MTAT.07.004 — Complexity Theory October 20th-21st, 2011

Lecture 7

Lecturer: Peeter Laud *Scribe(s): Riivo Talviste*

Polynomial hierarchy

1 Turing reducibility

From the algorithmics course, we know the notion of Turing reducibility, which states that if you have some problem that you cannot solve, you can try to generalize it to another problem that you might be able to solve.

Definition 1 (Turing reducibility) Let $A, B \subseteq \{0,1\}^*$. A is Turing-reducible to B, denoted $A \leq_{\mathrm{T}} B$, if there is a oracle TM $M^{(\cdot)}$, such that M^B recognizes A. ♢

I.e. $M^B(x)$ stops for all $x \in \{0,1\}^*$ and $M^B(x) = \mathsf{true} \Leftrightarrow x \in A$.

Similarily, we have polynomial-time and non-deterministic polynomial-time Turing reducibility.

Definition 2 (Polynomial-time Turing reducibility) Let $A, B \subseteq \{0,1\}^*$. A is Polynomial-time Turing-reducible to B, denoted $A \leq_{\mathrm{T}}^{\mathrm{P}} B$, if there is a oracle TM $M^{(\cdot)}$, such that M^B recognizes A in polynomial time. ♢

Definition 3 (Non-deterministic polynomial-time Turing reducibility) Let $A, B \subseteq \{0,1\}^*$. A is non-deterministic polynomial-time Turing-reducible to B, denoted $A \leq_{\mathrm{T}}^{\mathrm{NP}} B$, if there is a oracle NTM $M^{(\cdot)}$, such that M^B recognizes A in polynomial time. ♢

For example, $\mathsf{SAT}^{\mathrm{c}} \leq_{\mathrm{T}}^{\mathrm{P}} \mathsf{SAT}$, as we can just flip the answer.

1.1 Recursive hierarchy

Let $M_1, M_2, \ldots$ be an enumeration of all oracle turing machines.

Definition 4 Languages $A, B \subseteq \{0,1\}^*$ are *Turing equivalent* if $A \leq_{\mathrm{T}} B$ and $B \leq_{\mathrm{T}} A$. Denote $A \equiv_{\mathrm{T}} B$. ♢

Let $[A]$ be the equivalence class of $\equiv_{\mathrm{T}}$ containing A.

Definition 5 The *Turing jump* of a language A is

$$A' = \{i \mid M_i^A(i) \text{ stops }\}.$$

♢

The Turing jump can also be generalized to sets of languages.

Theorem 1 $A' \not\leq_{\mathrm{T}} A$.

Using Turing jump and equivalence classes, we can define an infinite hierarchy:

$$\text{Denote } \Sigma_0 = [\emptyset]; \quad \Sigma_i = \Sigma_{i-1} \cup \Sigma'_{i-1}.$$

2 Polynomial hierarchy

2.1 Exact problems

Consider the following problems:

- Given a graph G and an integer k. Does the largest clique of G have the size *exactly* k?
  - **Remark.** We know that finding a clique of size at least k is in NP, but showing that there are no larger cliques (with size $k+1$) seems to be in coNP.
- Given a propositional formula φ. Does there exist any smaller formula φ', such that $\varphi \equiv \varphi'$?
- Given a set $\varphi_1, \ldots, \varphi_m$ of formulas in $\mathcal{CNF}$ and a number k. Do there exist $i_1, \ldots, i_k$, such that $\varphi_{i_1} \wedge \ldots \wedge \varphi_{i_k}$ is unsatisfiable?

All of these problems seem not to be in NP, as short certificates seem hard to find. However, all on them are in PSPACE.

2.2 Classes Σ_2^p and Π_2^p

Definition 6 A language $L \subseteq \{0,1\}^*$ is in Σ_2^p, if there is a polynomial-time DTM M and a polynomial q, such that

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

♢

Definition 7 A language $L \subseteq \{0,1\}^*$ is in Π_2^p, if there is a polynomial-time DTM M and a polynomial q, such that

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

♢

Clearly, $\Sigma_2^p = \mathsf{co}\Pi_2^p$ and vice versa, as they are each other's negations (we also have to flip the answer of M). Also,

$$\mathsf{NP}, \mathsf{coNP} \subseteq \Sigma_2^p \cap \Pi_2^p.$$

Now, we provide more general definitions.

