Model of Hypercomputation: Closed Time Curve Turing Machines

Ariel Kelman & Dong Hao Ou Yang

December 4, 2020

1 Introduction

Algorithms have been studied for thousands of years. While perhaps not precise by the standards of modern mathematics, Euclid's *Elements* gives algorithms for constructing shapes using a straightedge and compass. In the twentieth century, mathematicians became interested in formalizing the intuitive notion of an algorithm. The *Entscheidungsproblem* ("decision problem"), posed by David Hilbert and Wilhelm Ackermann originally posed in 1928, asks whether there is an algorithm to determine whether a proposition from or precludes a given set of axioms. Godel's incompleteness result demonstrated that such an algorithm was impossible. Turing gave an essentially equivalent answer in his 1936 paper in which he proved the undecidability of the Halting problem.

To do so, Turing introduced an abstract machine (a Turing machine), whose functioning is taken to define computability, and to provide a precise notion of *algorithm*. Church provided an equivalent model, known as λ-calculus, that gives equivalent results. We therefore have the following definition:

Definition 1. A problem is called **computable** (Turing-solvable) if there exists some Turing machine M that is able to execute the algorithm that outputs an answer to the problem.

giving rise to the Church-Turing thesis,

Proposition 1. *A problem or a function is computable if and only if it is Turing-solvable.*

which is closer to a definition than a theorem.

Such models of computation are defined abstractly, without any reference to physical reality or natural laws. This raises the question of whether such laws could give rise to different models of computation. Might there be computational speedups? How large might they be? How do the answers to these questions depend on the natural laws (or physical parameters) of a universe? While computer scientists generally ignore the mundane considerations of how computers physically work, and traditionally ignore numerical factors that arise in the analysis of efficiency, it will turn out that the laws of nature can have effects beyond those mundane considerations. While the considerations are somewhat empirical, they change even standard results in computer science.

We shall see that such extended models of computation - hypercomputation - do in fact yield extra computational power. In the next subsection we review the basic models of computation to set the stage and indicate required background for what follows. In section 2 we discuss both the limits and gains to computation due to switching from a classical to a relativistic universe. In section 3 we introduce time travel modelled through closed timelike curves (possibly understood through the framework of general relativity, though one could propose other natural frameworks that allow time travel). After a qualitative introduction and more technical definitions, we discuss how even computability - not just computational complexity - is affected by time travel. In the concluding section we sum up some take-home lessons, and suggest some further models of hypercomputation.

1.1 Models of Computation and their Limitations

We've already mentioned Turing machines and λ-calculus, but before proceeding to hypercomputation, it will be useful to mentions several other models of computation. We also note that classical circuits provide a model computationally equivalent to Turing machines and λ-calculus.

The first of these is non-deterministic Turing machines, as contrasted with the ordinary deterministic Turing machines. On this model, the changing of states becomes probabilistic (or equivalently, one can allow a deterministic Turing machine to probabilistically write symbols on the tape, which are then read by a deterministic transition function). From a computability perspective, the two models are equivalent, i.e., an algorithm can be simulated on a deterministic Turing machine if and only if it can be simulated on a non-deterministic Turing machine. From a complexity standpoint, an important class is BPP, the class of problems solvable by a non-deterministic Turing machine in polynomial time with bounded error (this is the probabilistic analogue of P). Nevertheless, it is thought that $\mathsf{BPP} = \mathsf{P}$.

The next model is that of quantum computation. This model was first proposed (qualitatively) by Feynman [1], with the motivation that the world was quantum, but classical models of computation could not efficiently model quantum processes. While quantum Turing machines have been defined, the standard model of quantum computation utilizes quantum circuits (these models were proved to be equivalent in [2]). If (as widely suspected) quantum computers turn out to be more powerful than classical computers from a complexity standpoint, it will be an example of the general phenomena discussed in this paper - the revision of computational models based on physical reality.

2 Relativistic Considerations

The standard models of computation - Turing Machines, λ-calculus, classical circuits - to the extent that they assume a model of physics at all, the model is classical. This claim can be demonstrated by the "billiard ball" model of computation, in which classical balls bounce around to implement any classical algorithm [3, 4]. With that in mind, we can consider how our models of computation must change or become limited when the laws of physics change (or as we consider different laws of nature).

One such limitation arises from the (special) relativistic constraint that information cannot be transferred through space faster than light. Thus, as computers get bigger, communication between parts of the computer are limited by the speed of light, c. This will obviously not affect computability - a Turing machine performs only local operations, and obeys this constraint. Nevertheless, this does not lead to a fundamental limit even on physically realizable computation, as we have not (yet) postulated any constraints that prevent an arbitrarily small computer. While such limits might arise due to quantum theory, here we simply explore the limitations due to relativity.

