Advanced Computational Complexity

George Voutsadakis[1]

[1]Mathematics and Computer Science
Lake Superior State University

LSSU Math 600

1. The Polynomial Hierarchy and Alternations
   - The Class Σ_2^P
   - The Polynomial Hierarchy
   - Alternating Turing Machines
   - Time vs. Alternations: Time-Space Tradeoffs for SAT
   - Defining the Hierarchy via Oracle Machines

Introducing the Polynomial Time Hierarchy

- We introduce a new complexity class, called the polynomial hierarchy, denoted PH, which is a generalization of P, NP and coNP.
- It consists of an infinite number of subclasses, called levels, which are conjectured to be distinct, a stronger form of the conjecture $\mathrm{P} \neq \mathrm{NP}$.
- We provide three equivalent definitions of the polynomial hierarchy:
  1. As the set of languages defined via polynomial-time predicates, combined with a constant number of alternating for all ($\forall$) and exists ($\exists$) quantifiers, generalizing the definitions of NP and coNP.
  2. Via the use of alternating Turing machines, that are a generalization of nondeterministic Turing machines.
  3. Via the use of oracle Turing machines.
- A fourth characterization, using uniform families of circuits, will be postponed for later.
- These characterizations are used to show that SAT cannot be solved using simultaneously linear time and logarithmic space.

Subsection 1

The Class Σ_2^P

Independent Set and Exact Independent Set

- We focus on some computational problems that seem to not be captured by NP-completeness.
- Recall the following NP problem IndSet, for which we do have a short certificate of membership,

$$\textsc{IndSet} = \{\langle G, k\rangle : \text{graph } G \text{ has an independent set of size} \geq k\}.$$

- Consider a slight modification consisting of determining the largest independent set in a graph (phrased as a decision problem),

$$\textsc{ExactIndSet} = \{\langle G, k\rangle : \text{the largest independent set in } G \text{ has size exactly } k\}.$$

- Now there seems to be no short certificate for membership.
- $\langle G, k\rangle \in \textsc{ExactIndSet}$ iff *there exists* an independent set of size k in G and *every other* independent set has size at most k.

Smallest Equivalent DNF Formula

- Consider, also, the problem of determining the smallest Boolean formulas equivalent to a given formula,

$$\textsc{MinEqDNF} = \{\langle \varphi, k\rangle : \exists \text{DNF formula } \psi \text{ of size} \leq k \text{ that is equivalent to the DNF formula } \varphi\},$$

  where:
  - A **DNF formula** is a Boolean formula that is an OR of ANDs;
  - Two formulas are **equivalent** if they agree on all possible assignments.
- The complement of this language is

$$\overline{\textsc{MinEqDNF}} = \{\langle \varphi, k\rangle : \forall \text{DNF formulas } \psi \text{ of size} \leq k\ \exists \text{assignment } u \text{ s.t. } \varphi(u) \neq \psi(u)\}.$$

- Again, there is no obvious notion of a certificate of membership for $\textsc{MinEqDNF}$.
- To capture these languages, we seem to need to allow not only a single "exists" or "for all" quantifier, but a combination of both.

The Class Σ_2^p

Definition (The Class Σ_2^p)

The class Σ_2^p is the set of all languages L for which, there exists a polynomial-time TM M and a polynomial q, such that

$$x \in L \quad \Leftrightarrow \quad \exists u \in \{0,1\}^{q(|x|)} \forall v \in \{0,1\}^{q(|x|)} M(x,u,v) = 1,$$

for every $x \in \{0,1\}^*$.

- Note that Σ_2^p contains both the classes NP and coNP.

Examples (Cont'd)

- The language $\textsc{ExactIndSet}$ is in Σ_2^p.

  A pair $\langle G, k\rangle$ is in $\textsc{ExactIndSet}$ iff:
  - There exists a size-k subset S of G's vertices, such that:
  - For every size-$(k+1)$ subset S':

  We have:
  - S is an independent set in G;
  - S' is not an independent set in G.
- The language $\textsc{MinEqDNF}$ is also in Σ_2^p.

