NP Completeness

NP-completeness was introduced by Stephen Cook in 1971 in a foundational paper.


Leonid Levin independently introduced the same concept and proved that a variant of SAT is NP-complete.

---

1. S. Cook. The Complexity of Theorem-Proving Procedures. STOC, 1971.
2. L. Levin. Universal Search Problems. PINFTRANS: Problems of Information Transmission, 1973. (Translated by Trakhtenbrot)

The question "$\mathbf{P} \stackrel{?}{=} \mathbf{NP}$" became known since 1971.

Synopsis

1. NP-Completeness
2. Cook-Levin Theorem
3. Karp Theorem
4. Ladner Theorem
5. Baker-Gill-Solovay Theorem
6. On Relativization

NP-Completeness

Gödel proved (1931): Theorem Proving is undecidable.

Gödel questioned (1956): What happens if we are only interested in theorems with short proofs?

From now on “polynomial time” is abbreviated to “P-time”.

On the Definition of **NP**

Theorem. A language $L \subseteq \{0,1\}^*$ is in **NP** if and only if there exists a polynomial $p : \mathbf{N} \to \mathbf{N}$ and a P-time TM $\mathbb{M}$, called a verifier for L, such that for every $x \in \{0,1\}^*$,

$$x \in L \text{ iff } \exists u{\in}\{0,1\}^{p(|x|)}.\mathbb{M}(x,u) = 1.$$

If $x \in L$ and $u \in \{0,1\}^{p(|x|)}$ satisfy $\mathbb{M}(x,u){=}1$, then we call u a certificate for x with respect to L and $\mathbb{M}$.

A sequence of successful choices can be seen as a certificate. Conversely a certificate can be generated nondeterministically.

NP and Efficient Verifiability

P: The class of efficiently solvable problems.

NP: The class of efficiently verifiable problems.

- A problem in **P** will be called a P-problem.
- A problem in **NP** will be called an NP-problem.

NP-Problems

1. `IndSet`, `TSP`, `Subset Sum`, $0/1$ `IntProg`, `dHamPath`
2. `Graph Isomorphism`, `Factoring`
   - Babai's result in 2015, 2^{polylog}.
   - Prime Factoring is NP-complete.
3. `Linear Programming`, `Primality`

One can reduce one problem to another without even knowing any solutions to either.

Karp Reduction

Suppose $L, L' \subseteq \{0,1\}^*$.

We say that L is Karp reducible to L', denoted by $L \leq_K L'$, if there is a P-time computable function $r : \{0,1\}^* \to \{0,1\}^*$ such that for every $x \in \{0,1\}^*$,

$$x \in L \text{ iff } r(x) \in L'.$$

If $L \leq_K L' \leq_K L''$ then $L \leq_K L''$.

Hardness and Completeness

We say that L' is NP-hard if $L \leq_K L'$ for every $L \in \mathbf{NP}$.

We say that L' is NP-complete if L' is NP-hard and $L' \in \mathbf{NP}$.

Fact.

1. If language L is NP-hard and $L \in \mathbf{P}$, then $\mathbf{P} = \mathbf{NP}$.
2. If language L is NP-complete, then $L \in \mathbf{P}$ iff $\mathbf{P} = \mathbf{NP}$.

Does there exist an NP-complete problem any way?

Bounded Halting Problem, the problem of all NP-problems

Theorem. The following language is NP-complete:

$$\mathtt{TMSAT} \stackrel{\text{def}}{=} \{\langle \alpha, x, 1^n, 1^t\rangle \mid \exists u{\in}\{0,1\}^n.\ \mathbb{M}_\alpha(x,u) \text{ outputs } 1 \text{ in } t \text{ steps}\}.$$

Let $L \in \mathbf{NP}$ be decided by a q-time TM $\mathbb{M}$ using p-size certificate. Then "$x \in L$" iff

$$\exists u \in \{0,1\}^{p(|x|)}.\ \mathbb{M}(x,u) \text{ outputs } 1 \text{ in } q(|x|+p(|x|)) \text{ steps}$$

iff "$\langle \llcorner\mathbb{M}\lrcorner, x, 1^{p(|x|)}, 1^{q(|x|+p(|x|))}\rangle \in \mathtt{TMSAT}$".

