Computational Complexity Theory

Lecture 15: Polynomial Hierarchy (contd.)

Department of Computer Science,
Indian Institute of Science

Recap: Class Σ_i

- Definition. A language L is in Σ_i if there's a polynomial function $q(.)$ and a poly-time TM M (the "verifier") s.t.

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \ \forall u_2 \in \{0,1\}^{q(|x|)} \ Q_i u_i \in \{0,1\}^{q(|x|)}$$

$$\text{s.t. } M(x,u_1,\ldots,u_i) = 1,$$

where Q_i is $\exists$ or $\forall$ if i is odd or even, respectively.

- Obs. $\Sigma_i \subseteq \Sigma_{i+1}$ for every i.

Recap: Polynomial Hierarchy

- Definition. A language L is in Σ_i if there's a polynomial function $q(.)$ and a poly-time TM M (the "verifier") s.t.

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)}\ \forall u_2 \in \{0,1\}^{q(|x|)}\ \cdots\ Q_i u_i \in \{0,1\}^{q(|x|)}$$

$$\text{s.t. } M(x, u_1, \ldots, u_i) = 1,$$

where Q_i is $\exists$ or $\forall$ if i is odd or even, respectively.

- Definition. *(Meyer & Stockmeyer 1972)*

$$PH = \bigcup_{i \in \mathbf{N}} \Sigma_i .$$

$\vdots$

Σ_3

$|$

Σ_2

$|$

$\Sigma_1 = NP$

$\Sigma_0 = P$

Recap: Class $\prod_i$

- Definition. $\prod_i$ = co-$\sum_i$ = $\{ L : \bar{L} \in \sum_i \}$.

- Obs. A language L is in $\prod_i$ if there's a polynomial function $q(.)$ and a poly-time TM M (the "verifier") s.t.

$$x \in L \iff \forall u_1 \in \{0,1\}^{q(|x|)}\ \exists u_2 \in \{0,1\}^{q(|x|)}\ \ Q_i u_i \in \{0,1\}^{q(|x|)}$$

$$\text{s.t. } M(x,u_1,\ldots,u_i) = 1,$$

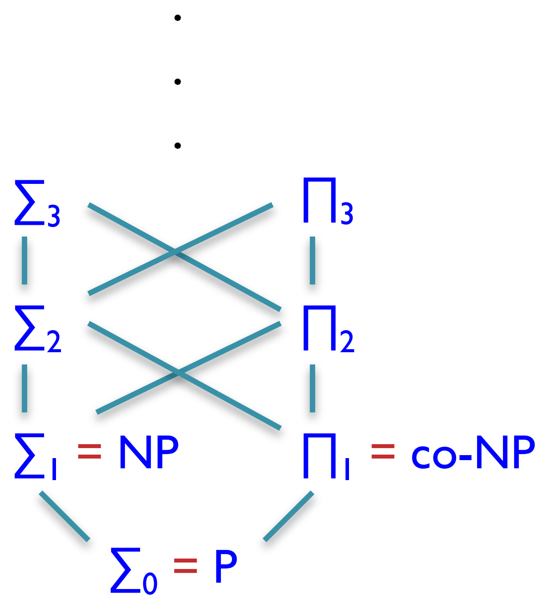
where Q_i is $\forall$ or $\exists$ if i is odd or even, respectively.

- Obs. $\sum_i \subseteq \prod_{i+1} \subseteq \sum_{i+2}$.

