Complexity Theory

VU 181.142, SS 2018

6. The Polynomial Hierarchy

Reinhard Pichler

Institut für Informationssysteme
Arbeitsbereich DBAI
Technische Universität Wien

15 May, 2018

Outline

EXACT TSP

Problem **EXACT TSP**

INSTANCE: n cities $1, \ldots, n$, a non-negative integer distance d_{ij} between any two cities i and j (such that $d_{ij} = d_{ji}$), and an integer B.

QUESTION: Is the length of the shortest tour equal to B?

Complexity of **EXACT TSP**

EXACT TSP can be considered as the intersection of two problems – one in NP and one in co-NP:

- in NP: **TSP(D)** (= asking if the shortest tour has length $\leq B$).
- in co-NP: **TSP COMPLEMENT** (= asking if the shortest tour has length $\geq B$).

The Class DP

Definition

A language L is in the class DP iff there are two languages $L_1 \in$ NP and $L_2 \in$ co-NP such that $L = L_1 \cap L_2$.

Remark. Note that DP *is not* NP $\cap$ co-NP!
(Most likely DP is not even contained in NP $\cup$ co-NP.)

Proposition

- **EXACT TSP** *is* DP*-complete.*
- *All exact cost versions of the* NP*-complete optimization problems studied in the lecture are* DP*-complete, e.g.* **INDEPENDENT SET** *(i.e.: is the size of the biggest independent set equal to some K?),* **VERTEX COVER**, **CLIQUE**, *etc.*

SAT-UNSAT

Problem **SAT-UNSAT**

INSTANCE: two Boolean expressions (φ, φ') (possibly both in 3-CNF).

QUESTION: Is it true that φ is satisfiable and φ' is unsatisfiable?

Proposition

SAT-UNSAT *is* DP*-complete.*

Proof of Membership

Let $L_1 = \{(\varphi, \psi) \mid \varphi$ satisfiable and ψ arbitrary propositional formula$\}$
Let $L_2 = \{(\varphi, \psi) \mid \varphi$ arbitrary propositional formula and ψ unsatisfiable$\}$
Clearly $L_1 \in$ NP, $L_2 \in$ co-NP, and **SAT-UNSAT** $= L_1 \cap L_2$.

Proof of Hardness

Let L be an arbitrary language in DP, i.e., there exists a language $L_1 \in$ NP and a language $L_2 \in$ co-NP with $L = L_1 \cap L_2$.

Let x be an arbitrary instance of L. We reduce x to the following instance $R(x)$ of **SAT-UNSAT**:

$L_1 \in$ NP and $L_2 \in$ co-NP $\Rightarrow$
there exists a reduction R_1 from L_1 to **3-SAT** and
there exists a reduction R_2 from L_2 to co-**3-SAT**.
We define $R(x) := (R_1(x), R_2(x))$.

Clearly, $R(x)$ is a positive instance of **SAT-UNSAT** $\Leftrightarrow$
$R_1(x)$ is satisfiable and $R_2(x)$ is unsatisfiable $\Leftrightarrow$
(by the correctness of R_1 and R_2) $x \in L_1$ and $x \in L_2$ $\Leftrightarrow$
$x \in L$.

Further DP-complete problems

"Critical Problems"

CRITICAL SAT
INSTANCE: Propositional formula φ in CNF
QUESTION: Is it true that φ is unsatisfiable but deleting any clause makes φ satisfiable?

CRITICAL HAMILTON PATH
INSTANCE: (Directed or undirected) graph $G = (V, E)$
QUESTION: Is it true that G has no Hamilton path but addition of any edge creates a Hamilton path?

CRITICAL 3-COLORABILITY
INSTANCE: Undirected graph $G = (V, E)$
QUESTION: Is it true that G has no 3-coloring but deletion of any node makes it 3-colorable?

Further DP-complete problems

Proposition

CRITICAL SAT, **CRITICAL HAMILTON PATH**, *and* **CRITICAL 3-COLORABILITY** *are* DP*-complete.*

Remark

The above problems are called "critical" because the input x is "critical" with respect to some property, i.e., x has some property but the slightest modification of x does not.

Oracle Machines

Motivation

- Intuitively, an oracle is a subroutine with 0 cost (we count the cost of the oracle as 1 for the call – but we neglect the cost of the computation carried out by the oracle). $\Rightarrow$ We can study complexity in a setting where a part of the computation comes "for free".
- Oracles allow us to isolate orthogonal (independent) sources of complexity, i.e. we can answer questions like: Suppose that we know the complexity of some sub-task A for solving problem B. What is the remaining complexity of problem B?

Oracle Machines

Definition

- An oracle Turing machine $M^?$ has the following additional features:
  – an additional tape (= query tape)
  – three additional states: query state $q?$, answer states q_{YES}, q_{NO}
- Suppose that $M^?$ has an oracle for the problem A. Then the call of the oracle works as follows: If $M^?$ is in state $q?$, then $M^?$ decides if the string z on the query tape is a positive instance of A or not. $\Rightarrow$ $M^?$ either enters state q_{YES} or q_{NO} in one step.
- Notation. For any *time* complexity class $\mathcal{C}$ and oracle A (where A is either a problem or a class of problems) we write $\mathcal{C}^A$ for the problems which can be decided by a TM within the time bound of $\mathcal{C}$, where the TM is allowed to use an oracle for (any problem in the class) A.
- Examples. $\mathrm{P}^{\mathbf{SAT}}$, $\mathrm{NP}^{\mathbf{SAT}}$, P^{NP}, $\mathrm{NP}^{\mathrm{NP}}$, ...,

The Polynomial Hierarchy

Definition

The polynomial hierarchy is a sequence of classes:

- $\Delta_0\mathrm{P} = \Sigma_0\mathrm{P} = \Pi_0\mathrm{P} = \mathrm{P}$
- $i \geq 0$: $\Delta_{i+1}\mathrm{P} = \mathrm{P}^{\Sigma_i\mathrm{P}}$
  $\Sigma_{i+1}\mathrm{P} = \mathrm{NP}^{\Sigma_i\mathrm{P}}$
  $\Pi_{i+1}\mathrm{P} = \text{co-NP}^{\Sigma_i\mathrm{P}}$
- Cumulative polynomial hierarchy: $\mathrm{PH} = \bigcup_{i\geq 0} \Sigma_i\mathrm{P}$