  A pair $\langle \varphi, k\rangle$ is in $\textsc{MinEqDNF}$ iff:
  - There exists a DNF formula ψ of size $\leq k$, such that:
  - For every assignment u:

  We have $\varphi(u) = \psi(u)$.
- The language $\textsc{MinEqDNF}$ is known to be Σ_2^p-complete.

Subsection 2

The Polynomial Hierarchy

The Polynomial Hierarchy

- The definition of the polynomial hierarchy generalizes those of NP, coNP and Σ_2^p.
- It consists of every language that can be defined via a combination of a polynomial time computable predicate and a constant number of $\forall/\exists$ quantifiers.

Definition (Polynomial Hierarchy)

For $i \geq 1$, a language L is in Σ_i^p if there exists a polynomial-time TM M and a polynomial q, such that

$$x \in L \quad \text{iff} \quad \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \cdots Q_i u_i \in \{0,1\}^{q(|x|)} \; M(x, u_1, \ldots, u_i) = 1,$$

where Q_i denotes $\exists$, if i is odd, and $\forall$, if i is even.

The Polynomial Hierarchy (Cont'd)

Definition (Polynomial Hierarchy Cont'd)

The **polynomial hierarchy** is the set

$$\mathrm{PH} = \bigcup_i \Sigma_i^p.$$

- Note that $\Sigma_1^p = \mathrm{NP}$.
- For every i, define

$$\Pi_i^p = \mathrm{co}\Sigma_i^p = \{\overline{L} : L \in \Sigma_i^p\}.$$

- Thus, $\Pi_1^p = \mathrm{coNP}$.
- For every i, $\Sigma_i^p \subseteq \Pi_{i+1}^p \subseteq \Sigma_{i+2}^p$.
- Therefore,

$$\mathrm{PH} = \bigcup_{i>0} \Pi_i^p.$$

Collapsing of the Polynomial Hierarchy

- We believe that $\mathsf{P} \neq \mathsf{NP}$ and $\mathsf{NP} \neq \mathsf{coNP}$.
- An appealing generalization of these conjectures is that, for every i,

$$\Sigma_i^p \subsetneq \Sigma_{i+1}^p.$$

- This conjecture is used often in complexity theory and is, sometimes, stated as "the polynomial hierarchy does not collapse".
- The polynomial hierarchy is said to **collapse** if there is some i, such that

$$\Sigma_i^p = \Sigma_{i+1}^p.$$

- As we will see, this would imply $\Sigma_i^p = \bigcup_{j \geq 1} \Sigma_j^p = \mathsf{PH}$.
- In this case, we say that the polynomial hierarchy **collapses to the** i**-th level**.
- The smaller i is, the weaker, and, hence, more believable, it is to conjecture that PH does not collapse to the i-th level.

Properties of the Polynomial Hierarchy

Theorem

1. For every $i \geq 1$, if $\Sigma_i^p = \Pi_i^p$, then $\mathsf{PH} = \Sigma_i^p$, i.e., the hierarchy collapses to the i-th level.
2. If $\mathsf{P} = \mathsf{NP}$, then $\mathsf{PH} = \mathsf{P}$, i.e., the hierarchy collapses to P.

- We prove the second part.
  The first part follows by a similar reasoning.
  Suppose, first, that $\mathsf{P} = \mathsf{NP}$.
  We prove, by induction on i, that $\Sigma_i^p, \Pi_i^p \subseteq \mathsf{P}$.
  For $i = 1$, we have $\Sigma_1^p = \mathsf{NP}$ and $\Pi_1^p = \mathsf{coNP}$.
  So, by assumption, $\Sigma_1^p, \Pi_1^p \subseteq \mathsf{P}$.
  Assume the inclusions are true for $i - 1$.
  We prove that $\Sigma_i^p \subseteq \mathsf{P}$.
  Since Π_i^p consists of complements of languages in Σ_i^p and P is closed under complementation, it would follow that $\Pi_i^p \subseteq \mathsf{P}$.