When we add in general relativistic considerations, rigid limits can be given. One limit arises from spatial limitations, such as the theoretical impossibility of measuring anything to a finer accuracy than the Planck length, given by

$$\begin{aligned} l_p &= \sqrt{\frac{\hbar G}{c^3}} \\ &\approx 1.62(10^{-35})\text{m} \end{aligned} \tag{1}$$

where $\hbar$ is the reduced Planck constant and G is the gravitational constant. Roughly speaking, confining a particle (e.g. a photon) to such a small distance would require such an enormous uncertainty in momentum that the energy density would be sufficient to create a (small) black hole [5]. There is therefore a limit on the density of a computer (before it collapses into a black hole), and as a computer gets larger (as required to avoid this collapse), the speed of light constraint becomes important. However, such limitations are heuristic and somewhat speculative, as there is no workable theory of physics on such scales - we therefore await a theory of quantum gravity before analyzing this limit any further.

There is another bound that places a constraint on the amount of information that can be stored in any region of a given size. Known as Bekenstein's bound, it was originally argued in [6] based on

considerations involving the second law of thermodynamics. There has been debate about the precise formulation of the bound, but we follow the logic of the original paper; differences in numberical constants and such details (such as those arising from accounting for quantum effects near a black hole) do not affect the main computational conclusions. The interested readers are referred to [7] as a starting point.

Start by taking the entropy of a black hole to be proportional to its surface area ($S \propto A/4$ where S is the entropy and A the surface area; in Planck units this is equality) and consider dropping in a body with energy E and effective radius R. The surface area of the black hole must increase by $8\pi ER$, thereby increasing its entropy. Unless the entropy of the body is below (or equal to) $2\pi ER$ (simply divide the previous expression by 4), entropy will decrease through this process, violating the second law. We therefore have that for any body

$$S \leq 2\pi ER = \frac{2\pi kER}{\hbar c} \tag{2}$$

where with the equality we've switched from natural units ($k = \hbar = c = G = 1$). Since entropy S of some object is related to the amount of information I in this object via $S = kI\ln(2)$, this gives a bound on the amount of information that can be stored in a region of radius R containing energy E:

$$I \leq \frac{2\pi ER}{\hbar c \ln(2)} \tag{3}$$

and since E is constrained (both in practice and by the need to avoid gravitational collapse of the computer), a computer that must work with a large amount of information will need a large volume, and the speed of light constraint becomes relevant.

So much for relativistic constraints on computation; what about relativistic benefits for computation? Several models have been proposed to increase the computational power of computers through harnessing relativistic effects. These are typified through by following example:

1. Program a brute-force 3SAT solver into a computer, and start the computation.
2. Go spend some time near a black hole, where time passes much slower for you than it does for the computer.
3. Come back, and get the results of your computation: hopefully, you've experienced a polynomial amount of time, while the computer has had an exponential period of time to run.

Similar "algorithms" can be cooked up using only special relativistic effects, and we discuss these simpler cases. The analysis involving general relativity ends with the same result [5]. The main idea is for the computation to take place in one reference frame (or on one worldline) in which more time passes than for the observer who wants the answer to a computational problem. The Lorentz transformation specifying the effects of time dilation is given by

$$\begin{aligned} \Delta t' &= \gamma(\Delta t) \\ &= \frac{1}{\sqrt{1 - \frac{v^2}{c^2}}}(\Delta t) \end{aligned} \tag{4}$$

where γ is the Lorentz factor. This describes a stationary computer experiencing time $\Delta t'$ while an observer travelling at constant speed v experiences time Δt (we ignore the effects of acceleration, which do not change the analysis, but are discussed below). We have $\Delta t' > \Delta t$, giving some extra time for the computer to operate. It is interesting to note that this procedure requires one who wished to gain a computational speedup to do the travelling (the same is true when one tries to use gravitational time dilation - e.g. going near a black whole slows down time, so the user must leave the computer far away during the travels).

While a computational speedup can be obtained from harnessing this effect, the gain in time shown is polynomial in v. Therefore, to achieve an exponential speedup (e.g. allowing one to solve NP-complete problems in polynomial time), one would require an exponential amount of energy - just to achieve the required speed, since energy scales polynomially with speed. This introduces another issue, namely that even if such energy could be found, it would require an exponential volume to store. At that point, it would require exponential time for distant fuel to contribute (due to the speed of light limitation), obviating the theoretically possible gain [5]. Note that this does not eliminate the exponential gain once these speeds have been reached, but it means that one could not use this method to run an exponential-time algorithm in polynomial time.

This discussion somewhat pushes the boundaries on the reasonableness of ignoring physical constraints when considering computational power. When the governing computational consideration is about the time for fuel to contribute, the theoretical computer science issues are bound up in limitations of natural laws (though we are still not discussing technology - only the theoretical limits on what technology can do). Perhaps this is to be expected, as we explicitly invoked physical considerations, but it does mean that we've left the CS ivory tower somewhat.

As mentioned above, accounting for acceleration does not change the analysis. In that case, the time dilation is given by

$$\Delta t = \int \sqrt{1 - \left(\frac{v(t)}{c}\right)^2} dt \tag{5}$$

where again Δt is the time experienced by the traveller moving at $v(t)$. This velocity is measured in the frame of the (stationary) computer, which experiences time $\Delta t' > \Delta t$. Determining this velocity can be somewhat involved; in the case of constant proper acceleration a, it is given by

$$v(t) = \frac{at}{\sqrt{1 + \left(\frac{at}{c}\right)^2}}. \tag{6}$$

The same result can be seen above - to get an exponential speedup, one must accelerate to exponential speeds. Proving that this is in fact that case for arbitrary $a(t)$ seems to be a lacuna in the literature. Nevertheless, it is true, e.g., for simple cases, such as 4 periods of constant acceleration as described at [8].