In the literature also the following notation is used: Δ_i^{p}, Σ_i^{p}, Π_i^{p}

The Polynomial Hierarchy

Properties of the Polynomial Hierarchy

- special case $i = 1$:
  $\Delta_1\mathrm{P} = \mathrm{P}^{\Sigma_0\mathrm{P}} = \mathrm{P}^{\mathrm{P}} = \mathrm{P}$
  $\Sigma_1\mathrm{P} = \mathrm{NP}^{\Sigma_0\mathrm{P}} = \mathrm{NP}^{\mathrm{P}} = \mathrm{NP}$
  $\Pi_1\mathrm{P} = \text{co-NP}^{\Sigma_0\mathrm{P}} = \text{co-NP}$
- special case $i = 2$:
  $\Delta_2\mathrm{P} = \mathrm{P}^{\Sigma_1\mathrm{P}} = \mathrm{P}^{\mathrm{NP}}$
  $\Sigma_2\mathrm{P} = \mathrm{NP}^{\Sigma_1\mathrm{P}} = \mathrm{NP}^{\mathrm{NP}}$
  $\Pi_2\mathrm{P} = \text{co-NP}^{\Sigma_1\mathrm{P}} = \text{co-NP}^{\mathrm{NP}}$
- $\Delta_i\mathrm{P} \subseteq \begin{matrix}\Sigma_i\mathrm{P}\\ \Pi_i\mathrm{P}\end{matrix} \subseteq \Delta_{i+1}\mathrm{P} \subseteq \begin{matrix}\Sigma_{i+1}\mathrm{P}\\ \Pi_{i+1}\mathrm{P}\end{matrix} \subseteq \Delta_{i+2}\mathrm{P}$

Characterization via Certificates

Theorem

- *Let L be a language and $i \geq 1$. Then $L \in \Sigma_i\mathrm{P}$ iff there is a polynomially balanced relation R (i.e., there exists k, s.t. $(x,y) \in R$ implies $|y| \leq |x|^k$), such that the language $\{x\#y \mid (x,y) \in R\}$ is in $\Pi_{i-1}\mathrm{P}$ and*

$$L = \{x \mid \text{there exists a } y \text{ with } |y| \leq |x|^k \text{ s.t. } (x,y) \in R\}$$

- *Let L be a language and $i \geq 1$. Then $L \in \Pi_i\mathrm{P}$ iff there is a polynomially balanced relation R such that the language $\{x\#y \mid (x,y) \in R\}$ is in $\Sigma_{i-1}\mathrm{P}$ and*

$$L = \{x \mid \text{for all } y \text{ with } |y| \leq |x|^k, (x,y) \in R\}$$

Remark. Of course, in the definition of $\Sigma_i\mathrm{P}$, we could omit the condition $|y| \leq |x|^k$, since we talk about a polynomially balanced relation R.

Proof

It suffices to prove the correctness of the characterization of $\Sigma_i\mathrm{P}$ for every i. The correctness of the characterization of $\Pi_i\mathrm{P}$ follows immediately by the equality $\Pi_i\mathrm{P} = \text{co-}\Sigma_i\mathrm{P}$.

The correctness proof for $\Sigma_i\mathrm{P}$ proceeds by induction on i.
Recall that $\Sigma_1\mathrm{P} = \mathrm{NP}$. Hence, for $i = 1$, the theorem corresponds to the characterization of NP via succinct certificates. For $i > 1$, we show both directions separately:

"$\Leftarrow$" Suppose that such a relation R exists. We must show that $L \in \Sigma_i\mathrm{P}$. Indeed, L is decided by the following non-deterministic, polynomial-time Turing machine with $\Sigma_{i-1}\mathrm{P}$-oracle:

1. On input x, guess an appropriate y.
2. Check by means of a $\Pi_{i-1}\mathrm{P}$ oracle if $(x,y) \in R$ (or, equivalently, check by a $\Sigma_{i-1}\mathrm{P}$ oracle if $(x,y) \notin R$).

Proof (continued)

"$\Rightarrow$" Suppose that $L \in \Sigma_i\mathrm{P}$, i.e., L is decided by a non-deterministic, polynomial-time TM M with an oracle for some language $K \in \Sigma_{i-1}\mathrm{P}$. We must show that an appropriate relation R exists.

By the induction hypothesis, there exists a binary relation S, s.t. the language $\{u\#v \mid (u,v) \in S\}$ is in $\Pi_{i-2}\mathrm{P}$ and $K = \{u \mid \text{there exists a } v \text{ with } |v| \leq |u|^k \text{ s.t. } (u,v) \in S\}$.

We construct a relation R as follows: We know that $x \in L$ iff there exists an accepting computation of M (with oracle for K) on x. We define R, s.t. $(x,y) \in R$, iff y is a "certificate" of x in the following sense:

1. y encodes the non-deterministic choices of a successful computation of the TM M as in the construction of succinct certificates for NP.
2. In addition, y contains a certificate v_j for every *successful* call u_j to the oracle $K \in \Sigma_{i-1}\mathrm{P}$.

Proof (continued)

It remains to show that $(x,y) \in R$ can indeed be decided in $\Pi_{i-1}\mathrm{P}$:

1. We must check that the first part of y indeed encodes a successful computation of the TM M. This can be done in polynomial time (as in the construction of succinct certificates for NP).
2. We must check for polynomially many pairs (u_j, v_j) that $(u_j, v_j) \in S$ holds. Each such test is in $\Pi_{i-2}\mathrm{P}$. Hence, in total, these tests are in $\Delta_{i-1}\mathrm{P} \subseteq \Pi_{i-1}\mathrm{P}$ (actually, they are even in $\Pi_{i-2}\mathrm{P}$).
3. For all "no"-queries u_j to the K-oracle, we must check that indeed $u_j \notin K$. Since $K \in \Sigma_{i-1}\mathrm{P}$, each test $u_j \notin K$ is in $\Pi_{i-1}\mathrm{P}$.

   All these tests together can be done by a single $\Pi_{i-1}\mathrm{P}$ computation since the co-query "there exists a j with $u_j \in K$" is in $\Sigma_{i-1}\mathrm{P}$: Guess j and v_j and check in $\Pi_{i-2}\mathrm{P}$ that $(u_j, v_j) \in S$.

Characterization via Certificates and Alternation

Definition

A relation $R \subseteq (\Sigma^*)^{i+1}$ is said to be polynomially balanced if whenever $(x, y_1, \ldots, y_i) \in R$, it holds that $|y_1|, \ldots, |y_i| \leq |x|^k$ for some k.

Corollary, part 1

Let L be a language and $i \geq 1$. Then $L \in \Sigma_i \mathrm{P}$ iff there is a polynomially balanced, polynomial-time decidable $(i+1)$-ary relation R such that

$$L = \{x \mid \exists y_1 \forall y_2 \exists y_3 \cdots Q y_i \text{ such that } (x, y_1, \ldots, y_i) \in R\}$$

where Q is $\forall$ if i is even and $\exists$ if i is odd.

Characterization via Certificates and Alternation

Corollary, part 2

Let L be a language and $i \geq 1$. Then $L \in \Pi_i \mathrm{P}$ iff there is a polynomially balanced, polynomial-time decidable $(i+1)$-ary relation R such that

$$L = \{x \mid \forall y_1 \exists y_2 \forall y_3 \cdots Q y_i \text{ such that } (x, y_1, \ldots, y_i) \in R\}$$

where Q is $\exists$ if i is even and $\forall$ if i is odd.