Definition 8 A language $L \subseteq \{0,1\}^*$ is in Σ_k^p, if there is a polynomial-time DTM M and a polynomial q, such that

$$x \in L \Leftrightarrow \underbrace{\exists v_1 \in \{0,1\}^{q(|x|)}\, \forall v_2 \in \{0,1\}^{q(|x|)} \ldots Q_k v_k \in \{0,1\}^{q(|x|)}}_{k \text{ quantifiers}} : M(x, v_1, \ldots, v_k) = \mathsf{true}.$$

Here Q_k is $\forall$ if k is even, and $\exists$ otherwise. ♢

Definition 9 A language $L \subseteq \{0,1\}^*$ is in Π_k^p, if there is a polynomial-time DTM M and a polynomial q, such that

$$x \in L \Leftrightarrow \underbrace{\forall v_1 \in \{0,1\}^{q(|x|)}\, \exists v_2 \in \{0,1\}^{q(|x|)}\, \ldots\, Q_k v_k \in \{0,1\}^{q(|x|)}}_{k \text{ quantifiers}} : M(x, v_1, \ldots, v_k) = \mathsf{true}.$$

Here Q_k is $\forall$ if k is odd, and $\exists$ otherwise. ♢

By definition,

$$\mathsf{NP} = \Sigma_1^p \quad \text{and} \quad \mathsf{coNP} = \Pi_1^p.$$

Also, $M(x, v_1, \ldots, v_k)$ is working in polynomial time and thus cannot use more than polynomial amount of memory. As $v_1, \ldots, v_k$ are all polynomial size, then

$$\Sigma_k^p, \Pi_k^p \subseteq \mathsf{PSPACE}.$$

With these definitions, we can formulate polynomial hierarchy:

$$\mathsf{PH} = \bigcup_{i \in \mathbb{N}} \Sigma_i^p.$$

As $\Sigma_i^p \subseteq \Pi_{i+1}^p$, we also have $\mathsf{PH} = \bigcup_{i \in \mathbb{N}} \Pi_i^p$.

From the generalization of $\mathsf{P} \neq \mathsf{NP}$ and $\mathsf{NP} \neq \mathsf{coNP}$, we get the following conjecture.

Conjecture 2 *Polynomial hierarchy does not collapse.*

In other words, there is belief that for all i, $\Sigma_i^p \neq \Sigma_{i+1}^p$ (actually it is a separate conjecture for each i).

2.3 Complete problems for Σ_i^p and Π_i^p

Prevously, we found complete problems for many complecity classes: NP, PSPACE, NL. Thus we are also interested in Σ_i^p-complete and Π_i^p-complete problems. As we are dealing with problems that are somewhere between NP and PSPACE, we will still use the polynomial-time reduction $\leq_{\mathrm{m}}^{\mathrm{P}}$.

Definition 10

$$\Sigma_i\mathsf{SAT} = \{\exists u_1 \forall u_2 \ldots Q_i u_i : \varphi(u_1, u_2, \ldots, u_i) = \mathsf{true}\},$$
$$\Pi_i\mathsf{SAT} = \{\forall u_1 \exists u_2 \ldots Q_i u_i : \varphi(u_1, u_2, \ldots, u_i) = \mathsf{true}\},$$

where

- φ is a Boolean formula;
- $u_1, \ldots, u_i$ are vectors of Boolean variables;
- quantifications ($\forall$ and $\exists$) are alternating.

♢

Theorem 3

$$\Sigma_i\mathsf{SAT} \textit{ is } \Sigma_i^p\textit{-complete.} \quad \Pi_i\mathsf{SAT} \textit{is } \Pi_i^p\textit{-complete.}$$

This theorem is a special case of the TQBF-completeness problem introduced in previous lectures.

2.4 Defining Σ_i^p and Π_i^p through oracle TMs

Theorem 4

$$\Sigma_i^p = \mathsf{NP}^{\Sigma_{i-1}\mathsf{SAT}}. \quad \Pi_i^p = \mathsf{co}\Sigma_i^p.$$

Proof In the proof, we use k instead of i. To prove $\Sigma_k^p = \mathsf{NP}^{\Sigma_{k-1}\mathsf{SAT}}$ it is sufficient to show that $\Sigma_k^p \subseteq \mathsf{NP}^{\Sigma_{k-1}\mathsf{SAT}}$ and $\Sigma_k^p \supseteq \mathsf{NP}^{\Sigma_{k-1}\mathsf{SAT}}$.

