Complexity Theory
The Polynomial Hierarchy

Daniel Borchmann, Markus Krötzsch

Computational Logic

2016-01-12

The Polynomial Hierarchy

Bounding Alternation

For ATMs, alternation itself is a resource. We can distinguish problems by how much alternation they need to be solved.

We first classify computations by counting their quantifier alternations:

Definition 16.1

Let $\mathcal{P}$ be a computation path of an ATM on some input.

- $\mathcal{P}$ is of type Σ_1 if it consists only of existential configurations (with the exception of the final configuration)
- $\mathcal{P}$ is of type Π_1 if it consists only of universal configurations
- $\mathcal{P}$ is of type Σ_{i+1} if it starts with a sequence of existential configurations, followed by a path of type Π_i
- $\mathcal{P}$ is of type Π_{i+1} if it starts with a sequence of universal configurations, followed by a path of type Σ_i

Alternation-Bounded ATMs

We apply alternation bounds to every computation path:

Definition 16.2

A Σ_i Alternating Turing Machine is an ATM for which every computation path on every input is of type Σ_j for some $j \leq i$.
A Π_i Alternating Turing Machine is an ATM for which every computation path on every input is of type Π_j for some $j \leq i$.

Note that it's always ok to use fewer alternations ("$j \leq i$") but computation has to start with the right kind of quantifier ($\exists$ for Σ_i and $\forall$ for Π_i).

Example 16.3

A Σ_1 ATM is simply an NTM.

Alternation-Bounded Complexity

We are interested in the power of ATMs that are both time/space-bounded and alternation-bounded:

Definition 16.4

Let $f : \mathbb{N} \to \mathbb{R}^+$ be a function. $\Sigma_i\textsc{Time}(f(n))$ is the class of all languages that are decided by some $O(f(n))$-time bounded Σ_i ATM. The classes $\Pi_i\textsc{Time}(f(n))$, $\Sigma_i\textsc{Space}(f(n))$ and $\Pi_i\textsc{Space}(f(n))$ are defined similarly.

The most popular classes of these problems are the alternation-bounded polynomial time classes:

$$\Sigma_i\mathrm{P} = \bigcup_{d\geq 1} \Sigma_i\textsc{Time}(n^d) \qquad \text{and} \qquad \Pi_i\mathrm{P} = \bigcup_{d\geq 1} \Pi_i\textsc{Time}(n^d)$$

Hardness for these classes is defined by polynomial many-one reductions as usual.

Basic Observations

Theorem 16.5

$\Sigma_1\mathrm{P} = \mathrm{NP}$ *and* $\Pi_1\mathrm{P} = \mathrm{coNP}$.

Proof.

Immediate from the definitions. □

Theorem 16.6

$\mathrm{co}\Sigma_i\mathrm{P} = \Pi_i\mathrm{P}$ *and* $\mathrm{co}\Pi_i\mathrm{P} = \Sigma_i\mathrm{P}$.

Proof.

We observed previously that ATMs can be complemented by simply exchanging their universal and existential states. This does not affect the amount of time or space needed. □

Example

MinFormula

Input: A propositional formula φ.

Problem: Is φ the shortest formula that is satisfied by the same assignments as φ?

One can show that MinFormula is $\Pi_2\mathrm{P}$-complete. Inclusion is easy:

```
MinFormula(formula φ) :
  universally choose ψ := formula shorter than φ
  exist. guess I := assignment for variables in φ
  if φ^I = ψ^I :
    return FALSE
  else :
    return TRUE
```


The Polynomial Hierarchy

Like for NP and coNP, we do not know if $\Sigma_i\mathrm{P}$ equals $\Pi_i\mathrm{P}$ or not. What we do know, however, is this:

Theorem 16.7

- $\Sigma_i\mathrm{P} \subseteq \Sigma_{i+1}\mathrm{P}$ *and* $\Sigma_i\mathrm{P} \subseteq \Pi_{i+1}\mathrm{P}$
- $\Pi_i\mathrm{P} \subseteq \Pi_{i+1}\mathrm{P}$ *and* $\Pi_i\mathrm{P} \subseteq \Sigma_{i+1}\mathrm{P}$

Proof.

