Lecture #13: What If...? More about Nondeterministic Computation

More about Relativization and Relativized Proofs

This document includes a proof of the result, which was included at the end of the notes for Lecture #13.

Theorem (Baker, Gill, and Solovay)**.**

(a) There exists a language $A \subseteq \Sigma_A^\star$ (for some alphabet Σ_A) such that $\mathcal{P}^A = \mathcal{NP}^A$.

(b) There exists a language $B \subseteq \Sigma_B^\star$ (for some alphabet Σ_B) such that $\mathcal{P}^B \neq \mathcal{NP}^B$.

Once again, this is "for interest only:" CPSC 511 will not need to read this material in order to do well in this course.

A Language A Such That $\mathcal{P}^A = \mathcal{NP}^A$

Let Σ_{UTM} be the alphabet used to encode deterministic Turing machines and their inputs, described in Lecture #4. Consider the language $L_{\mathit{EXPTIME}} \subseteq \Sigma_{\text{UTM}}^\star$ consisting of encodings of

- a deterministic Turing machine M,
- an input string ω for M, and
- a positive integer t — encoded using its ***binary*** representation

such that M accepts ω after taking at most t steps.

Easy Exercise: Prove that L_{EXPTIME} is complete for *EXPTIME* with respect to polynomial-time many-one reductions.

Now let $A = L_{\mathit{EXPTIME}}$.

On input $\omega \in \Sigma^\star$:

1. Write onto the query tape a string in $\Sigma^\star_{\text{UTM}}$ that encodes
    - the deterministic Turing machine M,
    - the input string ω, and
    - the integer $t = c \cdot 2^{(n^k)}$ (in binary), where $n = |\omega|$.

   Then enter the query state.
2. If the machine is now in state q_Y then ***accept*** ω. Otherwise, ***reject*** ω.

Figure 1: Oracle Computation to Decide Membership in L

Claim 1. EXPTIME $\subseteq \mathcal{P}^A$*.*

Proof. Since $L \in$ *EXPTIME* there exists a deterministic Turing machine M and positive integer constants c and k such that

- M decides L, and
- M uses at most $c \cdot 2^{(n^k)}$ steps on every string $\omega \in \Sigma^\star$ with length n, for every integer $n \geq 0$.

Consider a ***deterministic oracle Turing machine*** $\widehat{M}$ — with an oracle for A — that implements the algorithm shown in Figure 1. This is a deterministic oracle Turing machine, with an oracle for $L_{\textit{EXPTIME}}$, that decides the language L. Furthermore, since c and k are positive integer *constants* the binary representation of $t = c \cdot 2^{(n^k)}$ is a string with length polynomial in n that can be computed from n, deterministically, using time polynomial in n as well.

Since M is a *fixed* Turing machine, its (smallest) encoding is a fixed string, and step 1 of this algorithm can be carried out deterministically using time polynomial in $n = |\omega|$ in the worst case. Since step 2 only requires a constant number of steps, it follows that $L \in \mathcal{P}^A$.

Since L was an arbitrarily chosen language in *EXPTIME* it follows that *EXPTIME* $\subseteq \mathcal{P}^A$. as claimed. ☐

Certainly, $\mathcal{P}^A \subseteq \mathcal{NP}^A$. If $\mathcal{NP}^A \subseteq$ *EXPTIME* then it will follow that

$$\textit{EXPTIME} \subseteq \mathcal{P}^A \subseteq \mathcal{NP}^A \subseteq \textit{EXPTIME}$$

so that $\mathcal{P}^A = \mathcal{NP}^A$, as claimed. Hence the following result is all that is needed to complete a proof of part (a) of the theorem.

On input $\omega \in \Sigma^\star$:

1. Use exhaustive search to traverse the computation tree for M on input ω. Go to step 2 whenever M enters its query state.

   If an accepting configuration is found in the computation tree being traversed then ***accept*** ω. Otherwise (after the entire computation tree has been traversed) ***reject*** ω.
2. If the non-blank part of M's query tape stores a single string in $\Sigma^\star_{\text{UTM}}$ then execute $\widehat{M}$, using this as the input string. If $\widehat{M}$ *accepts* this string then erase the query tape, move the query tape head back to the leftmost cell, and enter state q_Y — returning control to the simulation of M, and resuming step 1.

