TECHNISCHE
UNIVERSITÄT
DRESDEN

COMPLEXITY THEORY

Lecture 15: P vs. NP and Diagonalisation

Markus Krötzsch

Knowledge-Based Systems

TU Dresden, 3rd Dec 2019

Review

Ladner's Theorem

Theorem 15.1 (Ladner, 1975): If $\mathsf{P} \neq \mathsf{NP}$, then there are problems in NP that are neither in P nor NP-complete.

Such problems are called NP-intermediate.

- No natural problem is known to be NP-intermediate
- Indeed, this would imply that $\mathsf{P} \neq \mathsf{NP}$

BIRD

BIRD

There was this emperor and he asks this shepherd's boy:

"How many seconds in eternity?"

The shepherd's boy says:

"There's this mountain of pure diamond.
It takes an hour to climb it, and an hour to go around it.
Every hundred years, a little bird comes,
and sharpens its beak on the diamond mountain.
And when the entire mountain is chiselled away,
the first second of eternity will have passed."

(from a story by the Brothers Grimm/Steven Moffat)

Lazy diagonalisation

A powerful proof idea:

- Don't try to construct a diagonalisation that tries to flip the behaviour for a TM $\mathcal{M}_i$ that is given in the input
- Rather, for each $\mathcal{M}_i$ that you want to be different from, keep on behaving "sufficiently different" for a large range of inputs – until the inputs given are big enough to detect a difference with $\mathcal{M}_i$ (on much smaller inputs)
- To know which $\mathcal{M}_i$ we are working on, simply recompute $f(0), f(1), \ldots$ as far as possible in each step

We observed:

- Progress is really slow with this method.
- However, it does not matter how inefficient the actual computations are internally. Lazy diagonalisation has all the time in the world.

(the little bird keeps chiseling away diamond mountains for eternity)

Schöning's Generalisation

Intermediate problems between other classes

Uwe Schöning established the following interesting generalisation of Ladner's approach:

Theorem 15.2 (Schöning 1982): Consider two classes C_1 and C_2 of decidable languages such that for either class C_k:

- We can effectively enumerate TMs $\mathcal{M}_0^k, \mathcal{M}_1^k, \ldots$ that halt on all inputs and such that $\mathsf{C}_k = \{\mathbf{L}(\mathcal{M}_i^k) \mid i \geq 0)\}$.
- If $\mathbf{L} \in \mathsf{C}_k$ and **L'** differs from **L** on only a finite number of words, then $\mathbf{L'} \in \mathsf{C}_k$

If there are decidable languages $\mathbf{L_1} \notin \mathsf{C}_1$ and $\mathbf{L_2} \notin \mathsf{C}_2$, then there is a decidable language $\mathbf{L_d} \notin \mathsf{C}_1 \cup \mathsf{C}_2$.
Moreover, if $\mathbf{L_1} \in \mathsf{P}$ and $\mathbf{L_2}$ is not trivial (i.e., $\mathbf{L_2} \notin \{\emptyset, \Sigma^*\}$), then $\mathbf{L_d} \leq_p \mathbf{L_2}$.

This can be used for proving the existence of many other classes of intermediate problems.

The result has an elegant, not very long proof slightly different from our proof of Ladner's theorem, but using related ideas. See Uwe Schöning: A Uniform Approach to Obtain Diagonal Sets in Complexity Classes, Theor. Comput. Sci. 18, 95–103.

Example: Ladner's Theorem via Schöning

We obtain the previous result as a special case:

Corollary 15.3: Consider the classes $\mathrm{C}_1 = \mathrm{NPC}$ (NP-complete problems) and $\mathrm{C}_2 = \mathrm{P}$. We find that for either class C_k:

- We can effectively enumerate TMs $\mathcal{M}_0^k, \mathcal{M}_1^k, \ldots$ that halt on all inputs and such that $\mathrm{C}_k = \{\mathbf{L}(\mathcal{M}_i^k) \mid i \geq 0)\}$.
- If $\mathbf{L} \in \mathrm{C}_k$ and $\mathbf{L'}$ differs from $\mathbf{L}$ on only a finite number of words, then $\mathbf{L'} \in \mathrm{C}_k$

If $\mathrm{P} \neq \mathrm{NP}$, then $\mathbf{L_1} = \emptyset \notin \mathrm{NPC}$ and $\mathbf{L_2} = \textsc{Sat} \notin \mathrm{P}$, hence there is a decidable language $\mathbf{L_d} \notin \mathrm{NPC} \cup \mathrm{P}$.
Moreover, as $\emptyset \in \mathrm{P}$ and $\textsc{Sat}$ is not trivial, $\mathbf{L_d} \leq_p \textsc{Sat}$ and hence $\mathbf{L_d} \in \mathrm{NP}$.

