Introduction to Complexity Theory*
Lecture 9: The Polynomial Hierarchy (PH)

Notes taken by Ronen Mizrahi

November 14, 1999

Summary: In this lecture we define a hierarchy of complexity classes starting from $\mathcal{NP}$ and yet contained in PSPACE. This is done in two ways, the first by generalizing the notion of Cook reductions, and the second by generalizing the definition of $\mathcal{NP}$. We show that the two are equivalent. We then try to make some observations regarding the hierarchy, our main concern will be to learn when does this hierarchy collapse, and how can we relate it to complexity classes that we know already such as BPP and P/Poly.

1 The Definition of the class PH

In the literature you may find three common ways to define this class, two of those ways will be
$\boxed{1}$ $\boxed{34}$ $\Rightarrow$ presented here. (The third, via "alternating" machines is omitted here.)

1.1 First definition for PH: via oracle machines

1.1.1 Intuition

Recall the defintion of a Cook reduction, the reduction is done using a polynomial time machine that has access to some oracle. Requiring that the oracle will belong to a given complexity class C, will raise the question:

> What is the complexity class of all those languages that are Cook reducable to some language from C?

For example:

> Let us set the complexity class of the oracle to be $\mathcal{NP}$, then for Karp reduction we know that every language L, that is Karp reducable to some language in $\mathcal{NP}$ (say SAT), will also be in $\mathcal{NP}$. However it is not clear what complexity class will a Cook reduction (to $\mathcal{NP}$) yield.

1.1.2 Perliminary definitions

Definition 1 (the language $L(M^A)$): *The language $L(M^A)$ is the set of inputs accepted by machine M given access to oracle A.*

---

* Lecture Notes for a course given by Oded Goldreich at the Weizmann Institute of Science, Israel.

Notations:

- M^A : The orcale machine M with access to oracle A.
- $M^A(x)$: The output of the orcale machine M^A on input x.

We note the following interesting cases for the above definition:

1. M is a deterministic polynomial time machine. Then M is a Cook reduction of $L(M^A)$ to A.
2. M is a probabilistic polynomial time machine. Then M is a randomized Cook reduction of $L(M^A)$ to A.
3. M is a non-deteministic polynomial time machine (note that the non determinism is related only to M, A is an oracle and as such it always gives the right answer). When we define the polynomial hierarchy we will use this case.

Observe that given one of the above cases, knowing the complexity class of the oracle, will define another complexity class which is the set of languages $L(M^A)$, where A is an oracle from the given comlexity class. The resulting complexity class may be one that is known to us (such as $\mathcal{P}$ or $\mathcal{NP}$), or a new class.

Definition 2 (the class M^C): *Let M be an oracle machine. Then M^C is the set of languages obtained from the machine M given access to an oracle from the class of languages C. That is,*

$$M^C \stackrel{\text{def}}{=} \{L(M^A) : A \in C\}$$

For example:

- $M^{\mathcal{NP}} = \{L(M^A) : A \in \mathcal{NP}\}$

  Note: we do not gain much by using $\mathcal{NP}$, rather than any $\mathcal{NP}$-complete language (such as SAT). That is, we know that any language, A, in $\mathcal{NP}$ is Karp reducable to SAT, by using this reduction we can alter M, and obtain a new machine $\tilde{M}$, such that $L(M^A) = L(\tilde{M}^{SAT})$.

In the following definition we abuse notation a little. We write ${C_1}^{C_2}$ but refer to machines natually associated with the class C_1, and to their natural extension to orale machines. We note that not every class has a natural enumeration of machines associated with it, let allow a natural extension of such machines to oracle machines. However, such associations and extensions do hold for the main classes we are interested in such as $\mathcal{P}$, $\mathcal{NP}$ and $\mathcal{BPP}$.

Definition 3 (the class ${C_1}^{C_2}$ – a fuzzy framework): *Assume that C_1 and C_2 are classes of languages, and also that for each language L in C_1, there exists a machine M_L, such that $L = L(M_L)$. Furthermore, consider the extension of M_L into an oracle machine M so that given access to the empty oracle M behaves as M_L (i.e., $L(M_L) = L(M^{\emptyset})$). Then ${C_1}^{C_2}$ is the set of languages obtained from such machines M_L, where $L \in C_1$, given access to an oracle for a language from the class of languages C_2. That is,*

$${C_1}^{C_2} = \{L(M^A) : L(M^{\emptyset}) \in C_1 \;\&\; A \in C_2\}$$

The above framework can be properly instantiated in some important cases. For example:

- $\mathcal{P}^C = \{L(M^A) : M \text{ is deterministic polynomial-time oracle machine } \& \ A \in C\}$
- $\mathcal{NP}^C = \{L(M^A) : \text{same as above but } M \text{ is non-deterministic}\}$
- $\mathcal{BPP}^C = \{L(M^A) : \text{same as above but } M \text{ is probabilistic}\}$

  Here we mean that with probability at least $2/3$, machine M on input x and oracle aceess to $A \in C$ correctly decides whether $x \in L(M^A)$.