In the final analysis, relativistic considerations do not seem to lead to computational speedups of the sort that interest the theoretical computer scientist. While polynomial speedups may be nice, we leave it to the engineers to decide if this will ever be the most efficient way to get that limited speedup. We therefore turn to examining other forms of hypercomputation.

3 Closed Timelike Curves (CTCs)

3.1 Introducing CTCs

Now we can consider a more exotic modification to the laws of physics - time travel. Naively, one might consider a computer that can send itself arbitrary messages (perhaps limited by memory storage, which we conveniently ignore in the tradition of theoretical computer science) back in time. On this model, everything (computable) becomes constant time: the computer takes as much time as is required, and then sends the result back in time. Perhaps even the halting problem can be solved, it time travel allows the computer to run for an infinite amount of time while only a finite amount of time elapses for the user.

However, this model is boring (everything is computable in constant time), and is a little too generous with what we allow from time travel. For instance: do we require any sort of consistency condition to avoid grandfather paradoxes? For illustration, at the very least we might require that after the user gets an answer, they don't turn off the computer until the computation-time elapses -

otherwise the answer to a difficult computation question simply popped into existence without ever having actually been solved!

We therefore restrict our analysis to more limited time travel, and require some actual model (even if not physically complete) that addresses these sorts of paradoxes[1]. In particular, we consider closed timelike curves (CTCs) in a quantum universe, as proposed by Duetsch [9]. [2] In this model, roughly speaking, it is required that we model the state of the universe probabilistically, and that state is a "fixed point" of the evolution operator of the universe. Below, we flesh out the idea in more detail, but it is useful first to have a qualitative picture. One solves the Grandfather paradox on this theory by showing that A is born is born with probability 0.5, and that *if* one is born, one (certainly) goes back in time to kill A's grandfather. We therefore have that the grandfather dies with probability 0.5, and so A is born with probability 0.5. Probabilistic consistency!

Strictly speaking, this may not even require positing a change to the laws of nature, as closed timelike curves are compatible with the field equations of general relativity. The first such solutions were found by Godel in 1949 [10], but that solution, known as the Godel metric, has characteristics that are thought not to hold for our universe (such as no Hubble expansion). As mentioned in the discussion of relativity, however, physics awaits a model that can accommodate both general relativity and quantum mechanics, so it is premature to conclude that CTC-computers are a physically realizable possibility.

Before turning to the quantum case, we introduce the classical Deutschian CTC Turing Machine. What follows is based on [11]; more discussion, including of other CTC complexity classes, can be found in [12].

Definition 2. *A **classical Turing Machine with a Deutschian CTC**, or* $\mathsf{TM}_{\mathrm{CTC}}$*, is a non-deterministic (i.e. probabilistic) Turing machine M whose memory tape is divided into two parts:*

- *a "causality-respecting" register (tape) $\mathcal{R}_{CR}$, which contains the inputs of M,*
- *a "closed timelike curve" register $\mathcal{R}_{\mathrm{CTC}}$,*

and that has an operation that set the current square to 0 *or* 1 *with equal probability of* $1/2$*.*

Both registers contain infinitely many squares and, just as ordinary Turning machine, finitely many steps in a computation can only access to finitely many squares. Thus, the tape contents of a $\mathsf{TM}_{\mathrm{CTC}}$ is a pair of binary strings (x, y), whose lengths are finite, such that x is on $\mathcal{R}_{CR}$ and y is placed on $\mathcal{R}_{CTC}$. In the following, we shall write $(x, y) \in \mathcal{R}_{CR} \times \mathcal{R}_{CTC}$ to indicate that input x is written on $\mathcal{R}_{CR}$ and y is written on $\mathcal{R}_{CTC}$.

In addition, we require that the machine M satisfy the following condition to be a $\mathsf{TM}_{\mathrm{CTC}}$:

(i) M must halt at some point and output 0 or 1 for any input pair $(x, y) \in \mathcal{R}_{CR} \times \mathcal{R}_{CTC}$.

This requires that for any input x to $\mathcal{R}_{CR}$, there will be an infinite dimensional stochastic matrix $S_x : \{0,1\}^* \to \{0,1\}^*$ that M induces on $\mathcal{R}_{CTC}$. Let $\mathcal{P}$ be a probability distribution on $\{0,1\}^*$. Then, we define $\mathcal{P}$ to be a **fixed point** of S_x if

$$S_x(\mathcal{P}) = \mathcal{P}. \tag{7}$$

[1] Different models of time travel are used in various fictional narratives. The Harry Potter and Artemis Fowl series/universes require a consistency condition - "if someone goes back in time, it's as if they've already gone back" - though that's done in a deterministic universe (or at least, no lack of determinism is specifically stated). The Umbrella Academy adopts some other model, and it seems as though the timeline actually changes each time time travel occurs.