First, we show that $\Sigma_k^p \subseteq \mathsf{NP}^{\Sigma_{k-1}\mathsf{SAT}}$ and to make it less abstract in the beginning, we will start with a specific case, where $k = 2$. Hence, we show that

$$\Sigma_2^p \subseteq \mathsf{NP}^{\Sigma_1\mathsf{SAT}} = \mathsf{NP}^{\mathsf{SAT}}.$$

If $L \in \Sigma_2^p$ then

$$\exists M : x \in L \Leftrightarrow \exists u \forall v : M(x, u, v) = \mathsf{true}.$$

We want to construct a NTM M'^{SAT} that accepts the same language L. We start by replacing the clause $\exists u$ with just non-deterministically choosing the right u. We get the following construction for M'^{SAT}:

$M'^{\mathsf{SAT}}(x)$:
  $u :=$ **choose** -- non-deterministic
  $b := \forall v : M(x, u, v) \stackrel{?}{=} \mathsf{true}$
  return b

Now we need the oracle SAT to provide the value for b. However, the SAT problem is commonly phrased more like "$\exists u : M(x, u, v) = \mathsf{true}$", that is with the $\exists$ quantifier. So we just negate b and get

$$\neg b = \neg \forall v : M(x, u, v) \stackrel{?}{=} \mathsf{true} \equiv \exists v : M(x, u, v) \stackrel{?}{=} \mathsf{false}.$$

To be able to use the SAT oracle, we have to raplace false with true. We can do that by defining a new TM $\tilde{M}$ that just flips the answer of M. Hence, $\tilde{M}$ accepts the language

$$\tilde{L} = \{(x, u) \,|\, \exists v : M(x, u, v) = \mathsf{true}\}.$$

Language $\tilde{L}$ belongs to NP. Hence there exists a function f that many-one reduces it to SAT in polynomial time:

$$f(x, u) \in \mathsf{SAT} \Leftrightarrow (x, u) \in \tilde{L}.$$

Finally, the constructed Turing machine works like this:

$M'^{\mathsf{SAT}}(x)$:
  $u :=$ **choose** -- non-deterministic
  $b :=$ **query** oracle SAT with $f(x, u)$
  return $\neg b$

Now, for the general case, we show that

$$\Sigma_k^p \subseteq \mathsf{NP}^{\Sigma_{k-1}\mathsf{SAT}}.$$

If $L \in \Sigma_k^p$ then

$$\exists M : x \in L \Leftrightarrow \exists u_1 \forall u_2 \exists u_3 \dots Q_k u_k : M(x, u_1, \dots, u_k) = \mathsf{true}.$$

Again, we want to construct a NTM $M'^{\Sigma_{k-1}\mathsf{SAT}}$ that accepts the same language L. Just like before, we define a new TM $\tilde{M} = \neg M$, that accepts the language

$$\tilde{L} = \{(x, u_1) \,|\, \exists u_2 \forall u_3 \dots \bar{Q_k} u_k : M(x, u_1, \dots, u_k) = \mathsf{true}\},$$

where $\bar{Q_k}$ is $\exists$ if Q_k is $\forall$ and vice versa.
We also have a reduction

$$f : \{\{0,1\}^*\}^2 \to Q_{k-1}\mathsf{Frm},$$

where $Q_{k-1}\mathsf{Frm}$ stands for a Boolean formula with $k-1$ quantifiers, and

$$f(x, u) \in \Sigma_{k-1}\mathsf{SAT} \Leftrightarrow (x, u) \in \tilde{L}.$$

And the construction of $M'^{\Sigma_{k-1}\mathsf{SAT}}$ is the following

```
M'^{Σ_{k-1}SAT}(x) :
    u_1 := choose      -- non-deterministic
    b := query oracle Σ_{k-1}SAT with f(x, u_1)
    return ¬b
```

Secondly, we show that $\Sigma_k^p \supseteq \mathsf{NP}^{\Sigma_{k-1}\mathsf{SAT}}$. As before, we will start with a simpler and more intuitive case, where $k = 2$:

$$\mathsf{NP}^{\Sigma_1\mathsf{SAT}} = \mathsf{NP}^{\mathsf{SAT}} \subseteq \Sigma_2^p.$$

In this case we have

$$\exists \text{ NTM } M^{(\cdot)} : x \in L \Leftrightarrow M^{\mathsf{SAT}}(x) = \mathsf{true}$$

and we want to show that $L \in \Sigma_2^p$. For that, we construct a deterministic TM M' working in polynomial time, such that

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

where q is a polynomial.

Since M^{SAT} is a non-deterministic TM, we can think of its work as traversing a tree, with at least one leaf containing `accept`. While traversing the tree, M^{SAT} can make any number of queries to its SAT oracle. We will encode all that in u and v for out DTM M'.