The Karp reduction maps x of size n onto $\langle \llcorner\mathbb{M}\lrcorner, x, 1^{p(n)}, 1^{q(n+p(n))}\rangle$.

A language $L \subseteq \{0,1\}^*$ is in **coNP** $\stackrel{\text{def}}{=} \left\{L \mid \overline{L} \in \mathbf{NP}\right\}$ if there exists a polynomial $p : \mathbf{N} \to \mathbf{N}$ and a P-time TM $\mathbb{M}$ such that

$$x \in L \text{ iff } \forall u{\in}\{0,1\}^{p(|x|)}.\mathbb{M}(x,u){=}1$$

for every x.

$$\mathbf{P} \subseteq \mathbf{coNP} \cap \mathbf{NP}.$$

P $=$ **NP** Implies **coNP** $=$ **NP**.

Theorem. If $\mathbf{P} = \mathbf{NP}$ then $\mathbf{EXP} = \mathbf{NEXP}$.

Proof.

Suppose $L \in \mathbf{NEXP}$ is accepted by an NDTM in 2^{n^c} time. Then

$$L_{\mathrm{pad}} = \left\{ x01^{2^{|x|^c}} \mid x \in L \right\} \in \mathbf{NP}.$$

By assumption $L_{\mathrm{pad}} \in \mathbf{P}$. But then $L \in \mathbf{EXP}$. $\square$

The padding technique used in this proof was introduced by Berman and Hartmanis.

1. Berman and Hartmanis. On Isomorphisms and Density of NP and Other Complete Sets. STOC, 1976.

Cook-Levin Theorem

Conjunctive Normal Form

A CNF formula has the form

$$\bigwedge_i \left(\bigvee_j v_{i_j} \right),$$

where v_{i_j} is called a literal, and $\bigvee_j v_{i_j}$ a clause. A literal is either a propositional variable, or the negation of a propositional variable.

A kCNF formula is a CNF formula in which all clauses contain at most k literals.

The First Natural **NP**-Complete Problem

SAT is the language of all satisfiable CNF formulae.

- The set of the unsatisfiable DNF's is the complement of SAT.
- The set of the satisfiable DNF's is in **P**.

3SAT

Fact. $2\mathtt{SAT} \in \mathbf{P}$.

Fact. $\mathtt{SAT} \leq_K 3\mathtt{SAT}$.

Proof.
For example $u_1 \vee u_2 \vee u_3 \vee u_4 \vee u_5$ is satisfiable iff

$$(u_1 \vee u_2 \vee v) \wedge (\neg v \vee u_3 \vee w) \wedge (\neg w \vee u_4 \vee u_5)$$

is satisfiable. □

Corollary. $3\mathtt{SAT}$ is NP-complete if $\mathtt{SAT}$ is NP-complete.

Cook-Levin Theorem

Theorem (Cook, 1971; Levin, 1973). `SAT` is NP-complete.

Formula for Equality

Intuitively the formula

$$(x_1 \vee \overline{y_1}) \wedge (\overline{x_1} \vee y_1) \wedge \ldots \wedge (x_n \vee \overline{y_n}) \wedge (\overline{x_n} \vee y_n)$$

is equivalent to

$$(x_1 = y_1) \wedge \ldots \wedge (x_n = y_n).$$

Formula for Boolean Function

Claim. For every Boolean function $f\colon \{0,1\}^\ell \to \{0,1\}$, there is an ℓ-variable CNF formula φ_f of size at most $\ell 2^\ell$ such that $\varphi_f(u) = f(u)$ for every $u \in \{0,1\}^\ell$, where the size of a CNF formula is defined to be the number of $\vee/\wedge$ symbols.