Recap: Polynomial Hierarchy

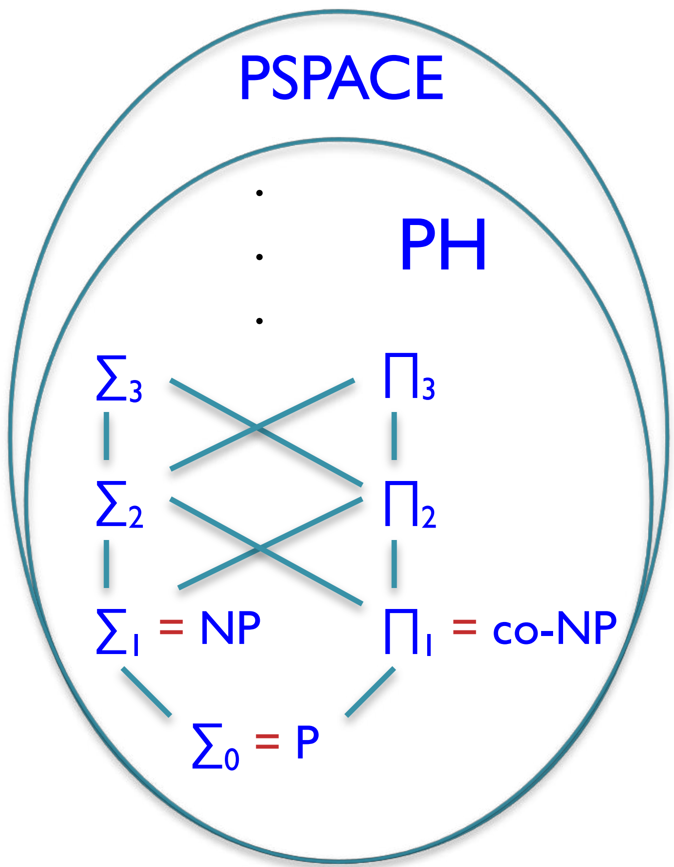
- Obs. $\mathrm{PH} = \bigcup_{i \in \mathbf{N}} \Sigma_i = \bigcup_{i \in \mathbf{N}} \Pi_i$.

PH =



Recap: Polynomial Hierarchy

- Claim. PH $\subseteq$ PSPACE .
- Proof. Similar to the proof of TQBF $\in$ PSPACE.



Recap: Does PH collapse?

- **General belief.** Just as many of us believe $P \neq NP$ (i.e. $\Sigma_0 \neq \Sigma_1$) and $NP \neq co\text{-}NP$ (i.e. $\Sigma_1 \neq \Pi_1$), we also believe that for every i, $\Sigma_i \neq \Sigma_{i+1}$ and $\Sigma_i \neq \Pi_i$.

- **Definition.** We say PH <u>collapses to the i-th level</u> if $\Sigma_i = \Sigma_{i+1}$. (justified in the next theorem)

- **Conjecture.** There is no i such that PH collapses to the i-th level.

This is stronger than the $P \neq NP$ conjecture.

Recap: PH collapse theorems

- Theorem. If $\Sigma_i = \Sigma_{i+1}$ then $PH = \Sigma_i$.
- Theorem. If $\Sigma_i = \Pi_i$ then $PH = \Sigma_i$.

Recap: Complete problems in PH ?

- Recall, to define completeness of a complexity class, we need an appropriate notion of a *reduction*.
- What kind of reductions will be suitable is guided by *a complexity question*, like a comparison between the complexity class under consideration & another class.
- Is P = PH ? …use poly-time Karp reduction!

- Definition. A language L' is *PH-hard* if for every L in PH, $L \leq_p L'$. Further, if L' is in PH then L' is *PH-complete*.

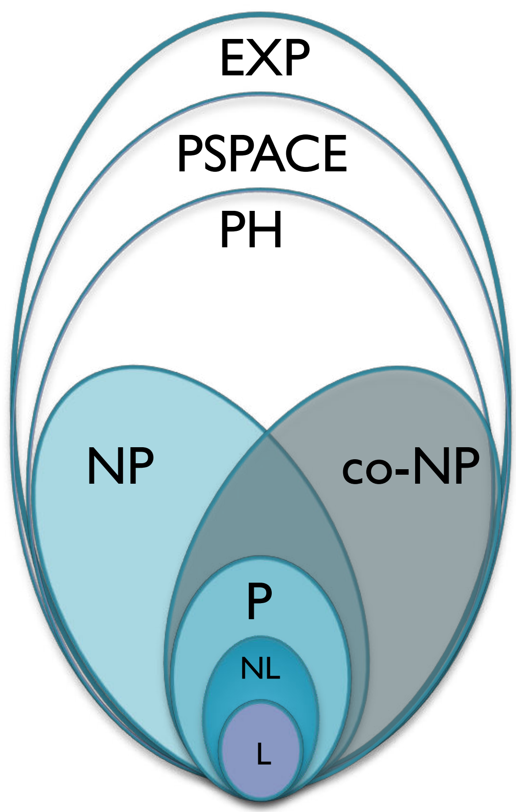
Recap: Complete problems in PH ?