Proof of the Induction Step

- Let $L \in \Sigma_i^p$.

  By definition, there is a polynomial-time Turing machine M and a polynomial q, such that

$$x \in L \quad \text{iff} \quad \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \cdots Q_i u_i \in \{0,1\}^{q(|x|)} \; M(x, u_1, \ldots, u_i) = 1,$$

  where Q_i is $\exists/\forall$ according to the parity of i.

  Define the language L' by stipulating that

$$\langle x, u_1 \rangle \in L' \quad \text{iff} \quad \forall u_2 \in \{0,1\}^{q(|x|)} \cdots Q_i u_i \in \{0,1\}^{q(|x|)} \; M(x, u_1, u_2, \ldots, u_i) = 1.$$

  Clearly, $L' \in \Pi_{i-1}^p$.

Proof of the Induction Step (Cont'd)

- We have $L' \in \Pi_{i-1}^p$.

  So, by our assumption, L' is in P.

  This implies that there is a polynomial-time TM M' computing L'.

  Plugging M' in the defining condition for L, we get

  $$x \in L \quad \text{iff} \quad \exists u_1 \in \{0,1\}^{q(|x|)} M'(x, u_1) = 1.$$

  But this means $L \in \text{NP}$.

  Therefore, under our assumption, $\text{P} = \text{NP}$, we get $L \in \text{P}$.

Complete problems for Levels of PH

- We defined the notion of a language B reducing to a language C via a polynomial-time Karp reduction, denoted $B \leq_p C$, by the existence of a polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$, such that, for every x,

$$x \in B \quad \text{iff} \quad f(x) \in C.$$

- We say that a language L is Σ_i^p-**complete** if:
  - $L \in \Sigma_i^p$;
  - For every $L' \in \Sigma_i^p$, $L' \leq_p L$.
- We define Π_i^p-**completeness** and PH-**completeness** in the same way.
- We set out to show the following.
  - The polynomial hierarchy is believed not to have a complete problem.
  - For every $i \in \mathbb{N}$, both Σ_i^p and Π_i^p have complete problems.

PH is Believed to Lack Complete Problems

Claim

If there exists a language L that is PH-complete, then there exists an i, such that $\mathrm{PH} = \Sigma_i^p$ (and, hence, the hierarchy collapses to its i-th level).

- We only provide a sketch of the proof.

  Suppose that there exists a language L that is PH-complete.

  By definition $L \in \mathrm{PH} = \bigcup_i \Sigma_i^p$.

  Thus, there exists i, such that $L \in \Sigma_i^p$.

  Since L is PH-complete, we can reduce every language of PH to L.

  But every language that is polynomial-time reducible to a language in Σ_i^p is itself in Σ_i^p.

  Hence, $\mathrm{PH} \in \Sigma_i^p$.

PH and PSPACE

- Just like NP and coNP, PH is also contained in PSPACE,

$$\mathrm{PH} \subseteq \mathrm{PSPACE}.$$

- Thus, unless the polynomial hierarchy collapses, PH $\neq$ PSPACE.

  We use contraposition.

  Assume that PH $=$ PSPACE.

  Then the PSPACE-complete problem TQBF is PH-complete.

  By the claim, the polynomial hierarchy collapses.

Complete Problems for Different Levels

- For every $i \geq 1$, we consider the class Σ_i^p.
- We also consider the following problem involving quantified Boolean expressions of the following type, with a limited number of alternations,

$$\Sigma_i \text{SAT} = \exists u_1 \forall u_2 \exists \cdots Q_i u_i (\varphi(u_1, u_2, \ldots, u_i) = 1),$$

  where:
  - φ is a Boolean formula not necessarily in CNF form (though the form does not make any difference);
  - Each u_i is a vector of Boolean variables;
  - Q_i is $\forall$ or $\exists$, depending on the parity of i.
- It turns out that, for all i, $\Sigma_i \text{SAT}$ is Σ_i^p-complete.
- For every i, $\Sigma_i \text{SAT}$ is a special case of the TQBF problem.
- One can similarly define a problem $\Pi_i \text{SAT}$, which is Π_i^p-complete.