Proof idea

Use the above theorem and proceed by induction on i. Repeatedly replace languages in $\Pi_j \mathrm{P}$ and $\Sigma_j \mathrm{P}$ by their certificate forms as in the theorem.

Properties of PH

Definition

We say that the polynomial hierarchy collapses to the i-th level if $\Delta_j \mathrm{P} = \Sigma_j \mathrm{P} = \Pi_j \mathrm{P} = \Sigma_i \mathrm{P}$ holds for every $j > i$.

Remark. It is unknown whether PH is indeed an infinite hierarchy, i.e.: $\Sigma_0 \mathrm{P} \subset \Sigma_1 \mathrm{P} \subset \Sigma_2 \mathrm{P} \subset \ldots$. is generally believed but not known.

Proposition

- *If for some $i \geq 1$, $\Sigma_i \mathrm{P} = \Pi_i \mathrm{P}$, then the polynomial hierarchy collapses to the i-th level. In particular, if $\mathrm{NP} = \text{co-NP}$, then the polynomial hierarchy collapses to the first level.*
- $\mathrm{P} = \mathrm{NP}$ *iff* $\mathrm{P} = \mathrm{PH}$.
- *Notice that it can be the case that $\mathrm{P} \neq \mathrm{NP}$ and $\mathrm{NP} \neq \text{co-NP}$ but PH collapses to the second level (not expected to happen, though).*

QBFs: Quantified Boolean Formulae

QSAT$_i$

"quantified satisfiability with i alternating blocks of quantifiers":

INSTANCE: Boolean expression φ with the Boolean variables partitioned into i sets $X_1, \ldots, X_i$

QUESTION: Is it true that
there exists a partial truth assignment for the variables X_1 such that
for all partial truth assignments for X_2
there exists a partial truth assignment for X_3
$\ldots \varphi$ is satisfied by the overall truth assignment?

Notation

A QSAT$_i$-formula is given in the form $\exists X_1 \forall X_2 \exists X_3 \cdots Q X_i \, \varphi$, where Q is $\forall$ if i is even and $\exists$ if i is odd.

Σ_iP-Completeness

Theorem

For all $i \geq 1$, QSAT$_i$ is Σ_iP-complete.

Proof of Σ_iP-membership

Recall the characterization of Σ_iP via certificates: A language L is in Σ_iP iff there is a polynomially balanced, polynomial-time decidable $(i+1)$-ary relation R, s.t. $L = \{x \mid \exists y_1 \forall y_2 \exists y_3 \cdots Q y_i \text{ such that } (x, y_1, \ldots, y_i) \in R\}$.

For a QSAT$_i$-formula $\psi = \exists X_1 \forall X_2 \exists X_3 \cdots Q X_i\ \varphi$, we take as certificates $(y_1, \ldots, y_i)$ combinations of variable assignments to the alternating blocks $X_1, X_2, \ldots, X_i$ of variables (where each y_j is an assignment on the variables in X_j), s.t. the formula φ is true in the overall assignment.

Proof of Σ_iP-hardness

We only consider the case that i is odd. The case that i is even is treated analogously. Let L be an arbitrary language in Σ_iP. Hence, there exists a polynomially balanced, polynomial-time decidable $(i+1)$-ary relation R, s.t. $L = \{x \mid \exists y_1 \forall y_2 \exists y_3 \cdots Q y_i \text{ such that } (x, y_1, \ldots, y_i) \in R\}$.

There exists a polynomial-time deterministic TM M accepting $x\#y_1\#\ldots\#y_i$ iff $(x, y_1, \ldots, y_i) \in R$. Following the proof of the Cook-Levin Theorem, there exists a Boolean formula φ that captures the computations of M. We split the variables in φ into $i+2$ classes:

- Variable set X: all propositional variables in φ encoding the first part (before the first $\#$) of the input string to M.
- Variable sets Y_1 to Y_i: encode the remaining input string.
- Variable set Z captures all other aspects of the computation of M.

Proof of Σ_iP-hardness (continued)

Now let x be an arbitrary instance of the decision problem corresponding to the language L. Moreover, let $\hat{X}$ denote the values of the variables in X corresponding to the string x. By $\varphi(\hat{X})$ we denote the result of substituting in φ the corresponding Boolean values $\hat{X}$ for X. We define the desired instance of QSAT$_i$ as $\psi = \exists Y_1 \forall Y_2 \ldots \exists Y_i \exists Z \varphi(\hat{X})$.

For the correctness proof, we observe: Let $y_1, \ldots, y_i$ be arbitrary strings with Boolean "encoding" $\hat{Y}_1, \ldots, \hat{Y}_i$ and suppose that we substitute these values $\hat{Y}_1, \ldots, \hat{Y}_i$ for the variables $Y_1, \ldots, Y_i$ in φ. Then the resulting expression $\varphi(\hat{X}, \hat{Y}_1, \ldots, \hat{Y}_i)$ is satisfiable (i.e., there exist an appropriate assignment to the variables in Z) iff $(x, y_1, \ldots, y_i) \in R$.

It remains to show that $x \in L \Leftrightarrow \psi$ is a positive instance of QSAT$_i$.

Indeed, $x \in L$ iff there is a y_1 s.t. for all $y_2, \ldots$ there is a y_i s.t. $(x, y_1, \ldots, y_i) \in R$. In terms of ψ, this means that for these values of $\hat{X}$ there are values $\hat{Y}_1$ for Y_1 s.t. for all values $\hat{Y}_2$ for $Y_2, \ldots$ there are values $\hat{Y}_i$ for Y_i and there are values $\hat{Z}$ for Z s.t. the resulting formula $\varphi(\hat{X}, \hat{Y}_1, \ldots, \hat{Y}_i, \hat{Z})$ is **true**, i.e., ψ is a positive instance of QSAT$_i$.

Further Complete Problems

Theorem

For all $i \geq 1$ even, the QSAT$_i$ problem remains Σ_iP-complete even if the instances $\exists X_1 \forall X_2 \exists X_3 \cdots \forall X_i\ \varphi$ are restricted s.t. φ is in 3-DNF.

For all $i \geq 1$ odd, the QSAT$_i$ problem remains Σ_iP-complete even if the instances $\exists X_1 \forall X_2 \exists X_3 \cdots \exists X_i\ \varphi$ are restricted s.t. φ is in 3-CNF.

Theorem

MINIMAL MODEL SAT: *Given a propositional formula φ in CNF and an atom x, is x* **true** *in some (subset) minimal model of φ?*
MINIMAL MODEL SAT *is Σ_2P-complete.*

MINIMAL MODEL SAT

Proof of the Σ_2P-membership

We have to show that **MINIMAL MODEL SAT** can be decided by an NP-algorithm using an NP-oracle (or, equivalently, a co-NP-oracle).
Let ψ be an arbitrary CNF-formula with variables in X.