   In all other cases erase the query tape, move the query tape head back to the leftmost cell, and enter state q_N — returning control to the simulation of M and resuming step 1, once again.

Figure 2: Algorithm to Establish Membership in *EXPTIME*

Claim 2. $\mathcal{NP}^A \subseteq$ EXPTIME.

Proof. The result will be established by showing that it is possible to deterministically complete a nondeterministic polynomial-time computation, that includes oracle queries for A, in singly exponential time.

With that noted, let $L \subseteq \Sigma^\star$ such that $L \in \mathcal{NP}^A$. It is necessary and sufficient to show that $L \in$ *EXPTIME*.

Since $L \in \mathcal{NP}^A$ there exists a ***nondeterministic*** oracle Turing machine M with an oracle for A, that decides L, using time at most $c_1 n^k + c_0$ for every input string $\omega \in \Sigma^\star$ with length n — for positive integer constants c_1, k, and c_0. It follows that there is also a positive integer constant $\widehat{c}$ (which depends on M's transition function) such that the computation tree for M, on an input string $\omega \in \Sigma^\star$ with length n, has ***size*** at most $\widehat{c}^{(c_1 n^k + c_0)}$.

This computation tree includes ***oracle queries.*** However, since $A = L_{\textit{EXPTIME}} \in$ *EXPTIME*, there is a deterministic Turing machine $\widehat{M}$ that decides $L_{\textit{EXPTIME}}$, using time at most $d \cdot 2^{(n^{\widehat{k}})}$, for any input string with length n, for positive integer constants d and $\widehat{k}$.

Now consider a ***deterministic*** Turing machine, $\widetilde{M}$, with input alphabet Σ, that implements the algorithm shown in Figure 2, above. It can be shown that $\widetilde{M}$ is a (standard) deterministic Turing machine that decides the language L

Since $\widehat{M}$ is a *fixed* deterministic Turing machine, $\widetilde{M}$ can (essentially) include a copy of it, so that the simulation of each move of $\widehat{M}$, included in executions of step 2, can be carried out using a constant number of moves of $\widetilde{M}$. A simulation of $\widehat{M}$ also includes an "initialization" phase, in which the query tape is used to set up $\widehat{M}$'s input tape. The cost to "clean up"' the oracle tape and reposition the tape head, at the end, should also be considered. With a bit of work, it can be argued that the cost of this is at most linear in the number of moves made by M since the *last* time the oracle got queried — so the total cost of this is at most linear in the cost needed to simulate M, and can now be ignored: It will be dominated by the cost of other steps that *are* being counted.

Each tape of M can be included as a tape of $\widetilde{M}$. M's "current state"' can either be remembered using $\widetilde{M}$'s finite control or stored as a symbol on a separate tape.

One more tape of $\widetilde{M}$ can be used to represent a ***stack***, representing the branch in M's computation tree that is currently being checked — including sequence of moves that were made. If this *also* stores information about when "moves left" caused a tape head to stay where it is, because the tape head was already at the leftmost cell, then this allows a move to be ***reversed*** or ***rolled back*** when information is popped off this stack.

This is all you need for ***depth-first search*** to be used to traverse M's computation tree on the input string, using a number of steps — excluding the cost for executions of $\widehat{M}$ — that is linear in the size, $\widehat{c}^{(c_1 n^k + c_0)}$, of this tree.

Next consider any *branch* (path from the roof down to a leaf) in this tree. The number of these is certainly also at least $\widehat{c}^{(c_1 n^k + c_0)}$, and the sum of the lengths of all strings (in $\Sigma_{\mathsf{UTM}}^\star$) that are written onto M's query tape and asked about, as this branch down the tree is executed, must be at most the ***depth*** of this branch — at most $c_1 n^k + c_0$.