Proof: Most properties are clear. The enumeration of NP-complete languages can be constructed by using ideas we have used to prove Ladner's theorem.

Enumerate all pairs of polytime TMs and polytime reductions, and construct a new machine for each pair: on input w, check if the reduction correctly reduces **Sat** to the given TM for all inputs up to this length. If yes, behave like the TM; if no, behave like **Sat**. Therefore, if a reduction is not working, the resulting machine will be like **Sat** in all but a finite number of places.

□

Example: Separating coNP from NPC

We obtain another result as a special case:

Corollary 15.4: Consider the classes $C_1 = $ NPC (NP-complete problems) and $C_2 = $ coNP. We find that for either class C_k:

- We can effectively enumerate TMs $\mathcal{M}_0^k, \mathcal{M}_1^k, \ldots$ that halt on all inputs and such that $C_k = \{\mathbf{L}(\mathcal{M}_i^k) \mid i \geq 0)\}$.
- If $\mathbf{L} \in C_k$ and $\mathbf{L'}$ differs from $\mathbf{L}$ on only a finite number of words, then $\mathbf{L'} \in C_k$

If NP $\neq$ coNP, then $\mathbf{L_1} = \emptyset \notin$ NPC and $\mathbf{L_2} = \textsc{Sat} \notin$ coNP, hence there is a decidable language $\mathbf{L_d} \notin$ NPC $\cup$ coNP.

Moreover, as $\emptyset \in$ P and $\textsc{Sat}$ is not trivial, $\mathbf{L_d} \leq_p \textsc{Sat}$ and hence $\mathbf{L_d} \in$ NP.

In words: There are problems in NP which are not in coNP and yet not NP-complete.

This is a stronger statement than Ladner's theorem, but it also uses a stronger assumption (namely that NP $\neq$ coNP).

Discussion: Schöning's Result

Many further classes of problems could be separated in this way.

The most critical questions for applying the theorem are:

- Can we really effectively enumerate TMs for the respective classes? (this is not trivial and not always true)
- Are the classes really closed under finite variations?
- Under which assumptions can we be sure that $\mathbf{L_1} \notin \mathcal{C}_1$ and $\mathbf{L_2} \notin \mathcal{C}_2$?

Hierarchies of intermediate problems

Another generalisation of Ladner's theorem comes about by applying similar arguments to find problems that are intermediate to, e.g., P and some intermediate problem.

This shows a lot of complicated structure in, e.g., NP:

- We can define classes of mutually $\leq_p$-reducible problems (even if not NP-complete), called polynomial many-one degrees
- These classes can be orderd by $\leq_p$ on their representatives

Hierarchies of intermediate problems

Another generalisation of Ladner's theorem comes about by applying similar arguments to find problems that are intermediate to, e.g., P and some intermediate problem.

This shows a lot of complicated structure in, e.g., NP:

- We can define classes of mutually $\leq_p$-reducible problems (even if not NP-complete), called polynomial many-one degrees
- These classes can be orderd by $\leq_p$ on their representatives

Fact 15.5: If P $\neq$ NP, then the order of polynomial many-one degrees is dense and non-total.

Recall:
Dense: Between any two elements is another one distinct from both
Non-total: There are incomparable elements

The Limits of Diagonalisation

The Power of Diagonalisation

We have established powerful results using diagonalisation arguments:

- Time and Space Hierarchy: even fine-grained time and space classes differ
- Ladner's Theorem: NP-intermediate problems differ from both P and NP-complete

What next?

Are there any other interesting open questions about the potential difference of some complexity classes that we would like to resolve?

The Power of Diagonalisation

We have established powerful results using diagonalisation arguments:

- Time and Space Hierarchy: even fine-grained time and space classes differ
- Ladner's Theorem: NP-intermediate problems differ from both P and NP-complete

What next?
Are there any other interesting open questions about the potential difference of some complexity classes that we would like to resolve?

How about separating P from NP?

The Power of Diagonalisation

We have established powerful results using diagonalisation arguments:

- Time and Space Hierarchy: even fine-grained time and space classes differ
- Ladner's Theorem: NP-intermediate problems differ from both P and NP-complete

What next?
Are there any other interesting open questions about the potential difference of some complexity classes that we would like to resolve?

How about separating P from NP?

- This has not been resolved using diagonalisation so far
- Indeed, we know some things that make it seem very unlikely that diagonalisation alone could succeed there

$\leadsto$ Coming up next . . .

