Indian Association for the Cultivation of Science
(Deemed to be University under *de novo* Category)

Master's/Integrated Master's-PhD Program/ Integrated Bachelor's-Master's Program/PhD Course

Theory of Computation II: COM 5108

Lecture VI

Instructor: Goutam Biswas *Autumn Semester 2023*

0.1 Arithmetic Hierarchy

The *arithmetic hierarchy* of undecidable problems classifies the problems to different grades of undecidability. We consider four languages.

$$A_{TM} = \{< M, x >: \text{ the TM } M \text{ accepts } x\}.$$

The language A_{TM} is *recursively enumerable (r.e.)*. Another name of this class is Σ_1^0. If we assume that a TM halts only when it *accepts* the input, the phrase "the TM M accepts x" can be written as "$\exists t,$ M halts on x within t steps". A_{TM} is Turing *recognizable* language but not decidable (not *recursive*). In general a language L in Σ_1^0 can be written as $L = \{w \in \Sigma^* : \exists y R(w, y)\}$, where R is a *recursive predicate* i.e. its truth value can be decided by a TM. In our example $w =< M, x >, y = t$.

The second language is $\overline{A_{TM}}$, the complement of A_{TM}.

$$\overline{A_{TM}} = \{< M, x >: \text{ the TM } M \text{ does not accepts } x\}.$$

The language $\overline{A_{TM}}$ is not *r.e.*. The phrase "the TM M does not accepts x" can be written as "$\forall t,$ M does not halt on x within t steps". There is no TM that *recognizes* it. This language belongs to the class Π_1^0. Any complement of a *r.e.* belongs to this class. In general a language L in Π_1^0 can be written as $L = \{w \in \Sigma^* : \forall y R(w, y)\}$, where R is a *recursive predicate*. In our example $w =< M, x >, y = t$.

The third language is

$$A_{fin} = \{< M >: \text{ the language of the TM } M \text{ is finite}\}.$$

This language is neither in Σ_1^0 nor in Π_1^0. The phrase "the language of the TM M is finite" can be written as "$\exists n \forall x \forall t,$ $|x| \geq n \Rightarrow M$ does not halt on x in t steps". Note that "$\forall x \forall t$" can be combined to a single universal quantifier[1].

[1]Both x and t are positive integers. A pair of positive integers (x, t) can be encoded as a single integer e.g. $m = 2^x \cdot 3^t$, where $\pi_1(m) = x$ and $\pi_2(m) = t$. So $\forall x \forall t \equiv \forall m$ where x and t are replaced by $\pi_1(m)$ and $\pi_2(m)$ respectively.

This language belongs to Σ_2^0. In general a language L in Σ_2^0 can be written as $L = \{w \in \Sigma^* : \exists y \forall z R(w, y, z)\}$, where R is a *recursive predicate*. In our example $w =< M >, y = n, z =< x, t >$.

The fourth language A_{inf} is complement of A_{fin}.

$$A_{inf} = \{< M >: \text{ the language of the TM } M \text{ is infinite}\}.$$

The phrase "the language of the TM M is infinite" can be written as "$\forall n \exists x \exists t,\ |x| \geq n$ and M halt on x in t steps". Two existential quantifiers can be combined to a single existential quantifier. This language belongs to the class Π_2^0. In general a language L in Π_2^0 can be written as $L = \{w \in \Sigma^* : \forall y \exists z R(w, y, z)\}$, where R is a *recursive predicate*. In our example $w =< M >, y = n, z =< x, t >$.

There is a hierarchy $\Sigma_1^0, \Sigma_2^0, \cdots$ and the class of their complements $\Pi_1^0, \Pi_2^0, \cdots$. Also there is $\Delta_1^0, \Delta_2^0, \cdots$, where $\Delta_i^0 = \Sigma_i^0 \cap \Pi_i^0$ for all $i = 1, 2, \cdots$. The class Δ_1^0 is the class of *recursive* languages.

A set is in Σ_n^0 if there is a *decidable* $(n+1)$*-ary predicate* R such that $L = \{x : \exists y_1 \forall y_2 \cdots Q y_n\ R(x, y_1, \cdots, y_n)\}$, $Q = \exists$ if n is odd and $\forall$ if n is even.

A set is in Σ_n^0 if there is a *decidable* $(n+1)$*-ary predicate* R such that $L = \{x : \forall y_1 \exists y_2 \cdots Q y_n\ R(x, y_1, \cdots, y_n)\}$, $Q = \exists$ if n is even and $\forall$ if n is odd.

0.2 Polynomial Hierarchy

In a similar line *polynomial hierarchy*, **PH**, is *conjectured* to divide the problems harder than **NP** and **coNP**. The class **PH** is conjectured to have infinite layers of subclasses beyond **P**, **NP** and **coNP**. We have the following motivating example.