First, we can take the accepting computatipn path of M^{SAT} and encode all of the decision points (branchings in the tree) in u. Also, we encode in u all the oracle queries (Boolean formulas φ) and all the satisfying valuations (certificates) for all the queries, that got the yes-reply. Secondly, v with the $\forall$ quantifier is able to hold all the possible valuations for all of the no-replies from the oracle ($\tilde{q}$ is a polynomial):

$$v \equiv v_1 | v_2 | \dots | v_{\tilde{q}(|x|)}.$$

Now, we can construct our DTM M' in the following way:

$M'(x, u, v)$:
Simulate $M^{(\cdot)}(x)$:
When
- $M^{(\cdot)}$ makes a non-deterministic step
  - get the "right" choice from u;
- $M^{(\cdot)}$ queries the oracle and the reply would be yes
  - u has the corresponding query (φ) and the certificate,
  so we can check it in polynomial time;
- $M^{(\cdot)}$ queries the oracle and the reply would be no
  - u has the corresponding query and v encodes all the valuations,
  so check that $\forall i : \varphi(v_i) = \mathsf{false}$.

return output of simulated $M^{(\cdot)}(x)$

Now, for the general case, we show that

$$\mathsf{NP}^{\Sigma_{k-1}\mathsf{SAT}} \subseteq \Sigma_k^p.$$

In this case we have

$$\exists \text{ NTM } M^{(\cdot)} : x \in L \Leftrightarrow M^{\Sigma_{k-1}\mathsf{SAT}}(x) = \mathsf{true}$$

and we want to show that $L \in \Sigma_k^p$. This means that

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{q(|x|)} : (x, u) \in \hat{L},$$

where q is a polynomial and $\hat{L} \in \Pi_{k-1}^p$ (we get Π_{k-1}^p by just removing the first $\exists$ quantifier in Σ_k^p). Now we just have to construct a TM that accepts $\hat{L}$.

Like before, let u encode all the decision points of the accepting computation path of $M^{\Sigma_{k-1}\mathsf{SAT}}$. Additionally, u encodes all the oracle queries (φ) with their replies and also the certificates for the yes-replies. This lets us construct the TM that accepts $\hat{L}$ as a Turing machine $\hat{M}$ with the oracle $Q_{k-1}\mathsf{Frm}$ (this is the language of Boolean formulas with $k-1$ quantifiers), that works as follows:

$\hat{M}^{Q_{k-1}\mathsf{Frm}}(x, u)$:
Simulate $M^{(\cdot)}(x)$:
When
- $M^{(\cdot)}$ makes a non-deterministic step
  - get the "right" choice from u;
- $M^{(\cdot)}$ queries the oracle and the reply would be yes
  - u has the corresponding query φ and the certificate c,
  so we can check if $\varphi(c) \stackrel{?}{\in} \Pi_{k-2}\mathsf{SAT}$;
- $M^{(\cdot)}$ queries the oracle and the reply would be no
  - u has the corresponding query,
  so check that $\neg\varphi(c) \stackrel{?}{\in} \Pi_{k-1}\mathsf{SAT}$.

return output of simulated $M^{(\cdot)}(x)$

We have shown that

$$\Sigma_k^p \subseteq \mathsf{NP}^{\Sigma_{k-1}\mathsf{SAT}} \text{ and } \Sigma_k^p \supseteq \mathsf{NP}^{\Sigma_{k-1}\mathsf{SAT}}.$$

Hence,

$$\Sigma_k^p = \mathsf{NP}^{\Sigma_{k-1}\mathsf{SAT}}.$$

The second part of the theorem, showing that

$$\Pi_i^p = \mathsf{co}\Sigma_i^p$$

is trivial, as Π_i^p and Σ_i^p are each other's negations (we also have to flip the output of TM M, but in the case of DTM it is easy). ■

2.5 Collapsing

Theorem 5

$$\textit{If } \Sigma_i^p = \Pi_i^p, \textit{ then } \Sigma_{i+1}^p = \Sigma_i^p.$$

This states that if for some i $\Sigma_i^p = \Pi_i^p$ then the polynomial heirarchy collapses, i.e. it is not an infinite hierarchy. For example, we know that $\mathsf{NP} = \Sigma_1^p$ and that $\mathsf{coNP} = \Pi_1^p$. Thus, if $\mathsf{NP} = \mathsf{coNP}$ then the polynomial hierarchy collapses.