1. Guess a truth assignment $\mathcal{I}$, s.t. x is true in $\mathcal{I}$. Let $Y \subseteq X$ denote the variables which are true in $\mathcal{I}$.
2. Check ψ is true in $\mathcal{I}$.
3. Check (with an oracle) that there does not exist a "smaller" satisfying truth assignment $\mathcal{J}$ of ψ, i.e., let Z denote the variables true in $\mathcal{J}$, then $Z \not\subset Y$ for any satisfying truth assignment $\mathcal{J}$ of ψ.

The check in step 3 can be done by a co-NP-oracle, i.e.: checking that there does exist a "smaller" satisfying truth assignment $\mathcal{J}$ of ψ can be clearly done in NP.

MINIMAL MODEL SAT

Proof of the Σ_2P-hardness

By reduction from QSAT$_2$. Let an arbitrary instance of QSAT$_2$ be given by the QBF

$$\psi = (\exists x_1, \dots, x_k)(\forall y_1, \dots, y_\ell)\varphi$$

Now let $\{x'_1, \dots, x'_k, z\}$ be fresh propositional variables. We construct an instance of **MINIMAL MODEL SAT** by the variable z and the formula

$$\chi = \left(\bigwedge_{i=1}^{k} (\neg x_i \leftrightarrow x'_i) \right) \wedge \left(\neg\varphi \vee (y_1 \wedge \dots \wedge y_\ell \wedge z) \right)$$

We have to show that ψ is **true** $\Leftrightarrow$ z is **true** in a minimal model of χ. (We only prove the "$\Rightarrow$"-direction here).

Idea. The conjuncts $\neg x_i \leftrightarrow x'_i$ make truth assignments on the variables $\{x_1, \dots, x_k, x'_1, \dots, x'_k\}$ incomparable (since any satisfying truth assignment assigns **true** to exactly k out of these $2k$ variables).

MINIMAL MODEL SAT

Proof of the Σ_2P-hardness (continued)

"$\Rightarrow$" Suppose that $\psi = (\exists x_1, \dots, x_k)(\forall y_1, \dots, y_\ell)\varphi$ is **true**. Then there exists a partial assignment $\mathcal{I}$ on $\{x_1, \dots, x_k\}$, s.t. for any values assigned to $\{y_1, \dots, y_\ell\}$, the formula φ is **true** (or, equivalently, $\neg\varphi$ is **false**).

We define the truth assignment $\mathcal{J}$ appropriate to χ as follows:

$\mathcal{J}(x_i) = \mathcal{I}(x_i)$ and $\mathcal{J}(x'_i) = \mathcal{I}(\neg x_i)$ for every i,
$\mathcal{J}(y_j) =$ **true** for every j, and $\mathcal{J}(z) =$ **true**.

We claim that $\mathcal{J}$ is a minimal model of χ where z is **true**.

Clearly, $\mathcal{J}$ is a model (i.e., satisfying truth assignment) of χ since all conjuncts $(\neg x_i \leftrightarrow x'_i)$ and the disjunct $(y_1 \wedge \dots \wedge y_\ell \wedge z)$ are **true** in $\mathcal{J}$. Moreover, $\mathcal{J}(z) =$ **true** by definition. It remains to show that there does not exist a strictly "smaller" model of χ.

Suppose to the contrary that there exists a model $\mathcal{J}'$ of χ, s.t. $\mathcal{J}'$ is strictly smaller than $\mathcal{J}$. Then there exists a variable x_i, x'_i, y_j or z, s.t. this variable is **true** in $\mathcal{J}$ and **false** in $\mathcal{J}'$. We distinguish 3 cases:

MINIMAL MODEL SAT

Proof of the Σ_2P-hardness (continued)

Case 1. Suppose that there exists a variable x_i with $\mathcal{J}(x_i) =$ **true** and $\mathcal{J}'(x_i) =$ **false**. Since both $\mathcal{J}$ and $\mathcal{J}'$ satisfy χ, we thus have $\mathcal{J}(x'_i) =$ **false** and $\mathcal{J}'(x'_i) =$ **true**. Hence, $\mathcal{J}'$ cannot be smaller than $\mathcal{J}$.

Case 2. Suppose that there exists a variable x'_i with $\mathcal{J}(x'_i) =$ **true** and $\mathcal{J}'(x'_i) =$ **false**. Analogously to case 1, this leads to a contradiction since $\mathcal{J}(x_i) =$ **false** and $\mathcal{J}'(x_i) =$ **true**.

Case 3. Suppose that either $\mathcal{J}'(y_j) =$ **false** for some j or $\mathcal{J}'(z) =$ **false**. Then clearly the disjunct $(y_1 \wedge \dots \wedge y_\ell \wedge z)$ is **false** in $\mathcal{J}'$. For $\mathcal{J}'$ to be a model of χ, it must be a model of $\neg\varphi$, i.e., $\mathcal{J}'$ is an extension of $\mathcal{I}$ to $\{y_1, \dots, y_\ell\}$, s.t. $\neg\varphi$ is **true** (or, equivalently, φ is **false**) in $\mathcal{J}'$.
This contradicts the assumption that $\mathcal{I}$ is a partial assignment on $\{x_1, \dots, x_k\}$, s.t. for any values of $\{y_1, \dots, y_\ell\}$, the formula φ is **true**.

Exercise. Prove also the other direction of the equivalence between QSAT$_2$ and **MINIMAL MODEL SAT**.

Further Properties of PH

Theorem

If there is a PH*-complete problem, then the polynomial hierarchy collapses to some finite level.*

Proof

Assume L is PH-complete. Then $L \in \Sigma_i$P for some i. But then any $L' \in \Sigma_{i+1}$P reduces to L. This means that $\Sigma_i\mathrm{P} = \Sigma_{i+1}\mathrm{P}$ since each level is closed under reductions. Thus PH collapses to the i-th level.

- By the above theorem, PH probably has no complete problems. But of course each level of PH does (namely QSAT$_i$).
- If L is a Σ_{i+1}P-complete language and $L \in \Sigma_i$P, then PH collapses to the i-th level. Hence, if PH does not collapse, then problems on an upper level are strictly harder than on a lower level.

Restrictions on the Oracle Calls

Motivation

We consider two kinds of restrictions:

1. Number of oracle calls.
   - In DP, only 2 calls to an oracle are allowed.
   - Many natural problems require only $O(\log n)$ oracle calls, since they come down to finding the optimal value via binary search, e.g.: max. size of a clique, max./min. cardinality of a model, etc.
2. Adaptive vs. non-adaptive calls:
   - adaptive: The i-th question to the oracle may depend on the result of the previous $(i-1)$ calls to the oracle.
   - non-adaptive: otherwise.