Example 1. Consider the **NP**-*complete* problem of *independent set*.

$$INDSET = \{< G, k >: \text{ graph } G \text{ has an independent set of size} \geq k\}.$$

This problem has a *short certificate*, a set of k vertices that are not connected by edges among themselves.
Its complement problem is

$$\overline{INDSET} = \{< G, k >: \text{ graph } G \text{ has no independent set of size} \geq k\}.$$

It does not have any obvious short certificate and it belongs to **coNP**.

Following is a similar problem which also does not have any obvious short certificate (no one knows).

$$MAX{-}INDSET = \{< G, k >: \text{ the size of the largest independent set of } G \text{ is } k\}.$$

In this case we may write, "the undirected graph G has an independent set u of size k such that all independent sets of G are of size less than or equal to k."

In the similar line of undecidable problems, the characterization of these languages can be written using existential and an universal quantifiers. The only difference is the Turing machine is polynomial time bounded and the quantifiers also polynomial bounded.

A problem in class **NP** is characterized by a polynomial bounded[2] existential quantifier corresponding to certificate. This class is called Σ_1^P, a subclass (conjectured) of *polynomial hierarchy* (**PH**).

[2]In the *arithmetic hierarchy* the quantifiers are unbounded.

A language $L \subseteq \{0,1\}^*$ is in **NP** if there is a polynomial time Turing machine and a polynomial $p : \mathbb{N}_0 \to \mathbb{N}_0$, such that for all $x \in \{0,1\}^*$,

$$x \in L \Leftrightarrow \exists w \in \{0,1\}^{p(|x|)}, \text{ s.t. } M \text{ accepts } < x, w >.$$

In general $x \in L \Leftrightarrow \exists^p w R(x,w)\}$, where the quantifier is polynomial ($p(|x|)$) bounded and the predicate can be computed in polynomial time.

Similarly, $\overline{L}$ is in **coNP** if L is in **NP** i.e.

$$x \in \overline{L} \Leftrightarrow \forall w \in \{0,1\}^{p(|x|)}, \text{ s.t. } \overline{M} \text{ accepts } < x, w >.$$

The machine $\overline{M}$ is same as M of **NP** with the *accept* and *reject* states interchanged.

There is a polynomial bounded universal quantifier for a problem in **coNP** and the class is called Π_1^P of the **PH**.
The intersection of Σ_1^P and Π_1^P, is $\Delta_1^P = \mathbf{P}$.

The problem of MAX-INDSET is specified by an existential quantifier, followed by an universal quantifier, is in the class Σ_2^P. Both the quantifiers are polynomial bounded.
$< G, k > \in$ MAX-INDSET if and only if $\exists$ an independent set $V_k = \{v_{i_1}, \cdots, v_{i_k}\}$ of G s.t.
$\forall$ independent sets $V_l = \{v_{j_1}, \cdots, v_{j_l}\}$ of G, $l \leq k$. It is to be noted that any subset of $V(G)$ can be encoded as a binary string of finite length.

So it is possible to have a polynomial time bounded Turing machine M and a polynomial $p : \mathbb{N}_0 \to \mathbb{N}_0$, (which is cn here), such that M accepts $<< G, k > , V_k, V_l >$ where V_k is an independent subset of $V(G)$ of size k and V_l is any independent subset of $V(G)$ of size l such that $l \leq k$. We have the following formal definition of Σ_2^P.

Definition 1. Σ_2^P is the class of languages L such that there is a polynomial time bounded Turing machine M and a polynomial q (both depends on L) such that for all $x \in \{0,1\}^*$,

$$x \in L \text{ if and only if } \exists u \in \{0,1\}^{q(|x|)} \; \forall v \in \{0,1\}^{q(|x|)} \; M \text{ accepts } < x, u, v >.$$

MAX-INDSET is in Σ_2^P. By definition Σ_2^P contains both **NP** and **coNP**. In general a language $L \in \Sigma_2^P$ is defined as

$$x \in L \text{ if and only if } \exists^p y \forall^p z R(x, y, z).$$

Now we are in a position to generalize the definition of such classes and define the *polynomial hierarchy*.

Definition 2. The class Σ_n^P, for $n \geq 1$, is the collection of languages L so that there is a polynomial time bounded Turing machine M and a polynomial q (both depends on L) such that for all $x \in \{0,1\}^*$, $x \in L$ if and only if

$$\exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \cdots Q u_n \in \{0,1\}^{q(|x|)} \; s.t. \; M \text{ accepts } < x, u_1, \cdots, u_n >,$$

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

The *polynomial hierarchy* $\mathbf{PH} = \bigcup_{i \geq 1} \Sigma_i^P$, the union of all these classes. We define

$$\Pi_n^P = co\Sigma_n^P = \{\overline{L} \in \{0,1\}^* : \; L \in \Sigma_n^P\}.$$

We observe that $\mathbf{NP} = \Sigma_1^P$ and $\mathbf{coNP} = \Pi_1^P$. It is not difficult to establish that for all $n \geq 1$, $\Sigma_n^P \subseteq \Pi_{n+1}^P \subseteq \Sigma_{n+2}^P$. So $\mathbf{PH} = \bigcup_{i \geq 1} \Pi_i^P$.

