Need more RAM? Just invent time travel!

Robert Andrews*[1], Mitchell Jones*[1], Patrick Lin*[1], and the UIUC Theory CS Group†[1]

[1]Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA.

April 1, 2019

Abstract

This paper is an example of what happens when a bunch of theory students procrastinate.

1 Introduction

Since the introduction of the Turing machine by Alan Turing, there has been an uncountable amount of literature studying these machines. In addition, there are many variations of Turing machines and computational models, such as non-determinism, non-uniform computation and circuit complexity, and interactive proofs. In this paper, we introduce a new type of Turing machine—what we call a Time Traveling Turing machine, also known as a TM™. We note that the notion of time travel in computing has been studied before (of course in a much more formal and rigorous setting) [1].

In the following sections we: (i) formally define such a time traveling machine, (ii) define a new collection of complexity classes, whose languages are recognized by Time Traveling Turing machines, and (iii) relate these classes to classical complexity classes.

2 Motivation

On the eve of October 31st 2018, Alan Turing, then 27 years of age, was sitting in a dimly lit room, scrawling many equations on various pieces of paper around his room. He sat back in his chair as he tried to slowly verify his calculations in excitement. He had just invented time travel. What he called a Time Traveling Turing machine. In his haste, he quickly built the machine the next day, November 1st. As he stepped into the machine, he began to feel the machine shake and rumble. In the last second before the time travel began, the machine code suffered from a segfault, and Turing traveling back in time to September 1st, 1939. Turing soon after failed to build a machine which would send him *back to the future*, and was stuck in this period of time forever. Of course, as we now know, Turing was soon recruited to Bletchly Park. The technology at the time was not powerful enough to build any sort of sophisticated Turing machine. His discovery of time travel was lost, and whenever he tried to share his discoveries after the war, people simply did not listen, dismissing his ideas as infeasible. Turing realized he would never return back to 2018.

*{rgandre2, mfjones2, plin15}@illinois.edu

†https://publish.illinois.edu/theory-cs/

A few months after this event, unbeknownst to all, some students from UIUC stumbled upon scraps of Turing's calculations on that spooky evening of 2018. In this paper we attempt to relay to you some of the ideas and proofs that Turing had.

3 Formal definition

A Time Traveling Turing machine, or to introduce as much ambiguity as possible, abbreviated as TM, is a regular deterministic Turing machine with the following additional feature: At any point during computation, the TM can enter a special time travel state $\circlearrowleft$. On entering this state, the machine is allowed to write $a(n)$ bits (where n is the length of the original input) onto a special *advice* tape. The machine then teleports back in time, restarting computation from scratch: it begins in the start state, and the original input of length n is on the work tape. The key difference is that through the magic of time travel, the TM now has access to an additional advice tape with $a(n)$ bits, which was sent into the past mysteriously from its future self.

Formally, a language L belongs to the class $\mathsf{TIMEYWIMEY}[a(n), t(n)]$ if there exists a Time Traveling Turing machine TM such that on input x:

1. After at most $t(n)$ steps from the start state, the machine either halts (and correctly decides if $x \in L$ or $x \notin L$) or must travel back in time by entering the time travel state $\circlearrowleft$.
2. Each time the TM time travels back, it can write at most $a(n)$ bits of information onto the advice tape for its past self.

Note that there is no restriction on the *number* of times a TM can travel back in time, only on *how much* information can be sent back to its past self.

4 Results

In this section we show how the class of languages recognized by Time Traveling Turing machines with $a(n)$ advice and $t(n)$ time relate to more traditional time and space-complexity classes. We also show that both adding randomness and non-determinism does not give TM accepting a language in $\mathsf{TIMEYWIMEY}[\text{poly}(n), \text{poly}(n)]$ more computational power.