[2] It's not clear whether this model really solves the paradoxes of time travel, without a guarantee that the same result occurs on each run through the timeline (such a guarantee would go against the spirit of the model). Otherwise, paradoxes can still occur. Perhaps this issue is solved by a Many Worlds interpretation of quantum mechanics, since on that picture all possibilities are always (in a timeless sense) present. Deutsch explicitly argues for Many Worlds on this basis in [9]. In any case, this would still require that in a particular World, different results can happen the "second time through the timeline". As I recall reading somewhere, English grammar doesn't have the resources to handle time travel.

In other words, $\mathcal{P}$ is a fixed point of S_x - the action of S_x leaves $\mathcal{P}$ unchanged. S_x essentially defines the part of the computation that involves time travel.

We further require that the machine M must satisfy the following conditions to be a $\mathsf{TM}_{\mathrm{CTC}}$:

(ii) the stochastic matrix S_x has at least one fixed point $\mathcal{P}$ for every input $x \in \{0,1\}^*$ (this condition is required to handle the case where S_x is infinite dimensional; in the finite case a fixed point is guaranteed),

(iii) for any input (x, y), the operator S_x either accepts on all of its fixed points or rejects on all of its fixed points.

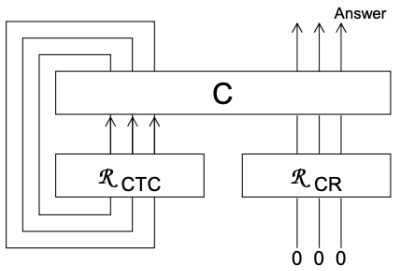

Figure 1: A graphic representation of CTC machine. The circuit C acts on both $\mathcal{R}_{CR}$ and $\mathcal{R}_{CTC}$ registers. The condition (ii) indicates that Nature must find a probability distribution on $\mathcal{R}_{CTC}$ that is a fixed point and then output from $\mathcal{R}_{CR}$.

By **accept**, we mean that, given an input x and a probability distribution $\mathcal{P}$ on $\{0,1\}^*$,

$$\Pr_{y\sim\mathcal{P}}\big(M(x,y) \text{ outputs } 1\big) \geq \frac{2}{3},$$

and, by **reject**, we mean that

$$\Pr_{y\sim\mathcal{P}}\big(M(x,y) \text{ outputs } 0\big) \geq \frac{2}{3}.$$

Finally, let $L \subseteq \{0,1\}^*$ be some language. The machine M **decides** L if M accepts any $x \in L$ and rejects any $x \notin L$. We denote the class of all languages that can be decided by some $\mathsf{TM}_{\mathrm{CTC}}$ as $\mathsf{Computable}_{\mathsf{CTC}}$. This suffices as a model of classical (probabilistic) computation with CTCs.

Before continuing, it may be helpful to give a higher level overview of how such a $\mathsf{TM}_{\mathrm{CTC}}$. It accepts input x on a classical register, and writes an output y on the CTC register such that when the $\mathsf{TM}_{\mathrm{CTC}}$ runs, via the probabilistic transitions S_x, (x, y) is a fixed point of the computation. Thus, in a sense y is both the input and output to the computation - it is the only stable thing that can be written on $\mathcal{R}_{CTC}$. This raises the question of when/where/by who the actual answer was computed, but it is difficult to make this into a precise paradox. We therefore chalk it up to the paradoxes of time travel.

We can also define the quantum analogue, which is a generalization of the classical version.

Definition 3. *A **quantum Turing machine with Deutschian CTC**, or $\mathsf{QTM}_{\mathrm{CTC}}$, is a machine that consists of a sequence of unitary operation on the tensor product of Hilbert spaces $\mathcal{R}_{RC} \otimes \mathcal{R}_{CTC}$, where $\mathcal{R}_{RC}$ and $\mathcal{R}_{CTC}$ has the same function as in the classical case.*

Both registers are of countably infinite dimensional size whose bases consists of vectors represented by binary strings $|p_1, p_2, ...\rangle$, where $p_i \in \{0,1\}$. For a given normalized state $|\psi\rangle = \sum_{x\in\{0,1\}^*} \alpha_x |x\rangle$,

we can use a quantum Turing machine to act on it, which is a finite control. That is, there is an internal finite-dimensional Hilbert space such that any unitary transformation contained in this quantum Turing machine can act on only $O(1)$ contiguous qubits. This corresponds to the standard limitations places on quantum Turing machines.

Besides tape qubits, the unitary transformation in a quantum Turing machine can also act on the qubits in the internal Hilbert space as well as the subspace spanned by the states $|L\rangle, |R\rangle, |Acc\rangle$ and $|Rej\rangle$, which are states that corresponding to moving left, right, accepting, or rejecting respectively.