Complexity Classes

Definition

- $\Delta_2\mathrm{P} = \mathrm{P}^{\mathrm{NP}}$: no restrictions on the oracle calls
- $\Delta_2\mathrm{P}[\log n] = \mathrm{P}^{\mathrm{NP}[\log n]}$: only $O(\log n)$ oracle calls allowed
  $\Delta_2\mathrm{P}[\log n]$ is also referred to as Θ_2P.
- $\mathrm{P}^{\mathrm{NP}}_{\|}$: polynomially many, non-adaptive oracle calls
- analogous classes for function problems: $\mathrm{FP}^{\mathrm{NP}[\log n]}$, $\mathrm{FP}^{\mathrm{NP}}_{\|}$

Remark

Recognizing a language $L \in \mathrm{P}^{\mathrm{NP}}_{\|}$: Machine M computes on input x in polynomial time a polynomial number of **SAT**-instances (or any other problem in NP) and then calls the oracle for all these instances at once. Based on the answers, M decides in polynomial time whether $x \in L$.

Examples in $\mathrm{P}^{\mathrm{NP}[\log n]}$

CARD-MINIMAL MODEL SAT

INSTANCE: Boolean formula φ and an atom z.
QUESTION: Is z true in a cardinality-minimal model of φ?

Proof of $\mathrm{P}^{\mathrm{NP}[\log n]}$-membership

1. Compute the size K of a cardinality-minimal model of φ. This can be done by a binary search asking questions like "Does φ have a model of size $\leq k$?". For this task, we need $\log n$ calls to an NP-oracle, where n = number of variables in φ.
2. Finally, ask an NP-oracle: "Is z true in some model $\mathcal{I}$ of φ, s.t. $\mathcal{I}$ sets exactly K variables to true?"

Analogously: **CARD-MAXIMAL MODEL SAT**

Examples of Optimization Problems in $\mathrm{FP}^{\mathrm{NP}[\log n]}$

Some graph problems

- **MIN-VERTEX COVER**, **MAX-CLIQUE**, **MAX-INDEP.-SET**: Given a graph $G = (V, E)$, what is the size of the smallest vertex cover (resp. of the biggest clique or the biggest independent set)?
- **CHROMATIC NUMBER**: Given a graph $G = (V, E)$, what is the smallest number k, s.t. G has a k-coloring?

Some SAT-related problems

- **CARD-MINIMAL-MODEL**, **CARD-MAXIMAL-MODEL**: Given a Boolean formula φ, what is the size of a minimal (resp. maximal) model of φ?
- **MAX-SAT**: Given a Boolean formula φ in CNF, what is the maximal number of clauses that can be satisfied by a truth assignment?

Examples in P^{NP}

WEIGHT-MINIMAL MODEL SAT

INSTANCE: Boolean formula φ, vector $(w_1, \dots, w_n)$ of weights (i.e., positive integers) of the variables $(x_1, \dots, x_n)$ in φ and an atom x_i.
QUESTION: Is x_i true in a weight-minimal model of φ?

Proof of P^{NP}-membership

1. Compute the minimal weight W of all models of φ. This can be done by a binary search asking questions like “Does φ have a model of weight $\leq w$?”. For this task, we need logarithmically many calls (w.r.t. the total weight) to an NP-oracle. These are polynomially many calls w.r.t. the representation of the weights $w_1, \dots, w_n$.
2. Finally, ask an NP-oracle: “Is x_i true in some model $\mathcal{I}$ of φ, s.t. the total weight of the variables true in $\mathcal{I}$ is W?”

Analogously: **WEIGHT-MAXIMAL MODEL SAT**

Examples in P^{NP} (Continued)

LEX-MINIMAL MODEL SAT

INSTANCE: Boolean formula φ, order $(x_1, \dots, x_n)$ of the variables in φ.
QUESTION: Is x_n true in the lexicographically smallest model of φ?

Proof of P^{NP}-membership

LEX-MINIMAL MODEL SAT can be decided by the following program with n calls to an NP-oracle.

for $i := 1$ to n do {
  check if φ has a model $\mathcal{I}$, s.t. (for all $j < i$: $\mathcal{I}(x_j) = v_j$) and $\mathcal{I}(x_i) = 0$;
  if yes then set $v_i := 0$, otherwise set $v_i := 1$;
}
if $v_n = 1$ then return true else return false.

Analogously: **LEX-MAXIMAL MODEL SAT**

Examples of Optimization Problems in $\mathrm{FP}^{\mathrm{NP}}$

Some graph problems

- **MIN-WEIGHT-VERTEX COVER**, **MAX-WEIGHT-CLIQUE**, **MAX-WEIGHT-INDEP.-SET**: Given a graph $G = (V, E)$ and weights w_i of the vertices, what is the size of the minimal total weight of a vertex cover, etc.?
- **TSP**: What is the length of the shortest tour through the n cities?

Some SAT-related problems

- **WEIGHT-MINIMAL-MODEL**, **WEIGHT-MAXIMAL-MODEL**
- **MAX-WEIGHT-SAT**: Given a Boolean formula φ in CNF and vector $(w_1, \dots, w_m)$ of weights of the *clauses* $(c_1, \dots, c_m)$ in φ, what is the maximal total weight of clauses that can be simultaneously satisfied by a truth assignment?

$\mathrm{P}^{\mathrm{NP}[\log n]}$ vs. $\mathrm{P}^{\mathrm{NP}}_{\parallel}$

Theorem

$\mathrm{P}^{\mathrm{NP}[\log n]} = \mathrm{P}^{\mathrm{NP}}_{\parallel}$

Proof

Both inclusions are shown separately:

"$\subseteq$": Suppose that a machine M makes $k \log n$ adaptive queries. For each of these queries, there are 2 possible outcomes. Hence, in total there are at most $2^{k \log n} = n^k$ queries in the whole computation. Hence, the computation of M can be simulated by first computing the n^k possible queries and asking all of them at once to the oracle.

Proof (continued)

"$\supseteq$": Suppose that a language L is decided by a TM M with polynomially many non-adaptive **SAT** queries. Then L can be decided with logarithmically many adaptive NP queries as follows:

- In $\mathrm{O}(\log n)$ queries determine the precise number K of "yes" answers to the non-adaptive queries. This can be done by binary search using the oracle: "Given a set of Boolean expressions, does it have satisfying truth assignments for at least k of them?
- Ask the NP query: "Do there exist K satisfiable Boolean expressions such that if all other expressions were unsatisfiable (at this point, we know that they must be), then M would end up accepting?"

  Remark. A succinct certificate for the last query consists of indices $i_1, \ldots, i_K$ of Boolean expressions and models $\mathcal{I}_1, \ldots, \mathcal{I}_K$ of them.