Let φ_f be

$$\bigwedge_{v\in\{0,1\}^\ell \wedge f(v)=0} C_v(z_1,\ldots,z_\ell)$$

where for each v, $C_v(v) = 0$ and $C_v(u) = 1$ for every $u \neq v$.

Proof of Cook-Levin Theorem

Suppose $L \in \mathbf{NP}$.

Let $\mathbb{M}$ be an oblivious P-time TM and p a polynomial such that for every $x \in \{0,1\}^*$,

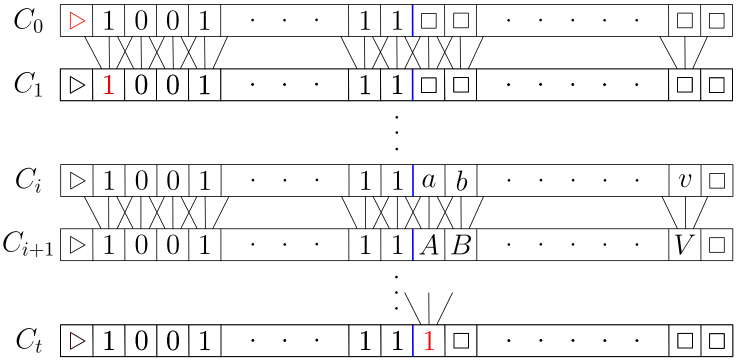
$$x \in L \text{ iff } \exists u \in \{0,1\}^{p(|x|)}.\mathbb{M}(x,u) = 1.$$

We shall construct a P-time computable function $\varphi\_ : L \to \mathtt{SAT}$ such that "$\varphi_x \in \mathtt{SAT}$ iff $x \in L$", which is equivalent to saying that

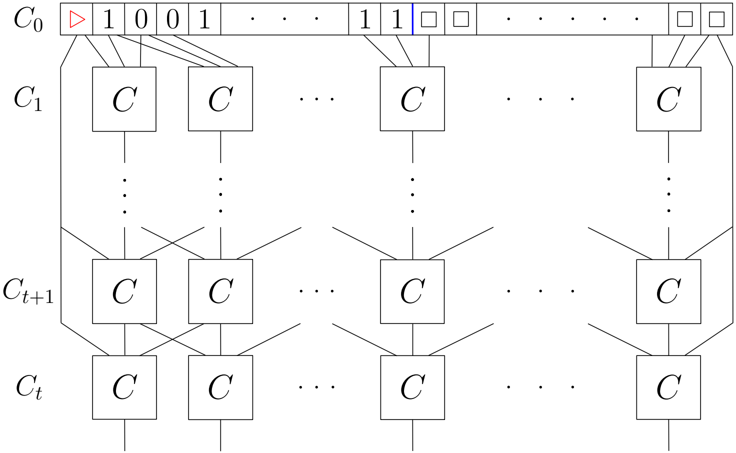
$$\varphi_x \in \mathtt{SAT} \text{ iff } \exists u \in \{0,1\}^{p(|x|)}.\mathbb{M}(xu) = 1.$$

We may assume that $\mathbb{M}$ is oblivious.

The First Proof



The First Proof



The Second Proof

The idea is that φ_x should be a CNF formula such that

- it codes up the execution sequence of $\mathbb{M}(xu)$ from the initial configuration to the final configuration, and
- $\mathbb{M}(xu) = 1$ iff the CNF formula is satisfiable.

The Second Proof

We choose to encode snapshots rather than configurations.

- "Computation is local."

The i-th snapshot z_i depends only on z_{i-1}, $\mathrm{prev}(i)$, $\mathrm{inputpos}(i)$.

- The current state is calculated from z_{i-1}.
- The symbol at the input tape is read at $\mathrm{inputpos}(i)$.
- The other symbols are collected from $\mathrm{prev}(i)$'s.

The Second Proof

Now $x \in L$ if and only if

$$\exists y \in \{0,1\}^{|x|+p(|x|)}.\exists z_1 \ldots z_{T(|x|)}.\varphi_x(y, z_1, \ldots, z_{T(|x|)})$$

where $\varphi_x(y, z_1, \ldots, z_{T(|x|)})$ is the conjunction of the following propositions.