Back to the motivating question: Observe that saying that L is Cook-reducible to SAT (i.e., $L \propto_C SAT$) is equivalent to writing $L \in \mathcal{P}^{\mathcal{NP}}$. We may now re-address the question regarding the power of Cook reductions. Observe that $\mathcal{NP} \cup \text{co}\mathcal{NP} \subseteq \mathcal{P}^{\mathcal{NP}}$, this is because:

- $\mathcal{NP} \subseteq \mathcal{P}^{\mathcal{NP}}$ holds, because for $L \in \mathcal{NP}$ we can take the oracle A to be an oracle for the language L, and the machine $M \in \mathcal{P}$ to be a trivial machine that takes its input asks the oracle about it, and outputs the oracle's answer.
- $\text{co}\mathcal{NP} \subseteq \mathcal{P}^{\mathcal{NP}}$ holds, because we can take the same oracle as above, and a different (yet still trivial) machine $M \in \mathcal{P}$ that asks the oracle about its input, and outputs the boolean complement of the oracle's answer.

We conclude that under the assumption that $\mathcal{NP} \neq \text{co}\mathcal{NP}$, Cook-reductions to $\mathcal{NP}$ give us more power than Karp-reductions to the same class.

> Oded's Note: *We saw such a result already, but it was quite artificial. I refer to that fact that $\mathcal{P}$ is Cook-reducible to the class of trivial langugaes (i.e., the class $\{\emptyset, \{0,1\}^*\}$), whereas non-trivial languages can not be Karp-reduced to trivial ones.*

1.1.3 Actual definition

Definition 4 (the class Σ_i): *Σ_i is a sequence of sets and will be defined inductively:*

- $\Sigma_1 \stackrel{\text{def}}{=} \mathcal{NP}$
- $\Sigma_{i+1} \stackrel{\text{def}}{=} \mathcal{NP}^{\Sigma_i}$

Notations:

- $\Pi_i \stackrel{\text{def}}{=} \text{co}\Sigma_i$
- $\Delta_{i+1} \stackrel{\text{def}}{=} \mathcal{P}^{\Sigma_i}$

Definition 5 (The hierarchy – PH): $\mathsf{PH} \stackrel{\text{def}}{=} \cup_{i=1}^{\infty} \Sigma_i$

The arbitrary choice to use the Σ_i's (rather than the Π_i's or Δ_i's) is justified by the following observations.

1.1.4 Almost syntaxtic observations

Proposition 1.1 $\Sigma_i \cup \Pi_i \subseteq \Delta_{i+1} \subseteq \Sigma_{i+1} \cap \Pi_{i+1}$.

Proof: We prove each of the two containments:

1. $\Sigma_i \cup \Pi_i \subseteq \Delta_{i+1} = \mathcal{P}^{\Sigma_i}$.
   The reason for that is the same as for $\mathcal{NP} \cup \mathrm{co}\mathcal{NP} \subseteq \mathcal{P}^{\mathcal{NP}} = \Delta_2$ (see above)
2. $\mathcal{P}^{\Sigma_i} \subseteq \Sigma_{i+1} \cap \Pi_{i+1}$.
   $\mathcal{P}^{\Sigma_i} \subseteq \mathcal{NP}^{\Sigma_i} = \Sigma_{i+1}$ is obvious. Since $\mathcal{P}^{\Sigma_i}$ is closed under complementation, $L \in \mathcal{P}^{\Sigma_i}$ implies that $\overline{L} \in \mathcal{P}^{\Sigma_i} \subseteq \Sigma_{i+1}$ and so $L \in \Pi_{i+1}$.

■

Proposition 1.2 *$\mathcal{P}^{\Sigma_i} = \mathcal{P}^{\Pi_i}$ and $\mathcal{NP}^{\Sigma_i} = \mathcal{NP}^{\Pi_i}$.*

Proof: Given a machine M and an oracle A, it is easy to modify M to $\tilde{M}$ such that: $L(M^{\overline{A}}) = L(\tilde{M}^A)$. The way we build $\tilde{M}$ is by taking M and flipping every answer obtained from the oracle. In particular, if M is deterministic (resp. non-deterministic) polynomail-time then so is $\tilde{M}$. Thus,
$\boxed{1}$ $\boxed{39}$ $\Rightarrow$ for such M and any class $\mathcal{C}$ the classes $M^{\mathrm{co}\mathcal{C}}$ and $\tilde{M}^{\mathcal{C}}$ are identical. ■

1.2 Second definition for PH: via quantifiers

1.2.1 Intuition

The approach taken here is to recall one of the definitions of $\mathcal{NP}$ and try to generalize it.