Similarly, given a quantum Turing machine M and an initial state $|x\rangle$ on $\mathcal{R}_{CR}$, one can associate a superoperator $\overline{S}_x$, which maps a mixture of states, i.e. a density operator, to another one, induced on $\mathcal{R}_{CTC}$. Again, a density operator ρ is a fixed point of $\overline{S}_x$ if $\overline{S}_x(\rho) = \rho$. As mentioned above, for a finite-dimensional case, one can always find a fixed point for each superoperator but this is not true for infinite-dimensional case. In analogous with the classical machine, a quantum machine M is a $\mathsf{QTM}_{\mathrm{CTC}}$ if the following conditions are true:

(i) M always halts and "outputs" either $|Acc\rangle$ or $|Rej\rangle$ on any input $|x\rangle \otimes |y\rangle \in \mathcal{R}_{RC} \otimes \mathcal{R}_{CTC}$,

(ii) the superoperator $\overline{S}_x$ has a fixed point for any $x \in \{0,1\}^*$,

(iii) and, for any $x \in \{0,1\}^*$, the fixed point ρ of $\overline{S}_x$ either accepts, i.e.,

$$\Pr(M\,|x\rangle\,\langle x| \otimes \rho \text{ outputs } |Acc\rangle) \geq \frac{2}{3},$$

or rejects, i.e.,

$$\Pr(M\,|x\rangle\,\langle x| \otimes \rho \text{ outputs } |Rej\rangle) \geq \frac{2}{3}.$$

Again, a quantum machine M **decides** a language $L \subseteq \{0,1\}^*$ if M accepts $x \in L$ and reject $x \notin L$. We denote the class of all languages that can be decided by some $\mathsf{QTM}_{\mathrm{CTC}}$ as $\mathsf{QComputable}_{\mathsf{CTC}}$.

We should stress two striking features about the above definition of the CTC Turing machine before we move onto the extent and limit of its computability power. First of all, it is possible for the machine M to induce some stochastic operator with fixed points without a finite support, i.e., a probabilistic distribution over infinitely many strings in $\{0,1\}^*$. Also, even though each fixed point $\mathcal{P} = \{p_x\}_{x\in\{0,1\}^*}$ might have finite *expected string length*, i.e., $\sum_{x\in\{0,1\}^*} p_x|x| < \infty$, there is no upper bound that can be determined a priori.

In principle, these problems rise the issue of whether it is reasonable to construct a computer that can accept or reject (instantaneously!) a string with an enormous length, something like $10^{10^{10^{10}}}$ bits? In classical computability theory these issues are avoided by taking limits to infinity and the like. However, with CTCs, because time is not a concern, the few-step computation can use arbitrarily large space - one can't argue that only in the limit of long computation times are large tapes required. Possibly somewhat mitigating this issue is that while CTC Turing machines can solve the halting problem (as we will see shortly), they always output a finite string, and always halt. To the extent that issues remain, they are intricately bound up with the weirdness of time travel.

3.2 Tackling Uncomputable Problems with CTCs

In the previous section, we have defined both classical and quantum version of Deutschian CTC Turing machine. To demonstrate the enhance of compubability power, we will begin by showing that a Deutschian CTC Turing machine can solve the halting problem, described in section 2.

Proposition 2. $\mathrm{HALT} \in \mathsf{Computable}_{\mathrm{CTC}}$.

Proof. Let P be some Turing machine with description $\langle P\rangle$ and suppose P runs for at least n steps. We denote the string that records the first n steps of execution history by P as s_n. Then, s_0 would the empty string, indicating the blank tape and P is at its initial state, and s_{n+1} can be obtained

recursively by appending s_n with the $(n+1)$-th execution step by P. It follows that, given any string y and knowledge of P, it is easy to determine whether $y = s_n$ for some $n \geq 0$ and, if it is equal, which n it is. We will call the string s_n the **halting history** if P halts after the n-th execution and a **non-halting history** otherwise.

Now, we define our $\mathsf{TM}_{\mathrm{CTC}}$, M, that takes $(\langle P\rangle, y)$ as input on its $\mathcal{R}_{RC} \times \mathcal{R}_{CTC}$, where y is some string, and does the following:

1. If y is a halting history, then y is unchanged on $\mathcal{R}_{CTC}$ and M outputs `HALT`.
2. If $y = s_n$ for some n and is a non-halting history, then write s_{n+1} onto $\mathcal{R}_{CTC}$ with probability $1/2$, or write s_0 onto $\mathcal{R}_{CTC}$ with probability $1/2$ and output `LOOP`.
3. If $y \neq s_n$ for any n, then rewrite s_0 into R_{CTC} and output `LOOP`.

Based on these rules, we will have two possible cases. In the case that P halts, then $y = s_n$ for some finite n on $\mathcal{R}_{CTC}$. The induced stochastic matrix S_y, on the input $y \in \mathcal{R}_{CTC}$, will have a unique fixed point at the probability distribution $\mathcal{P} = \delta(x - y)$, where $x \in \{0, 1\}^*$ is any string and $\delta(x - y)$ is the Dirac function and thus will output `HALT` with certainty.

On the other hand, if P runs forever, one can easily check that S_y has geometric distribution $\mathcal{P}$ with $\Pr(y = s_n) = (1/2)^{n+1}$ as its unique fixed point. Again, M will output `LOOP` with certainty. Therefore, HALT $\in \mathsf{Computable}_{\mathrm{CTC}}$. □

3.3 Computability Power of $\mathsf{TM}_{\mathrm{CTC}}$

In the previous subsection, we have demonstrated that CTC Turing machines are much more powerful than ordinary Turning machines since they can compute a problem that is not Turing computable. In this subsection, we will show that, in fact, Deutschian CTCs have the exactly same computability power as an oracle for the halting problem, i.e.,

$$\mathtt{Computable}_{\mathrm{CTC}} = \mathtt{Computable}^{\mathrm{HALT}},$$

where $\mathtt{Computable}^{\mathrm{HALT}}$ the set of all problems that can be Turing-reducible to the halting problem, i.e., a language L is Turing-reducible to HALT if, let M be an oracle machine that decides HALT, M decides L.