1. x is a prefix of y;
2. z_1 encodes the initial snapshot;
3. $z_{T(|x|)}$ encodes the final snapshot with output 1;
4. $z_{\mathrm{prev}(i)}, z_{i-1}, z_i$ and $y_{\mathrm{inputpos}(i)}$ must satisfy the relationship imposed by the transition function.

The size of z_i is bounded by $(3c + \log n)2^{3c+\log n}$, where c is the size of snapshot. Conclude that $\varphi_x(y, \widetilde{z})$ is of polynomial length.

The Second Proof

$\varphi_x(y, \widetilde{z})$ is constructed in P-time.

- Calculate $|x \circ u|$ in P-time, and
- Simulate $\mathbb{M}$ on $0^{|x \circ u|}$ in P-time to calculate
  - the head positions at the i-th step, and
  - for each work tape the largest $j < i$ such that at the j-th step the head of the work tape was scanning the same cell as it is scanning at the i-th step.

Property of Cook-Levin Reduction

The reduction is a Levin reduction.

- A reduction from an NP language to another is a Levin reduction if it provides an efficient method to transform certificates to certificates forward and backward.

The reduction is parsimonious.

- A reduction r from an NP language to another NP language is parsimonious if the number of the certificates of x is equal to the number of the certificates of $r(x)$.

The reduction can be done in logspace.

- Confer Papadimitriou's book.

Decision vs. Search

Search problems are evidently harder than the corresponding decision problems. They are however equivalent in some sense.

Theorem. If $\mathbf{P} = \mathbf{NP}$, then for every $\mathbf{NP}$ language L, there is a P-time TM that on input $x \in L$ outputs a certificate for x.

Proof.

Construct a Levin reduction from $L \in \mathbf{NP}$ to $\mathtt{SAT}$.

Given a CNF formula with n variables, we can call upon a P-time algorithm for $\mathtt{SAT}$ for $n+1$ times to find a certificate for the formula if it is satisfiable. We then apply the Levin reduction to get a certificate for x. $\square$

Theorem. The following language is co-NP-complete:

$$\texttt{TAUTOLOGY} = \{\varphi \mid \varphi \text{ is a tautology}\}.$$

"$L \in \mathbf{coNP}$" $\Rightarrow$ "$\overline{L} \in \mathbf{NP}$" $\Rightarrow$ "$\overline{L} \leq_K \texttt{SAT}$" $\Rightarrow$ "$L \leq_K \overline{\texttt{SAT}}$".

Karp Theorem

Soon after Cook's paper, Richard Karp showed in 1972 that 21 combinatorial problems are NP-complete.

1. Reducibility Among Combinatorial Problems. In Complexity of Computer Computations, R. Miller and J. Thatcher (editors), 85-103, 1972.

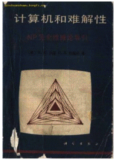
Karp's 21 NP-Complete Problems

SAT

- 0-1 Integer Programming
- Clique
  - Set Packing
  - Vetex Cover
    - Set Covering, Feedback Node Set, Feedback Arc Set
    - Directed Hamilton Circuit
      - Undirected HC
- 3-SAT
  - Chromatic Number
    - Clique Cover
    - Exact Cover
      - Hitting Set, Steiner Tree, 3-Dimensional Matching
      - Knapsack
        - Job Sequencing
        - Partition
          - Max Cut

Michael R. Garey and David S. Johnson.

- Computers and Intractability – A Guide to the Theory of NP-Completeness, 1979.



Gödel's Question Answered

Let $\mathcal{A}$ be a familiar axiomatic system. Let THEOREM be the problem

$$\left\{(\varphi, 1^{|\varphi|^c}) \mid \varphi \text{ has a formal proof of length } \leq |\varphi|^c \text{ in } \mathcal{A}\right\}.$$

This is the finite version of Hilbert's problem.