Succinct Set Cover

- Consider the problem SUCCINCTSETCOVER
- The input consists of:
  - A collection
$$S = \{\varphi_1, \varphi_2, \ldots, \varphi_m\}$$
    of 3-DNF formulas on n variables;
  - An integer k.
- We must determine whether there exists a subset $S' \subseteq \{1, 2, \ldots, m\}$ of size at most k for which
$$\bigvee_{i \in S'} \varphi_i$$
  is a tautology.
- By its definition it is clear that SUCCINCTSETCOVER is in Σ_2^p.
- It has been shown that SUCCINCTSETCOVER is Σ_2^p-complete.

Subsection 3

Alternating Turing Machines

From NDTMs to Alternating Turing Machines

- **Alternating Turing machines** (**ATM**s) are generalizations of nondeterministic Turing machines.
- Even though NDTMs are not a realistic computational model, they help us understand the processes of guessing and verifying answers.
- ATMs play a similar role for certain languages for which there is no obvious short certificate for membership.
- The absence of such a certificate implies that such languages cannot be characterized using nondeterminism alone.

Features of Alternating Turing Machines

- In an alternating Turing machine:
  - Two transition functions are available to choose from at each step;
  - Every internal state, except q_{accept} and q_{halt}, is labeled with either $\exists$ or $\forall$.
- An ATM's computation can evolve at every step in two ways.
- A non-deterministic TM accepts its input if there exists some sequence of choices that leads it to the state q_{accept}.
- By analogy, in an ATM, the existential quantifier of an NDTM over each choice is replaced with the quantifier corresponding to the label at each state.

Alternating Acceptance

Definition (Alternating Acceptance)

We define an alternating Turing Machine M **accepting an input** x.
Let $G_{M,x}$ denote the directed acyclic configuration graph of M on input x.
In $G_{M,x}$, there is an edge from a configuration C to configuration C' iff C' can be obtained from C by one step of M's transition function.
We label some of the vertices in this graph by "ACCEPT" by repeatedly applying the following rules until they cannot be applied anymore:

- Configuration C_{accept}, with the machine in q_{accept}, is labeled "ACCEPT".

Alternating Acceptance (Cont'd)

Definition (Alternating Acceptance Cont'd)

- If a configuration C is in a state labeled $\exists$ and there is an edge from C to a configuration C' labeled "ACCEPT", then we label C "ACCEPT".
- If a configuration C is in a state labeled $\forall$ and both configurations C', C'' reachable from it in one step are labeled "ACCEPT", then we label C "ACCEPT".

We say that M **accepts** x if, at the end of this process, the starting configuration C_{start} is labeled "ACCEPT".

Alternating time

Definition (Alternating Time)

- For $T : \mathbb{N} \to \mathbb{N}$, we say that an alternating TM M **runs in $T(n)$-time** if, for every input $x \in \{0,1\}^*$ and for every possible sequence of transition function choices,

$$M \text{ halts in at most } T(|x|) \text{ steps.}$$

- We say that a language L is in $\text{ATIME}(T(n))$ if there is a constant c and a $c \cdot T(n)$-time ATM M, such that, for every $x \in \{0,1\}^*$,

$$M \text{ accepts } x \quad \text{iff} \quad x \in L.$$

Σ_iTIME and Π_iTIME

Definition (Σ_iTIME and Π_iTIME)

For $i \in \mathbb{N}$, we define $\Sigma_i\mathrm{TIME}(T(n))$ (resp. $\Pi_i\mathrm{TIME}(T(n))$) to be the set of languages accepted by a $T(n)$-time ATM M, such that:

- M's initial state is labeled "$\exists$" (resp. "$\forall$");
- On every input and on every path from the starting configuration in the configuration graph, M can alternate at most $i-1$ times from states with one label to states with the other label.