Let $f(m) = d \cdot 2^{(m^{\widehat{k}})}$ — and notice that if $m_1, m_2 \in \mathbb{N}$ then

$$f(m_1) + f(m_2) \leq f(m_1 + m_2).$$

This can be used to show that the total time needed, for *all* simulations of $\widehat{M}$, as any branch in M's computation tree is checked, is at most

$$f(c_1 n^k + c_0) = d \cdot 2^{((c_1 n^k + c_0)^{\widehat{k}})}$$

Adding everything up, one can see that the time used by $\widetilde{M}$ is at most linear in

$$\widehat{c}^{(c_1 n^k + c_0)} \left(c_1 n^k + c_0 + d 2^{((c_1 n^k + c_0)^{\widehat{k}})} \right) \in O\left(2^{n^\ell} \right)$$

for some integer ***constant*** ℓ such that

$$\ell \geq (k+1) \cdot \lceil \log_2 \widehat{c}\, \rceil + (k+1)(\widehat{k}+1).$$

It follows that $L \in$ *EXPTIME*. Since L was arbitrarily chosen from $\mathcal{NP}^A$, this implies that $\mathcal{NP}^A \subseteq$ *EXPTIME* — as claimed. $\square$

As noted above it now follows that $\mathcal{P}^A = \mathcal{NP}^A$, as needed to establish part (a) of the theorem.

A Language B such that $\mathcal{P}^B \neq \mathcal{NP}^B$

Now let $\Sigma = \{\mathtt{0}, \mathtt{1}\}$. Let $B \subseteq \Sigma^\star$ be a language to be described later, and let $L_B \subseteq \{\mathtt{1}\}^\star$ such that, for every integer $n \geq 0$, $\mathtt{1}^n \in L_B$ ***if and only if*** B includes ***at least one*** string in $\Sigma^\star$ with length n:

$$\mathtt{1}^n \in L_B \iff B \cap \Sigma^n \neq \emptyset.$$

Claim 3. *$L_B \in \mathcal{NP}^B$, for* **every** *language $B \subseteq \Sigma^\star$.*

Proof. Let $B \subseteq \Sigma^\star$. Consider a nondeterministic oracle Turing machine, with an oracle for B, that sweeps over its input ***guessing*** symbols and writing them onto the query tape as it goes, so that if the input is $\mathtt{1}^n$ for $n \geq 0$ then a string in $\Sigma^\star$ with length n is written onto it.

If the oracle Turing machine then enters its query state and, one step later.

- ***accepts*** if the next state entered is q_Y, and
- ***rejects*** if the next state entered is q_N,

then this is a nondeterministic oracle Turing machine, with an oracle for B, that decides L_B, using at most $n + 3$ steps when the input has length n.

It follows that $L_B \in \mathcal{NP}^B$. Since B was an arbitrarily chosen language over Σ, this establishes the claim. $\square$

It is important that $L_B \in \mathcal{NP}^B$ for ***every*** language $B \subseteq \Sigma^\star$, because a ***diagonalization*** argument will be used to describe the language B whose existence is asserted in part (b) of the theorem — and this language will be constructed "on the fly."

It is not hard to modify the encoding of deterministic Turing machines to produce an encoding for ***deterministic oracle Turing machines*** — you simply need to include encodings of the query state, "Yes" state and "No state."[1] We may therefore assume that

$$M_0, M_1, M_2, \ldots$$

is an enumeration (or "listing") of ***all*** deterministic oracle Turing machines.

The set $\mathbb{N} \times \mathbb{N}$ is also enumerable — and this implies that it is possible to enumerate, or "list" ***all*** of the ordered pairs (M_i, j) for $i, j \in \mathbb{N}$.

[1]You can also ***require*** these states to be encoded as `qQ`, `qY`, and `qN`, respectively.

The goal of the construction to be described will be to ensure that, after (M_i, j) has been processed, make sure that ***either***

(a) M_i does not decide L_B,

or

(b) M_i uses more than n^j steps when executed on the input string $\mathtt{1}^n$, for some integer $n \geq 1$.

Assuming that this goal has been accomplished, suppose that $B \in \mathcal{P}^B$. Then there must exist positive integers c_1, c_0 and k, and a deterministic oracle Turing machine M_i, with an oracle for B, that decides L_B — using at most $c_1 n^k + c_0$ steps, on the input $\mathtt{1}^n$, for every integer $n \geq 0$. However, a ***contradiction*** is obtained: If we set $j = c_1 + k + c_0$ then

$$c_1 n^k + c_0 \leq n^{c_1+k+c_0} = n^j$$

for all $n \in \mathbb{N}$ such that $n \geq 1$, and neither property (a) nor (b) is satisfied for this choice of i and j. It will therefore follow that $B \notin \mathcal{P}^B$, so that $\mathcal{P}^B \neq \mathcal{NP}^B$.