Fact. THEOREM is NP-complete.

Proof.

1. In familiar axiomatic systems, such as PA and ZF, verification runs in time polynomial in the length of proof.
2. Arithmetization, for example. □

Gödel essentially raised the question "$\mathbf{P} \stackrel{?}{=} \mathbf{NP}$" as early as in 1956 in a private communication to John von Neumann.

Berman-Hartmanis Conjecture

Conjecture. NP-complete problems are polynomially isomorphic.

Evidence:

- $A \leq_K^1 B$ for all known NP-complete problems A, B.
- If $A \leq_K^1 B \leq_K^1 A$ then $A \cong_p B$.

  The proof is similar to that of Myhill Isomorphism Theorem.

Cook-Levin reductions are $\leq_K^1$.

Berman-Hartmanis Conjecture

Fact. Berman-Hartmanis Conjecture implies $\mathbf{P} \neq \mathbf{NP}$.

A dense language cannot be p-isomorphic to a sparse language.

- $\texttt{SAT}$ is dense.
- $\{1^n \mid n \in \mathbf{N}\}$ is sparse.

A set $S \subseteq \{0,1\}^*$ is dense if $|S^{\leq n}| = 2^{n^{O(1)}}$; it is sparse if $|S^{\leq n}| = n^{O(1)}$.

Ladner Theorem

We have seen many NP-complete problems.

Is there an NP-problem that is neither in **P** nor NP-complete?

Finding such a problem is at least as hard as proving "$\mathbf{P} \neq \mathbf{NP}$".

Ladner Theorem. If $\mathbf{P} \neq \mathbf{NP}$, then there is a language neither in $\mathbf{P}$ nor NP-complete.

---

1. Richard Ladner. On the Structure of Polynomial Time Reducibility. J. ACM, 1975.

Motivation

We know that $\mathtt{SAT}$ is NP-complete whereas $2\mathtt{SAT}$ is in **P**.

The idea is to remove from $\mathtt{SAT}$ just enough instances so that the remaining set is not NP-complete, but not enough to make it in **P**.

Technically think of $\psi 01^n$ as $\psi \wedge v_1 \wedge \ldots \wedge v_n$. Observe that

- $\left\{\psi 01^{|\psi|^c} \mid \psi \in \mathtt{SAT}\right\}$ is NP-complete, whereas
- $\left\{\psi 01^{2^{|\psi|}} \mid \psi \in \mathtt{SAT}\right\}$ is in **P**.

We will find some $H(x)$ such that $|\psi|^{H(|\psi|)}$ does the job.

The problem

$$\mathtt{SAT}_H = \left\{ \psi 0 1^{|\psi|^{H(|\psi|)}} \mid \psi \in \mathtt{SAT} \right\},$$

and the function

$$H(n) = \begin{cases} i, & \begin{array}{l} i < \log\log n \text{ is the smallest index of a TM such that} \\ \mathbb{M}_i(x) \text{ outputs } \mathtt{SAT}_H(x) \text{ in } i|x|^i \text{ steps for all } x \text{ with } |x| \leq \log n, \end{array} \\ \log\log n, & \text{otherwise} \end{cases}$$

are inter-dependent by definition.

---

1. $\mathtt{SAT}_H$ and $H(n)$ are well defined. $H(n)$ is nondecreasing since $i < H(n)$ implies $i < H(n+1)$.
2. $\mathtt{SAT}_H \in$ **NP** because $H(n)$ is computable in $o(n^3)$-time. Confer

$$T(n) = (\log\log n) 2^{\log n}(c \cdot C \log C + 2^{\log n} + T(\log n) + \ldots),$$

where $C = (\log\log n)(\log n)^{\log\log n}$ and $c = o(\log n)$.

Fact. If $H(\mathbf{N})$ is finite, then some $\mathbb{M}_i$ decides $\mathtt{SAT}_H$ in in^i-steps.

Proof.