Proof It is sufficient to show that $\Sigma_{i+1}^p \subseteq \Sigma_i^p$. (the inclusion in the other direction is trivial). First, let us show that $\Sigma_{i+1}^p \subseteq \Sigma_i^p$. For that, pick a language $L \in \Sigma_{i+1}^p$, so we have

$$x \in L \Leftrightarrow \exists u_1 \forall u_2 \dots Q_{i+1} u_{i+1} : M(x, u_1, \dots, u_{i+1}) = \mathsf{true}.$$

Now, define a new language by packing the two first arguments of M together:

$$L' = \{(x, u_1) \mid \underbrace{\forall u_2 \exists u_3 \dots Q_{i+1} u_{i+1}}_{i \text{ quantifiers}} : M((x, u_1), u_2, \dots, u_{i+1}) = \mathsf{true}\}.$$

By definition, $L' \in \Pi_i^p$ and because $\Sigma_i^p = \Pi_i^p$, then also $L' \in \Sigma_i^p$. This is, $\exists M'$, such that

$$L' = \{(x, u_1) \mid \exists u_2 \forall u_3 \dots \bar{Q}_{i+1} u_{i+1} : M'((x, u_1), u_2, \dots, u_{i+1}) = \mathsf{true}\},$$

where $\bar{Q}$ is just the opposite of Q.

Next we just "unpack" the two arguments we put together beforehand and get

$$x \in L \Leftrightarrow \exists u_1 \exists u_2 \forall u_3 \dots \bar{Q}_{i+1} u_{i+1} : M'(x, u_1, u_2, \dots, u_{i+1}) = \mathsf{true}.$$

As there are two $\exists$ quantifiers in a row, there are just $i-1$ alternations of quantifiers. Hence, $L \in \Sigma_i^p$. ■

Corollary 6

$$\textit{If } \Sigma_i^p = \Pi_i^p, \textit{ then } \Sigma_j^p = \Sigma_i^p \textit{ for all } j \geq i.$$

2.6 PH-completeness

Theorem 7 *If* PH *has complete problems then ploynomial hierarchy collapses.*

Sketch of Proof (Informal) If there exists a language that is PH-complete then this language belongs to some Σ_i^p. By the definition of completeness, all other languages in PH can be reduced to that language in Σ_i^p. This means that there cannot be any languages that are harder than this language. ■

Corollary 8 *If* PH = PSPACE *then polynomial hierarchy collapses.*

The idea behind the corollary lies in the fact that there exist complete languages in PSPACE, e.g. TQBF. Hence, by the theorem, the polynomial hiererchy collapses.

2.7 Σ_k^p and game-playing

Imagine a two-player game with perfect information:

- a set of possible *states*, a *starting state*, possible *ending states* with the indication who *won* and who *lost*;
- for each state there are possible legal moves for both player;
- and both players always know the state the game is in.

With this construction, it is easy to express "*Can the first player win in k half-moves?*" with the following:

$$\underbrace{\exists\text{my move}\,\forall\text{opponent's move}\,\exists\text{my move}\ldots}_{k\text{ quantifications}}:\ \text{I win!}\ .$$

For example, in chess, the "*checkmate in three moves*" could be expressed in Σ_5^p (because we would have 5 half-moves).

Similarily, the Π_k^p could be used to express the situation where the first player is determined to lose in k half-moves.

3 Alternating Turing Machines

Alternating Turing machines (ATM) are similar to non-deterministic Turing machines, as they can be tought as having the same tree-shape computation path. However, in each node of the tree (i.e. in each state) ATMs have a quantifier: either $\exists$ or $\forall$. The acceptance on input $x \in \{0,1\}^*$ is given as a recursive condition:

- a configuration with state $q_{\texttt{acc}}$ leads to accepting configuration;
- a configuration with a state labeled with $\exists$ leads to accepting configuration if *at least one* of its successors lead to accepting configuration.

- a configuration with a state labeled with $\forall$ leads to accepting configuration if *all* of its successors lead to accepting configuration.
- $x \in \{0,1\}^*$ is accepted if starting configuration with x leads to accepting configuration.

For simplicity we assume that each configuration of ATM M has exactly two possible successors. Hence, we get a binary tree.

3.1 Classes $\mathsf{ATIME}(T)$, $\Sigma_i\mathsf{TIME}$ and $\Pi_i\mathsf{TIME}$

Similarily to DTIME and NTIME, we define a new class ATIME:

Definition 11 $L \in \mathsf{ATIME}(T)$ if exists an ATM M and constant c, such that for all $x \in \{0,1\}^*$:

- all paths in the configuration graph of M, starting from the initial configuration of x, have length at most $c \cdot T(|x|)$;
- $x \in L$ iff M accepts x.