Consider the construction shown in Figure 3 on page 7.

Throughout the construction the following properties will be satisfied.

(c) $B \subseteq D \subseteq \Sigma^\star$ where $\Sigma = \{\mathtt{0}, \mathtt{1}\}$.

(d) D is a ***finite*** subset of $\Sigma^\star$. We call the strings in D ***committed*** — because the question of whether they might eventually be included in B has already been decided. That is, no string $\omega \in \Sigma^\star$ is ever added to B if it already belongs to D. All other strings in $\Sigma^\star$ are ***uncommitted,*** and it is possible that they might eventually be included in B later on.

(e) If any deterministic oracle Turing machine M_i has been run on input $\mathtt{1}^m$ before this, during this construction, then D includes ***every*** string in $\Sigma^\star$ with length m — so it has already been "decided" whether $\mathtt{1}^m \in L_B$.

(f) On the other hand, if *no* deterministic oracle Turing machine M_i has been run on input $\mathtt{1}^m$ before this, during this construction, that B does not include ***any*** strings in $\Sigma^\star$ with length m.

(g) E is a finite subset of $\mathbb{N}$. For $i \in \mathbb{N}$, if $i \in E$ then M_i has been ***eliminated***: It has already been proved that M_i does not decide L_B, because property (a) was established when (M_i, j) was processed for some integer $j \geq 0$.

Note that properties (c)–(g) *do* initially hold.

Consider, now, the processing of $M_{i,j}$. If $i \in E$, as the figure should suggest, it is not necessary to do anything because machine M_i has been eliminated. If M_i's input alphabet has size greater than one then one can simply include i in E and do nothing more, because B cannot possibly the the language of M_i.

```
1.  $B$ := $\emptyset$
2.  $D$ := $\{\lambda\}$
3.  $E$ := $\emptyset$
4.  while processing $(M_i, j)$ {
5.    if $(i \notin E)$ {
6.      if ($M_i$'s tape alphabet has size greater than one) {
7.        $E := E \cup \{i\}$
        } else {
8.        Set $n$ to be a non-negative integer such that strictly more than $n^j$ strings
          in $\{0, 1\}^\star$ do not belong to $D$.
9.        Carry out the first $n^j$ steps of the execution of $M_i$ on input $1^j$.

          (a) If $M_i$ asks whether a string $\omega \in D$ belongs to $B$ during these steps
              then say "Yes" (moving to state $q_Y$) if $\omega$ has already been included
              in $B$, and say "No" (moving to state $q_N$) otherwise — without chang-
              ing either $B$ or $D$.

          (b) If $M_i$ asks whether a string $\omega \in \Sigma^\star$ belongs to $B$, where $\omega \notin D$,
              then include $\omega$ in $D$ (replacing this set with $D \cup \{\omega\}$), without chang-
              ing $B$, and say "No" (moving to state $q_N$).

10.       if ($M_j$ accepted $1^n$ after taking at most $n^j$ steps) {
11.         Include all uncommitted strings in $\Sigma^\star$ with length $n$ in $D$, but not in $B$
            — so that $1^n \notin L_B$ and $i$ has been added to $E$.
12.       } else if ($M_j$ rejected 1 after taking at most $n^j$ steps) {
13.         Choose a string $\zeta \in \Sigma^\star$ with length $n$ such that $\zeta \notin D$. Add $\zeta$ to
            both $B$ and $D$ — so that $1^n \in L_B$ and $i$ has been added to $E$. Then
            add all other uncommitted strings in $\Sigma^\star$ with length $n$ to $D$, but not
            to $B$.
          } else {
14.         Include all uncommitted strings in $\Sigma^\star$ with length $n$ in $D$ but not in $B$
            — so that $1 \notin L_B$.
          }
        }
      }
    }
```

Figure 3: Construction of the Desired Language B

Suppose, therefore, that $i \notin E$ and $M_i's$ tape alphabet has size one so that, in effect, M_i's tape alphabet is $\{1\}$. Then, since D is a ***finite*** subset of $\Sigma^\star$, there are 2^n strings in $\Sigma^\star$ with length n for every non-negative integer n, and $n^j \in o(2^n)$, there exists a non-negative integer such that strictly more than n^j strings in $\Sigma^\star$ still do not belong to D, as required at line 8.