Theorem 1. Let $a : \mathbb{N} \to \mathbb{N}$ and $t : \mathbb{N} \to \mathbb{N}$ be time-constructible functions. Then,

$$\mathsf{TIMEYWIMEY}[a(n), t(n)] \subseteq \mathsf{SPACE}[a(n) + t(n)] \subseteq \mathsf{TIMEYWIMEY}[a(n) + t(n), a(n) + t(n) + O(1)]\,.$$

Proof. We begin by proving the first containment. Let $L \in \mathsf{TIMEYWIMEY}[a(n), t(n)]$ be an arbitrary language and N be a TM deciding L. We design a regular Turing machine M deciding L using $a(n) + t(n)$ space. The machine M starts by copying the input onto a new work tape. Then M simulates N as normal. Any time N enters the state $\circlearrowleft$, M writes the advice tape onto its work tape according to N, M then clears the contents of its work tape (except for the advice string). Finally, M writes the contents of the original input back onto the work tape, along with the advice string, and resumes computation of N as if a time travel step had occurred. Clearly M only ever uses at most $a(n) + t(n)$ space, since the length of the advice tape is $a(n)$, and N only ever writes at most $t(n)$ bits onto its tape before entering $\circlearrowleft$.

For the second containment, let $L \in \mathsf{SPACE}[a(n)+t(n)]$ and M be a regular Turing machine deciding L in space $a(n)+t(n)$. Consider the following TM N. On input x, N simulates M for $a(n)+t(n)$ steps. It then writes the contents of its tape (which is of size $a(n)+t(n)$), current position of the head, and the state of M onto the advice tape (which requires an additional $O(1)$ space, where the hidden constant depends on M). Then N invokes the special state $\circlearrowleft$ to teleport back to its original state, now with access to its advice tape. If the advice tape is non-empty, N copies the advice tape onto the work tape, and restores its configuration. It then continues simulating M in this way until M halts and return an answer. Thus $L \in \mathsf{TIMEYWIMEY}[a(n)+t(n), a(n)+t(n)+O(1)]$. $\square$

We can obtain the following corollaries by setting parameters in Theorem 1.

Corollary 2. $\mathsf{TIMEYWIMEY}[\mathrm{poly}(n), \mathrm{poly}(n)] := \bigcup_{c\in\mathbb{N}} \mathsf{TIMEYWIMEY}[n^c, n^c] = \mathsf{PSPACE}$.[1]

Corollary 3. $\mathsf{TIMEYWIMEY}[O(\log(n)), O(\log(n))] = \mathsf{L}$.

We now show the relation between $\mathsf{TIMEYWIMEY}[a(n), t(n)]$ and time-complexity classes.

Theorem 4. Let $a : \mathbb{N} \to \mathbb{N}$ and $t : \mathbb{N} \to \mathbb{N}$ be time-constructible functions. Then,

1. $\mathsf{TIME}[t(n)] \subseteq \mathsf{TIMEYWIMEY}[0, t(n)]$, and
2. $\mathsf{TIMEYWIMEY}[a(n), t(n)] \subseteq \mathsf{TIME}\left[2^{O(a(n))} t(n)\right]$.

Proof. The first statement follows easily from the definitions, since any Time Traveling Turing machine can choose to never time travel.

As for the second statement, let $L \in \mathsf{TIMEYWIMEY}[a(n), t(n)]$ and let N be a TM deciding L. We can simulate N by a standard Turing machine M that exactly mimics the behavior of N. The machine N runs for at most $t(n)$ steps before time traveling, so to show M runs in the desired time, all we need to do is show that N travels back in time at most $2^{O(a(n))}$ times.

Fix some string x as input to N. Consider the set of possible advice tape contents, which are (without loss of generality) strings in $\{0, 1, \square\}^{a(n)}$ where $\square$ is the blank symbol. We can build a directed graph G_x on this set by adding an edge from w to z if, starting on input x and advice contents w, the advice tape contains z when N next enters the $\circlearrowleft$ state. As N is deterministic, every vertex of G_x has outdegree at most 1. Moreover, since N must halt, the subgraph reachable from $\square^{a(n)}$ (i.e., the initial advice tape contents) must be acyclic, as otherwise N would enter an infinite loop. This implies the subgraph reachable from $\square^{a(n)}$ is a path of size at most $3^{a(n)} = 2^{O(a(n))}$. Thus N travels back in time at most $2^{O(a(n))}$ times, so M runs in $2^{O(a(n))}t(n)$ time total. The result follows. $\square$

One immediate corollary of the above Theorem is the following characterization of the class P.