Definition 6 (polynomially-bounded relation): *a k-ary relation R is called* polynomially bounded *if there exists a polynomial $p(.)$ such that:*

$$\forall (x_1, \ldots, x_k)\,,\ [(x_1, \ldots, x_k) \in R \Longrightarrow (\forall i)\, |x_i| \leq p(|x_1|)]$$

Note: our definition requires that all the elements of the relation are not too long with regard to the first element, but the first element may be very long. We could even require a stronger condition: $\forall i \forall j |x_i| \leq p(|x_j|)$, this will promise that every element of the relation is not too long with regard to every one of the others. We do not make this requirement because the above definition will turn out to be satisfactory for our needs, this is because in our relations the first element is the input word, and we need the rest of the elements in the relation to be bounded in the length of the input. Also the complexity classes, that we will define using the notion of a polynomially bounded k-ary relation, will turn out the same for both the weak and the strong definition of the relation.

We now state again the definition of the complexity class $\mathcal{NP}$:

Definition 7 ($\mathcal{NP}$): *$L \in \mathcal{NP}$ if there exists a polynomially bounded and polynomial time recognizable binary relation R_L such that:*

$$x \in L \;\; \textit{iff} \;\; \exists y \text{ s.t. } (x, y) \in R_L$$

The way to generalize this definition will be to use a k-ary relation instead of just a binary one.

1.2.2 Actual definition

What we redefine is the sequence of sets Σ_i such that Σ_1 will remain $\mathcal{NP}$. The definition for PH remains the union of all the Σ_i's.

Definition 8 (Σ_i): *$L \in \Sigma_i$ if there exists a polynomially bounded and polynomial time recognizable (i+1)-ary relation R_L such that:*

$$x \in L \textit{ iff } \exists y_1 \forall y_2 \exists y_3 \ldots Q_i y_i, \textit{ s.t. } (x, y_1, \ldots, y_i) \in R_L$$

- $Q_i = \forall$ *if i is even*
- $Q_i = \exists$ *otherwise*

$\boxed{1}$ $\boxed{42}$ $\Rightarrow$

1.3 Equivalence of definitions

We have done something that might seem a mistaken; that is, we have given the same name for an object defined by two different definitions. However, we now intend to prove that the classes produced by the two definitions are equal. A more conventional way to present those two definitions is to state one of them as the definition for PH, and then prove an "if and only if" theorem that characterizes PH according to the other definition.

Theorem 9 : *The above two definitions of PH are equivalent. Furthermore, for every i, the class Σ_i as in Definition* **??** *is identical to the one in Definition* **??**.

Proof: We will show that for every i, the class Σ_i by the two definitions is equal. In order to distinguish between the clases produced by the two definitions we will introduce the following notation:

- Σ_i^1 is the set Σ_i produced by the first definition.
- Σ_i^2 is the set Σ_i produced by the second definition.
- Π_i^1 is the set Π_i produced by the first definition.
- Π_i^2 is the set Π_i produced by the second definition.

Part 1: We prove by induction on i that $\forall i$, $\Sigma_i^2 \subseteq \Sigma_i^1$:

- Base of induction: Σ_1 was defined to be $\mathcal{NP}$ in both cases so there is nothing to prove.
- We assume that the claim holds for i and prove for $i+1$: suppose $L \in \Sigma_{i+1}^2$ then by definition it follows that there exists a relation R_L such that:

  $$x \in L \text{ iff } \exists y_1 \forall y_2 \exists y_3 \ldots Q_i y_i Q_{i+1} y_{i+1}, \text{ s.t. } (x, y_1, \ldots, y_i, y_{i+1}) \in R_L$$

  In other words this means that:

  $$x \in L \text{ iff } \exists y_1, \text{ s.t. } (x, y_1) \in L_i$$

  where L_i is defined as follows:

  $$L_i \stackrel{\text{def}}{=} \{(x', y') : \forall y_2 \exists y_3 \ldots Q_i y_i Q_{i+1} y_{i+1}, \text{ s.t. } (x', y', \ldots, y_i, y_{i+1}) \in R_L\}$$

We claim that $L_i \in \Pi_i^2$, this is by complementing the definition of Σ_i^2. If we do this complementation for $L \in \Sigma_i^2$ we get:

$$x \in L \text{ iff } \exists y_1 \forall y_2 \ldots Q_i y_i, \text{ s.t. } (x, y_1, \ldots, y_i) \in R_L$$

$$x \in \overline{L} \text{ iff } \forall y_1 \exists y_2 \ldots \overline{Q}_i y_i, \text{ s.t. } (x, y_1, \ldots, y_i) \notin R_L$$

This is almost what we had in the definition of L_i except for the "$\notin R_L$" as opposed to "$\in R_L$". Remember that deciding membership in R_L is polynomial time recognizable, and therefore its complement is also so. Now that we have that $L_i \in \Pi_i^2$, we can use the inductive hypothesis $\Pi_i^2 \subseteq \Pi_i^1$. So far we have managed to show that:

$$x \in L \text{ iff } \exists y_1, \text{ s.t. } (x, y_1) \in L_i$$

Where L_i belongs to Π_i^1. We now claim that $L \in \mathcal{NP}^{\Pi_i^1}$, this is true because we can write a non-deterministic, polynomial-time machine, that decides membership in L, by guessing y_1, and using an oracle for L_i. Therefore we can further conclude that:

$$L \in \mathcal{NP}^{\Pi_i^1} \equiv \mathcal{NP}^{\Sigma_i^1} \equiv \Sigma_{i+1}^1.$$