Note that, during the simulation of the first n^j steps of M_i on input 1^n at line 9, all responses by the oracle are consistent with answers given in the past, since a queried string's membership in B is checked to determine the oracle's answer whenever the queried string belongs to D. Note, as well that, since queried strings that do not belong to D are added to this set, while saying "No" and without adding them to B, no strings are ever included in B during this part of the process.

Suppose, now, that properties (c)–(g) held before the execution of the step at line 9. Then properties (c) and (d) still hold after the execution of this step, because only finitely many strings in $\Sigma^\star$ were added during this execution (at most one for each step taken in the M_i's execution on its first n^j steps) and no string was added to B unless it was added to D at the same time. Property (e) holds — for it concerns executions on inputs $1^{\widehat{n}}$ that have taken place before this execution of the step at line 9, because n was chosen to be a non-negative integer such that D *does not* include all strings with length n— and no strings have been removed from D during the execution of this step. Similarly, property (f) must still hold because the execution of the step at line 9 did not add any new strings to B. The execution of the step at line 9 does not change the set E, so property (g) is satisfied after the execution of this step if it was executed before it.

Consider, next, the steps at lines 10–14, and suppose, again, that properties (c)–(g) held before the execution of these steps. Once again, since only finitely many strings have been added to D and none were added to B unless they were added to D at the same time, properties (c) and (d) hold, once again, after these steps have been executed. Since all uncommitted strings with length n are added to D when one of the steps at lines 11, 13 or 14 are executed, property (e) is satisfied at the end of the execution of these steps as well. Since the only string added to B, and its length is n (as M_i has been run on 1^n during the construction), property (f) is also satisfied after these steps if it was satisfied before them. Since at most one non-negative integer, i has been added to E during the execution of these steps, and it has been proved that M_i does not compute L_B at this point in the construction, property (g) is also satisfied at the end of the execution of these steps.

Thus properties (c)–(g) are all satisfied after each step in the execution of this construction (after the initialization of B, D and E) as claimed. Furthermore, property (a) is established for i if either of the steps at lines 11 or 13 was reached and executed when processing (M_i, j), while property (b) was established in the step at line 14 was reached and executed instead.

This establishes the following — as needed to establish part (b) of the theorem, as well.

Claim 4. *Let $B \subseteq \Sigma^\star$ be the set of strings that are eventually included in the set called "B" as the constructions shown in Figure 3 is applied. Then $\mathcal{P}^B \neq \mathcal{NP}^B$.*

Additional Results

Computations with a ***random oracle*** — which responds to every unique query with a (truly) random response — have also been considered. Furthermore, it was conjectured by Bennet and Gill [1] that two "acceptable" complexity classes are equal if and only if they are equal (with probability 1) under a random oracle — Bennet and Gill's paper can be consulted for a more precise definition of this. If this conjecture was true, then it would have followed from other results in the paper that $\mathcal{P} \neq \mathcal{NP}$.

However, as later established by Chang, Chor et al. [2], this conjecture is false. Consequently, Bennet and Gill's result is now considered to be, at best, "weak" evidence that $\mathcal{P} \neq \mathcal{NP}$, and this question remains open.

References

[1] C. H. Bennett and J. Gill. Relative to a random oracle a, $\mathcal{P}^A$!= $\mathcal{NP}^A$!= co-$\mathcal{NP}^A$ with probability 1. *SIAM Journal on Computing*, 10:96–113, 1981.

[2] Richard Chang, Benny Chor, Oded Goldreich, Juris Hartmanis, Johan Hastad, Desh Ranjan, and Pankaj Rohtagi. The random oracle hypothesis is false. *Journal of Computer and System Sciences*, 49:24–39, 1994.