$\diamondsuit$

And similarily to the classes Σ_i^p and Π_i^p, we define:

Definition 12 $L \in \Sigma_i\mathsf{TIME}(T)$, if exist M, c satisfying the conditions in the definition of ATIME, and

- the initial state of M is labeled with $\exists$;
- on all paths in the configuration graph of M, there are at most $i-1$ switches between $\exists$ and $\forall$.

$\diamondsuit$

Definition 13 $L \in \Pi_i\mathsf{TIME}(T)$, if exist M, c satisfying the conditions in the definition of ATIME, and

- the initial state of M is labeled with $\forall$;
- on all paths in the configuration graph of M, there are at most $i-1$ switches between $\exists$ and $\forall$.

$\diamondsuit$

3.2 Equivalences

Theorem 9

$$\Sigma_i^p = \bigcup_c \Sigma_i\mathsf{TIME}(\lambda n.n^c)$$

Sketch of Proof (Informal) First, we will show that $\Sigma_i^p \subseteq \Sigma_i \mathsf{TIME}(\lambda n.n^c)$, for some c. Let us pick a language $L \in \Sigma_i^p$, then

$$x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots Q_i u_i : M(x, u_1, \ldots, u_i) = \mathsf{true}.$$

To show that L is also in $\Sigma_i \mathsf{TIME}(n^c)$, we can just take all the quantifiers and put them in the states of the corresponding ATM. Then, in the end, M just works as a usual DTM.

Secondly, we show that $\Sigma_i \mathsf{TIME}(\lambda n.n^c) \subseteq \Sigma_i^p$, for some c. Again, let us pick a language $L \in \Sigma_i \mathsf{TIME}(n^c)$, then we have an ATM M' that accepts L. We show that the same L can also be accepted by the M in the following construction:

$$x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots Q_i u_i : M(x, u_1, \ldots, u_i) = \mathsf{true}.$$

M can simulate M' and whenever M' has to make a choice in a given state with one of the quantifiers, M can just take the value of the corresponding u_i and determine the "correct" choice.

We have shown that $\Sigma_i^p \subseteq \Sigma_i \mathsf{TIME}(\lambda n.n^c)$ and $\Sigma_i \mathsf{TIME}(\lambda n.n^c) \subseteq \Sigma_i^p$. Hence $\Sigma_i \mathsf{TIME}(\lambda n.n^c) = \Sigma_i^p$. ■

Theorem 10

$$\Pi_i^p = \bigcup_c \Pi_i \mathsf{TIME}(\lambda n.n^c)$$

The proof is analogous to the previous one.

Theorem 11

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

4 Time-space tradeoffs for SAT

Thus far we have talked about complexity classes with various time limits or space limits. For example, a Turing machine accepting a language in NP is not concerned about how much memory it uses and a TM accepting a language in NL can take any amount of time to finish. Next, we will consider a complexity class, where limits on both time and space exist.

Definition 14 $L \in \mathsf{TISP}(T, S)$ is exists DTM M that accepts L in time $O(T)$ and in space $O(S)$. ♢

Note, that the name TISP comes from `TIme` and `SPace`.

Theorem 12

$$\mathsf{SAT} \notin \mathsf{TISP}(\lambda n.n^{1.1}, \lambda n.n^{0.1}).$$

This theorem gives a lower bound for algorithms solving SAT. It is not proven that there are no algoritms that solve SAT in linear time, nor is it proven that there exist no algorithms that solve SAT in logarithmic memory. However, this theorem states that SAT cannot be solved in linear time **and**, at the same time, using logarithmic memory.

4.1 Efficiency of reduction

Lemma 13 *If* $\mathsf{SAT} \in \mathsf{TISP}(\lambda n.n^{1.1}, \lambda n.n^{0.1})$ *then*

$$\mathsf{NTIME}(\lambda n.n) \subseteq \mathsf{TISP}(\lambda n.n^{1.1} \cdot polylog(n), \lambda n.n^{0.1} \cdot polylog(n)).$$

Here, $polylog(n)$ means "polynomial in the length of n".

Claim 14 *If* $L \in \mathsf{NTIME}(T)$*, then* L *can be recognized by some* oblivious *NTM in time* $\lambda n.T(n) \log T(n)$*.*

The head movement of an *oblivious* Turing machine depends only on the language L it is recognizing, and on the *length* of the input $x \in \{0,1\}^*$, but not on the actual input x. Also, in the case of oblivious TM, position of head at each step and previous step when the head was at a certain position, can be computed in $polylog(n)$.

