HIERARCHY THEOREMS

Def: A function $f : \mathbf{N} \longrightarrow \mathbf{N}$, where $f(n) \geq \log(n)$, is called **space constructible** if the function that maps 1^n to the binary representation of $f(n)$ is computable in space $O(f(n))$.

Eg., $\log(n), n\log(n), n^2, n^k$ for any k, etc.

Space Hierarchy Theorem: For any space constructible function f, there exists a language A that is decidable in space $O(f(n))$ but not in space $o(f(n))$.

Recall that:

1. $g(n) = O(f(n))$ if $\exists C, n_0$ s.t. $\forall n \geq n_0$, $g(n) \leq C \cdot f(n)$.

2. $g(n) = o(f(n))$ if $\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$.

PROOF: The following $O(f(n))$ space algorithm B decides a language A that is not decidable in $o(f(n))$.

B is given as follows: on input w,

1. Let $n = |w|$.

2. Compute $f(n)$, and mark off this much tape.
   (If later stages attempt to use more, *reject*.)

3. If $w \neq \langle M \rangle 10^*$, *reject*.

4. Simulate M on w, counting number of steps.
   If steps exceed $2^{f(n)}$, *reject*.

5. If M accepts, *reject*.
   If M rejects, *accept*.

Machine B is a decider.

Let $A = L(B)$, so A is decided in space $O(f(n))$.

Next we show that A cannot be decided in space $o(f(n))$.

Suppose that M decides A in space $g(n) = o(f(n))$.

$\exists n_0$, such that $\forall n \geq n_0$, $f(n) > g(n)$. Thus, on input $w = \langle M \rangle 10^{n_0}$ (which has length $> n_0$), B can simulate the computation of M on w without exceeding the alloted space.

But then B will do the opposite of M, so M cannot decide A, which contradicts our assumption.

Corollary: For any two functions $g, f : \mathbf{N} \longrightarrow \mathbf{N}$, where $g(n) = o(f(n))$,

$$\mathsf{SPACE}(g(n)) \subsetneq \mathsf{SPACE}(f(n))$$

In particular, for $0 \leq c_1 < c_2$,

$$\mathsf{SPACE}(n^{c_1}) \subsetneq \mathsf{SPACE}(n^{c_2})$$

Corollary: NL $\subsetneq$ PSPACE.

PROOF: Savitch's Theorem shows that NL $\subseteq$ SPACE($\log^2(n)$). On the other hand, we now know that SPACE($\log^2(n)$) $\subsetneq$ SPACE(n).

Another **Corollary:** PSPACE $\subsetneq$ EXPSPACE.

Def: A function $t : \mathbf{N} \longrightarrow \mathbf{N}$, where $t(n) \geq n \log(n)$, is called **time constructible** if the function that maps 1^n to the binary representation of $t(n)$ is computable in time $O(t(n))$.

Time Hierarchy Theorem: For any time constructible function $t : \mathbf{N} \longrightarrow \mathbf{N}$, there exists a language A that is decidable in time $O(t(n))$, but not in time $o(t(n)/\log(t(n)))$.

PROOF: Consider the following $O(t(n))$ algorithm B, which on input w works as follows:

1. $n = |w|$.

2. Compute $t(n)$, and store the value $\lceil t(n)/\log(t(n)) \rceil$ in a binary counter. Decrement this counter before steps 3.,4., and 5. If it hits 0, *reject*.

3. If $w \neq \langle M \rangle 10^*$ for some TM M, *reject*.

4. Simulate M on w.

5. If M accepts, *reject*.
   If M rejects, *accept*.

Let $A = L(B)$, so A is decidable in time $O(t(n))$.

To show that A is *not* decidable in time $o(t(n)/\log(t(n)))$, we use an argument similar to the proof of the space hierarchy theorem:

Assume M decides A in time $g(n) = o(t(n)/\log(t(n)))$, and get a contradiction by considering an input $\langle M \rangle 10^n$ for some $n \geq n_0$.

Corollary: If $t, s : \mathbf{N} \longrightarrow \mathbf{N}$, where $t(n) = o(s(n)/\log(s(n)))$, and s is time constructible, then:

$$\mathrm{TIME}(t(n)) \subsetneq \mathrm{TIME}(s(n))$$

Corollary: If $1 \leq c_1 < c_2$, then

$$\mathrm{TIME}(n^{c_1}) \subsetneq \mathrm{TIME}(n^{c_2})$$

Corollary: $\mathrm{P} \subsetneq \mathrm{EXPTIME}$.

Recall that regular expressions are built up from $\emptyset, \varepsilon, a \in \Sigma$, by using the regular operations $\cup, \circ, *$.