Review: Oracles

Recall the following defintions from Lecture 3:

Definition 3.15: An Oracle Turing Machine (OTM) is a Turing machine $\mathcal{M}$ with a special tape, called the oracle tape, and distinguished states $q_?$, q_{yes}, and q_{no}. For a language **O**, the oracle machine $\mathcal{M}^{\mathbf{O}}$ can, in addition to the normal TM operations, do the following:

Whenever $\mathcal{M}^{\mathbf{O}}$ reaches $q_?$, its next state is q_{yes} if the content of the oracle tape is in **O**, and q_{no} otherwise.

Observe that invoking the oracle always takes one step only

Definition 3.16: A problem **P** is Turing reducible to a problem **Q** (in symbols: $\mathbf{P} \leq_T \mathbf{Q}$), if **P** is decided by an OTM $\mathcal{M}^{\mathbf{Q}}$ with oracle **Q**.

The following is immediate from the definition:

Proposition 15.6: Turing reducibility $\leq_T$ is transitive.

Cook vs. Karp

One can talk about polynomial-time Turing reductions: if $\mathcal{M}^{\mathbf{Q}}$ is a polynomially time bounded OTM that decides **L** we may write $\mathbf{L} \leq_T^p \mathbf{Q}$.

Polynomial Turing reductions are also known as Cook reductions, while polynomial many-one reductions are known as Karp reductions.

Cook vs. Karp

One can talk about polynomial-time Turing reductions: if $\mathcal{M}^{\mathbf{Q}}$ is a polynomially time bounded OTM that decides **L** we may write $\mathbf{L} \leq_T^p \mathbf{Q}$.

Polynomial Turing reductions are also known as Cook reductions, while polynomial many-one reductions are known as Karp reductions.

Cook reductions are more powerful than Karp reductions, since they can make use of solutions to another problem many times within a single run, rather than only once at the very end. We observe:

Proposition 15.7: $\leq_p \subseteq \leq_T^p$.

Cook vs. Karp

One can talk about polynomial-time Turing reductions: if $\mathcal{M}^{\mathbf{Q}}$ is a polynomially time bounded OTM that decides **L** we may write $\mathbf{L} \leq_T^p \mathbf{Q}$.

Polynomial Turing reductions are also known as Cook reductions, while polynomial many-one reductions are known as Karp reductions.

Cook reductions are more powerful than Karp reductions, since they can make use of solutions to another problem many times within a single run, rather than only once at the very end. We observe:

Proposition 15.7: $\leq_p \subseteq \leq_T^p$.

Example 15.8: It is easy to see that $\textsc{Tautology} \leq_T^p \textsc{Sat}$, whereas $\textsc{Tautology} \leq_p \textsc{Sat}$ is neither known nor expected.

Complexity classes with oracles

Definition 15.9: Consider a language $\mathbf{O} \subseteq \Sigma^*$:

- $\mathrm{P}^{\mathbf{O}}$ is the set of all languages decidable by a deterministic polynomial-time OTM with oracle $\mathbf{O}$,
- $\mathrm{NP}^{\mathbf{O}}$ is the set of all languages decidable by a non-deterministic polynomial-time OTM with oracle $\mathbf{O}$.

Complexity classes with oracles

Definition 15.9: Consider a language $\mathbf{O} \subseteq \Sigma^*$:

- $\mathsf{P}^{\mathbf{O}}$ is the set of all languages decidable by a deterministic polynomial-time OTM with oracle $\mathbf{O}$,
- $\mathsf{NP}^{\mathbf{O}}$ is the set of all languages decidable by a non-deterministic polynomial-time OTM with oracle $\mathbf{O}$.

Some simple observations:

Proposition 15.10: If $\mathbf{O} \in \mathsf{P}$ then $\mathsf{P}^{\mathbf{O}} = \mathsf{P}$.

Complexity classes with oracles

Definition 15.9: Consider a language $\mathbf{O} \subseteq \Sigma^*$:

- $\mathsf{P}^{\mathbf{O}}$ is the set of all languages decidable by a deterministic polynomial-time OTM with oracle $\mathbf{O}$,
- $\mathsf{NP}^{\mathbf{O}}$ is the set of all languages decidable by a non-deterministic polynomial-time OTM with oracle $\mathbf{O}$.

Some simple observations:

Proposition 15.10: If $\mathbf{O} \in \mathsf{P}$ then $\mathsf{P}^{\mathbf{O}} = \mathsf{P}$.

Proposition 15.11: $\mathsf{NP} \subseteq \mathsf{P}^{\text{SAT}}$.

Complexity classes with oracles

Definition 15.9: Consider a language $\mathbf{O} \subseteq \Sigma^*$:

- $\mathrm{P}^{\mathbf{O}}$ is the set of all languages decidable by a deterministic polynomial-time OTM with oracle $\mathbf{O}$,
- $\mathrm{NP}^{\mathbf{O}}$ is the set of all languages decidable by a non-deterministic polynomial-time OTM with oracle $\mathbf{O}$.