Claim 15 *If* L *is recognized in time* T *by some oblivious NTM, then there exists a reduction* f *from* L *to* SAT*, such that*

- $|f(x)| \in O(T)$*;*
- *each bit of* $f(x)$ *can be computed in time* $polylog(|x|)$*.*

The idea behind the last claim is that by using an oblivious TM, the reduction takes less time, as we do not have to explicitly express all the details in f. For example, f does not have to encode the movement of heads, as this does not depend on the input. Now, f only has to describe $O(T)$ steps, each of which takes a constant number of bits, so $|f(x)| \in O(T)$. Without an oblivous TM the description of the configuration at each step of the computation normally takes $O(T)$ bits, hence such reduction takes $O(T^2)$ time.

4.2 Time to alternation

Lemma 16

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

Proof For a language $L \in \mathsf{TISP}(n^{12}, n^2)$ we have a TM M that accepts $x \in L$ in space $c \cdot |x|^2$ and time $c \cdot |x|^{12}$. We have to show that $L \in \Sigma_2\mathsf{TIME}(n^8)$ also holds, so

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{c \cdot |x|^8} \, \forall v \in \{0,1\}^{c \cdot |x|^8} : M'(x,u,v) = \mathsf{true},$$

where M' is a DTM working in time $O(n^8)$.

As M is working in time $O(n^{12})$, it also has $O(n^{12})$ possible configurations. We encode those configurations in u and v of M' so that M' accepts L iff

- exist configurations $C_0, C_1, \ldots, C_{c \cdot |x|^6}$ of M (encoded in u of M'), such that

- for each $i \in \{0, \ldots, c \cdot |x|^6\}$ (encoded on v of M')
- C_i is reachable from C_{i-1} in $|x|^6$ steps. Also, C_0 and $C_{c\cdot|x|^6}$ are initial and final configurations.

So M' makes a chain of the configurations of M and splits it into $c \cdot |x|^6$ parts with length $c \cdot |x|^6$:

$$\underbrace{C_0 \to\to\to C_1}_{C_1 \text{ is reachable from } C_0 \text{ in } c\cdot|x|^6 \text{ steps}} \to\to\to C_2 \to\to \ldots \to C_{c\cdot|x|^6}$$

The third point in the list above can be checked by M' in time $|x|^6$. However, the configurations take space $|x|^8$, so M' also works in $O(|x|^8)$ time. ■

4.3 The Padding Argument

Let $\mathsf{CL}_1(f(n))$ and $\mathsf{CL}_2(g(n))$ be complexity classes that are characterized by the amount of resources (time or space) they allow to spend to the machines that acceps languages belonging to these classes. The resources are measored by functions $f(n)$ and $g(n)$ respectively (with O-precision), where n is the input size.

Theorem 17 *If* $\mathsf{CL}_1(f(n)) \subseteq \mathsf{CL}_2(g(n))$ *then* $\mathsf{CL}_1(f(n^c)) \subseteq \mathsf{CL}_2(g(n^c))$ *for every constant* $c \in \mathbb{N}$.

Proof Let M be a Turing machine that decides $L \in \mathsf{CL}_1(f(n^c))$ in $f(n^c)$ time (or space). If $c = 1$, the statement is trivial. Otherwise, if $c \geq 2$, we define a new language:

$$L' = \{x01^{|x|^c-|x|-1} : x \in L\}.$$

The notation "$x01^{|x|^c-|x|-1}$:" here stands for the initial argument x (bitstring), followed by a zero and a number of 1-s so that all of the bitstrings (elements) in L' have length of $|x|^c$.

Now, define a new TM $M'(y)$ that accepts iff y is in the form $x01^{|x|^c-|x|-1}$ and $M(x) = \mathsf{true}$. Machine M' works with resources $f(|x|^c) = f(|y|)$ and hence,

$$L' \in \mathsf{CL}_1(f(n)) \subseteq \mathsf{CL}_2(g(n)).$$

Consequently, there is a CL_2-machine N' that desides L' in time (or space) $g(n)$. So we define a new machine $N(x) \equiv N(x01^{|x|^c-|x|-1})$, which decides L with resources

$$g(|x01^{|x|^c-|x|-1}|) = g(|x|^c).$$

Hence, $L \in \mathsf{CL}_2(g(|x|^c))$. ■

From the book "Computational Complexity: A Modern Approach" by Sanjeev Arora and Boaz Barak:

> The padding technique used in this proof, whereby we transform a language by "padding" every string in a language with a string of (useless) symbols, is also used in several other results in complexity theory. In many settings it can be used to show that equalities between complexity classes "scale up"; that is, if two

different type of resources solve the same problems within bound $T(n)$ then this also holds for functions T' larger than T. Viewed contrapositively, padding can be used to show that inequalities between complexity classes involving resurce bound $T'(n)$ "scale down" to resource bound $T(n)$.