Part 2: We prove by induction on i that $\forall i$, $\Sigma_i^1 \subseteq \Sigma_i^2$:

- Base of induction: as before.
- Induction step: suppose $L \in \Sigma_{i+1}^1$ then there exists a non-deterministic polynomial time machine M such that $L \in L(M^{\Sigma_i^1})$ which means that:

  $$\exists L' \in \Sigma_i^1, \text{ s.t. }, L = L(M^{L'})$$

  From the definition of $M^{L'}$ it follows that:

  $$x \in L \text{ iff } \exists y, q_1, a_1, \ldots, q_t, a_t \text{ s.t. :}$$

  1. Machine M, with non-determinstic choices y, interacts with its oracle in the following way:
     - 1^{st} query $= q_1$ and 1^{st} answer $= a_1$

       .

       .

       .
     - t^{th} query $= q_t$ and t^{th} answer $= a_t$
  2. for every $1 \leq j \leq t$:
     - $(a_j = 1) \Longrightarrow q_j \in L'$
     - $(a_j = 0) \Longrightarrow q_j \notin L'$

  where y is a description of the non-determinstic choices of M.

  Let us view the above according to the second definition, that is, according to the definition with quantifiers, then the first item is a polynomial time predicate and therefore this potentially puts L in $\mathcal{NP}$. The second item involves L'. Recall that $L' \in \Sigma_i^1$ and that by the inductive hypothesis $\Sigma_i^1 \subseteq \Sigma_i^2$, and therefore we can view membership in L' according to the second definition, and embed this result within what we have above. This will yield that for every $1 \leq j \leq t$:

- $(a_j = 1) \Longrightarrow \exists y_1^{(j,1)} \forall y_2^{(j,1)} \ldots Q_i y_i^{(j,1)}$, s.t. $(q_j, y_1^{(j,1)}, \ldots, y_i^{(j,1)}) \in R_{L'}$
- $(a_j = 0) \Longrightarrow \forall y_1^{(j,2)} \exists y_2^{(j,2)} \ldots Q_i y_i^{(j,2)}$, s.t. $(q_j, y_1^{(j,2)}, \ldots, y_i^{(j,2)}) \in \overline{R}_{L'}$

Let us define:

- w_1 is the concatenation of:
  $y, q_1, a_1, \ldots, q_t, a_t$, and $y_1^{(j,1)}$ for all j s.t. $(a_j = 1)$.
- w_2 is the concatenation of:
  $y_1^{(j,2)}$ for all j s.t. $(a_j = 0)$, and $y_2^{(j,1)}$ for all j s.t. $(a_j = 1)$.

  .

  .

  .
- w_i is the concatenation of:
  $y_{i-1}^{(j,2)}$ for all j s.t. $(a_j = 0)$, and $y_i^{(j,1)}$ for all j s.t. $(a_j = 1)$.
- w_{i+1} is the concatenation of:
  $y_i^{(j,2)}$ for all j s.t. $(a_j = 0)$.

R_L will be the $(i+1)$-ary relation defined in the following way: $(w_1, \ldots, w_{i+1}) \in R_L$ iff for every $1 \leq j \leq t$:

- $(a_j = 1) \Longrightarrow (q_j, y_1^{(j,1)}, \ldots, y_i^{(j,1)}) \in R_{L'}$
- $(a_j = 0) \Longrightarrow (q_j, y_1^{(j,2)}, \ldots, y_i^{(j,2)}) \in \overline{R}_{L'}$

where the w_i's are parsed analogously to the above.

Since $R_{L'}$ and $\overline{R}_{L'}$ where polynomially bounded, and polynomial time recognizable, so is R_L.

Altogether we have:

$$x \in L \text{ iff } \exists w_1, \forall w_2, \ldots Q_{i+1} w_{i+1}, \text{ s.t. } (w_1, \ldots, w_{i+1}) \in R_L$$

It now follows from the definition of Σ_{i+1}^2 that $L \in \Sigma_{i+1}^2$ as needed.

$\boxed{1}$ $\boxed{49}$ ⇒ ■

2 Easy Computational Observations

Proposition 2.1 $PH \subseteq PSPACE$

Proof: We will show that $\Sigma_i \subseteq PSPACE$ for all i. Let $L \in \Sigma_i$, then we know by the definition with quantifiers that:

$$x \in L \text{ iff } \exists y_1 \forall y_2 \exists y_3 \ldots Q_i y_i, \text{ s.t. } (x, y_1, \ldots, y_i) \in R_L$$

Given x we can use i variables to try all the possibilities for $y_1, \ldots, y_i$ and make sure that they meet the above requirement. Since the relation R_L is polynomially bounded, we have a polynomial bound on the length of each of the y_i's that we are checking. Thus we have constructed a deterministic machine that decides L.

