1 Notes from January 20, 2000

1.1 Time Constructible Functions

The set of time constructible functions is defined as

{t:$\exists$ a TM that runs for exactly $t(n)$ steps on all inputs of length n}

While this may seem fairly restrictive, all standard functions like $n^{1.5}$, $n^{\log n}$, n^k, 2^{n^2}, etc, are time constructible. However, the fact that these are time constructible will not be proven here. We will now use the concept of time constructibility to create a hierarchy theorem.

Theorem. *If* $\mathbf{T}$ *is time constructible and* $t(n)\log t(n) = o(T(n))$ *then* *DTime(t(n))*$\subsetneq$*DTime(T(n))*

Program 1 On input x

  compute n=$|x|$
  compute $T(n)$
  compute the number of leading 1's in x, which implies $x = 1^i0x'$
  Spend $T(n)$ steps simulating $M_i(x)$
  if this halts and accepts **then**
    reject
  else
    accept
  end if

Proof. We can construct program 1, which runs in total time $2n + 2T(n)$ (n from computing the lengths of the various inputs, and $T(n)$ from computing $T(n)$ and running the simulation).

Let A be the set accepted by program 1. If $A \in DTime(t(n))$, accepted by M_i, then for large enough inputs x of the form $1^i0x'$ the simulation can be run in $t(n)\log t(n)$ time on a universal Turing machine, which would mean $M_i(x) = 1 \Leftrightarrow x \notin L(M_i)$, and thus $A \notin DTime(t(n))$.

Since $A \in DTime(3T)$, then it follows that $A \in DTime(T)$ since $DTime(3T(n)) = DTime(T(n))$

This follows from the "linear speed-up theorm," which we will not prove here (it follows from the fact that Turing machines can have very large worktape alphabets.) $\square$

However, this does present a problem: In the worst case, it works, but many times it will halt very quickly (e.g. when x is of the form $1^i0x'$ and M_i corresponds to a machine that halts in time linear with n) This is an example of an *i.o. hierarchy theorem* (infinitely often).

Another class happens to be the *a.e. hierarchy theorem* (almost everywhere).

1.2 The A.E. Space Hierarchy Theorem

Theorem. *If $s(n)=\omega(1)$ is space-constructible then $\exists A \in Dspace(s)$ s.t.* $\forall i \quad L(M_i) = A \Rightarrow s_i(x) = \Omega(s(|x|))$

First, some facts.

Definition. $s(n)=\omega(1) \Leftrightarrow \forall c \overset{\infty}{\forall} n \; s(n) \geq c$

Definition. $s_i(x) = \Omega(s(|x|)) \Leftrightarrow \exists c > 0 \overset{\infty}{\forall} x \quad s_i(x) \geq c \cdot s(|x|)$

Program 2 On input (M, x)

---

compute $s(|x|)$
for $i = 1$ to $s(x)$ **do**
  use space $\leq s(|x|)$ to search for a y s.t. $(|y| \leq s(x)) \wedge (y < x) \wedge (y \in L(M) \Leftrightarrow y \notin L(M_i))$
  if no such y is found **then**
    if $M_i(x)$ can be simulated in $\leq s(|x|)$ space **then**
      accept $x \Leftrightarrow x \notin L(M_i) \bigstar$
    end if
  end if
end for
accept x

Proof. We construct a program for a set A as shown above in program 2. For each choice of M, program M defines a language A_M.

$$\forall M \quad A_M \in DSpace(s)$$

Assume that M is a machine accepting A_M. Then we will show

$$\forall i \left(L(M_i) = A_M \Rightarrow \overset{\infty}{\forall} x \quad s_i(x) \geq \frac{s(|x|)}{i} \right)$$

Note that this implies the theorm, since $s_i(x) \geq \frac{s(|x|)}{i} \Rightarrow s_i(x) = \Omega(s(|x|))$

To see this, assume for the sake of a contradiction that $L(M_i) = A_M$ and $\overset{\infty}{\exists} x \, s_i(x) < \frac{s(|x|)}{i}$

Since there may be many i for which this holds, in this case we choose the least such i. In other words, for every $j < i$, either $L(M_j) \neq A_M$ or

$$\overset{\infty}{\forall} x \quad s_j(x) \geq \frac{s(|x|)}{j}$$

$\{j | (j < i)$ and line $\bigstar$ is executed on some input $x\}$ is a finite set.