For example, using the padding argument, we get the following corollary.

Corollary 18 *If* $\mathsf{NTIME}(n) \subseteq \mathsf{DTIME}(n^{1.2})$ *then*

$$\mathsf{NTIME}(n^{10}) \subseteq \mathsf{DTIME}(n^{12}).$$

4.4 Relationship on DTIME, NTIME, Σ_2TIME

Lemma 19 *If* $\mathsf{NTIME}(n) \subseteq \mathsf{DTIME}(n^{1.2})$ *then*

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

Proof First, we notice that $L \in \Sigma_2\mathsf{TIME}(n^8)$ iff exists DTM M working in time (n^8), such that

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{c|x|^8}\, \forall v \in \{0,1\}^{c|x|^8} : M(x,u,v) = \mathsf{true}.$$

This is equivalent to the same statement where both sides are negated and hence, the quantifiers have changed:

$$x \notin L \Leftrightarrow \forall u \in \{0,1\}^{c|x|^8}\, \exists v \in \{0,1\}^{c|x|^8} : M(x,u,v) = \mathsf{false}.$$

However, the part "$\exists v \in \{0,1\}^{c|x|^8} : M(x,u,v) = \mathsf{false}$" looks like a common construction of a non-deterministic TM. Hence, exists a NTM M' working in time $O(n^8)$, such that

$$x \notin L \Leftrightarrow \forall u \in \{0,1\}^{c|x|^8} : M'(x,u) = \mathsf{true}.$$

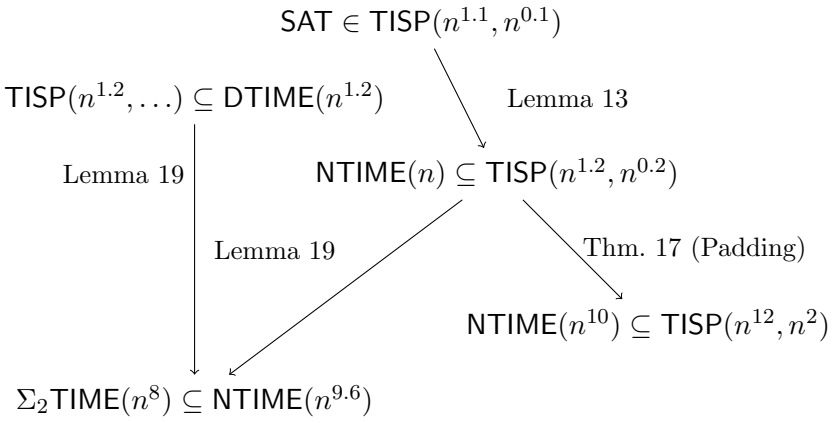
The padding argument gives us that $\mathsf{NTIME}(n^8) \subseteq \mathsf{DTIME}(n^{9.6})$. Hence, exists a DTM M'' working in time $O(n^{9.6})$, such that

$$x \notin L \Leftrightarrow \forall u \in \{0,1\}^{c|x|^8} : M''(x,u) = \mathsf{false}$$

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{c|x|^8} : M''(x,u) = \mathsf{true}.$$

Again, we see the common construction for a NTM. Hence, exists NTM M''' working in time $O(n^{9.6})$, that recognizes L. ■

Finally, we have enough knowledge to prove Theorem 12, stating that $\mathsf{SAT} \notin \mathsf{TISP}(\lambda n.n^{1.1}, \lambda n.n^{0.1})$.
Sketch of Proof (Informal) For the moment, let us assume that $\mathsf{SAT} \in \mathsf{TISP}(\lambda n.n^{1.1}, \lambda n.n^{0.1})$. Then, using the results form this section, we can draw a map, where the arrows indicate implications between the results.



Lemma 16: $\mathsf{TISP}(n^{12}, n^2) \subseteq \Sigma_2\mathsf{TIME}(n^8)$

From the leaves of this "tree" we get that

$$\mathsf{NTIME}(n^{10}) \subseteq \mathsf{TISP}(n^{12}, n^2) \subseteq \Sigma_2\mathsf{TIME}(n^8) \subseteq \mathsf{NTIME}(n^{9.6}),$$

that brings us to a contradiction. ■

4.5 Separating PH and PSPACE

Theorem 20 *There exists $A \subseteq \{0,1\}^*$, such that $\mathsf{PH}^A \neq \mathsf{PSPACE}^A$.*

More generally, for each k there exists oracle, relative to which the polynomial hierarchy has exactly k levels.