Δ_2P and Θ_2P-Hardness Proofs

Motivation

- Most problems in Δ_2P and Θ_2P are about optimization (asking if a minimum/maximum "solution" has certain properties).
- The complexity classes Δ_2P and Θ_2P are defined via oracle calls.
- We need to draw a connection between the result of a sequence of oracle calls and optimization.
- To this end, we define auxiliary problems **NP-MAX** and **LogNP-MAX**, which will allow us to draw this connection.

Remark. In principle, algorithms in Θ_2P are allowed to make $\mathrm{O}(\log n)$ oracle calls (where n is the size of the input). For the sake of simplicity, we assume below that there are only $\log n$ oracle calls.

Problems **NP-MAX** and **LogNP-MAX**

Problem **NP-MAX**

INSTANCE: $(M;x)$,
where M is a non-deterministic, polynomial-time Turing machine producing a binary string as output, and x is an input string to M.

QUESTION: Does the last bit of w have value 1,
where w denotes the lexicographically maximal output string over all computation paths of M on input x?

Problem **LogNP-MAX**

INSTANCE: $(M;x)$,
where M is a non-deterministic, polynomial-time Turing machine producing a binary string as output whose length is logarithmically bounded in the size of $(M;x)$, and x is an input string to M.

QUESTION: Does the last bit of w have value 1,
where w denotes the lexicographically maximal output string over all computation paths of M on input x?

Problems **NP-MAX** and **LogNP-MAX**

Theorem

- *The problem* **NP-MAX** *is* Δ_2P*-complete.*
- *The problem* **LogNP-MAX** *is* Δ_2P[log n]*-complete.*

Proof of Membership

Maintain a bit vector $(v_1, v_2, \ldots)$ of the lexicographically maximal prefix of possible outputs of TM M on input x:

initialize i to 0 and ask the following kind of questions to an NP-oracle: Does there exist a computation path of TM M on input x, such that the first i output bits are $(v_1, \ldots, v_i)$ and M outputs yet another bit?

If the answer to this oracle call is “no” then
– the algorithm stops with acceptance if ($i \geq 1$ and $v_i = 1$) holds
– otherwise (i.e., $i = 0$ or $v_i = 0$) it stops with rejection.

Continuation of the Membership Proof

If the oracle call yields a “yes” answer, then our algorithm calls another NP-oracle with the question: Does there exist a computation path of TM M on input x, such that the first $i+1$ output bits are $(v_1, \ldots, v_i, 1)$?

If the answer to this oracle call is “yes”,
– then we set $v_{i+1} = 1$;
– otherwise we set $v_{i+1} = 0$.

In either case, we then increment i by 1 and continue with the first oracle question (i.e., does there exist a computation path of TM M on input x, such that the first i output bits are $(v_1, \ldots, v_i)$ and M outputs yet another bit?).

Summary. Suppose that the lexicographically maximal output produced by M on input x has m bits. Then our algorithm needs in total $2m+1$ oracle calls (each working in non-deterministic polynomial time).
If the size of the output string of M is logarithmically bounded, then the number of oracle calls is logarithmically bounded as well.

Δ_2P-Hardness Proof of **NP-MAX**

Consider an arbitrary problem $\mathcal{P}$ in Δ_2P,
i.e., $\mathcal{P}$ is decided in deterministic polynomial time by a Turing machine N with access to an oracle N_{SAT} for the SAT-problem.

Let x be an arbitrary instance of problem $\mathcal{P}$. We construct instance $M; x$ of **NP-MAX**, where we leave x unchanged and we define M as follows:

(1) In principle, M simulates the execution of N on input x. However, whenever N reaches a call to the SAT-oracle, with some input φ_i say, then M non-deterministically executes N_{SAT} on φ_i.

Intuition. In the computation tree of M, the subtree corresponding to this non-deterministic execution of N_{SAT} on φ_i is precisely the computation tree of N_{SAT} on φ_i.

(2) On every computation path ending in acceptance (resp. rejection) of N_{SAT}, the TM M writes 1 (resp. 0) to the output. After that, M continues with the execution of N as if it had received a “yes” (resp. a “no”) answer from the oracle call.

(3) After the last oracle call, M executes N to the end. If N ends in acceptance, then M outputs 1; otherwise it outputs 0 as the final bit.

Conclusion of the Hardness Proof

It remains to show the following properties:

1. Correctness. Let w denote the lexicographically maximal output string over all computation paths of M on input x. Then the last bit of w has value 1 if and only if x is a positive instance of $\mathcal{P}$.
2. Polynomial time. The total length of each computation path of M on input x is polynomially bounded in $|x|$.
3. Logarithmically bounded output. If the number of oracle calls of TM N is logarithmically bounded in its input (i.e., problem $\mathcal{P}$ is in Θ_2P), then the size of the output produced by M on input x is also logarithmically bounded.

Correctness of the problem reduction

Claim. For every $i \geq 1$ and for every bit vector w_i of length i: w_i is a prefix of the lexicographically maximal output string over all computation paths of M on input x if and only if w_i encodes the correct answers of the first i oracle calls of TM N on input x,

i.e., for every $j \in \{1, \dots, i\}$:
– $w_i[j] = 1$ if the j-th call of the N_{SAT} oracle yields a "yes" answer and
– $w_i[j] = 0$ if the j-th call of the N_{SAT} oracle yields a "no" answer.

Proof of the Claim. By an easy induction on i.

Conclusion. Let m denote the number of oracle calls carried out by TM N on input x and let w denote the lexicographically maximal output of TM M over all its computation paths.

- Then w has length $m + 1$ such that the first m bits encode the correct answers of the oracle calls of TM N on input x.
- By the construction of M, we indeed have that the last bit of w is 1 (resp. 0), if and only if x is a positive (resp. negative) instance of $\mathcal{P}$.

Polynomial time

Suppose that N on input x with $|x| = n$ holds after $\leq p(n)$ steps and that N_{SAT} on input φ with $|\varphi| = m$ holds after $\leq q(m)$ steps for polynomials $p()$ and $q()$.

Then the total length of the computation of N counting also the computation steps of the oracle machine N_{SAT} is bounded by a polynomial $r(n)$ with $r(n) = O(p(n) * q(p(n)))$.

Observation. In our simulation of N with oracle N_{SAT} by a non-deterministic computation of M on input x, we possibly produce computation paths which N on input x can never reach, e.g.: if the correct answer of N_{SAT} on oracle input φ_1 is "yes", then the continuation of the simulation of N on input x on all computation paths where answer "no" on oracle input φ_1 is assumed, can never be reached by a computation of N (with oracle N_{SAT}) on input x.

Problem. How can we be sure that the upper bound $r(n)$ applies to every branch of the computation tree of M on input x?

Simple solution. Extend TM M by a counter such that M outputs 0 and halts if more than $r(n)$ steps have been executed.