For each $j < i$, let x_j be the least such input on which line $\bigstar$ is executed on input x_j (if such an input exists). Now consider a large input x. If x is much larger than any x_j, we'll find these x_j's as y's (in line 2 of Program 2), because the For loop will discover that those small M_j's do not accept $L(M)$, and thus M_i will be considered. Since we can diagonalize on this M_i, we find a contradiction.

Now consider a large input x. If x is much larger than any x_j, then the **for** loop in Program 2 will have variable y take on the value of each such x_j in turn, and it will discover that M_j does not accept $L(M)$, if x_j exists. If x_j does not exist, the **if** statement guarding line ★ will determine that M_j cannot be simulated in small space (or else line ★ would be executed, which would imply that x_j exists, contrary to assumption). Thus eventually the **for** loop will reach i, and M_i will be considered. However, by assumption there are many large strings x on which the space bound is small enough for the simulation of $M_i(x)$ to succeed, and thus line ★ will be executed, which implies that $L(M_i) \neq A_M$, contrary to assumption. □

One point that was glossed over in this proof, however, is the ability to say that we can construct a machine M that will accept A_M. In other words, a machine that will accept itself as input. This is a result of the *Complexity-theoretic Recursion Theorem.*

1.3 The Complexity-Theoretic Recursion Theorem

Let $M_i(x, y)$ be a TM computing a two place function. Then

$$\begin{aligned} \exists e\, M_e(x) &= M_i(e, x) \text{ where} \\ s_e(x) &= O(s_i(e, x)) \\ t_e(x) &= O(t_i(e, x)) \end{aligned}$$

These time and space bounds are easily verifiable.

Proof. We will use the following facts to prove this:

Fact. *There is a computable s s.t.*
$\forall x \forall y \forall z \quad M_{s(x,y)}(z) = M_x(y, z)$

Proof. To compute s, On input (x, y) output "On input z, compute $M_x(y, z)$" □

Fact. $\exists$ *computable c s.t.* $\forall i \forall j \forall x \forall y \quad M_{c(i,j)} = M_i(M_j(x), y)$

Proof. The machine to do this is
On input (i, j)
Build "On input (x, y), let $z := M_j(x)$, Run $M_i(z, y)$"

□

Fact. $\exists M_\alpha$ *s.t.* $\forall y \quad M_\alpha(y) = s(y, y)$

Using these three facts, we'll now show the existence of a machine M_i s.t. $M_e(x) = M_i(e, x)$.

Let $e = s(c(i, \alpha), c(i, \alpha))$

$$
\begin{aligned}
\text{So, } M_e(x) =& M_{s(c(i,\alpha),c(i,\alpha))}(x) \\
=& M_{c(i,\alpha)}(c(i, \alpha), x) \\
=& M_i(M_\alpha(c(i, \alpha)), x) \\
=& M_i(s(c(i, \alpha), c(i, \alpha)), x) \\
=& M_i(e, x)
\end{aligned}
$$

Therefore we can assume that a machine M can accept A_M. □

1.4 The A.E. Time Hierarchy Theorem

This is very similar to the previous proof, with a couple of minor changes. First, the statement of the Theorem:

Theorem. *If $t(n) = \omega(n)$ is time constructible, then*
$\exists A \in Dtime(t)$ *s.t.* $\forall i \quad L(M_i) = A \Rightarrow t_i(x) \log t_i(x) = \Omega(t(|x|))$

And the program corresponding to this theorem also needs to change. The main reason is that in the space problem, we could reuse space many times, however we can only use a slice of time once. This changes the program for program 2 to that of program 3. The rest of the proof is the same as for the A.E. space hierarchy theorem.

Program 3 On input (M, x)

```
compute t(|x|)
for i = 1 to log(|x|) do
  use time ≤ t(|x|)/2^i to search for a y s.t.
  (|y| ≤ log*(|x|)) ∧ (y ∈ L(M) ⇔ y ∉ L(M_i))
  if no such y is found then
    if M_i(x) can be simulated in ≤ t(|x|)/2^i time then
      accept x ⇔ x ∉ L(M_i)★
    end if
  end if
end for
accept x
```