Relativisation Barrier

Last revised: 2026-07-02

We use diagonalisation to prove a time hierarchy theorem separating P and EXP. However, it turns out that this proof technique does not help us prove or disprove $\mathsf{P} = \mathsf{NP}$. Using oracle machines, this result is formulated as the *relativisation barrier*.

Notation. For a Turing machine (TM) M, we define

$$M(x) = \begin{cases} 1 & \text{if } M \text{ accepts } x \\ 0 & \text{otherwise.} \end{cases}$$

For a string $x \in \{0,1\}^*$, we write $\langle x \rangle \in \mathbb{N}$ to denote its encoding.

1 Diagonalisation and Time Hierarchy Theorems

Recall the complexity class EXP defined as the collection of languages decidable by a (deterministic) Turing machine in exponential time:

$$\mathsf{EXP} \stackrel{\text{def}}{=} \bigcup_{c>0} \mathsf{DTIME}\left(2^{n^c}\right)$$

where for $f : \mathbb{N} \to \mathbb{N}$, the class $\mathsf{DTIME}(f(n))$ consists of languages decided by a TM in time $O(f(n))$.

In general, the Time Hierarchy Theorem says that

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

for $f(n) \ll g(n)$[5]. Here, we prove for the specific case for P vs EXP.

Theorem 1A. $\mathsf{P} \subsetneq \mathsf{EXP}$.

Proof. Since strings and TMs are both effectively enumerable, let M_x denote the $\langle x \rangle$th TM and WLOG assume it halts within $2^{|x|}$ steps on all inputs. Consider the Turing machine D: "On input x, run M_x on x for $2^{|x|}$ steps and output the opposite answer of M_x." This machine accepts the language

$$L := \{x \in \{0,1\}^* : M_x \text{ rejects } x\},$$

[5] [AB16] Theorem 3.1

and it follows that $L \in \mathsf{EXP}$ since simulating a TM using the universal Turing machine only costs a logarithmic slowdown.

Next, we prove $L \notin \mathsf{P}$. Suppose a Turing machine M decides L in time n^c for some $c > 0$. Let x be a sufficiently large string such that $|x|^c < 2^{|x|}$ and $M_x(x) = M(x)$. Such string exists because there are infinitely many TMs that accept the same language as M. It follows that M_x accepts x iff $x \in L$, a contradiction. □

This proof technique, called *diagonalisation*, relies only on two basic properties of Turing machines: (1) TMs are effectively enumerable, and (2) we appeal to the ability to simulate TMs without much overhead. This is essentially treating TMs as *blackboxes* as we never look at their internal workings. Later, we will show that this proof technique won't help us separate P and NP. This phenomenon is captured by the concept of *relativisation*, which we shall get into after introducing *oracles*.

2 Oracles and Relativised Theorems

The idea of relativisation is to consider the *relative* difficulty between decision problems. In (polynomial-time) many-one reductions, we noted that $A \leqslant_m^p B$ means A is roughly "as hard as" B, in the sense that if B can be solved efficiently, then so does A. We generalise this idea using (a relatively informal definition of) *oracles*.

Definition 2A. An **oracle machine** is a TM with access to a blackbox "oracle" that can decide a given decision problem in constant time. We denote a machine with oracle access to the language $O \subseteq \{0,1\}$ by superscript M^O.

An oracle machine M^O can ask the question "Does x belong to the language O?" and get its answer for an arbitrary number of times. If O is a difficult language, then M^O will generally be more powerful than M.

Definition 2B. For every $O \subseteq \{0,1\}^*$, let P^O denote the class of languages decidable by a polynomial-time Turing machine with oracle access to O. Similarly, NP^O is the class of languages decidable by a nondeterministic polynomial-time Turing machine with oracle access to O.

A key observation is that Turing machines with oracle access to some O are still enumerable and can be simulated (by a universal Turing machine with the same

oracle access). This means that any theorem about normal TMs that only relies on these two properties will also hold for TMs with any oracle O. In other words, these theorem can be *relativised*.

3 Relativisation Barrier

The relativisation barrier says that there cannot be a relativising result that proves or disproves $\mathsf{P} = \mathsf{NP}$.

Theorem 3A. There exists oracles A, B such that $\mathsf{P}^A = \mathsf{NP}^A$ and $\mathsf{P}^B \neq \mathsf{NP}^B$.

Proof. Let A be the following language.

$$A := \{\langle M, x, 1^n\rangle : M \text{ outputs 1 on } x \text{ within } 2^n \text{ steps}\}$$

Clearly, a TM with oracle access to A can perform an exponential-time computation at the cost of one query to the oracle, and so $\mathsf{EXP} \subseteq \mathsf{P}^A$. On the other hand, within exponential time, we can simulate both the execution of a nondeterministic polynomial-time TM (by enumerating all the nondeterministic choices) and queries to A through simulation, thus $\mathsf{NP}^A \subseteq \mathsf{EXP}$. With $\mathsf{P}^A \subseteq \mathsf{NP}^A$, it follows that $\mathsf{P}^A = \mathsf{NP}^A$.

Before constructing B, we first define

$$U_B := \left\{1^n \in \{0,1\}^* : \text{some string of length } n \text{ is in } B\right\}.$$

For any oracle B, the language U_B is clearly in NP^B as we can guess nondeterministically a string x such that $x \in B$. We now construct an oracle B such that $U_B \notin \mathsf{P}^B$, completing the proof that $\mathsf{P}^B \neq \mathsf{NP}^B$.

Let M_k^B be the kth oracle machine that runs in $2^n/10$ time. We construct B step by step, starting with no strings. The idea is to, at each step k, consider finitely many strings and declare if each of them should belong to B. We then pick some 1^n so that M_k^B disagrees with U_B. Ultimately the construction of B and our choices of 1^n ensure no M_k^B decides U_B.

Simulate the computation of M_k^B on 1^n, and whenever it queries $x \in \{0,1\}^*$ whose fate is already known, we answer the query consistently; otherwise we declare that x is not in B. Consequently, when the computation finishes, we have declared the fate of at most $2^n/10$ strings in $\{0,1\}^n$, all of them are not in B. Therefore, to make M_k^B disagree with U_B on 1^n, we declare that, if M_k^B accepts

1^n, no strings in $\{0,1\}^n$ are in B. Conversely, if M_k^B rejects 1^n, we pick a string $x \in \{0,1\}^n$ that M_k^B has not queried (such a string exists because M_k^B can only do at most $2^n/10$ queries) and declared it to be in B.

Since ant polynomial $p(n)$ is smaller than $2^n/10$ for some large enough n, having no M_k^B that accepts U_B means we have that $U_B \notin \mathsf{P}^B$. □

Bibliography

[AB16] S. Arora and B. Barak, *Computational complexity: A Modern Approach*, 4th printing 2016. New York: Cambridge University Press, 2016.