Time hierarchy.
Diagonalization arguments.

Reminder. Encoding TM-s as bit-strings

- A k-tape DTM or NTM $(\Gamma, Q, \delta, q_0, Q_F)$ can be encoded as a bit-string α. One has to mention
  - the number of tapes k; the sizes of Γ and Q;
  - the element q_0, elements of Q_F;
  - the points of δ.
- Let the encoding $M \leftrightarrow \alpha$ satisfy the following:
  - each $\alpha \in \{0, 1\}^*$ encodes some TM;
  - each TM M is encoded by an infinite number of bit-strings.
- Let M_α be the TM encoded by α.

Warmup. The halting problem

Consider the language

$$\mathrm{HALT} = \{\langle \alpha, x \rangle \mid M_\alpha \text{ stops on input } x\} \ .$$

Theorem. HALT is not accepted by any TM.

- Proof by contradiction. Assume M_{HALT} accepts HALT.
- Let $M'(x)$ first invoke $M_{\mathrm{HALT}}(\langle x, x \rangle)$.
  - If M_{HALT} accepts, then M' diverges.
  - If M_{HALT} rejects, then M' returns 1.
- Let β be an encoding of M'.
- What does $M'(\beta)$ do?

(Deterministic) time hierarchy theorem

- Let f and g be two time-constructible functions, such that $f(n) > n$ and $\lambda n.f(n) \log f(n) \in o(g)$.
- **Theorem.** DTIME$(f) \subsetneq$ DTIME(g).
- Remark. The logarithmic factor comes from the universal TM.

Proof. Let h be a function computable in time $O(g)$, such that

- $h \in \omega(f)$;
- $\lambda n.h(n) \log h(n) \in O(g)$.

(you can pick $h(n) = \lfloor g(n) / \log g(n) \rfloor$)

Proof

Define the language D as follows:

$$D = \{\alpha \in \{0,1\}^* \mid M_\alpha \text{ accepts } \alpha \text{ in } \leq h(|\alpha|) \text{ steps}\}^{\mathrm{c}} \ .$$

We show that $D \notin \mathsf{DTIME}(f)$.

- Let $L \in \mathsf{DTIME}(f)$. Let M accept L in time $c \cdot f$ for some constant c.
- Let $M = M_\alpha$ for some $\alpha \in \{0,1\}^*$, where $h(|\alpha|)/f(|\alpha|) > c$.
- We get $\alpha \in L$ iff $\alpha \notin D$. Hence $D \neq L$.

At the same time, $D \in \mathsf{DTIME}(g)$. A universal machine working in time $\lambda n.h(n)\log h(n)$ can accept D.

Non-deterministic time hierarchy theorem

- Let f and g be two time-constructible functions, such that $f(n) > n$ and $\lambda n. f(n+1) \in o(g)$.
- **Theorem.** NTIME$(f) \subsetneq$ NTIME(g).
- Note: no logarithmic factor!
- **Exercise.** Show that a k-tape NTM can be simulated in linear time on a 3-tape NTM.

Proof. Let h and h' be functions computable in time $O(g)$, such that

- $\lambda n. f(n+1) \in o(h')$
- $h' \in o(h)$ (E.g. take $h = \sqrt{fg}$ and $h' = \sqrt{fh}$)
- $h \in o(g)$.

Proof

Define a function φ as follows

$$\varphi(1) = 2$$
$$\varphi(i+1) = 2^{h(\varphi(i))}$$

For each n, let $\tilde{\varphi}(n) = \max\{i \mid \varphi(i) \leq n\}$. Define

$$D = \left\{ 1^n \,\middle|\, \begin{array}{l} i := \tilde{\varphi}(n) - 1 \\ n \neq \varphi(i+1) \wedge M_i \text{ accepts } 1^{n+1} \text{ in time } h'(n) \\ \text{OR} \\ n = \varphi(i+1) \wedge M_i \text{ rejects } 1^{\varphi(i)+1} \text{ in time } g(\varphi(i)+1) \end{array} \right\}$$

We must show $D \notin \mathsf{NTIME}(f)$ and $D \in \mathsf{NTIME}(g)$.

$D \in \mathsf{NTIME}(g)$

$$D = \left\{ 1^n \,\middle|\, \begin{array}{l} i := \tilde{\varphi}(n) - 1 \\ n \neq \varphi(i+1) \wedge M_i \text{ accepts } 1^{n+1} \text{ in time } h'(n) \\ \text{OR} \\ n = \varphi(i+1) \wedge M_i \text{ rejects } 1^{\varphi(i)+1} \text{ in time } g(\varphi(i)+1) \end{array} \right\}$$

- To compute $\tilde{\varphi}(n)$, compute $\varphi(1), \varphi(2), \ldots$ until $\geq n$.
- If $n \neq \varphi(i+1)$ then nondeterministically simulate M_i.
- If $n = \varphi(i+1)$, then search through all $O(2^{g(\varphi(i)+1)})$ computation paths of M_i.
    - There is sufficient time for that, because n is exponentially larger than $\varphi(i)$.