Logarithmically bounded output

Suppose that $\mathcal{P}$ is an arbitrary problem in Θ_2P, i.e., TM N only has logarithmically many calls of the N_{SAT} oracle.

- Problem. Similarly to the polynomial-time bound of the computation of M on x (in the simulation of N with oracle N_{SAT} by a non-deterministic computation of M), we possibly produce computation paths which N on input x can never reach.
- We have to make sure that the size of the output on every computation path of M on input x is logarithmically bounded by adding a counter for the number of output bits.
- If the counter exceeds $\log |x|$, then M outputs 0 and halts.

SAT-Variants in Δ_2P

LEX-MAXIMAL MODEL SAT

INSTANCE: Boolean formula φ, order $(x_1, \dots, x_n)$ of the variables in φ.
QUESTION: Is x_n true in the lexicographically biggest model of φ?

LEX-MINIMAL MODEL SAT

INSTANCE: Boolean formula φ, order $(x_1, \dots, x_n)$ of the variables in φ.
QUESTION: Is x_n true in the lexicographically smallest model of φ?

WEIGHT-MAXIMAL MODEL SAT

INSTANCE: Boolean formula φ, vector $(w_1, \dots, w_n)$ of weights (i.e., positive integers) of the variables $(x_1, \dots, x_n)$ in φ and an atom x_i.
QUESTION: Is x_i true in a weight-maximal model of φ?

analogously: **WEIGHT-MINIMAL MODEL SAT**

SAT-Variants in Θ_2P

LogLEX-MAXIMAL MODEL SAT

INSTANCE: Boolean formula φ, order $(x_1, \ldots, x_n)$ on some of the variables in φ with $n \leq \log|\varphi|$.
QUESTION: Is x_n true in the lexicographically biggest bit vector $(b_1, \ldots, b_n)$ that can be extended to a model of φ?

LogLEX-MINIMAL MODEL SAT

INSTANCE: Boolean formula φ, order $(x_1, \ldots, x_n)$ on some of the variables in φ with $n \leq \log|\varphi|$.
QUESTION: Is x_n true in the lexicographically smallest bit vector $(b_1, \ldots, b_n)$ that can be extended to a model of φ?

CARD-MAXIMAL MODEL SAT

INSTANCE: Boolean formula φ and an atom z.
QUESTION: Is z true in a cardinality-maximal model of φ?

analogously: **CARD-MINIMAL MODEL SAT**

LEX-MAXIMAL MODEL SAT and LogLEX-MAXIMAL MODEL SAT

Theorem

LEX-MAXIMAL MODEL SAT *is* Δ_2P*-complete;*
LogLEX-MAXIMAL MODEL SAT *is* Θ_2P*-complete.*
Hardness holds even if formulas are restricted to 3-CNF.

Proof of Membership

Idea of a polynomial-time algorithm with NP-oracle.

- maintain a bit vector $(v_1, \ldots, v_n)$ of the lexicographically maximal (prefix of a possible) model of φ
- for i from 1 to n call an NP-oracle asking:

  Does there exist a model of φ, s.t. the truth values of the first $i-1$ variables $(x_1, \ldots, x_{i-1})$ are $(v_1, \ldots, v_{i-1})$ and variable x_i is set to 1?

  If the answer is "yes", then set $v_i = 1$; otherwise set $v_i = 0$.
- if $v_n = 1$, then stop with acceptance; otherwise stop with rejection.

Proof of Hardness

Reduction from NP-MAX resp. LogNP-MAX

Let $M; x$ be an arbitrary instance of **NP-MAX**(resp. **LogNP-MAX**), i.e.:

- M is a Turing machine running in non-deterministic polynomial time and producing a binary string as output;
- x is an input string to M.

Assumptions on the Turing machine M.

- every computation path of M on input x has length N with $N = p(|x|)$ for some polynomial $p(\cdot)$.
- M has two tapes, where tape 2 is the dedicated output tape.

Reduction from NP-MAX resp. LogNP-MAX

Analogously to the proof of the Cook-Levin Theorem, we construct a propositional formula φ over the following variables:

$symbol^{(i)}_{\sigma}[\tau, \pi]$ for $1 \leq i \leq 2$, $0 \leq \tau \leq N$, $0 \leq \pi \leq N$ and $\sigma \in \Sigma$.
Intuitive meaning: at instant τ of the computation, cell number π on tape i contains symbol σ.

$cursor^{(i)}[\tau, \pi]$ for $1 \leq i \leq 2$, $0 \leq \tau \leq N$ and $0 \leq \pi \leq N$.
Intuitive meaning: at instant τ, the cursor of tape i points to cell number π.

$state_s[\tau]$ for $0 \leq \tau \leq N$ and $s \in K$.
Intuitive meaning: at instant τ, the NTM T is in state s.

$\vec{x}, z$ additional variables $\vec{x} = (x_1, x'_1, x_2, x'_2, \ldots, x_m, x'_m)$ and z to encode the output string w and the last bit.

Form of the propositional formula φ

As in the proof of the Cook-Levin Theorem, formula φ consists of a big conjunction containing the following groups of conjuncts:

- initialization facts
- transition rules
- uniqueness constraints
- inertia rules
- Since this is a Turing machine producing output (and not a decision procedure), we have to drop the acceptance condition.

In addition, we now have a further conjunct to encode the output string w of length $\leq m$ and the value of the last bit of w.

Encoding of the output string and the last bit

$$\begin{aligned}\chi \quad = \quad & \bigwedge_{\pi=1}^{m} x_\pi \leftrightarrow \mathit{symbol}_1^{(2)}[N,\pi] \ \wedge \\ \wedge \quad & \bigwedge_{\pi=1}^{m} x'_\pi \leftrightarrow \neg \mathit{symbol}_\sqcup^{(2)}[N,\pi] \ \wedge \\ \wedge \quad & z \leftrightarrow \bigvee_{\pi=1}^{N} \Big(x_\pi \wedge \bigwedge_{\pi'=\pi+1}^{N} \neg x'_\pi \Big)\end{aligned}$$

- For every i, variable x_i is true in a model J of φ if the i-th bit of w (along the computation path of M corresponding to J) is 1.
- Consequently, truth value false of x_i can mean that, on termination of M, the symbol in the i-th cell of tape 2 is either 0 or $\sqcup$.
- Variable x'_i is used to distinguish these two cases: x'_i is false if and only if the symbol in the i-th cell of tape 2 is $\sqcup$.
- Variable z is true in model J of φ if and only if the last bit of w is 1.