- Fact. If L is poly-time reducible to a language in Σ_i then L is in Σ_i .   (we've seen a similar fact for NP)

- Observation. If PH has a complete problem then PH collapses.
- Proof. If L is *PH-complete* then L is in Σ_i for some i. Now use the above fact to infer that PH = Σ_i .

Recap: Complete problems in PH ?

- Fact. If L is poly-time reducible to a language in Σ_i then L is in Σ_i. (we've seen a similar fact for NP)
- Corollary. PH $\subsetneq$ PSPACE unless PH collapses.



Recap: Complete problems in $\sum_i$

- Recall, to define completeness of a complexity class, we need an appropriate notion of a *reduction*.
- What kind of reductions will be suitable is guided by *a complexity question*, like a comparison between the complexity class under consideration & another class.
- Is $P = \sum_i$? …use poly-time Karp reduction!

- Definition. A language L' is $\sum_i$ *-hard* if for every L in $\sum_i$, $L \leq_p L'$. Further, if L' is in $\sum_i$ then L' is $\sum_i$ *-complete*.

Recap: Complete problems in Σ_i

- Definition. The language Σ_i-SAT contains all *true* QBF with i alternating quantifiers starting with $\exists$.

- Theorem. Σ_i-SAT is Σ_i *-complete*. (Σ_1-SAT is just SAT)

- Observation. Owing to the proof of the Cook-Levin theorem, we can assume that the formula in a Σ_i-SAT instance is a CNF (if i is odd) or a DNF (if i is even).

Recap: Other complete problems in Σ_2

- Ref. *"Completeness in the Polynomial-Time Hierarchy: A Compendium"* by *Schaefer and Umans (2008)*.

- Theorem. Eq-DNF and Succinct-SetCover are Σ_2 *-complete*.

An alternate characterization of PH

Oracle definition of Σ_i

- Definition. A language L is in $NP^{\Sigma_i\text{-SAT}}$ if there is a poly-time NTM with oracle access to Σ_i-SAT that decides L.

- Theorem. $\Sigma_{i+1} = NP^{\Sigma_i\text{-SAT}}$.

Oracle definition of Σ_i

- **Definition.** A language L is in $NP^{\Sigma_i\text{-SAT}}$ if there is a poly-time NTM with oracle access to Σ_i-SAT that decides L.

- **Theorem.** $\Sigma_{i+1} = NP^{\Sigma_i\text{-SAT}}$.

- Observe that Σ_1-SAT = SAT. We'll prove the special case $\Sigma_2 = NP^{SAT}$. The proof of the theorem is similar.

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in Σ_2. There's a polynomial function q(.) and a poly-time TM M s.t.

$$x \in L \iff \exists u \in \{0,1\}^{q(|x|)}\ \forall v \in \{0,1\}^{q(|x|)} \text{ s.t. } M(x,u,v) = 1.$$

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = \mathrm{NP}^{\mathrm{SAT}}$.
- Proof. Let L be a language in Σ_2. There's a polynomial function $q(.)$ and a poly-time TM M s.t.

$$x \in L \iff \exists u \in \{0,1\}^{q(|x|)} \; \forall v \in \{0,1\}^{q(|x|)} \text{ s.t. } \underbrace{\phi(x,u,v)}_{\substack{\text{Boolean circuit} \\ (\textit{by Cook-Levin})}} = 1 .$$

- In fact, owing to the proof of the Cook-Levin theorem, we can assume that ϕ is a DNF.

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in Σ_2. There's a polynomial function $q(.)$ and a poly-time TM M s.t.

$$x \in L \iff \exists u \in \{0,1\}^{q(|x|)}\ \forall v \in \{0,1\}^{q(|x|)} \text{ s.t. } \neg\phi(x,u,v) = 0.$$

- Think of a NTM N that has the knowledge of M. On input x, it guesses $u \in \{0,1\}^{q(|x|)}$ non-deterministically and computes the circuit $\phi(x,u,v)$. Then, it queries the SAT oracle with $\neg\phi(x,u,v)$.