- One can show that, for every $i \in \mathbb{N}$,

$$\Sigma_i^p = \bigcup_c \Sigma_i\mathrm{TIME}(n^c) \quad \text{and} \quad \Pi_i^p = \bigcup_c \Pi_i\mathrm{TIME}(n^c).$$

The Class AP

- In defining $\Sigma_i\mathsf{TIME}(T(n))$ and $\Pi_i\mathsf{TIME}(T(n))$, we restricted attention to ATMs whose number of alternations is some fixed constant i independent of the input size.
- We now go back to considering polynomial-time alternating Turing machines with no a priori bound on the number of quantifiers.
- We define

$$\mathsf{AP} = \bigcup_c \mathsf{ATIME}(n^c).$$

Characterization of AP

Theorem

AP = PSPACE.

- We provide a sketch of the proof.

  TQBF is trivially in AP.

  We "guess" values for each:

  - Existentially quantified variable using an $\exists$ state;
  - Universally quantified variable using a $\forall$ state.

  Then do a deterministic polynomial-time computation at the end.

  Moreover, every PSPACE language reduces to TQBF.

  Thus, PSPACE $\subseteq$ AP.

  To show that AP $\subseteq$ PSPACE, we can use a recursive procedure similar to the one used to show that $\mathrm{TQBF} \in$ PSPACE.

Alternating Space

- It is also possible to consider alternating Turing machines that run in polynomial space.
- The class of languages accepted by such machines is called APSPACE.
- It turns out that
$$\mathrm{APSPACE} = \mathrm{EXP}.$$
- Similarly, the set of languages accepted by alternating logspace machines is equal to P.

Subsection 4

Time vs. Alternations: Time-Space Tradeoffs for Sat

Time/Space Tradeoff for SAT

- It is widely believed that, for its solution, SAT requires both:
  - Exponential (or at least superpolynomial) time;
  - Linear (or at least super-logarithmic) space.
- However, we currently have no way to prove these conjectures.
- It is in fact possible, as far as we know, that SAT may have both a linear time algorithm and a logarithmic space one.
- But we can rule out an algorithm that runs simultaneously in linear time and logarithmic space.

Time/Space Tradeoff for SAT

Theorem (Time/Space Tradeoff for SAT)

For every two functions $S, T : \mathbb{N} \to \mathbb{N}$, define

$$\mathsf{TISP}(T(n), S(n))$$

to be the set of languages decided by a TM M that, on every input x:

- Takes at most $O(T(|x|))$ steps;
- Uses at most $O(S(|x|))$ cells of its read-write tapes.

Then,

$$\textsc{Sat} \notin \mathsf{TISP}(n^{1.1}, n^{0.1}).$$

- $\mathsf{TISP}(T(n), S(n))$ is often defined with respect to TMs with RAM memory.
- The Tradeoff Theorem carries over to that model.

Plan of the Proof

- We show that $\mathsf{NTIME}(n) \not\subseteq \mathsf{TISP}(n^{1.2}, n^{0.2})$.

  A careful analysis of the proof of the Cook-Levin Theorem yields a reduction from the task of

  deciding membership in an $\mathsf{NTIME}(T(n))$-language

  to the task of

  deciding whether a $O(T(n)\log T(n))$-sized formula is satisfiable.
- Moreover, every output bit of this reduction can be computed in polylogarithmic time and space.
- This, combined with the result above, yields the proof of the tradeoff theorem.

  In fact, suppose $\text{SAT} \in \mathsf{TISP}(n^{1.1}, n^{0.1})$.

  Then $\mathsf{NTIME}(n) \subseteq \mathsf{TISP}(n^{1.1}\text{polylog}(n), n^{0.1}\text{polylog}(n))$.
- We start by showing how to replace time with alternations.

Time/Space and Alternating Time

Claim

$\mathrm{TISP}(n^{12}, n^2) \subseteq \Sigma_2\mathrm{TIME}(n^8)$.

- The proof is similar to the proofs of Savitch's Theorem and the PSPACE-completeness of TQBF.