This machine uses i variables, the length of each of them is polynomially bounded in the length of the input. Since i is a constant, the overall space used by this machine is polynomial. ■

Proposition 2.2 *$\mathcal{NP} = \text{co}\mathcal{NP}$ implies $PH \subseteq \mathcal{NP}$ (which implies $PH = \mathcal{NP}$).*

Intuitively the extra power that non-deterministic Cook reductions have over non-deterministic Karp reductions, comes from giving us the ability to complement the oracle's answers for free. What we claim here is that if this power is meaningless then the whole hierarchy collapses.

Proof: We will show by induction on i that $\forall i\,,\ \Sigma_i = \mathcal{NP}$:

1. $i = 1$: by definition $\Sigma_1 = \mathcal{NP}$.
2. Induction step: by the inductive hypothesis it follows that $\Sigma_i = \mathcal{NP}$ so what remains to be shown is that $\mathcal{NP}^{\mathcal{NP}} = \mathcal{NP}$. Containment in one direction is obvious so we focus on proving that $\mathcal{NP}^{\mathcal{NP}} \subseteq \mathcal{NP}$. Let $L \in \mathcal{NP}^{\mathcal{NP}}$ then there exist a non-deterministic, polynomial-time machine M, and an oracle $A \in \mathcal{NP}$, such that $L = L(M^A)$. Since $\mathcal{NP} = \text{co}\mathcal{NP}$ it follows that $\overline{A} \in \mathcal{NP}$ too. Therefore, there exist relations R_A and $R_{\overline{A}}$ ($\mathcal{NP}$ relations for A and $\overline{A}$ respectively) such that:

   - $q \in A$ iff $\exists w$, s.t. $(q, w) \in R_A$.
   - $q \in \overline{A}$ iff $\exists w$, s.t. $(q, w) \in R_{\overline{A}}$

   Using these relations, and the definition of $\mathcal{NP}^{\mathcal{NP}}$ we get:
   $x \in L$ iff $\exists y, q_1, a_1, \ldots, q_t, a_t$, such that, for all $1 \leq j \leq t$:

   - $a_j = 1 \Longleftrightarrow q_j \in A \Longleftrightarrow \exists w_j, (q_j, w_j) \in R_A$
   - $a_j = 0 \Longleftrightarrow q_j \in \overline{A} \Longleftrightarrow \exists w_j, (q_j, w_j) \in R_{\overline{A}}$.

   Define:

   - w is the concatenation of: $y, q_1, a_1, \ldots, q_t, a_t, w_1, \ldots, w_t$
   - R_L is a binary relation such that:
     $(x, w) \in R_L$ iff for all $1 \leq j \leq t$:
     - $a_j = 1 \Longrightarrow (q_j, w_j) \in R_A$
     - $a_j = 0 \Longrightarrow (q_j, w_j) \in R_{\overline{A}}$.

     Since M is a polynomial-time machine, t is polynomial in the length of x. Combining this fact with the fact that both R_A and $R_{\overline{A}}$ are polynomial-time recognizable, and polynomially bounded, we conclude that so is R_L.

   All together we get that there exists an $\mathcal{NP}$ relation R_L such that :

   $$x \in L \text{ iff } \exists w, \text{ s.t. } (x, w) \in R_L$$

   Thus, $L \in \mathcal{NP}$.

■

Generalizing Proposition **??**, we have

Proposition 2.3 *For every $k \geq 1$, if $\Pi_k = \Sigma_k$ then $PH = \Sigma_k$.*

$\boxed{1}$ $\boxed{55}$ $\Rightarrow$ A proof is presented in the appendix to this lecture.

3 BPP is contained in PH

Not knowing whether $\mathcal{BPP}$ is contained in $\mathcal{NP}$, it is of some confort to know that it is contained in the Polynomial-Hierarchy (which extends $\mathcal{NP}$).

Theorem 10 (Sipser and Lautemann): $BPP \subseteq \Sigma_2$.

Proof: Let $L \in BPP$ then there exists a probabilistic polynomial time machine $A(x,r)$ where x is the input and r is the random guess. By the definition of BPP, with some amplification we get, for some polynomial $p(.)$:

$$\forall x \in \{0,1\}^n, \text{ s.t. } \mathrm{Pr}_{r \in_R \{0,1\}^{p(n)}}[A(x,r) \neq \chi(x)] < \frac{1}{3p(n)}$$

where $\chi(x) = 1$ if $x \in L$ and $\chi(x) = 0$ otherwise.