Some simple observations:

Proposition 15.10: If $\mathbf{O} \in \mathrm{P}$ then $\mathrm{P}^{\mathbf{O}} = \mathrm{P}$.

Proposition 15.11: $\mathrm{NP} \subseteq \mathrm{P}^{\textsc{Sat}}$.

Proof: If $\mathbf{L} \in \mathrm{NP}$, then there is a polynomial many-one reduction f from $\mathbf{L}$ to **Sat**, i.e., $w \in \mathbf{L}$ iff $f(w) \in$ **Sat**. A polytime OTM for $\mathbf{L}$, on input w, simply computes $f(w)$, and invokes the oracle to decide $f(w) \in$ **Sat**. $\square$

Complexity classes with oracles

Definition 15.9: Consider a language $\mathbf{O} \subseteq \Sigma^*$:

- $\mathrm{P}^{\mathbf{O}}$ is the set of all languages decidable by a deterministic polynomial-time OTM with oracle $\mathbf{O}$,
- $\mathrm{NP}^{\mathbf{O}}$ is the set of all languages decidable by a non-deterministic polynomial-time OTM with oracle $\mathbf{O}$.

Some simple observations:

Proposition 15.10: If $\mathbf{O} \in \mathrm{P}$ then $\mathrm{P}^{\mathbf{O}} = \mathrm{P}$.

Proposition 15.11: $\mathrm{NP} \subseteq \mathrm{P}^{\textsc{Sat}}$.

Proof: If $\mathbf{L} \in \mathrm{NP}$, then there is a polynomial many-one reduction f from $\mathbf{L}$ to $\textsc{Sat}$, i.e., $w \in \mathbf{L}$ iff $f(w) \in \textsc{Sat}$. A polytime OTM for $\mathbf{L}$, on input w, simply computes $f(w)$, and invokes the oracle to decide $f(w) \in \textsc{Sat}$. □

Proposition 15.12: $\mathrm{P}^{\mathbf{O}}$ is closed under complement. In particular, $\mathrm{coNP} \subseteq \mathrm{P}^{\textsc{Sat}}$, and $\mathrm{NP} \neq \mathrm{P}^{\textsc{Sat}}$ unless $\mathrm{NP} = \mathrm{coNP}$.

Relativisation

Relativisation is the generalisation of a result about TMs to OTMs for a fixed oracle. It is interesting to ask which results hold true relative to particular oracles.

Relativisation

Relativisation is the generalisation of a result about TMs to OTMs for a fixed oracle. It is interesting to ask which results hold true relative to particular oracles.

Proofs by diagonalisation typically relativise: they remain correct if TMs are allowed oracle calls. Indeed, our proofs so far only relied on two properties:

- TMs can be represented as strings (and therefore also iterated over)
- A TM can simulate another one (given as string) without much overhead

Both remain true when looking at machines for a (fixed) oracle **O**.

Relativisation

Relativisation is the generalisation of a result about TMs to OTMs for a fixed oracle. It is interesting to ask which results hold true relative to particular oracles.

Proofs by diagonalisation typically relativise: they remain correct if TMs are allowed oracle calls. Indeed, our proofs so far only relied on two properties:

- TMs can be represented as strings (and therefore also iterated over)
- A TM can simulate another one (given as string) without much overhead

Both remain true when looking at machines for a (fixed) oracle **O**.

Example 15.13: Consider TMs that can use the Halting problem as an oracle. Such OTMs cannot decide the Halting problem for machines of their own type. Suppose for a contradiction that they could. We can then construct a diagonal TM $\mathcal{D}$ that, on input $\langle\mathcal{M}\rangle$, simulates $\mathcal{M}$ on $\langle\mathcal{M}\rangle$ and flips the result. $\mathcal{D}$ on input $\langle\mathcal{D}\rangle$ leads to a contradiction.

Note: OTMs obtained my providing oracles for the Halting problem of a simpler type of (O)TM give rise to so-called Turing jumps. They produce some of the families of undecidable languages considered in the theory of computability.

The limits of diagonalisation

The fact that standard diagonalisation relativises suggests that it is too weak to settle questions whose answers do not relativise.

The limits of diagonalisation

The fact that standard diagonalisation relativises suggests that it is too weak to settle questions whose answers do not relativise.

A prominent example was discovered in the 1970s:

Theorem 15.14 (Baker, Gill, Solovay, 1975): The answer to $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ does not relativise: there are languages **A** and **B** such that $\mathsf{P}^{\mathbf{A}} = \mathsf{NP}^{\mathbf{A}}$ and $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$.

Therefore, any proof that answers $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ must be based on some property of (normal) TMs that does not relativise.