The language $\mathrm{EQ}_{\mathrm{REX}}$ given by $\{\langle R, S\rangle |\ R, S$ are reg. exp. and $L(R) = L(S)\}$ is in PSPACE.

Introduce an **exponentiation operator**, $R^k = R \uparrow k = R \circ R \circ \cdots \circ R$.

Let $\mathrm{EQ}_{\mathrm{REX}\uparrow}$ be like the set above except we allow regular expressions with exponentiation.

Thm: $\mathrm{EQ}_{\mathrm{REX}\uparrow}$ is EXPSPACE-complete.

Cor: $\mathrm{EQ}_{\mathrm{REX}\uparrow}$ is not in PSPACE.

RELATIVIZATION

Def: An **oracle** is a language A. An **oracle TM** M^A is an ordinary TM with an extra tape called the **oracle tape**.

M writes a string on the oracle tape, queries the oracle, and gets the answer in one step.

Let $\mathbf{P}^A$ and $\mathbf{NP}^A$ be the languages in **P**, **NP** decidable with oracle A.

Eg.: $\mathbf{NP}, \textbf{co-NP} \subseteq \mathbf{P}^{\mathsf{SAT}}$.

Interestingly, $\mathbf{NP}^{\mathsf{SAT}}$ is believed to be bigger than $\mathbf{NP}$.

Consider the problem NONMIN-FORMULA which is the language:

$$\{\langle\phi\rangle|\phi \text{ is not minimal}\}$$

where "not minimal" means that there exists another ϕ', such that $\phi \equiv \phi'$, but ϕ' has less symbols.

Then NONMIN-FORMULA is in $\mathbf{NP}^{\mathsf{SAT}}$, but it seems not to be in $\mathbf{NP}$.

Thm: There exist oracles A, B such that:

1. $\mathbf{P}^A \neq \mathbf{NP}^A$
2. $\mathbf{P}^B = \mathbf{NP}^B$

PROOF: For 2. let $B =$ TQBF. Then:

$$\mathbf{NP}^{\mathrm{TQBF}} \overset{(1)}{\subseteq} \mathbf{NPSPACE} \overset{(2)}{\subseteq} \mathbf{PSPACE} \overset{(3)}{\subseteq} \mathbf{P}^{\mathrm{TQBF}}$$

For 1., given any oracle A, define L_A as follows:

$$\{w | \exists x \in A \text{ such that } |x| = |w|\}$$

Clearly, $L_A \in \mathbf{NP}^A$.

We now present an A which ensures that $L_A \notin \mathbf{P}$.

CIRCUIT COMPLEXITY

Def: A **Boolean circuit** is a collection of **gates** and **inputs** connected by **wires**. Cycles are not permitted. Gates take three forms: AND, OR, NOT.

To a Boolean circuit C with n input variables and k output gates we associate a function:

$$f_C : \{0,1\}^n \longrightarrow \{0,1\}^k$$

A **circuit family** C is an infinite list of circuits $(C_0, C_1, C_2, \ldots)$, where C_n has n input variables. We say that C decides a language A if

$$w \in A \iff C_n(w) = 1$$

where $n = |w|$.

The **size** of a circuit is the number of gates that it contains.

The **size complexity** of a circuit family C is $f(n)$ if $f(n) = |C_n|$ for all n.

The **depth** of a circuit is the length of the longest path from the inputs to an output gate.

The **circuit complexity** of a language is the size complexity of a minimal circuit family for that language. Same for **circuit depth complexity**.

Thm: Suppose that $A \in$ TIME($t(n)$), then A has a circuit complexity $O(t^2(n))$.

PROOF: Let M be the TM that decides A in time $t(n)$. For each n, build C_n to simulate M on inputs of size n: the gates of C_n are organized in $t(n)$ many rows. Each row is wired into the previous row so that it can compute the new configuration from the previous configuration. Each row has $O(t(n))$ many gates.

Let CIRCUIT-SAT be the language:

$$\{\langle C \rangle \mid C \text{ is a satisfiable boolean circuit } \}$$

Thm: CIRCUIT-SAT is **NP**-complete.

PROOF: CIRCUIT-SAT$\in$ **NP**, and SAT is polytime many-one reducible to CIRCUIT-SAT.

Note that alternatively, we could have used the proof of the previous theorem to show that $\forall A \in$ **NP**, $A \leq_p$ CIRCUIT-SAT.

This would have given us a second proof of Cook's Thm. Now, we would show that CIRCUIT-SAT $\leq_p$ 3SAT to complete this proof.