If $H(\mathbf{N})$ is finite, then since H is nondecreasing, some i exists such that $H(n) = i$ for all large n. But then $\mathbb{M}_i$ decides $\mathtt{SAT}_H$ in $i|x|^i$ steps. $\square$

Fact. $\mathtt{SAT}_H \in \mathbf{P}$ iff $H(\mathbf{N})$ is finite.

Proof.

Suppose $\mathbb{M}_i$ decides $\mathtt{SAT}_H$ in cn^c-steps for some c. Let i be such that $i \geq c$. Then $H(n) \leq i$ for all large n. $\square$

Lemma. If $\mathbf{P} \neq \mathbf{NP}$ then $\mathtt{SAT}_H \notin \mathbf{P}$.

Proof.

If $\mathtt{SAT}_H \in \mathbf{P}$, then $\mathtt{SAT} \leq_K \mathtt{SAT}_H$ by the above fact. $\square$

Lemma. If $\mathbf{P} \neq \mathbf{NP}$ then $\texttt{SAT}_H$ is not NP-complete.

Assume that $\texttt{SAT}_H$ were NP-complete.

- There would be some in^i time Karp reduction $r : \texttt{SAT} \rightarrow \texttt{SAT}_H$.
- Fix some N such that $|r(\varphi)| = |\psi 01^{|\psi|^{H(|\psi|)}}| > N$ implies $|\psi| < \sqrt{|\varphi|}$.

A P-time algorithm *Sat*(φ) for $\texttt{SAT}$ is defined as follows:

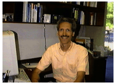
1. Compute $r(\varphi)$, which must be $\psi 01^{|\psi|^{H(|\psi|)}}$ for some ψ.
2. If $|r(\varphi)| > N$, call *Sat*(ψ) recursively, otherwise apply brutal force.

The depth of recursive call is bounded by $\log\log|\varphi|$.

Under plausible complexity theoretical assumptions, there are natural problems that are not NP-complete. However none of these problems has been shown to lie outside **P**.

Baker-Gill-Solovay Theorem

Isn't it tempting to look for a very clever use of diagonalization to construct an intermediate language without assuming $\mathbf{P} \neq \mathbf{NP}$?

Theodore Baker, John Gill and Robert Solovay published in 1975 a profound result that brought up the issue of relativization.[1]

---

1. Relativizations of the P =? NP Question. SIAM Journal on Computing, 4(4):431-442, 1975.

Oracle Turing Machine

An Oracle Turing Machine (OTM) is a TM $\mathbb{M}^?$ that has additionally one read-write oracle tape and states $q_{\mathtt{query}}, q_{\mathtt{yes}}, q_{\mathtt{no}}$.

- To execute $\mathbb{M}^?$ we need to specify an oracle $B \subseteq \{0,1\}^*$.
- During an execution whenever $\mathbb{M}^B$ enters the state $q_{\mathtt{query}}$, the machine moves into the state $q_{\mathtt{yes}}$ if $a \in B$ and $q_{\mathtt{no}}$ if $a \notin B$, where a is what is on the oracle tape.
- A query to B counts as a single computation step.

We write $\mathbb{M}^B(x)$ for the output of $\mathbb{M}^B$ on input x.

Nondeterministic Oracle TM's are defined in the same manner.

The Gödel encoding of the OTM's is independent of any oracle.

- $\mathbb{M}_0^?$, $\mathbb{M}_1^?$, $\mathbb{M}_2^?$, ...

$\mathbf{P}^O$ and $\mathbf{NP}^O$

Suppose $O \subseteq \{0,1\}^*$.

- $\mathbf{P}^O$ is the set of all languages decidable by P-time TM's with access to O.
- $\mathbf{NP}^O$ is the set of all languages acceptable by P-time NDTM's with access to O.

$\mathbf{NP}^{O[k]} \subseteq \mathbf{NP}^O$.

- The oracle can be queried for at most k times in any run.