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in Σ_2. There's a polynomial function $q(.)$ and a poly-time TM M s.t.

  $x \in L \Leftrightarrow \exists u \in \{0,1\}^{q(|x|)}\ \forall v \in \{0,1\}^{q(|x|)}$ s.t. $\neg\phi(x,u,v) = 0$.

- Think of a NTM N that has the knowledge of M. On input x, it guesses $u \in \{0,1\}^{q(|x|)}$ non-deterministically and computes the circuit $\phi(x,u,v)$. Then, it queries the SAT oracle with $\neg\phi(x,u,v)$.
- Note that $\neg\phi(x,u,v)$ is a CNF.

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most <u>**one**</u> query to the SAT oracle on every computation path on input x.

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most **one** query to the SAT oracle on every computation path on input x.
- We need to construct a Σ_2-statement that captures N's computation on input x.

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most one query to the SAT oracle on every computation path on input x.
- Think of a TM M that takes input x and $w \in \{0,1\}^{q(|x|)}$, $a_1 \in \{0,1\}$ and u_1, $v_1 \in \{0,1\}^{q(|x|)}$, where <u>$q(|x|)$ is the runtime of N on input x</u>, and does the following:

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most one query to the SAT oracle on every computation path on input x.
- Think of a TM M that takes input x and $w \in \{0,1\}^{q(|x|)}$, $a_1 \in \{0,1\}$ and $u_1, v_1 \in \{0,1\}^{q(|x|)}$, where $q(|x|)$ is the runtime of N on input x, and does the following:
- M simulates N on input x with w as the non-deterministic choices.

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most one query to the SAT oracle on every computation path on input x.
- Think of a TM M that takes input x and $w \in \{0,1\}^{q(|x|)}$, $a_1 \in \{0,1\}$ and $u_1, v_1 \in \{0,1\}^{q(|x|)}$, where $q(|x|)$ is the runtime of N on input x, and does the following:
- M simulates N on input x with w as the computation path. Suppose ϕ is the query asked by N on the path of computation defined by w.

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most one query to the SAT oracle on every computation path on input x.
- Think of a TM M that takes input x and $w \in \{0,1\}^{q(|x|)}$, $a_1 \in \{0,1\}$ and u_1, $v_1 \in \{0,1\}^{q(|x|)}$, where $q(|x|)$ is the runtime of N on input x, and does the following:

➢ If $a_1 = 1$ and $\phi(u_1) = 1$, M continues the simulation; else it stops and outputs 0. (In this case, M ignores v_1.)

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most one query to the SAT oracle on every computation path on input x.
- Think of a TM M that takes input x and $w \in \{0,1\}^{q(|x|)}$, $a_1 \in \{0,1\}$ and u_1, $v_1 \in \{0,1\}^{q(|x|)}$, where $q(|x|)$ is the runtime of N on input x, and does the following:
  - If $a_1 = 0$ and $\phi(v_1) = 0$, M continues the simulation; else it stops and outputs 0. (In this case, M ignores u_1.)

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most one query to the SAT oracle on every computation path on input x.
- Think of a TM M that takes input x and $w \in \{0,1\}^{q(|x|)}$, $a_1 \in \{0,1\}$ and u_1, $v_1 \in \{0,1\}^{q(|x|)}$, where $q(|x|)$ is the runtime of N on input x, and does the following:
- At the end of the simulation, M outputs whatever N outputs. Note: M is a poly-time TM.

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most one query to the SAT oracle on every computation path on input x.
- Observation. For any $w \in \{0,1\}^{q(|x|)}$ and $a_1 \in \{0,1\}$,

➢ N on computation path w gets answer a_1 from the SAT oracle and accepts x $\Leftrightarrow$

$$\exists u_1 \in \{0,1\}^{q(|x|)} \ \forall v_1 \in \{0,1\}^{q(|x|)} \text{ s.t. } M(x,w,a_1,u_1,v_1) = 1.$$

(…will prove the observation shortly. Let's finish the proof.)

