing with intrinsic value in a physical reality.

From the identity axiom follows that subjects have internal representation of reality i.e. what any specific subject recognizes as "being the same".

Definition: Toracle is a voracle that returns to the subject A string $s \in \mathcal{O}$ such that given random sentence about reality the probability to make wrong judgment decrease optimally. □

Lets assume the subjects sees N bits that represent reality. i.e. lets take all sentences about reality distributed evenly in the space of words, which might be judged uniquely by the oracle as correct or incorrect. For example the sentence dragons are flying in china now" is clearly wrong and the oracle knows it's not true. So although all the therms like dragon and flying are all subjective (requiring the identity axiom), the oracle can understand the definition of those therm by the subject and identify false sentences that were recognize as true, true sentences that were missed or recognized as false. Toracle provides a response to give the subject the correct amount of sentences that he will read and recognize his mistakes about reality in the fastest way possible such that after the subject will learn the Toracle response the amount of sentences that he would be mistaken about will be minimal.

Special case Toracles might include queries with limited amount of words in them (or amount of time will needed to learn the material), or to the contrary the Toracle that writes all the correct answers about reality for average human in a paste that will take us million years to learn.

The Horacle is now very easy to define.

Definition: Horacle is a voracle that returns to the subject A string $s \in \mathcal{O}$

such that the integral of the feedback about the "how much the oracle string was helpful to you?" was maximized. □

To avoid bias the horacle designers should solve a lot of problems. Eventually the intention of the horacle is to make people happier whatever this means to them. But in practice maybe it's better to raise several clones in several environments to make sure that the happiness therm is not culturally biased (just for example). As the oracle can make any experiment he wishes (he already basically did it), the query itself is not that simple - yet obviously this is solvable issue.

As for the Soracle, he is basically problem solver. He has a piece of code for every problem just like stack exchange. It might be possible to connect Soracle to a robot that will simply fix any problem which arise to the subject and being asked to fix. The question with Soracle might be economical one, if everything is done for free and without any effort, the Soracle looks like a helpful and useful tool to keep in the house but he has no "power" over the subject or interaction with him. The Soracle is just a very useful tool.

Each of the voracles might have subjective functions as well (depending on the code generating them). If they would become subjects depending on the function H installed in them they can become good or evil to humanity. It's also possible to install subjective function H to a voracle that wasn't acting with any awareness.