The complexity class $\mathbf{NP}^{\mathbf{NP}}$ for example is defined as follows:

$$\mathbf{NP}^{\mathbf{NP}} = \bigcup_{L \in \mathbf{NP}} \mathbf{NP}^{L}.$$

Example

1. $\overline{\mathtt{SAT}} \in \mathbf{P}^{\mathtt{SAT}}$.
2. $\mathbf{P}^{A} = \mathbf{P}$ if $A \in \mathbf{P}$.
3. $\mathbf{NP}^{\mathbf{NP}} = \mathbf{NP}^{\mathtt{SAT}}$.

Cook Reduction

A language L is Cook reducible to a language L' if there is a P-time OTM $\mathbb{M}^?$ such that L is decided by $\mathbb{M}^{L'}$.

- L is Cook reducible to L' iff L is Cook reducible to $\overline{L'}$.

Meditation on Reduction

Historically,

- Cook used the P-time Turing reduction in the 1971 paper,
- Karp restricted to the P-time m-reduction in the 1972 paper,
- Levin defined reduction for search problems in the 1973 paper.

Why Karp reduction in the definition of NP-completeness? Why not Cook reduction?

- Completeness is better defined in terms of Cook reduction.
  - Yet all known NP-completeness results can be established using Karp reduction.
- **NP** = **coNP** if reductions are understood as Cook reductions.

Lowness

A complexity class $\mathbf{B}$ is low for a complexity class $\mathbf{A}$ if $\mathbf{A}^{\mathbf{B}} = \mathbf{A}$.

- Problems in $\mathbf{B}$ are not only solvable, they are also easy to solve in $\mathbf{A}$'s viewpoint.

Lowness

If **A** is low for itself then **A** is closed under complement, provided it is powerful enough to negate boolean results.

- **P** is low for itself.
- **NP** is believed not to be low for itself.
- **PSPACE** is low for itself.
- **L** is low for itself.
- **EXP** is not low for itself, although $\overline{\mathbf{EXP}} = \mathbf{EXP}$.

Baker-Gill-Solovay Theorem. There are A, B such that $\mathbf{P}^A = \mathbf{NP}^A$ and $\mathbf{P}^B \neq \mathbf{NP}^B$.

Proof of Baker-Gill-Solovay Theorem, Case A

Let A be the following language

$$\{\langle \alpha, x, 1^n \rangle \mid \mathbb{M}_\alpha \text{ outputs } 1 \text{ on } x \text{ within } 2^n \text{ steps}\}.$$

Then $\mathbf{EXP} \subseteq \mathbf{P}^A \subseteq \mathbf{NP}^A \subseteq \mathbf{EXP}$. Hence $\mathbf{P}^A = \mathbf{NP}^A$.

Proof of Baker-Gill-Solovay Theorem, Case B

Basic idea of oracle B:

- A machine that only makes a polynomial number of queries can never get it right for every input.
- A machine that makes an exponential number of queries can always make the right judgments for all inputs.

Proof of Baker-Gill-Solovay Theorem, Case B

Let $B_0 = \emptyset$ and $n_0 = 0$. Construct B_{i+1} from B_i as follows:

- Let n_{i+1} be larger than $n_0, n_1, \ldots, n_i$.
  - Moreover n_{i+1} must be larger than the size of all strings that have been raised as questions by $\mathbb{M}_0^{B_0}(x_1), \ldots, \mathbb{M}_{i-1}^{B_{i-1}}(x_i)$, where $|x_1| = n_1 + 1, \ldots, |x_i| = n_i + 1$.
- Run $\mathbb{M}_i^{B_i}$ on $1^{n_{i+1}}$ for $2^{n_{i+1}-1}$ steps.
  - If $\mathbb{M}_i^{B_i}(1^{n_{i+1}})$ does not stop in $2^{n_{i+1}-1}$ steps, let $B_{i+1} = B_i$.
  - If $\mathbb{M}_i^{B_i}(1^{n_{i+1}})$ is accepted in $2^{n_{i+1}-1}$ steps, let $B_{i+1} = B_i$.
  - If $\mathbb{M}_i^{B_i}(1^{n_{i+1}})$ is rejected in $2^{n_{i+1}-1}$ steps, let $B_{i+1} = B_i \cup \{s\}$, where $|s| = n_{i+1}$ and s is not queried when executing $\mathbb{M}_i^{B_i}(1^{n_{i+1}})$.