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most one query to the SAT oracle on every computation path on input x.
- $x \in L \iff \exists w \in \{0,1\}^{q(|x|)}, a_1 \in \{0,1\}$ s.t
  - N on computation path w gets answer a_1 from the SAT oracle and accepts $x \iff \exists w \in \{0,1\}^{q(|x|)}, a_1 \in \{0,1\}$

$\exists u_1 \in \{0,1\}^{q(|x|)}\ \forall v_1 \in \{0,1\}^{q(|x|)}$ s.t. $M(x,w,a_1,u_1,v_1) = 1$.

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most one query to the SAT oracle on every computation path on input x.
- $x \in L \Leftrightarrow \exists w \in \{0,1\}^{q(|x|)}, a_1 \in \{0,1\}$ s.t

➢ N on computation path w gets answer a_1 from the SAT oracle and accepts $x \Leftrightarrow \exists w \in \{0,1\}^{q(|x|)}, a_1 \in \{0,1\}$

$$\exists u_1 \in \{0,1\}^{q(|x|)}\ \forall v_1 \in \{0,1\}^{q(|x|)} \text{ s.t. } M(x, w, \underbrace{a_1, u_1}_{\text{Call it } u}, v_1) = 1.$$

Oracle definition of $\sum_i$

- Theorem. $\sum_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most one query to the SAT oracle on every computation path on input x.
- $x \in L \Leftrightarrow \exists w \in \{0,1\}^{q(|x|)}, a_1 \in \{0,1\}$ s.t
  - N on computation path w gets answer a_1 from the SAT oracle and accepts x $\Leftrightarrow$

  $\exists u \in \{0,1\}^{2q(|x|)+1}\ \forall v_1 \in \{0,1\}^{q(|x|)}$ s.t. $M(x,u,v_1) = 1$.
- Therefore, L is in $\sum_2$.

Proof of the observation

- Observation. For any $w \in \{0,1\}^{q(|x|)}$ and $a_1 \in \{0,1\}$,

➢ N on computation path w gets answer a_1 from the SAT oracle and accepts x ⟺

$\exists u_1 \in \{0,1\}^{q(|x|)}\ \forall v_1 \in \{0,1\}^{q(|x|)}$ s.t. $M(x,w,a_1,u_1,v_1) = 1$.

- Proof.(⟹) M simulates N on computation path w. Let ϕ be the query asked by N to SAT.
- If $a_1 = 1$, $\exists u_1 \in \{0,1\}^{q(|x|)}\ \phi(u_1) = 1$ and N accepts x.

Proof of the observation

- Observation. For any $w \in \{0,1\}^{q(|x|)}$ and $a_1 \in \{0,1\}$,
- ➢ N on computation path w gets answer a_1 from the SAT oracle and accepts x $\Leftrightarrow$

  $\exists u_1 \in \{0,1\}^{q(|x|)}$ $\forall v_1 \in \{0,1\}^{q(|x|)}$ s.t. $M(x,w,a_1,u_1,v_1) = 1$.

- Proof.($\Rightarrow$) M simulates N on computation path w. Let ϕ be the query asked by N to SAT.
- If $a_1 = 1$, $\exists u_1 \in \{0,1\}^{q(|x|)}$ s.t. $M(x,w,a_1,u_1,v_1) = 1$.

In this case, M ignores v_1.

Proof of the observation

- Observation. For any $w \in \{0,1\}^{q(|x|)}$ and $a_1 \in \{0,1\}$,
- ➢ N on computation path w gets answer a_1 from the SAT oracle and accepts x $\Leftrightarrow$

  $\exists u_1 \in \{0,1\}^{q(|x|)}\ \forall v_1 \in \{0,1\}^{q(|x|)}$ s.t. $M(x,w,a_1,u_1,v_1) = 1$.
- Proof.($\Rightarrow$) M simulates N on computation path w. Let ϕ be the query asked by N to SAT.
- If $a_1 = 0$, $\forall v_1 \in \{0,1\}^{q(|x|)}$ $\phi(v_1) = 0$ and N accepts x.