> Oded's Note: *A word about the above is in place. Note that we do* **not** *assert that the error decreases as a fast fixed function of n, where the function is fixed before we determine the randomness complexity of the new algorithm. We saw result of that kind in Lecture 7; but here we claim something different. That is, that the error probability may depend on the randomness complexity of the new algorithm. Still, the dependency required here is easy to achieve. Specifically, suppose that the original algorithm uses $m = \mathrm{poly}(n)$ coins. Then by running it t times and ruling by majority we decrease the error probability to $\exp(-\Omega(t))$. The randomness complexity of the new algorithm is tm. So we need to set t such that $\exp(-\Omega(t)) < 1/3mt$, which can be satisfied with $t = O(\log m) = O(\log n)$.*

The key observation is captured by the following claim

Claim 3.1 *Denote $m = p(n)$ then, for every $x \in L \cap \{0,1\}^n$, there exist $s_1, \ldots, s_m \in \{0,1\}^m$ such that*

$$\forall r \in \{0,1\}^m, \bigvee_{i=1}^{m} A(x, r \oplus s_i) = 1 \tag{1}$$

Actually, the same sequence of s_i's may be used for all $x \in L \cap \{0,1\}^n$ (provided that $m \geq n$ which holds without loss of generality). However, we don't need this extra property.

Proof: We will show existence of such s_i's by the Probabilistic Method: That is, instead of showing that an object with some property exists we will show that a random object has the property with positive probability. Actually, we will upper bound the probability that a random object does not have the desired property. In our case we look for existence of s_i's satisfying Eq. (**??**), and so we will upper bound the probability, denoted P, that randomly chosen s_i's do not satisfy Eq. (**??**):

$$\begin{aligned}
P &\stackrel{\text{def}}{=} \mathrm{Pr}_{s_1,\ldots,s_m \in_R \{0,1\}^m}[\neg \forall r \in \{0,1\}^m, \bigvee_{i=1}^{m} (A(x, r \oplus s_i) = 1)] \\
&= \mathrm{Pr}_{s_1,\ldots,s_m \in_R \{0,1\}^m}[\exists r \in \{0,1\}^m, \bigwedge_{i=1}^{m} (A(x, r \oplus s_i) = 0)] \\
&\leq \sum_{r \in \{0,1\}^m} \mathrm{Pr}_{s_1,\ldots,s_m \in_R \{0,1\}^m}[\bigwedge_{i=1}^{m} (A(x, r \oplus s_i) = 0)]
\end{aligned}$$

where the inequality is due to the union bound. Using the fact that the events of choosing s_i's uniformly are independent, we get that the probability of all the events happening at once equals to the multiplication of the probabilities. Therefore:

$$P \leq \sum_{r \in \{0,1\}^m} \prod_{i=1}^{m} \mathrm{Pr}_{s_i \in_R \{0,1\}^m}[A(x, r \oplus s_i) = 0]$$

Since in the above probability r is fixed, and the s_i's are uniformly distributed then (by a property of the $\oplus$ operator), the $s_i \oplus r$'s are also uniformly distributed. Recall that we consider an arbitrary fixed $x \in L \cap \{0,1\}^n$. Thus,

$$\begin{aligned} P &\leq 2^m \cdot \mathrm{Pr}_{s \in_R \{0,1\}^m}[A(x, s) = 0]^m \\ &\leq 2^m \cdot \left(\frac{1}{3m}\right)^m \ll 1 \end{aligned}$$

The claim holds. ■

Claim 3.2 *For any $x \in \{0,1\}^n \setminus L$ and for all $s_1, \ldots, s_m \in \{0,1\}^m$, there exists $r \in \{0,1\}^m$ so that $\bigvee_{i=1}^m A(x, r \oplus s_i) = 0$.*

Proof: We will actually show that for all $s_1, \ldots, s_m$ there are many such r's. Let $s_1, \ldots, s_m \in \{0,1\}^m$ be arbitrary.

$$\mathrm{Pr}_{r \in \{0,1\}^m}[\bigvee_{i=1}^{m} A(x, r \oplus s_i) = 0] = 1 - \mathrm{Pr}_{r \in \{0,1\}^m}[\bigvee_{i=1}^{m} A(x, r \oplus s_i) = 1]$$

However, since $x \notin L$ and $\mathrm{Pr}_{r \in \{0,1\}^m}[A(x, r) = 1] < 1/3m$, we get

$$\begin{aligned} \mathrm{Pr}_{r \in \{0,1\}^m}[\bigvee_{i=1}^{m} A(x, r \oplus s_i) = 1] &\leq \sum_{i=1}^{m} \mathrm{Pr}_{r \in \{0,1\}^m}[A(x, r \oplus s_i) = 1] \\ &\leq m \cdot \frac{1}{3m} = \frac{1}{3} \end{aligned}$$

and so,

$$\mathrm{Pr}_{r \in \{0,1\}^m}[\bigvee_{i=1}^{m} A(x, r \oplus s_i) = 0] \geq \frac{2}{3}$$

Therefore there exist (many) such r's and the claim holds. ■

Combining the results of the two claims together we get:

$$x \in L \text{ iff } \exists s_1, \ldots s_m \in \{0,1\}^m, \forall r \bigvee_{i=1}^{m} A(x, r \oplus s_i) = 1$$

This assertion corresponds to the definition of Σ_2, and therefore $L \in \Sigma_2$ as needed. ■

Comment: The reason we used the $\oplus$ operator is because it has the property that given an arbitrary fixed r, if s is uniformly distributed then $r \oplus s$ is also uniformly distributed. Same for fixed s and random r. Any other efficient binary operation with this property may be used as well.