Ironically, we will prove the second part of the theorem by diagonalisation!

Proving the theorem: $\mathsf{P}^{\mathbf{A}} = \mathsf{NP}^{\mathbf{A}}$

Theorem 15.15 (Baker, Gill, Solovay, 1975): The answer to $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ does not relativise: there are languages $\mathbf{A}$ and $\mathbf{B}$ such that $\mathsf{P}^{\mathbf{A}} = \mathsf{NP}^{\mathbf{A}}$ and $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$.

Proof: It is not so hard to find a suitable $\mathbf{A}$ for the first part of the theorem: the power of the oracle must be such that deterministic and non-deterministic computations coincide.

Proving the theorem: $\mathsf{P}^{\mathbf{A}} = \mathsf{NP}^{\mathbf{A}}$

Theorem 15.15 (Baker, Gill, Solovay, 1975): The answer to $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ does not relativise: there are languages **A** and **B** such that $\mathsf{P}^{\mathbf{A}} = \mathsf{NP}^{\mathbf{A}}$ and $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$.

Proof: It is not so hard to find a suitable **A** for the first part of the theorem: the power of the oracle must be such that deterministic and non-deterministic computations coincide.

We know such complexity classes, e.g., PSpace. Therefore set $\mathbf{A} = \textbf{True QBF}$. Then:

$$\mathsf{NP}^{\textbf{True QBF}} \stackrel{(1)}{\subseteq} \mathsf{NPSpace} \stackrel{(2)}{\subseteq} \mathsf{PSpace} \stackrel{(3)}{\subseteq} \mathsf{P}^{\textbf{True QBF}}$$

Inclusion (1) follows from the observation that every nondeterministic OTM with oracle **True QBF** running in polytime can be simulated by an NTM running in polynomial space. Inclusion (2) is Savitch's Theorem. (3) follows from the PSpace-hardness of **True QBF**. Since $\mathsf{P}^{\textbf{True QBF}} \subseteq \mathsf{NP}^{\textbf{True QBF}}$, we obtain $\mathsf{NP}^{\textbf{True QBF}} = \mathsf{P}^{\textbf{True QBF}}$.

Proving the theorem: $\mathbf{P}^{\mathbf{B}} \neq \mathbf{NP}^{\mathbf{B}}$

Theorem 15.16 (Baker, Gill, Solovay, 1975): The answer to $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ does not relativise: there are languages **A** and **B** such that $\mathrm{P}^{\mathbf{A}} = \mathrm{NP}^{\mathbf{A}}$ and $\mathrm{P}^{\mathbf{B}} \neq \mathrm{NP}^{\mathbf{B}}$.

Proof (continued): To show $\mathrm{P}^{\mathbf{B}} \neq \mathrm{NP}^{\mathbf{B}}$, we define from an arbitrary language $\mathbf{C} \subseteq \Sigma^*$, a language $\mathbf{L_C}$ as follows:

$$\mathbf{L_C} = \{1^n \mid \text{there is } v \in \mathbf{C} \text{ with } |v| = n\}$$

Proving the theorem: $\mathbf{P}^{\mathbf{B}} \neq \mathbf{NP}^{\mathbf{B}}$

Theorem 15.16 (Baker, Gill, Solovay, 1975): The answer to $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ does not relativise: there are languages **A** and **B** such that $\mathsf{P}^{\mathbf{A}} = \mathsf{NP}^{\mathbf{A}}$ and $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$.

Proof (continued): To show $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$, we define from an arbitrary language $\mathbf{C} \subseteq \Sigma^*$, a language $\mathbf{L_C}$ as follows:

$$\mathbf{L_C} = \{1^n \mid \text{there is } v \in \mathbf{C} \text{ with } |v| = n\}$$

- Clearly, $\mathbf{L_C} \in \mathsf{NP}^{\mathbf{C}}$: Given an input of form 1^n, an NTM can guess a suitable word v and use the **C**-oracle to check the guess
- Whether $\mathbf{L_C} \in \mathsf{P}^{\mathbf{C}}$ or not depends on **C**. (We have already shown that $\mathbf{L}_{\textsc{True QBF}} \in \mathsf{P}^{\textsc{True QBF}}$ must hold)

Proving the theorem: $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$

Theorem 15.16 (Baker, Gill, Solovay, 1975): The answer to $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ does not relativise: there are languages **A** and **B** such that $\mathsf{P}^{\mathbf{A}} = \mathsf{NP}^{\mathbf{A}}$ and $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$.