  Suppose L is decided by a machine M using n^{12} time and n^2 space.

  For every $x \in \{0,1\}^*$, consider the configuration graph $G_{M,x}$ of M on input x.

  Each configuration in this graph can be described by a string of length $O\left(n^2\right)$.

  Moreover, x is in L if and only if there is a path of length n^{12} in this graph from the starting configuration C_{start} to an accepting configuration.

Time/Space and Alternating Time

- There is such a path if and only if there exist n^6 configurations

$$C_1, \ldots, C_{n^6}$$

(requiring a total of $O(n^8)$ bits to specify), such that, if we let $C_0 = C_{\text{start}}$, then:

  - C_{n^6} is accepting;
  - For every $i \in [n^6]$, the configuration C_i is computed from C_{i-1} within n^6 steps.

The latter condition can be verified in, say, $O(n^7)$ time.

So we get a $O(n^8)$-time Σ_2-TM for deciding membership in L.

From Alternating to Non-Deterministic Time

- We next show that, if, contrary to hypothesis,

$$\mathsf{NTIME}(n) \subseteq \mathsf{TISP}(n^{1.2}, n^{0.2}) \subseteq \mathsf{DTIME}(n^{1.2}),$$

then we can replace alternations with time.

Claim

If $\mathsf{NTIME}(n) \subseteq \mathsf{DTIME}(n^{1.2})$, then

$$\Sigma_2\mathsf{TIME}(n^8) \subseteq \mathsf{NTIME}(n^{9.6}).$$

- Using the equivalence between alternating time and the polynomial hierarchy, L is in $\Sigma_2\mathsf{TIME}(n^8)$ if and only if there is a TM M, such that

$$x \in L \quad \text{iff} \quad \exists u \in \{0,1\}^{c|x|^8} \forall v \in \{0,1\}^{d|x|^8} \; M(x,u,v) = 1,$$

for some constants c, d, where M runs in time $\mathrm{O}\left(|x|^8\right)$.

From Alternating to Non-Deterministic Time (Cont'd)

- Suppose $\text{NTIME}(n) \subseteq \text{DTIME}(n^{1.2})$.

  Then, by a simple padding argument, we have a deterministic algorithm D that, on inputs x, u, with $|x| = n$ and $|u| = cn^8$:
  - Runs in time $\text{O}\left((n^8)^{1.2}\right) = \text{O}\left(n^{9.6}\right)$-time;
  - Returns 1 if and only if, there exists some $v \in \{0,1\}^{dn^8}$, such that

$$M(x, u, v) = 0.$$

  Thus,

$$x \in L \quad \text{iff} \quad \exists u \in \{0,1\}^{c|x|^8} D(x, u) = 0.$$

  This implies that $L \in \text{NTIME}(n^{9.6})$.

Proof of the Time/Space Tradeoff for SAT

- Putting together the two claims shows that

  the assumption $\mathsf{NTIME}(n) \subseteq \mathsf{TISP}(n^{1.2}, n^{0.2})$ leads to contradiction.

  The assumption plus a simple padding argument implies that

  $$\mathsf{NTIME}(n^{10}) \subseteq \mathsf{TISP}(n^{12}, n^2).$$

  Now we have

  $$\begin{array}{rcll} \mathsf{NTIME}(n^{10}) & \subseteq & \mathsf{TISP}(n^{12}, n^2) & \\ & \subseteq & \Sigma_2\mathsf{TIME}(n^8) & \text{(by the First Claim)} \\ & \subseteq & \mathsf{NTIME}(n^{9.6}). & \text{(by the Second Claim)} \end{array}$$

  This contradicts the nondeterministic Time Hierarchy Theorem.

Subsection 5

Defining the Hierarchy via Oracle Machines

Oracle Characterization of the Polynomial Hierarchy

- Recall that **oracle machines** are machines with access to a special tape that they can use to make queries of the form "is $q \in O$?", for some language O.
- For every $O \subseteq \{0,1\}^*$, oracle TM M and input x, we denote by $M^O(x)$ the output of M on x with access to O as an oracle.