We have not yet mentioned computational oracles, but an intuitive explanation will suffice: an oracle is a black-box that (usually in one time-step) provides the solution to a class of problems. Thus, an oracle for HALT provides a solution to the halting problem when queried (with appropriate input).

Before we proceed, we shall mention an important fact. Similar to the case for ordinary classical and quantum Turing machines, for CTCs, we have

$$\mathsf{Computable}_{\mathrm{CTC}} = \mathsf{QComputable}_{\mathrm{CTC}}.$$

Thus, the above result can be extended to

$$\mathsf{Computable}_{\mathrm{CTC}} = \mathsf{QComputable}_{\mathrm{CTC}} = \mathtt{Computable}^{\mathrm{HALT}}.$$

First, we shall consider the inclusion on the left.

Theorem 1. $\mathtt{Computable}^{\mathrm{HALT}} \subseteq \mathtt{Computable}_{\mathrm{CTC}}$.

Proof. Let P be an oracle Turing machine that repeatedly queries a HALT oracle. We need to determine whether P accepts or rejects, provided one of these is the case. Let $\tau_1 < \tau_2 < ...$ be the time step on which P queries the HALT oracle. Without loss of generality, we assume that interval between each consecutive step is sufficiently large so that P can always delay a query or insert an additional query when needed.

Let $\{B_i\}$ be a sequence of ordinary (non-oracle) Turing machines and s_{i,n_i} be the history of the first n_i steps of the machine B_i, defined as in the proof of proposition 2. The sequence is ordered such that B_i is the i-th Turing machine that P submits to its HALT oracle to check whether it halts. Then, consider

$$s = \langle s_{1,n_1}, ..., s_{k,n_k} \rangle$$

the k-tuple of execution histories. For $i = 1, 2, ..., k$, we define $q_i = 1$ if s_{i,n_i} is a halting history and $q_i = 0$ otherwise. s is **valid** if:

- Each s_{i,n_i} is a valid execution history (not necessarily a halting history),
- If we run P, treating i-th oracle query as returning q_i, then $B_1, ..., B_k$ are the first k Turing machine that P submits to the HALT oracle at times $\tau_1, ..., \tau_k$.

Then, we call s **halting** if P will halt by the time step τ_{k+1}. In this case, we call s **accepting** if P accepts and **rejecting** if P rejects. Otherwise s is called **non-halting**. In the case where s is non-halting, we let B_{k+1} be the Turing machine that P asks at the time step τ_{k+1}.

Let δ be the k-tuple when $k = 0$, which is the string P encodes in its initial state, and let S_B be the stochastic matrix that maps $\mathcal{R}_{CTC}$ to itself assuming that $\mathcal{R}_{CTC}$ is initialized to $\langle B \rangle$. We can now define M (a $\texttt{TM}_{\text{CTC}}$):

1. If s is not valid, then set $s = \delta$.
2. If $s = \langle s_{1,n_1}, ..., s_{k,n_k} \rangle$ is valid and halting, then set $s = \langle S_{B_1}(s_{1,n_1}), ..., S_{B_k}(s_{k,n_k}) \rangle$ and accept if s is accepting and reject otherwise.
3. If $s = \langle s_{1,n_1}, ..., s_{k,n_k} \rangle$ is valid and non-halting, then set $s = \langle S_{B_1}(s_{1,n_1}), ..., S_{B_k}(s_{k,n_k}), s_{k+1,0} \rangle$

Now, for a valid execution, P makes exactly k queries to its HALT oracle, about $B_1, ..., B_k$ respectively. Let $\mathcal{P}_1, ..., \mathcal{P}_k$ be the fixed points of $S_{B_1}, ..., S_{B_k}$ respectively, and let S be the stochastic mapping that $M(\langle P \rangle, s)$ induces on s acts on its $\mathcal{R}_{\text{CTC}}$. Then, it suffices to show that $\langle \mathcal{P}_1, ..., \mathcal{P}_k \rangle$ is the unique fixed point of S since, if s is sampled from $\langle \mathcal{P}_1, ..., \mathcal{P}_k \rangle$, then $M(\langle P \rangle, s)$ accepts with certainty if oracle accepts and rejects with certainty if oracle rejects.

It is clear that $\langle \mathcal{P}_1, ..., \mathcal{P}_k \rangle$ is a fixed point. For uniqueness, we consider the following facts. First, any fixed point must have support only on valid k-tuples. Second, if $\mathcal{P} = \langle \mathcal{P}_1, ..., \mathcal{P}_k \rangle$ is a fixed point, then $\mathcal{P}$ marginalized to its first coordinate must be a unique point of S_{B_1}. Then, by induction, $\mathcal{P}$ is the unique fixed point. Therefore, $\texttt{Computable}^{\text{HALT}} \subseteq \texttt{Computable}_{\text{CTC}}$. □

To show the converse inclusion, we will borrow one of the most important theorem from functional analysis, the *Rietz representation theorem*.