Proof (continued): To show $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$, we define from an arbitrary language $\mathbf{C} \subseteq \Sigma^*$, a language $\mathbf{L}_{\mathbf{C}}$ as follows:

$$\mathbf{L}_{\mathbf{C}} = \{1^n \mid \text{there is } v \in \mathbf{C} \text{ with } |v| = n\}$$

- Clearly, $\mathbf{L}_{\mathbf{C}} \in \mathsf{NP}^{\mathbf{C}}$: Given an input of form 1^n, an NTM can guess a suitable word v and use the **C**-oracle to check the guess
- Whether $\mathbf{L}_{\mathbf{C}} \in \mathsf{P}^{\mathbf{C}}$ or not depends on **C**. (We have already shown that $\mathbf{L}_{\textsc{True QBF}} \in \mathsf{P}^{\textsc{True QBF}}$ must hold)

We will construct the required language **B** such that $\mathbf{L}_{\mathbf{B}} \notin \mathsf{P}^{\mathbf{B}}$.

We only consider the input alphabet $\Sigma = \{0, 1\}$. Let $\mathcal{M}_0, \mathcal{M}_1, \ldots$ be some enumeration polynomial time-bounded OTMs (with oracle **B**, but this is immaterial for the description of the OTM) for Σ. Let p_i be a polynomial that provides a time bound for $\mathcal{M}_i$.

Proving the theorem: $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$

Theorem 15.17 (Baker, Gill, Solovay, 1975): The answer to $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ does not relativise: there are languages **A** and **B** such that $\mathsf{P}^{\mathbf{A}} = \mathsf{NP}^{\mathbf{A}}$ and $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$.

Proof (continued): We construct **B** such that $\mathbf{L_B} = \{1^n \mid \text{there is } v \in \mathbf{B} \text{ with } |v| = n\} \notin \mathsf{P}^{\mathbf{B}}$.

Proving the theorem: $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$

Theorem 15.17 (Baker, Gill, Solovay, 1975): The answer to $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ does not relativise: there are languages **A** and **B** such that $\mathsf{P}^{\mathbf{A}} = \mathsf{NP}^{\mathbf{A}}$ and $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$.

Proof (continued): We construct **B** such that $\mathbf{L_B} = \{1^n \mid \text{there is } v \in \mathbf{B} \text{ with } |v| = n\} \notin \mathsf{P}^{\mathbf{B}}$.

We define **B** from short words to longer words, proceeding in stages $0, 1, \ldots$

Proving the theorem: $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$

Theorem 15.17 (Baker, Gill, Solovay, 1975): The answer to P $\stackrel{?}{=}$ NP does not relativise: there are languages **A** and **B** such that $\mathsf{P}^{\mathbf{A}} = \mathsf{NP}^{\mathbf{A}}$ and $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$.

Proof (continued): We construct **B** such that $\mathbf{L_B} = \{1^n \mid \text{there is } v \in \mathbf{B} \text{ with } |v| = n\} \notin \mathsf{P}^{\mathbf{B}}$.

We define **B** from short words to longer words, proceeding in stages $0, 1, \ldots$
In stage $i \geq 0$, do the following:

- Pick a number n such that (1) n is longer than the longest string w for which $w \stackrel{?}{\in} \mathbf{B}$ has been defined yet, and (2) $2^n > p_i(n)$.

Proving the theorem: $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$

Theorem 15.17 (Baker, Gill, Solovay, 1975): The answer to $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ does not relativise: there are languages **A** and **B** such that $\mathsf{P}^{\mathbf{A}} = \mathsf{NP}^{\mathbf{A}}$ and $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$.

Proof (continued): We construct **B** such that $\mathbf{L_B} = \{1^n \mid \text{there is } v \in \mathbf{B} \text{ with } |v| = n\} \notin \mathsf{P}^{\mathbf{B}}$.

We define **B** from short words to longer words, proceeding in stages $0, 1, \ldots$
In stage $i \geq 0$, do the following:

- Pick a number n such that (1) n is longer than the longest string w for which $w \stackrel{?}{\in} \mathbf{B}$ has been defined yet, and (2) $2^n > p_i(n)$.
- Simulate the (deterministic) run of $\mathcal{M}_i$ on 1^n. For oracle calls, $w \stackrel{?}{\in} \mathbf{B}$:
  - If $w \stackrel{?}{\in} \mathbf{B}$ was already defined, answer as defined.
  - If $w \stackrel{?}{\in} \mathbf{B}$ was not defined before, answer no (and define $w \notin \mathbf{B}$).

Proving the theorem: $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$

Theorem 15.17 (Baker, Gill, Solovay, 1975): The answer to $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ does not relativise: there are languages **A** and **B** such that $\mathsf{P}^{\mathbf{A}} = \mathsf{NP}^{\mathbf{A}}$ and $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$.

Proof (continued): We construct **B** such that $\mathbf{L_B} = \{1^n \mid \text{there is } v \in \mathbf{B} \text{ with } |v| = n\} \notin \mathsf{P}^{\mathbf{B}}$.