Immediate from the definitions. □

Thus, the classes $\Sigma_i\mathrm{P}$ and $\Pi_i\mathrm{P}$ form a kind of hierarchy: the Polynomial (Time) Hierarchy. Its entirety is denoted PH:

$$\mathrm{PH} := \bigcup_{i\geq 1} \Sigma_i\mathrm{P} = \bigcup_{i\geq 1} \Pi_i\mathrm{P}$$

Problems in the Polynomial Hierarchy

The "typical" problems in the Polynomial Hierarchy are restricted forms of True QBF:

> **True Σ_kQBF**
>
> *Input:* A quantified Boolean formula of the form $\varphi = \exists X_1.\forall X_2.\cdots \mathsf{Q}_k.\psi$.
>
> *Problem:* Is φ true?

True Π_kQBF is defined analogously, using $\varphi = \forall X_1.\exists X_2.\cdots \mathsf{Q}_k.\psi$.

Theorem 16.8

For every k, True Σ_kQBF is $\Sigma_k\mathrm{P}$-complete and True Π_kQBF is $\Pi_k\mathrm{P}$-complete.

Note: It is not known if there is any PH-complete problem.

Alternative Views on the Polynomial Hierarchy

Certificates

For NP, we gave an alternative definition based on polynomial-time verifiers that use a given polynomial certificate (witness) to check acceptance. Can we extend this idea to alternation-bounded ATMs?

Notation: Given an input word w and a polynomial p, we write $\exists^p c$ as abbreviation for "there is a word c of length $|c| \leq p(|w|)$." Similarly for $\forall^p c$.

We can rephrase our earlier characterisation of polynomial-time verifiers:

$\mathcal{L} \in \mathrm{NP}$ iff there is a polynomial p and language $\mathcal{V} \in \mathrm{P}$ such that

$$\mathcal{L} = \{w \mid \exists^p c \text{ such that } (w\#c) \in \mathcal{V}\}$$

Certificates for bounded ATMs

Theorem 16.9

$\mathcal{L} \in \Sigma_k\mathrm{P}$ iff there is a polynomial p and language $\mathcal{V} \in \mathrm{P}$ such that

$$\mathcal{L} = \{w \mid \exists^p c_1.\forall^p c_2 \ldots \mathsf{Q}^p_k c_k \text{ such that } (w\#c_1\#c_2\#\ldots\#c_k) \in \mathcal{V}\}$$

where $\mathsf{Q}_k = \exists$ if k is odd, and $\mathsf{Q}_k = \forall$ if k is even.

An analoguous result holds for $\mathcal{L} \in \Pi_k\mathrm{P}$.

Proof sketch.

$\Rightarrow$: Similar as for NP. Use c_i to encode the non-deterministic choices of the ATM. With all choices given, the acceptance on the specified path can be checked in polynomial time.

$\Leftarrow$: Use an ATM to implement the certificate-based definition of $\mathcal{L}$, by using universal and existential choices to guess the certificate before running a polynomial time verifier. $\square$

Oracles (Revision)

Recall how we defined oracle TMs:

Definition 16.10

Let $\mathcal{O}$ be a language. An Oracle Turing Machine with oracle $\mathcal{O}$ is a TM with a special oracle tape and three special states $q_?, q_{yes}, q_{no}$. Whenever the state $q_?$ is reached, the TM changes to state q_{yes} if the word on the oracle tape is in $\mathcal{O}$ and to q_{no} otherwise; and the oracle tape is cleared.

Let C be a complexity class:

- For a language $\mathcal{O}$, we write $\mathrm{C}^{\mathcal{O}}$ for the class of all problems that can be solved by a C-TM with oracle $\mathcal{O}$.
- For a complexity class O, we write C^{O} for the class of all problems that can be solved by a C-TM with an oracle from class O.

The Polynomial Hierarchy – Alternative Definition

We recursively define the following complexity classes:

Definition 16.11

- $\Sigma^{\mathrm{P}}_0 := \mathrm{P}$ and $\Sigma^{\mathrm{P}}_{k+1} := \mathrm{NP}^{\Sigma^{\mathrm{P}}_k}$
- $\Pi^{\mathrm{P}}_0 := \mathrm{P}$ and $\Pi^{\mathrm{P}}_{k+1} := \mathrm{coNP}^{\Pi^{\mathrm{P}}_k}$

Remark:

Complementing an oracle (language/class) does not change expressivity: we can just swap states q_{yes} and q_{no}. Therefore $\Sigma^{\mathrm{P}}_{k+1} = \mathrm{NP}^{\Pi^{\mathrm{P}}_k}$ and $\Pi^{\mathrm{P}}_{k+1} := \mathrm{coNP}^{\Sigma^{\mathrm{P}}_k}$. Hence, we can also see that $\Sigma^{\mathrm{P}}_k = \mathrm{co}\Pi^{\mathrm{P}}_k$.