Proof of the observation

- Observation. For any $w \in \{0,1\}^{q(|x|)}$ and $a_1 \in \{0,1\}$,
- ➢ N on computation path w gets answer a_1 from the SAT oracle and accepts x ⟺

$\exists u_1 \in \{0,1\}^{q(|x|)}$ $\forall v_1 \in \{0,1\}^{q(|x|)}$ s.t. $M(x,w,a_1,u_1,v_1) = 1$.

- Proof.(⟹) M simulates N on computation path w. Let ϕ be the query asked by N to SAT.
- If $a_1 = 0$, $\forall v_1 \in \{0,1\}^{q(|x|)}$ s.t. $M(x,w,a_1,u_1,v_1) = 1$.

In this case, M ignores u_1.

Proof of the observation

- Observation. For any $w \in \{0,1\}^{q(|x|)}$ and $a_1 \in \{0,1\}$,
- ➢ N on computation path w gets answer a_1 from the SAT oracle and accepts x $\Leftrightarrow$

  $\exists u_1 \in \{0,1\}^{q(|x|)}\ \forall v_1 \in \{0,1\}^{q(|x|)}$ s.t. $M(x,w,a_1,u_1,v_1) = 1$.
- Proof.($\Rightarrow$) M simulates N on computation path w. Let ϕ be the query asked by N to SAT.
- Irrespective of the value of a_1,

  $\exists u_1 \in \{0,1\}^{q(|x|)}\ \forall v_1 \in \{0,1\}^{q(|x|)}$ s.t. $M(x,w,a_1,u_1,v_1) = 1$.

Proof of the observation

- **Observation.** For any $w \in \{0,1\}^{q(|x|)}$ and $a_1 \in \{0,1\}$,

➢ N on computation path w gets answer a_1 from the SAT oracle and accepts x $\Leftrightarrow$

$\exists u_1 \in \{0,1\}^{q(|x|)}\ \forall v_1 \in \{0,1\}^{q(|x|)}$ s.t. $M(x,w,a_1,u_1,v_1) = 1$.

- Proof.($\Leftarrow$) Need to show that N on computation path w <u>gets answer a_1</u> from the SAT oracle. (*Homework*)

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- General case: N asks at most $q(|x|)$ queries to SAT oracle on every computation path on input x.
- *Homework*: Prove the general case. Define the poly-time machine M appropriately.

Oracles versus efficient algorithms

- Definition. A language L is in P^{SAT} if there is a poly-time TM with oracle access to SAT that decides L.
- $\Delta_2 := P^{SAT} \subseteq \Sigma_2 \cap \Pi_2$.
- A SAT oracle gives us the ability to solve SAT efficiently "much like" a poly-time algorithm for SAT.

Oracles versus efficient algorithms

- Definition. A language L is in P^{SAT} if there is a poly-time TM with oracle access to SAT that decides L.
- $\Delta_2 := P^{SAT} \subseteq \Sigma_2 \cap \Pi_2$.
- A SAT oracle gives us the ability to solve SAT efficiently "much like" a poly-time algorithm for SAT.
- Yet, in the first case we believe $P^{SAT} \neq NP^{SAT}$, (otherwise, PH collapses to Σ_2)

Oracles versus efficient algorithms

- Definition. A language L is in P^{SAT} if there is a poly-time TM with oracle access to SAT that decides L.
- $\Delta_2 := P^{SAT} \subseteq \Sigma_2 \cap \Pi_2$.
- A SAT oracle gives us the ability to solve SAT efficiently "much like" a poly-time algorithm for SAT.

- Yet, in the first case we believe $P^{SAT} \neq NP^{SAT}$, whereas in the second case PH collapses to P, i.e., $P^{SAT} = NP^{SAT}$.

Oracles versus efficient algorithms

- Definition. A language L is in P^{SAT} if there is a poly-time TM with oracle access to SAT that decides L.
- $\Delta_2 := P^{SAT} \subseteq \Sigma_2 \cap \Pi_2$.
- A SAT oracle gives us the ability to solve SAT efficiently "much like" a poly-time algorithm for SAT.

- Yet, in the first case we believe $P^{SAT} \neq NP^{SAT}$, whereas in the second case PH collapses to P, i.e., $P^{SAT} = NP^{SAT}$.
- Why? Think to understand the difference between oracles and poly-time algorithms for SAT (*Homework*).