We define **B** from short words to longer words, proceeding in stages $0, 1, \ldots$
In stage $i \geq 0$, do the following:

- Pick a number n such that (1) n is longer than the longest string w for which $w \stackrel{?}{\in} \mathbf{B}$ has been defined yet, and (2) $2^n > p_i(n)$.
- Simulate the (deterministic) run of $\mathcal{M}_i$ on 1^n. For oracle calls, $w \stackrel{?}{\in} \mathbf{B}$:
  - If $w \stackrel{?}{\in} \mathbf{B}$ was already defined, answer as defined.
  - If $w \stackrel{?}{\in} \mathbf{B}$ was not defined before, answer no (and define $w \notin \mathbf{B}$).
- For all w with $|w| \leq n$ where $w \stackrel{?}{\in} \mathbf{B}$ was not defined yet, define
  - $w \in \mathbf{B}$ if $\mathcal{M}_i$ has rejected 1^n
  - $w \notin \mathbf{B}$ if $\mathcal{M}_i$ has accepted 1^n

Proving the theorem: $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$

Theorem 15.17 (Baker, Gill, Solovay, 1975): The answer to $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ does not relativise: there are languages **A** and **B** such that $\mathsf{P}^{\mathbf{A}} = \mathsf{NP}^{\mathbf{A}}$ and $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$.

Proof (continued): We construct **B** such that $\mathbf{L_B} = \{1^n \mid \text{there is } v \in \mathbf{B} \text{ with } |v| = n\} \notin \mathsf{P}^{\mathbf{B}}$.

We define **B** from short words to longer words, proceeding in stages $0, 1, \ldots$
In stage $i \geq 0$, do the following:

- Pick a number n such that (1) n is longer than the longest string w for which $w \stackrel{?}{\in} \mathbf{B}$ has been defined yet, and (2) $2^n > p_i(n)$.
- Simulate the (deterministic) run of $\mathcal{M}_i$ on 1^n. For oracle calls, $w \stackrel{?}{\in} \mathbf{B}$:
  - If $w \stackrel{?}{\in} \mathbf{B}$ was already defined, answer as defined.
  - If $w \stackrel{?}{\in} \mathbf{B}$ was not defined before, answer no (and define $w \notin \mathbf{B}$).
- For all w with $|w| \leq n$ where $w \stackrel{?}{\in} \mathbf{B}$ was not defined yet, define
  - $w \in \mathbf{B}$ if $\mathcal{M}_i$ has rejected 1^n
  - $w \notin \mathbf{B}$ if $\mathcal{M}_i$ has accepted 1^n

Stage i therefore ends with $w \stackrel{?}{\in} \mathbf{B}$ defined for all words w up to length n.

Proving the theorem: $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$

Theorem 15.18 (Baker, Gill, Solovay, 1975): The answer to $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ does not relativise: there are languages **A** and **B** such that $\mathsf{P}^{\mathbf{A}} = \mathsf{NP}^{\mathbf{A}}$ and $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$.

Proof (continued): The definitions of stage i ensure that $\mathcal{M}_i$ does not decide $\mathbf{L}_{\mathbf{B}}$, since (for the chosen n) we have $\mathbf{L}_{\mathbf{B}}(1^n) \neq \mathcal{M}_i(1^n)$:

Proving the theorem: $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$

Theorem 15.18 (Baker, Gill, Solovay, 1975): The answer to $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ does not relativise: there are languages **A** and **B** such that $\mathsf{P}^{\mathbf{A}} = \mathsf{NP}^{\mathbf{A}}$ and $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$.

Proof (continued): The definitions of stage i ensure that $\mathcal{M}_i$ does not decide $\mathbf{L}_{\mathbf{B}}$, since (for the chosen n) we have $\mathbf{L}_{\mathbf{B}}(1^n) \neq \mathcal{M}_i(1^n)$:

- If $1^n \in \mathbf{L}(\mathcal{M}_i)$

Proving the theorem: $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$

Theorem 15.18 (Baker, Gill, Solovay, 1975): The answer to $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ does not relativise: there are languages **A** and **B** such that $\mathsf{P}^{\mathbf{A}} = \mathsf{NP}^{\mathbf{A}}$ and $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$.

Proof (continued): The definitions of stage i ensure that $\mathcal{M}_i$ does not decide $\mathbf{L_B}$, since (for the chosen n) we have $\mathbf{L_B}(1^n) \neq \mathcal{M}_i(1^n)$:

- If $1^n \in \mathbf{L}(\mathcal{M}_i)$, then we have defined **B** to contain no words of length n, so $1^n \notin \mathbf{L_B}$.