$D \notin \mathsf{NTIME}(f)$

- Let $L \in \mathsf{NTIME}(f)$. Let L be accepted by M_i for some i. Assume $L = D$.

- We have

$$
\begin{gathered}
1^{\varphi(i)+1} \in L \Leftrightarrow 1^{\varphi(i)+1} \in D \Leftrightarrow 1^{\varphi(i)+2} \in L \\
1^{\varphi(i)+2} \in L \Leftrightarrow 1^{\varphi(i)+2} \in D \Leftrightarrow 1^{\varphi(i)+3} \in L \\
\cdots\cdots\cdots\cdots\cdot \\
1^{\varphi(i+1)} \in L \Leftrightarrow 1^{\varphi(i+1)} \in D \Leftrightarrow 1^{\varphi(i)+1} \notin L
\end{gathered}
$$

P . . . ??? . . . NP-completeness

Theorem (Ladner). If P $\neq$ NP then there exists a language $A \in$ NP$\backslash$P that is not NP-complete.

Proof. We will construct such an A. Let

- $M_1, M_2, \ldots$ be all polynomial-time DTMs.
  - Let L_i be the language accepted by M_i.
- $f_1, f_2, \ldots$ be all polynomial-time computable functions.
  - M_i works in, f_i is computable in time $O(n^i)$.
  - From i, it is easy to find M_i, f_i.
- Let $B_0, B_1, \ldots$ be the enumeration of all bit-strings

These $M_1, M_2, \ldots$ and $f_1, f_2, \ldots$ are given. We will now construct A, such that. . .

Proof

- Claim $\mathcal{R}_i$: $A \neq L_i$
- Claim $\mathcal{S}_i$: there is an $x \in \{0,1\}^*$, such that $x \in$ SAT **XOR** $f_i(x) \in A$.
  - i.e. f_i does not polynomially many-one reduce SAT to A.
- Claim: $A \in$ NP

We set

$$A = \{x \in \{0,1\}^* \mid x \in \text{SAT and } g(|x|) \text{ is even}\}$$

for a function $g : \mathbb{N} \to \mathbb{N}$ that we define below.
If $g(n)$ is computable in time $O(p(n))$, then $A \in$ NP.

For sets X and Y define $X \triangle Y = (X \backslash Y) \cup (Y \backslash X)$.

Computing $g(n)$

$g(0) = g(1) = 2$. If $n \geq 2$ then $g(n)$ is computed as follows.

- **1st stage** Compute $g(0), g(1), g(2), \ldots$.
  - Stop after n time units.
  - Let u be the largest value, such that $g(u)$ was computed. Let $k = g(u)$.
- **2nd stage** Let $i = \lfloor \frac{k}{2} \rfloor$. For $j = 0, 1, 2, \ldots$, check whether

$B_j \in L_i$ **XOR** $B_j \in A$	$B_j \in$ SAT **XOR** $f_i(B_j) \in A$
(k is even)	(k is odd)

  - Stop after n time units.
  - If found such B_j then **return** $k+1$. Else **return** k.

Claim: $A \notin \mathrm{P}$

- Otherwise: $\exists i : A = L_i$. Consider smallest such i.
- Then $g(n)$ never grows past $2i$.
- If $g(n) = 2i$ almost always, then $A \triangle \mathrm{SAT}$ is finite. Hence $\mathrm{SAT} \in \mathrm{P}$.
- $g(n) = 2i'$ almost always, where $i' < i$ is impossible by minimality of i.
- $g(n) = 2i' + 1$ almost always would mean that $f_{i'}$ reduces SAT to A. I.e. $\mathrm{SAT} \in \mathrm{P}$.

Claim: SAT is not reducible to A

- Otherwise: $\exists i : f_i(\text{SAT}) = A$. Consider smallest such i.
- Then $g(n)$ never grows past $2i + 1$.
- If $g(n) = 2i + 1$ almost always, then A is finite. Hence $A \in$ P and also SAT $\in$ P.
- $g(n) = 2i' + 1$ almost always, where $i' < i$ is impossible by minimality of i.
- $g(n) = 2i'$ almost always would mean $A = L_{i'}$. Hence $A \in$ P and also SAT $\in$ P.