$\boxed{1}$ $\boxed{62}$ $\Rightarrow$

4 If NP has small circuits then PH collpases

The following result shows that an unlikely event regarding non-uniform complexity (i.e., the class $\mathcal{P}$/poly) implies an unlikely event regarding uniform complexity (i.e., PH).

Theorem 11 (Karp & Lipton): *If $\mathcal{NP} \subseteq \mathcal{P}/\text{poly}$ then $\Pi_2 = \Sigma_2$, and so $PH = \Sigma_2$.*

Proof: We will only prove the first implication in the theorem. The second follows by Proposition **??**. Showing that Σ_2 is closed under complementation, gives us that $\Pi_2 = \Sigma_2$. So what we will actually prove is that $\Pi_2 \subseteq \Sigma_2$.

Let L be an arbitrary language in Π_2, then there exists a trinary polynomially bounded, and polynomial time recognizable relation R_L such that:

$$x \in L \text{ iff } \forall y \exists z \text{ s.t. }, (x, y, z) \in R_L$$

Let us define:

$$L' \stackrel{\text{def}}{=} \{(x', y') : \exists z, \text{ s.t. } (x', y', z) \in R_L\}$$

Then we get that:

- $x \in L$ iff $\forall y, (x, y) \in L'$
- $L' \in \mathcal{NP}$

Consider a Karp reduction of L' to 3SAT, call it f:

$$x \in L \text{ iff } \forall y, f(x, y) \in 3SAT$$

Let us now use the assumption that $\mathcal{NP} \subseteq P/Poly$ for 3SAT, then it follows that 3SAT has small circuits $\{C_m\}_m$, where m is the length of the input. We claim that also 3SAT has small circuits $\{C'_n\}_n$ where n is the number of variables in the formula. This claim holds since the length of a 3SAT formula is of $O(n^3)$ and therefore $\{C'_n\}$ can use the larger sets of circuits $C_1, \ldots, C_{O(n^3)}$. Let us embed the circuits in our statement regarding membership in L, this will yield:

$x \in L$ iff $\exists (C'_1, \ldots, C'_n)$ $(n \stackrel{\text{def}}{=} max_y\{\#var(f(x, y))\})$ s.t.:

- $C'_1, \ldots, C'_n$ correctly computes 3SAT, for formulas with a corresponding number of variables.
- $\forall y, \ C'_{\#var(f(x,y))}(f(x, y)) = 1$

The second item above gives us that $L \in \Sigma_2$, since the quantifiers are as needed. However it is not clear that the first item is also bahaving as needed. We will restate the first item as follows:

$$\forall \phi_1, \ldots, \phi_n, [\bigwedge_{i=2}^{n} C'_i(\phi_i) = (C'_{i-1}(\phi'_i) \vee C'_{i-1}(\phi''_i)) \bigwedge C'_1 \text{ } operates \text{ } correctly] \qquad (2)$$

Where:

$\phi_i(x_1, \ldots, x_i)$ is any formula over i variables.

$\phi'_i(x_1, \ldots, x_{i-1}) \stackrel{\text{def}}{=} \phi_i(x_1, \ldots, x_{i-1}, 0)$

$\phi''_i(x_1, \ldots, x_{i-1}) \stackrel{\text{def}}{=} \phi_i(x_1, \ldots, x_{i-1}, 1)$

A stupid technicality: Note that assigning a value to one of the variables, gives us a formula that is not in CNF as required by 3SAT (as its clauses may contain constants). However this can easily be achieved, by iterating the following process, where in each iteration one of the following rules is applied:

- $x \vee 0$ should be changed to x.
- $x \vee 1$ should be changed to 1.
- $x \wedge 0$ should be changed to 0.
- $x \wedge 1$ can be changed to x.
- $\neg 1$ can be changed to 0.
- $\neg 0$ can be changed to 1.

If we end-up with a formula in which some variables do not appear, we can augment it by adding clauses of the form $x \wedge \neg x$.

> Oded's Note: *An alternative resolution of the above technicality is to extend the definition of CNF so to allow constnats to appear (in clauses).*

Getting back to the main thing: We have given a recursive definition for a correct computation of the circuits (on 3SAT). The base of the recursion is checking that a single variable fromula is handled correctly by C'_1, which is very simple (just check if the single variable formula is satisfiable or not, and compare it to the output of the circuit). In order to validate the $(i+1)^{\text{th}}$ circuit, we wish to use the i^{th} circuit, which has already been validated. Doing so requires us to reduce the number of variables in the formula by one. This is done by assigning to one of the variables both possible values (0 or 1), and obtaining two formulas upon which the i^{th} circuit can be applied. The full formula is satisfiable iff at least one of the reduced formulas is satisfiable. Therefore we combine the results of applying the i^{th} circuit on the reduced formulas, with the $\vee$ operation. It now remains to compare it to the value computed by the $(i+1)^{\text{th}}$ circuit on the full formula. This is done for all formula over $i+1$ variables (by the quantification $\forall \phi_{i+1}$).