Proving the theorem: $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$

Theorem 15.18 (Baker, Gill, Solovay, 1975): The answer to $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ does not relativise: there are languages **A** and **B** such that $\mathsf{P}^{\mathbf{A}} = \mathsf{NP}^{\mathbf{A}}$ and $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$.

Proof (continued): The definitions of stage i ensure that $\mathcal{M}_i$ does not decide $\mathbf{L}_{\mathbf{B}}$, since (for the chosen n) we have $\mathbf{L}_{\mathbf{B}}(1^n) \neq \mathcal{M}_i(1^n)$:

- If $1^n \in \mathbf{L}(\mathcal{M}_i)$, then we have defined **B** to contain no words of length n, so $1^n \notin \mathbf{L}_{\mathbf{B}}$.
- If $1^n \notin \mathbf{L}(\mathcal{M}_i)$

Proving the theorem: $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$

Theorem 15.18 (Baker, Gill, Solovay, 1975): The answer to $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ does not relativise: there are languages **A** and **B** such that $\mathsf{P}^{\mathbf{A}} = \mathsf{NP}^{\mathbf{A}}$ and $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$.

Proof (continued): The definitions of stage i ensure that $\mathcal{M}_i$ does not decide $\mathbf{L}_{\mathbf{B}}$, since (for the chosen n) we have $\mathbf{L}_{\mathbf{B}}(1^n) \neq \mathcal{M}_i(1^n)$:

- If $1^n \in \mathbf{L}(\mathcal{M}_i)$, then we have defined **B** to contain no words of length n, so $1^n \notin \mathbf{L}_{\mathbf{B}}$.
- If $1^n \notin \mathbf{L}(\mathcal{M}_i)$, then there is a word w of length $|w| = n$ such that the run of $\mathcal{M}_i$ on 1^n did not make an oracle call for $w \stackrel{?}{\in} \mathbf{B}$. This follows since there are 2^n words of this length, but at most $p_i(n) < 2^n$ oracle calls. Hence, we have defined $w \in \mathbf{B}$, so $1^n \in \mathbf{L}_{\mathbf{B}}$.

Proving the theorem: $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$

Theorem 15.18 (Baker, Gill, Solovay, 1975): The answer to P $\stackrel{?}{=}$ NP does not relativise: there are languages **A** and **B** such that $\mathsf{P}^{\mathbf{A}} = \mathsf{NP}^{\mathbf{A}}$ and $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$.

Proof (continued): The definitions of stage i ensure that $\mathcal{M}_i$ does not decide $\mathbf{L_B}$, since (for the chosen n) we have $\mathbf{L_B}(1^n) \neq \mathcal{M}_i(1^n)$:

- If $1^n \in \mathbf{L}(\mathcal{M}_i)$, then we have defined **B** to contain no words of length n, so $1^n \notin \mathbf{L_B}$.
- If $1^n \notin \mathbf{L}(\mathcal{M}_i)$, then there is a word w of length $|w| = n$ such that the run of $\mathcal{M}_i$ on 1^n did not make an oracle call for $w \stackrel{?}{\in} \mathbf{B}$. This follows since there are 2^n words of this length, but at most $p_i(n) < 2^n$ oracle calls. Hence, we have defined $w \in \mathbf{B}$, so $1^n \in \mathbf{L_B}$.

Since $\mathbf{L_B} \neq \mathbf{L}(\mathcal{M}_i)$ holds for all polynomial time bounded determinsitic OTMs $\mathcal{M}_i$ with oracle **B**, we obtain $\mathbf{L_B} \notin \mathsf{P}^{\mathbf{B}}$. □

Discussion: The proof of Baker/Gill/Solovay

Note 1: Our proof for $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$ would work not just for $\mathsf{P}^{\mathbf{B}}$ but for any $\mathsf{DTime}(f)^{\mathbf{B}}$ where $f \in o(2^n)$.

Discussion: The proof of Baker/Gill/Solovay

Note 1: Our proof for $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$ would work not just for $\mathsf{P}^{\mathbf{B}}$ but for any $\mathsf{DTime}(f)^{\mathbf{B}}$ where $f \in o(2^n)$.

Note 2: The proof uses diagonalisation, but not as a method for specifying Turing machines. Indeed, we do not require **B** to be computable (though it is), and we do not have to ensure that a particular resource-bounded TM can compute it.

Summary and Outlook

Ladner's theorem can be generalised to find intermediate problems elsewhere

Many results in complexity theory relativise to oracle TMs for some oracle (the same for all TMs considered)

The P vs. NP question does not relativise, as a famous result of Baker, Gill, and Solovay tells us

What's next?

- Generalising NTMs with alternation
- A hierarchy between NP and PSpace