Question:

How do these relate to our earlier definitions of the PH classes?

Oracle TMs vs. ATMs

It turns out that this new definition leads to a familiar class of problems:[1]

Theorem 16.12

For $k \geq 1$, we have $\Sigma^{\mathrm{P}}_k = \Sigma_k\mathrm{P}$ and $\Pi^{\mathrm{P}}_k = \Pi_k\mathrm{P}$.

Proof.

We prove the case $\Sigma^{\mathrm{P}}_k = \Sigma_k\mathrm{P}$ (the other follows by complementation). The proof is by induction on k.

Base case: $k = 1$.
The claim follows since $\Sigma^{\mathrm{P}}_1 = \mathrm{NP}^{\mathrm{P}} = \mathrm{NP}$ and $\Sigma_1\mathrm{P} = \mathrm{NP}$ (as noted before).

[1] Because of this result, both of our notations are used interchangeably in the literature, independently of the definition used.

Oracle TMs vs. ATMs (2)

Induction step: assume the claim holds for k. We show $\Sigma^{\mathrm{P}}_{k+1} = \Sigma_{k+1}\mathrm{P}$.

"$\supseteq$" Assume $\mathcal{L} \in \Sigma_{k+1}\mathrm{P}$.

- By Theorem 16.9, for some language $\mathcal{V} \in \mathrm{P}$ and polynomial p:
  $\mathcal{L} = \{w \mid \exists^p c_1.\forall^p c_2 \ldots \mathcal{Q}^p_k c_k \text{ such that } (w\#c_1\#c_2\#\ldots\#c_k) \in \mathcal{V}\}$
- By Theorem 16.9, the following defines a language in $\Pi_k\mathrm{P}$:
  $\mathcal{L}' := \{(w\#c_1) \mid \forall^p c_2 \ldots \mathcal{Q}^p_k c_k \text{ such that } (w\#c_1\#c_2\#\ldots\#c_k) \in \mathcal{V}\}$.
- The following algorithm in $\mathrm{NP}^{\mathcal{L}'}$ decides $\mathcal{L}$:
  on input w, non-deterministically guess c_1; then check $(w\#c_1) \in \mathcal{L}'$ using the $\mathcal{L}'$ oracle
- By induction, $\mathcal{L}' \in \Pi^{\mathrm{P}}_k$. Hence, the algorithm runs in
  $\mathrm{NP}^{\Pi^{\mathrm{P}}_k} = \mathrm{NP}^{\Sigma^{\mathrm{P}}_k} = \Sigma^{\mathrm{P}}_{k+1}$

Oracle TMs vs. ATMs (3)

Induction step: assume the claim holds for k. We show $\Sigma_{k+1}^{\mathrm{P}} = \Sigma_{k+1}\mathrm{P}$.

"$\subseteq$" Assume $\mathcal{L} \in \Sigma_{k+1}^{\mathrm{P}}$.

- There is an $\Sigma_{k+1}^{\mathrm{P}}$-TM $\mathcal{M}$ that accepts $\mathcal{L}$, using an oracle $O \in \Sigma_k^{\mathrm{P}}$.
- By induction, $O \in \Sigma_k\mathrm{P}$ and thus $\bar{O} \in \Pi_k\mathrm{P}$ for its complement
- For an $\Sigma_{k+1}\mathrm{P}$ algorithm, first guess (and verify) an accepting path of $\mathcal{M}$ including results of all oracle queries.
- Then universally branch to verify all guessed oracle queries:
  - For queries $w \in O$ with guessed answer "no", use $\Pi_k\mathrm{P}$ check for $w \in \bar{O}$
  - For queries $w \in O$ with guessed answer "yes", use $\Pi_{k-1}\mathrm{P}$ check for $(w\#c_1) \in O'$, where O' is constructed as in the $\supseteq$-case, and c_1 is guessed in the first $\exists$-phase

□

More Classes in PH

We defined Σ_k^{P} and Π_k^{P} by relativising NP and coNP with oracles.

What happens if we start from P instead?

Definition 16.13

$\Delta_0^{\mathrm{P}} := \mathrm{P}$ and $\Delta_{k+1}^{\mathrm{P}} := \mathrm{P}^{\Sigma_k^{\mathrm{P}}}$.