So all together we get that:

$$x \in L \text{ iff } \exists (C'_1, \ldots, C'_n), \text{ s.t. } \forall y, (\phi_1, \ldots, \phi_n), \; (x, (C'_1, \ldots, C'_n), (y, \phi_1, \ldots, \phi_n)) \in R_L$$

where R_L is a polynomially-bounded 3-ary relation defined using the Karp reduction f, Eq. (**??**) and the simplifying process above. Specifically, the algorithm recognizing R_L computes the formula $f(x,y)$, determines the formulas ϕ'_i and ϕ''_i (for each i), and evaluates circuits (the description of which is given) on inputs which are also given. Cleraly, this algorithm can be implemented in polynomial-time, and so it follows that $L \in \Sigma_2$ as needed. ■

5 Appendix: Proof of Proposition ??

Recall that our aim is to prove the claim:

For every $k \geq 1$, if $\Pi_k = \Sigma_k$ then $PH = \Sigma_k$.

Proof: For an arbitrary fixed k, we will show by induction on i that $\forall i \geq k, \Sigma_i = \Sigma_k$:

1. Base of induction: when $i = k$, there is nothing to show.
2. Induction step: by the inductive hypothesis it follows that $\Sigma_i = \Sigma_k$, so what remains to be shown is that $\mathcal{NP}^{\Sigma_k} = \Sigma_k$. Containment in one direction is obvious so we focus on proving that $\mathcal{NP}^{\Sigma_k} \subseteq \Sigma_k$.

   Let $L \in \mathcal{NP}^{\Sigma_k}$, then there exist a non-deterministic, polynomial-time machine M, and an oracle $A \in \Sigma_k$, such that $L = L(M^A)$. Since $\Pi_k = \Sigma_k$ it follows that $\overline{A} \in \Sigma_k$ too. Therefore, there exist relations R_A and $R_{\overline{A}}$ ($k+1$-ary relations, polynomially bounded, and polynomial time recognizable, for A and $\overline{A}$ respectively) such that :

   - $q \in A$ iff $\exists w_1, \forall w_2, \ldots, Q_k w_k$ s.t. $(q, w_1, \ldots w_k) \in R_A$.
   - $q \in \overline{A}$ iff $\exists w_1, \forall w_2, \ldots, Q_k w_k$ s.t. $(q, w_1, \ldots w_k) \in R_{\overline{A}}$.

   Using those relations, and the definition of $\mathcal{NP}^{\Sigma_k}$ we get:
   $x \in L$ iff $\exists y, q_1, a_1, \ldots, q_t, a_t$ s.t. for all $1 \leq j \leq t$:

   - $a_j = 1 \iff q_j \in A \iff \exists w_1^{(j,1)}, \forall w_2^{(j,1)}, \ldots, Q_k w_k^{(j,1)}$ s.t. $(q_j, w_1^{(j,1)}, \ldots w_k^{(j,1)}) \in R_A$.
   - $a_j = 0 \iff q_j \in \overline{A} \iff \exists w_1^{(j,0)}, \forall w_2^{(j,0)}, \ldots, Q_k w_k^{(j,0)}$ s.t. $(q_j, w_1^{(j,0)}, \ldots w_k^{(j,0)}) \in R_{\overline{A}}$.

   Define:

   - w_1 is the concatenation of: $y, q_1, a_1, \ldots, q_t, a_t, w_1^{(1,0)}, \ldots, w_1^{(t,0)}, w_1^{(1,1)}, \ldots, w_1^{(t,1)}$.

     .

     .
   - w_k is the concatenation of: $w_k^{(1,0)}, \ldots, w_k^{(t,0)}, w_k^{(1,1)}, \ldots, w_k^{(t,1)}$
   - R_L is a $k+1$-ary relation such that:
     $(x, w_1, \ldots, w_k) \in R_L$ iff for all $1 \leq j \leq t$:
     - $a_j = 1 \implies (q_j, w_1^{(j,1)}, \ldots w_k^{(j,1)}) \in R_A$.
     - $a_j = 0 \implies (q_j, w_1^{(j,0)}, \ldots w_k^{(j,0)}) \in R_{\overline{A}}$.

     Since M is a polynomial machine, then t is polynomial in the length of x. R_A and $R_{\overline{A}}$ are polynomial time recognizable, and polynomially bounded relations.
     Therefore R_L is also so.

   All together we get that there exists a polynomially bounded, and polynomial time recognizable relation R_L such that :

   $$x \in L \text{ iff } \exists w_1, \forall w_2, \ldots, Q_k w_k \text{ s.t. } (x, w_1, \ldots, w_k) \in R_L$$

   By the definition of Σ_k, $L \in \Sigma_k$.

■