Corollary 5. $\mathsf{P} \subseteq \mathsf{TIMEYWIMEY}[0, \mathrm{poly}(n)] \subseteq \mathsf{TIMEYWIMEY}[O(\log n), \mathrm{poly}(n)] \subseteq \mathsf{P}$.

[1] Additionally, it is well known that $\mathsf{IP} = \mathsf{PSPACE}$. Therefore $\mathsf{TIMEYWIMEY}[\mathrm{poly}(n), \mathrm{poly}(n)] = \mathsf{IP}$. If the reader is still wondering why our model of computation should be taken somewhat-seriously, we remark that it is no less ridiculous than assuming access to an "all-powerful" prover, as in the IP model.

4.1 Does adding randomness give any more power?

Suppose a TM recognizing a language $L \in \mathsf{TIMEYWIMEY}[\mathrm{poly}(n), \mathrm{poly}(n)]$ now also had access to a random string of r with $|r| = \mathrm{poly}(n)$ bits. Specifically, every time the machine travels back in time, it receives *the same* set of random bits r as it did in all previous time periods. Put differently, it does not get a new sequence of r random bits every time it travels back. We say that such a random TM N accepts a language L, if for all $x \in L$, N accepts x with probability at least $2/3$ over the random choice of r. If $x \notin L$, N accepts x with probability at most $1/3$.

Theorem 6. Suppose N is a randomized TM with access to a random string r of $\mathrm{poly}(n)$ bits. Then every such TM can be simulated by a deterministic TM M.

Proof. The idea is that M will maintain two pieces of information on its advice tape at all times: (i) the current guess of the random string r, (ii) the number times the machine N has accepted a string x when simulated with the string r as its "random" input. Both of these strings are of $\mathrm{poly}(n)$ size. After M has simulated N on all possible choices for r, it then outputs whether or not x was accepted by N by counting the number of times N accepted x with the string r fed to N by M. □

4.2 What about Non-determinism?

A language L is in $\mathsf{NTIMEYWIMEY}[a(n), t(n)]$ if there exists a non-deterministic TM M which always writes at most $a(n)$ bits on its advice tape, and between time periods, can only execute, non-deterministically, $t(n)$ of steps before time traveling back. Following Theorem 1, it is not hard to show that a similar Theorem holds if the Time Traveling Turing machine is allowed to act non-deterministically.

Lemma 7.

$$\mathsf{NTIMEYWIMEY}[a(n), t(n)] \subseteq \mathsf{NSPACE}[a(n) + t(n)] \subseteq \mathsf{NTIMEYWIMEY}[a(n) + t(n), a(n) + t(n) + O(1)] .$$

A consequence of Savitch's theorem is that $\mathsf{PSPACE} = \mathsf{NPSPACE}$. Hence via transitivity, we conclude that non-determinism does not buy you any extra power when you have access to time travel.

Corollary 8. $\mathsf{TIMEYWIMEY}[\mathrm{poly}(n), \mathrm{poly}(n)] = \mathsf{NTIMEYWIMEY}[\mathrm{poly}(n), \mathrm{poly}(n)]$.

4.3 Ok, what about if the TM *also* has access to some kind of oracle?

"Hmm...", the authors think slowly about this for a few seconds. After what seemed like an excruciating amount of silence, one of us blurts out thoughtlessly "Wait, what's that over there?! Look behind you!" As you turn around, we quickly hop into our DeLorean and zoom away.

5 Open problems

As pointed out, it is open whether or not having access to an oracle buys you any additional computational power. Additionally, does this newly introduced class $\mathsf{TIMEYWIMEY}[a(n), t(n)]$ provide other characterizations of well-known complexity classes? For example, is there a set of parameters for which this class is NL? NP? PH?

Acknowledgements. We'd like to thank the many PhD students within the CS Theory group at UIUC. There were many fruitful discussions about Time Traveling Turing machines, which were a great way to procrastinate on research, grading, preparing for labs, research, doing homework, research, and research.

References

[1] Scott Aaronson and John Watrous. "Closed Timelike Curves Make Quantum and Classical Computing Equivalent". In: *CoRR* abs/0808.2669 (2008). URL: http://arxiv.org/abs/0808.2669.