Theorem 2 (**Rietz representation theorem**). *Let $\mathcal{H}$ be some Hilbert space, possibly infinite dimensional, and let $F : \mathcal{H} \to C$ be a linear functional. Then, there exists a unique $u \in \mathcal{H}$ such that $F(v) = \langle u, v \rangle$ for any $v \in \mathcal{H}$.*

Also, we need the following definition. For any mixture of states ρ, which are some $N \times N$ matrix (assuming that the Hilbert space is finite dimensional), we define the *vectorization* of ρ be the N^2 vector whose entries are the same as ρ (write components of ρ into a single column); we will denote it by $\text{vec}(\rho)$.

Then, we have the following lemmas.

Lemma 1. Let $\mathcal{H}$ be a Hilbert space and $F : \mathcal{H} \to C$ be a linear functional. For fixed $\epsilon > 0$ and integer $T \geq 1$, let $u \in \mathcal{H}$ be a unit vector such that $\|F^t u - u\| \leq \epsilon$. Then there exists a vector $w \in \mathcal{H}$ such that $\|u - w\| \leq \epsilon$ and w is a fixed point of F, i.e., $F(w) = w$.

Lemma 2. For any mixtures of states σ, ρ,

$$\|\mathrm{vec}(\sigma) - \mathrm{vec}(\rho)\| \leq \|\sigma - \rho\|_{\mathrm{tr}} = \mathrm{tr}\,|\sigma - \rho|. \tag{8}$$

Furthermore, if σ is a mixture of states on $\{0,1\}^{\leq k}$ and ρ is a mixture of states on $\{0,1\}^*$, then

$$\|\sigma - \rho\|_{\mathrm{tr}} \leq 2^{k/4+2}\sqrt{\|\mathrm{vec}(\sigma) - \mathrm{vec}(\rho)\|}. \tag{9}$$

The proofs are unimportant for our discussion; we shall refer readers to [11]. Now, we will finish the discussion by proving the converse inclusion.

Theorem 3. $\mathsf{Computable}^{\mathrm{HALT}} \supseteq \mathsf{Computable}_{\mathrm{CTC}} = \mathsf{QComputable}_{\mathrm{CTC}}$.

Proof. Given a language $L \in \mathsf{QComputable}_{\mathrm{CTC}}$ and an input $x \in \{0,1\}^*$, we have a superoperator $\overline{S}_x$ on the space $\{0,1\}^*$ and $\overline{S}_x$ has at least one fixed point. There is also a quantum CTC Turing machine M such that either

$$\Pr(M(\rho) \text{ accepts}) \geq \frac{2}{3} \text{ for all fixed points } \rho \text{ of } \overline{S}_x$$

or

$$\Pr(M(\rho) \text{ rejects}) \geq \frac{2}{3} \text{ for all fixed points } \rho \text{ of } \overline{S}_x.$$

The problem is then to decide which of the above inequalities is true.

Let $\mathcal{M}_k$ be the set of mixtures of states on $\{0,1\}^{\leq k}$ that has rational entries only. Then, we define $\mathcal{M} = \bigcup_{k\geq 1} \mathcal{M}_k$. Clearly, $\mathcal{M}$ is countably infinite and is dense in the set of all mixtures of states on $\{0,1\}^*$ since we can always a mixtures of states on $\{0,1\}^*$ arbitrarily well with elements in $\mathcal{M}$.

We will define an oracle Turing machine P with a HALT oracle that distinguish the above two cases, taking the descriptions of $\overline{S}_x$ and M as input. The machine P will do the following: For all $k \geq 1$ and all mixtures of states $\rho \in \mathcal{M}_k$, for each of the above cases

1. Use HALT oracle to check whether exists some $t \geq 0$ such that

$$\|\rho - \overline{S}_x^t(\rho)\|_{\mathrm{tr}} > \frac{1}{2^{k+12}}.$$

2. If no such t exist, then

   - HALT and accept if $\Pr(M(\rho) \text{ accepts}) \geq 0.55$.
   - HALT and reject if $\Pr(M(\rho) \text{ rejects}) \geq 0.55$.

Now, we shall show that P always halt. By the assumption, $\overline{S}_x$ has a fixed point ρ. Then, for a $k \geq 0$ sufficiently large, we can find $\sigma \in \mathcal{M}_k$ such that

$$\|\rho - \sigma\|_{\mathrm{tr}} \leq \frac{1}{2^{k+12}}.$$

This implies, for all $t \geq 0$,

$$\|\overline{S}_x^t(\sigma) - \rho\|_{\mathrm{tr}} = \|\overline{S}_x^t(\sigma) - \overline{S}_x^t(\rho)\|_{\mathrm{tr}} \leq \|\sigma - \rho\|_{\mathrm{tr}} \leq \frac{1}{2^{k+12}}.$$

Thus, by lemma 2, we have

$$\|\mathrm{vec}(\overline{S}_x^t(\sigma)) - \mathrm{vec}(\rho)\| \leq \frac{1}{2^{k+12}}.$$

Thus, P will halt when it reaches σ.