Some immediate observations:

- $\Delta_1^{\mathrm{P}} = \mathrm{P}^{\mathrm{P}} = \mathrm{P}$
- $\Delta_2^{\mathrm{P}} = \mathrm{P}^{\mathrm{NP}} = \mathrm{P}^{\mathrm{coNP}}$
- $\Delta_k^{\mathrm{P}} \subseteq \Sigma_k^{\mathrm{P}}$ (since $\mathrm{P} \subseteq \mathrm{NP}$) and $\Delta_k^{\mathrm{P}} \subseteq \Pi_k^{\mathrm{P}}$ (since $\mathrm{P} \subseteq \mathrm{coNP}$)
- $\Sigma_k^{\mathrm{P}} \subseteq \Delta_{k+1}^{\mathrm{P}}$ and $\Pi_k^{\mathrm{P}} \subseteq \Delta_{k+1}^{\mathrm{P}}$

Problems for Δ_k^{P}?

Δ_k^{P} seems to be less common in practice, but there are some known complete problems for $\mathrm{P}^{\mathrm{NP}} = \Delta_2^{\mathrm{P}}$:

Uniquely Optimal TSP [Papadimitriou, JACM 1984]

Input: Undirected graph G with edge weights (distances).

Problem: Is there exactly one shortest travelling salesman tour on G?

Divisible TSP [Krentel, JCSS 1988]

Input: Undirected graph G with edge weights; number k.

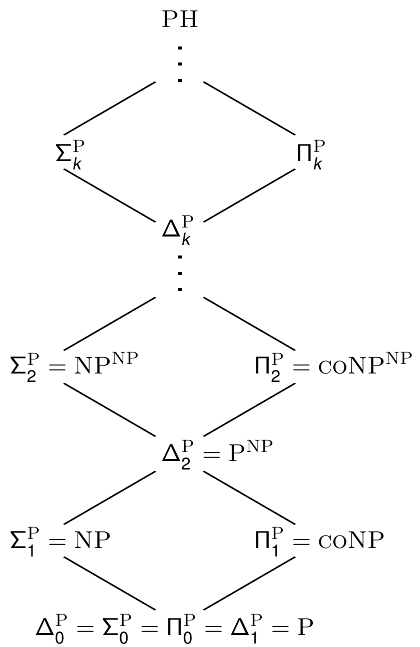
Problem: Is the shortest travelling salesman tour on G divisible by k?

Odd Final SAT [Krentel, JCSS 1988]

Input: Propositional formula φ with n variables.

Problem: Is X_n true in the lexicographically last assignment satisfying φ?

Is the Polynomial Hierarchy Real?



Questions:

Are all of these classes really distinct?
Nobody knows

Are any of these classes really distinct?
Nobody knows

Are any of these classes distinct from P?
Nobody knows

Are any of these classes distinct from PSPACE?
Nobody knows

What do we know then?

What We Know (Excerpt)

Theorem 16.14

If there is any k such that $\Sigma_k^{\mathrm{P}} = \Sigma_{k+1}^{\mathrm{P}}$ then $\Sigma_j^{\mathrm{P}} = \Pi_j^{\mathrm{P}} = \Sigma_k^{\mathrm{P}}$ for all $j > k$, and therefore $\mathrm{PH} = \Sigma_k^{\mathrm{P}}$.
In this case, we say that the polynomial hierarchy collapses at level k.

Proof.

Left as exercise (not too hard to get from definitions). □

Corollary 16.15

If $\mathrm{PH} \neq \mathrm{P}$ then $\mathrm{NP} \neq \mathrm{P}$.

Intuitively speaking: "The polynomial hierarchy is built upon the assumption that NP has some additional power over P. If this is not the case, the whole hierarchy collapses."

What We Know (Excerpt)

Theorem 16.16

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

Proof.

Left as exercise (induction over PH levels, using that $\mathrm{PSPACE}^{\mathrm{PSPACE}} = \mathrm{PSPACE}$). □

Theorem 16.17

If $\mathrm{PH} = \mathrm{PSPACE}$ then there is some k with $\mathrm{PH} = \Sigma_k^{\mathrm{P}}$.

Proof.

If $\mathrm{PH} = \mathrm{PSPACE}$ then TrueQBF $\in \mathrm{PH}$. Hence TrueQBF $\in \Sigma_k^{\mathrm{P}}$ for some k. Since TrueQBF is PSPACE-hard, this implies $\Sigma_k^{\mathrm{P}} = \mathrm{PSPACE}$. □

What We Believe (Excerpt)

"Most experts" think that

- The polynomial hierarchy does not collapse completely (same as $\mathrm{P} \neq \mathrm{NP}$)
- The polynomial hierarchy does not collapse on any level (in particular $\mathrm{PH} \neq \mathrm{PSPACE}$ and there is no PH-complete problem)

But there can always be surprises ...