Transformation into 3-CNF and ordering on the variables

- Finally, transform φ into φ^* in 3-CNF by a standard transformation (e.g., the Tseytin transformation).
- Let $\vec{y}$ denote the vector of variables $\mathit{symbol}_\sigma^{(i)}[\tau,\pi]$, $\mathit{cursor}^{(i)}[\tau,\pi]$, and $\mathit{state}_\tau[s]$ plus the additional variables introduced by the transformation into 3-CNF.
- Let the variables in $\vec{y}$ be arranged in arbitrary order with $\vec{y} = \{y_1,\dots,y_\ell\}$.
- In the reduction from **NP-MAX**, define the following order on all variables in φ^*: $x_1 > x'_1 > \cdots > x_m > x'_m > y_1 > \cdots > y_\ell > z$.
- In the reduction from **LogNP-MAX**, define an order only on part of the variables in φ^*, namely: $x_1 > x'_1 > \cdots > x_m > x'_m > z$, i.e., we ignore the variables in $\vec{y}$.

Correctness of the reduction from **NP-MAX** / **LogNP-MAX**

- Let w denote the lexicographically maximal output string produced by M on input x. We have to show: the last bit of w is 1 if and only if z is true in the lexicographically maximal model of φ^*.
- As in the proof of the Cook-Levin Theorem, every computation path of M corresponds to one or more models of φ^*.
- We construct a truth assignment J on the variables $\vec{x}$, $\vec{y}$, and z in φ^*:

  J restricted to $\vec{y}$ is chosen such that it is lexicographically maximal among all truth assignments on $\vec{y}$ corresponding to a computation path of M on input x and with output w.
- We claim that J is the lexicographically maximal model of φ^*.
- From this, the correctness of the reduction follows immediately.

  For the reduction from **LogNP-MAX**, we thus make use of the following observation: the truth assignments of J on $\vec{x}$ and z are uniquely defined by the output w of M (independently of $\vec{y}$ in J).

Conclusion of the correctness proof

It remains to show that J is the lexicographically maximal model of φ^*.

Suppose to the contrary that there exists a bigger model J' of φ^*. We distinguish 3 cases according to the group of variables where J' is bigger:

1. If J' is bigger than J on $\vec{x}$, then (since the truth values of $\vec{x}$ encode the output string of some computation of M on x) there exists a bigger output than w. This contradicts our assumption that w is maximal.
2. If J' coincides with J on $\vec{x}$ and J' is bigger than J on $\vec{y}$, then J' restricted to $\vec{y}$ corresponds to a computation path producing the same output string w as the computation path encoded by J on $\vec{x}$. This contradicts our choice of truth assignment J on $\vec{y}$.
3. The truth value of z in any model of φ^* is uniquely determined by the truth value of $\vec{x}$. Hence, it cannot happen that J' and J coincide on $\vec{x}$ but differ on z.

WEIGHT-MAXIMAL MODEL SAT is Δ_2P-complete

Idea of Δ_2P-hardness proof

- By reduction from **LEX-MAXIMAL MODEL SAT**:
  Consider an arbitrary instance $\varphi; (x_1, \dots, x_n)$, where φ is a Boolean formula over variables X and $(x_1, \dots, x_n)$ is an ordering of the variables in X.
- We define the instance $\varphi; (x_1, \dots, x_n); (w(x_1), \dots, w(x_n)); z$ of **WEIGHT-MAXIMAL MODEL SAT** as follows:
  - Formula φ (and, hence, also variable set X) is left unchanged.
  - For every $i \in \{1, \dots, n\}$, we define the weight $w(x_i) = 2^{n-i}$.
  - We set $z = x_n$.
- Idea. Simulate lexicographical ordering by weights of variables x_i: $w(x_i)$ is greater than the sum of weights $w(x_{i+1}) + \cdots + w(x_n)$.

CARD-MAXIMAL MODEL SAT is Θ_2P-complete

Idea of Θ_2P-hardness proof

- By reduction from **LogLEX-MAXIMAL MODEL SAT**:
  consider an arbitrary instance $\varphi; (x_1, \dots, x_n)$ with a Boolean formula φ and an ordering $(x_1, \dots, x_n)$ of logarithmically many variables in φ; let $Y = \{y_1, \dots, y_m\}$ denote the remaining variables in φ.
- To define an instance $\psi; z$ of **CARD-MAXIMAL MODEL SAT**, we add the following fresh variables:
  - "copies" of each variable x_i, i.e., for every $i \in \{1, \dots, n\}$, we introduce $2^{n-i} - 1$ new variables $x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(r_i)}$ with $r_i = 2^{\ell-i} - 1$.
  - a primed copy of each variable in Y: $Y' = \{y_1', \dots, y_m'\}$.

Instance of CARD-MAXIMAL MODEL SAT

Formula ψ and distinguished variable z.

$$
\begin{aligned}
\psi &= \varphi \land \\
&\land \bigwedge_{i=1}^{n} \left((x_i \leftrightarrow x_i^{(1)}) \land \cdots \land (x_i \leftrightarrow x_i^{(r_i)}) \right) \land \\
&\land \bigwedge_{i=1}^{m} y_i \leftrightarrow \neg y_i' \\
z &= x_n
\end{aligned}
$$

Idea.

- Simulate weight of x_i by "copies" $x_i^{(j)}$, which are forced to have identical truth values as x_i in every model of ψ.
- Use the primed variables y_i' to encode the dual of y_i. They thus make the cardinality of models of ψ indistinguishable on $Y \cup Y'$.

CARD-MINIMAL MODEL SAT is Θ_2P-complete

Idea of Θ_2P-hardness proof

- By reduction from **CARD-MAXIMAL MODEL SAT**:
  consider an arbitrary instance $\varphi; x_i$,
  where φ is a Boolean formula over the variables $X = \{x_1, \ldots, x_n\}$.
- Add primed and double-primed copies of the variables, i.e.,
  $X' = \{x'_1, \ldots, x'_n\}$ and $X'' = \{x''_1, \ldots, x''_n\}$.
- Define instance $\psi; x_i$ of **CARD-MINIMAL MODEL SAT** with

$$\psi = \varphi \wedge \bigwedge_{i=1}^{n} \left((x_i \leftrightarrow \neg x'_i) \wedge (x_i \leftrightarrow \neg x''_i) \right)$$

- Idea. The cardinality-minimal models of ψ restricted to the variables X are precisely the cardinality-maximal models of φ.

Learning Objectives

- Oracle machines
- Complexity classes: DP, Δ_iP, Σ_iP, Π_iP, PH
- The intuition of these classes and complete problems
- Restrictions on the oracle calls:
  $\Delta_2\mathrm{P}[\log n] = \mathrm{P}^{\mathrm{NP}[\log n]} = \mathrm{P}^{\mathrm{NP}}_{\|}$
- oracle calls vs. optimization
- Problem reductions in Σ_2P
- Properties of PH (sufficient conditions for PH to collapse)
- Characterization of Σ_iP and Π_iP via certificates
- The power of alternation: limited alternation in QSAT$_i$
  The classes Δ_2P and Δ_2P[log n]