Let $B = \bigcup_{i \in \mathbf{N}} B_i$.

Proof of Baker-Gill-Solovay Theorem, Case B

Let U_B be defined as follows:

$$U_B = \{1^n \mid B \text{ contains a string of length } n\}.$$

Lemma. $U_B \in \mathbf{NP}^B$ and $U_B \notin \mathbf{P}^B$.

Proof.

Assume that $\mathbb{M}_i^B$ decided U_B in P-time $T(n)$. Let i be such that $T(n_i) < 2^{n_i - 1}$. Then

$$\mathbb{M}_i^B(1^{n_{i+1}}) = 0 \Leftrightarrow \mathbb{M}_i^{B_{i+1}}(1^{n_{i+1}}) = 0 \Rightarrow \mathbb{M}_i^{B_i}(1^{n_{i+1}}) = 1 \Rightarrow \mathbb{M}_i^{B_{i+1}}(1^{n_{i+1}}) = 1 \Leftrightarrow \mathbb{M}_i^B(1^{n_{i+1}}) = 1.$$

This is a contradiction. $\square$

On Relativization

Relativization

A proof of $\mathbf{A} \neq \mathbf{B}$ ($\mathbf{A} = \mathbf{B}$) relativizes if it is also essentially a proof of $\mathbf{A}^O \neq \mathbf{B}^O$ ($\mathbf{A}^O = \mathbf{B}^O$) without specifying the oracle O.

"We feel that this is further evidence of the difficulty of the $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ question. ...

It seems unlikely that ordinary diagonalization methods are adequate for producing an example of a language in **NP** but not in **P**; such diagonalizations, we would expect, would apply equally well to the relativized classes. ...

On the other hand, we do not feel that one can give a general method for simulating nondeterministic machines by deterministic machines in polynomial time, since such a method should apply as well to relativized machines."

Baker, Gill, and Solovay

Relativization Barrier

Whatever a proof of $\mathbf{P} \neq \mathbf{NP}$ (or $\mathbf{P} = \mathbf{NP}$) is, it cannot be a proof that relativizes. It seems to rule out any possibility of using diagonalization/simulation to settle the issue.

- This appears very surprising since Recursion Theory does relativize and Complexity Theory can be seen as a resource bounded version of Recursion Theory.

Relativization Barrier in 1975-1988

Hopcroft (1984):

> This perplexing state of affairs is obviously unsatisfactory as it stands. No problem that has been relativized in two conflicting ways has yet been solved, and this fact is generally taken as evidence that the solutions of such problems are beyond the current state of mathematics.

The researchers were not aware of a true complexity theoretical statement whose negation is also true in some relativized world.

- The general practice was either to prove something or to refute it in some relativized world.

Relativization Barrier, Mystery or Myth

From early on researchers have noticed that separation by relativization often exploits difference of oracle access mechanisms.

- Baker-Gill-Solovay Theorem.
- Another example is Hopcroft, Paul and Valiant's result that $\mathbf{TIME}(S(n)) \subsetneq \mathbf{SPACE}(S(n))$ for space constructible $S(n)$.
  - Hartmanis, Chang, Chari, Ranjan, and Rohatgi have come up with an oracle A such that $\mathbf{TIME}^A(S(n)) = \mathbf{SPACE}^A(S(n))$.
- Speedup Theorem.

"This example demonstrates the danger in thinking of oracle worlds as a way of relativizing complexity classes – oracles do not relativize complexity classes, they only relativize the machines."

Hartmanis, Chang, Chari, Ranjan and Rohatgi, 1992.

We will see that some big proofs in complexity theory are non-relativzing.

$$\mathbf{P} \stackrel{?}{=} \mathbf{NP}$$

only time can tell.