Finally, we will finish the proof by showing that P always yields the correct result. Since P halts, then there exists some mixture of states $\sigma \in \mathcal{M}_k$ for some k such that

$$\|\overline{S}_x^t(\sigma) - \sigma\|_{\text{tr}} \leq \frac{1}{2^{k+12}}$$

for all $t \geq 0$. By lemma 2, this implies that

$$\|\text{vec}(\overline{S}_x(\sigma)) - \text{vec}(\sigma)\| \leq \frac{1}{2^{k+12}}.$$

Then, by lemma 1, there exists some fixed point ρ of $\overline{S}_x$ such that

$$\|\text{vec}(\rho) - \text{vec}(\sigma)\| \leq \frac{1}{2^{k+12}}.$$

Again, by lemma 2, this implies that

$$\|\sigma - \rho\|_{\text{tr}} \leq 2^{k/4+2}\sqrt{\frac{1}{2^{k+12}}} \leq \frac{1}{16}.$$

Thus, if $\Pr(M(\rho) \text{ accepts}) \geq \frac{2}{3}$, then we have

$$\Pr(M(\rho) \text{ accepts}) \geq \frac{2}{3} - \frac{1}{16}$$

and this is also true for the case $\Pr(M(\rho)$ rejects). Since this shows that either all fixed points satisfy the former case or all fixed points satisfy the later case, then P correctly decides whether $x \in L$.

Therefore, $\texttt{QComputable}_{\text{CTC}} \subseteq \texttt{Computable}^{\text{HALT}}$. □

We have reached our goal of discussion, finding that the extent and the limit of computability power of Deutschian CTCs. That is, $\texttt{Computable}_{\text{CTC}} = \texttt{Computable}^{\text{HALT}}$. Thus, CTCs allow one to define a model of computation that gives a possible construction of an oracle for HALT. Also, just as a final remark, the above result can also be expressed as

$$\texttt{Computable}_{\text{CTC}} = \texttt{QComputable}_{\text{CTC}} = \texttt{Computable}^{\text{HALT}} = \texttt{PSPACE}.$$

For detailed discussion for this result, we refer readers to [12].

4 Conclusion

We've covered several models of computation, up to the extreme of potentially physically realizable (in the sense of not violating a known law of nature) machines that can solve the halting problem. There are other ways to extend models of computation. One example is supertasks (see, e.g. [13]), in which a computer each successive operation takes less time (e.g. by factor of 1/2). Another model applies the notion of fuzzy logic to Turing machines, such as in [14]. Hopefully this paper provided an interesting introduction to hypercomputation, and the reader is inclined to add time-travel-based computers to the collection of fascinating models of computation. Who knows - maybe in 100 years, after the quantum computing skeptics are silenced, there will be CTC-computing skeptics, steadily back-peddling as CTC technology advances.

References

[1] Richard Feynman. "Simulating Physics with Computers". *International Journal of Theoretical Physics* vol. 21, (1982), pp. 467–488.

[2] A. Chi-Chih Yao. "Quantum circuit complexity". *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*. 1993, pp. 352–361. DOI: 10.1109/SFCS.1993.366852.

[3] Edward Fredkin and Tommaso Toffoli. "Conservative Logic". *International Journal of Theoretical Physics* vol. 21, (1982), pp. 219–253.

[4] Jérôme Durand-Lose. "Computing Inside the Billiard Ball Mode". *Collision-Based Computing*. Ed. by Andrew Adamatzky. 2002, pp. 135–160.

[5] Scott Aaronson. "NP-complete Problems and Physical Reality" (2005). URL: https://www.scottaaronson.com/papers/npcomplete.pdf.

[6] Jacob Bekenstein. "Universal upper bound on the entropy-to-energy ratio for bounded systems". *Physical Review D* vol. 23, no. 2 (1981), pp. 287–298.

[7] Jacob D. Bekenstein. "How does the Entropy/Information Bound Work?" *Foundations of Physics* vol. 35, no. 11 (2005), pp. 1805–1823.

[8] Wikipedia. *Twin Paradox*. Accessed: 28 Nov 2020. URL: https://en.wikipedia.org/wiki/Twin_paradox.

[9] David Deutsch. "Quantum mechanics near closed timelike lines". *Physical Review D* vol. 44, no. 10 (1991).

[10] Kurt Gödel. "An Example of a New Type of Cosmological Solutions of Einstein's Field Equations of Gravitation". *Rev. Mod. Phys.* vol. 21, (3), pp. 447–450.

[11] Scott Aaronson, Mohammad Bavarian, and Giulio Gueltrini. *Computability Theory of Closed Timelike Curves*. 2016. arXiv: 1609.05507 [quant-ph].

[12] Scott Aaronson and John Watrous. "Closed timelike curves make quantum and classical computing equivalent". *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* vol. 465, no. 2102 (2008), pp. 631–647. ISSN: 1471-2946.

[13] Oron Shagrir. "Super-tasks, accelerating Turing machines and uncomputability". *Theoretical Computer Science* vol. 317, no. 1 (2004), pp. 105 –114.

[14] Jiří Wiedermann. "Characterizing the Super-Turing Computing Power and Efficiency of Classical Fuzzy Turing Machines". *Theor. Comput. Sci.* vol. 317, no. 1–3 (June 2004), pp. 61–69.