Theorem (Characterization of the Polynomial Hierarchy)

For every $i \geq 2$,

$$\Sigma_i^p = \mathsf{NP}^{\Sigma_{i-1}\mathrm{SAT}},$$

where $\mathsf{NP}^{\Sigma_{i-1}\mathrm{SAT}}$ is the set of languages decided by polynomial-time NDTMs with access to the oracle $\Sigma_{i-1}\mathrm{SAT}$.

- We showcase the proof idea by showing that $\Sigma_2^p = \mathsf{NP}^{\mathrm{SAT}}$.

The Second Level is in $\mathrm{NP}^{\textsc{Sat}}$

- Suppose that $L \in \Sigma_2^p$.

  Then, there is a polynomial-time TM M and a polynomial q, such that $x \in L$ iff

  $$\exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} M(x, u_1, u_2) = 1.$$

  For every fixed u_1 and x, the statement

  $$\text{“for every } u_2,\ M(x, u_1, u_2) = 1\text{”}$$

  is the negation of an NP-statement.

  Hence, its truth can be determined using an oracle for $\textsc{Sat}$.

  So a simple NDTM N, given oracle access for $\textsc{Sat}$, can decide L.

  On input x, nondeterministically guess u_1;
  Use the oracle to decide if $\forall u_2 M(x, u_1, u_2) = 1$.

  We see that $x \in L$ iff there exists a choice u_1 that makes N accept.

$\mathsf{NP}^{\mathrm{SAT}}$ is in the Second Level: Idea

- Conversely, suppose that L is decidable by a polynomial-time NDTM N with oracle access to SAT.
  - N could make polynomially many queries to the SAT oracle.
  - Moreover, every query could depend upon all preceding queries.

  At first sight this seems to give N more power than a Σ_2^p machine, which has the capability to nondeterministically make a single query to a coNP language.

  We wish to replace N by an equivalent Σ_2^p machine.

  The main idea is to:
  - Nondeterministically guess all future queries as well as the SAT oracle's answers;
  - Then to make a single coNP query whose answer verifies that all this guessing was correct.

$\mathrm{NP}^{\mathrm{SAT}}$ is in the Second Level: Details

- x is in L if and only if there exists a sequence of nondeterministic choices and correct oracle answers that makes N accept x.

  That is, there are:

  - A sequence of choices $c_1, \ldots, c_m \in \{0, 1\}$;
  - Answers to oracle queries $a_1, \ldots, a_k \in \{0, 1\}$;

  such that, on input x, if the machine N uses choices $c_1, \ldots, c_m$ and receives a_i as the answer to its i-th query:

  (1) M reaches the accepting state q_{accept};
  (2) All the answers are correct.

$\mathrm{NP}^{\mathrm{SAT}}$ is in the Second Level: Details (Cont'd)

- Let φ_i denote the i-th query that M makes to its oracle when executing on x, while:
  - Using choices $c_1, \ldots, c_m$;
  - Receiving answers $a_1, \ldots, a_k$.

  Then, Condition (2) can be phrased as follows:
  - If $a_i = 1$, then there exists an assignment u_i, such that $\varphi_i(u_i) = 1$;
  - If $a_i = 0$, then, for every assignment v_i, $\varphi_i(v_i) = 0$.

  Thus,

$$\begin{array}{lll} x \in L & \text{iff} & \exists c_1, \ldots, c_m, a_1, \ldots, a_k, u_1, \ldots, u_k \forall v_1, \ldots, v_k \\ & & [N \text{ accepts } x \text{ using choices } c_1, \ldots, c_m \\ & & \text{and answers } a_1, \ldots, a_k \\ & & \text{AND } \forall i \in [k] \text{ if } a_i = 1, \text{ then } \varphi_i(u_i) = 1 \\ & & \text{AND } \forall i \in [k], \text{ if } a_i = 0, \text{ then } \varphi_i(v_i) = 0]. \end{array}$$

  This shows that $L \in \Sigma_2^p$.