NP-**intermediate problems**

Problems conjectured to be NP-intermediate are

- graph isomorphism
- factoring
  - Given n and an interval $[k, l]$. Does n have a factor in that interval?

Oracle Turing Machines

- An Oracle TM (det. or non-det.) M is a TM with
  - A designated tape — the query tape
  - Three designated states q_{query}, q_{yes}, q_{no}.
- An oracle $\mathcal{O}$ is a subset of $\{0,1\}^*$.
- Whenever M running together with $\mathcal{O}$ (denoted $M^{\mathcal{O}}$) goes into state q_{query},
  - the contents of query tape is interpreted as a bit-string x;
  - M goes to state q_{yes} if $x \in \mathcal{O}$. Otherwise M goes to state q_{no}.
  - This takes a single step.

An oracle $\mathcal{O}$ gives us relativized complexity classes $\mathsf{P}^{\mathcal{O}}$, $\mathsf{NP}^{\mathcal{O}}$, etc.

Limits of diagonalization

- The diagonalization proofs used the facts that
  - there is an efficient mapping between bit-strings and TMs
  - efficient universal TMs exist.
- The proofs did not really consider the internal workings of the TMs M_i from the enumeration of all TMs.
- All these proofs would go through also for oracle TMs.
- Can a similar proof decide $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$.

Theorem. There exist $A, B \subseteq \{0,1\}^*$, such that $\mathsf{P}^A = \mathsf{NP}^A$ and $\mathsf{P}^B \neq \mathsf{NP}^B$.

The language EXPCOM

Consider the language

$$\text{EXPCOM} = \{\langle M, x, 1^n\rangle \mid \text{DTM } M \text{ accepts } x \text{ in } \leq 2^n \text{ steps}\}\ .$$

This is a complete language for exponential-time computation.

A computation in $\text{NP}^{\text{EXPCOM}}$ on input of length n would

- non-deterministically choose a certificate of length $\leq p(n)$;
- (make up to $p(n)$ steps), solve up to $p(n)$ problems, each requiring up to $2^{p(n)}$ steps.

A deterministic algorithm would need at most
$2^{p(n)} \cdot p(n) \cdot 2^{p(n)} = 2^{(p(n))^2 \log p(n)}$ steps. Fits in EXPCOM.

Thus $P^{\text{EXPCOM}} = \text{NP}^{\text{EXPCOM}}$.

The oracle B

For any $B \subseteq \{0,1\}^*$ let

$$U_B = \{1^n \mid \exists x : |x| = n \wedge x \in B\} \ .$$

For any B we have $U_B \in \mathsf{NP}^B$. **Exercise.** Why?

We'll now construct a language B, such that $U_B \notin \mathsf{P}^B$.

The oracle B

Let $M_1, M_2, \ldots$ be the enumeration of oracle DTM-s.
We will now define partial functions $\varphi_0, \varphi_1, \ldots$ from $\{0,1\}^*$ to $\{\text{yes}, \text{no}\}$, such that

- φ_0 is always undefined.
- Each φ_i is defined only on a finite subset of $\{0,1\}^*$.
- If $\varphi_i(x)$ is defined, then $\varphi_{i+1}(x) = \varphi_i(x)$.
- For each $x \in \{0,1\}^*$ there exists i, such that $\varphi_i(x)$ is defined.

In the end we define

$$B = \{x \in \{0,1\}^* \mid \exists i : \varphi_i(x) = \text{yes}\} \ .$$

Let t be a superpolynomial function, such that $\forall n : t(n) < 2^n$

Constructing φ_{i+1}

- Set $\varphi_{i+1} = \varphi_i$.
- Let $n = (\max_{\varphi_i(x) \text{ is defined}} |x|) + 1$.
- Run $M_{i+1}^{(\cdot)}(1^n)$ for $t(n)$ steps. If M_i queries for x, then
  - If $\varphi_i(x)$ is defined, then answer $\varphi_i(x)$.
  - If $\varphi_i(x)$ is not defined, then answer no.
    - Set $\varphi_{i+1} = \varphi_{i+1}[x \mapsto \text{no}]$.
- If $M_{i+1}^{(\cdot)}$ stops in $t(n)$ steps, then
  - If $M_{i+1}^{(\cdot)}$ accepts, then set $\varphi_{i+1} = \varphi_{i+1}[\{0,1\}^n \mapsto \text{no}]$
  - If $M_{i+1}^{(\cdot)}$ rejects then pick $x \in \{0,1\}^n$ that $M_{i+1}^{(\cdot)}$ did not query, set $\varphi_{i+1} = \varphi_{i+1}[x \mapsto \text{yes}]$.