An alternate definition is as follows: A language $L \in \Sigma_n^P$ if there is a polynomial-time computable $(n+1)$-ary relation R such that for all $x \in \{0,1\}^*$, $x \in L$ if and only if $\exists u_1 \forall u_2 \cdots Q_n u_n R(x, u_1, \cdots, u_n)$, where $u_1, \cdots, u_n \in \{0,1\}^{q(|x|)}$.

0.3 PH and OTM

There is a definition of Σ_n^P, Π_n^P and Δ_n^P using *oracle Turing machine* (OTM).

$$\Sigma_0^P = \Pi_0^P = \Delta_0^P = \mathbf{P}.$$

And

$$\Delta_{i+1}^P = \mathbf{P}^{\Sigma_i^P},\ \Sigma_{i+1}^P = \mathbf{NP}^{\Sigma_i^P},\ \Pi_{i+1}^P = \mathbf{coNP}^{\Sigma_i^P}.$$

$\mathbf{P}^{\Sigma_i^P}$ is set of languages that can be decided by a polynomial time bounded OTM where the oracle is a *complete problem* of Σ_i^P. Similarly, $\mathbf{NP}^{\Sigma_i^P}$ is set of languages that can be decided by a nondeterministic polynomial time bounded OTM where the oracle is a *complete problem*.

Note that

- $\Delta_1^P = \mathbf{P}^{\Sigma_0^P} = \mathbf{P}^{\mathbf{P}} = \mathbf{P}$.
- Similarly, $\Sigma_1^P = \mathbf{NP}^{\mathbf{P}} = \mathbf{NP}$ and $\Pi_1^P = \mathbf{coNP}^{\mathbf{P}} = \mathbf{coNP}$.
- $\Delta_2^P = \mathbf{P}^{\Sigma_1^P} = \mathbf{P}^{\mathbf{NP}}$. If the oracle is a complete problem for **NP**, a language in **NP** or **coNP** can be decided in polynomial time (how?). So $\Sigma_1^P(\mathbf{NP}), \Pi_1^P(\mathbf{coNP}) \subseteq \Delta_2^P$.
- $\Sigma_2^P = \mathbf{NP}^{\Sigma_1^P} = \mathbf{NP}^{\mathbf{NP}}$. It is clear that $\mathbf{NP} = \Sigma_1^P \subseteq \Sigma_2^P$. Similarly, $\mathbf{coNP} = \Pi_1^P \subseteq \Pi_2^P$.
- In general $\Sigma_i^P \subseteq \Sigma_{i+1}^P$ and $\Delta_i^P \subseteq \Delta_{i+1}^P$

Now we have two definitions of Σ_n^p.

1. A language $L \in \Sigma_n^p$ if

   $$L = \{x \in \{0,1\}^* :\ \exists^p y_1 \forall y^p y_2 \cdots Q_n^p y_n R(x, y_1, \cdots y_n)\},$$

   where all quantifiers are polynomial ($p(|x|)$) bounded (strings over $\{0,1\}^{p(|x|))}$) and the predicate $R(x, y_1, \cdots y_n)$ can be computed in polynomial time, $Q_n = \exists$ if n is odd, $= \forall$ if n is even.

2. $\Sigma_0^p = \mathbf{P}$ and $\Sigma_n^p = \mathbf{NP}^{\Sigma_{n-1}^p}$, where the oracle set is is a complete problem of Σ_{n-1}^p.

It is necessary to prove their equivalence. Here we present the proof outline of the case where $n = 2$.

Example 2. We show that $\Sigma_2^p \subseteq \mathbf{NP}^{SAT}$ (SAT is a complete problem of Σ_1^p). $L = \{x \in \{0,1\}^* :\ \exists^p y_1 \forall^p y_2 R(x, y_1, y_2)\}$. We design the following polynomial time OTM with the oracle set SAT to decide L.

N : Input x

(i) Make a nondeterministic guess of y_1.

(ii) The negation of $R()$ is $\overline{R}()$. Find $\overline{R}(x, y_1)$. By Karp reduction ($\leq_m^p$) it reduces to $\overline{\phi}(y_2)$.

(iii) Send query $\overline{\phi}(y_2) \stackrel{?}{\in} SAT$.

(iv) If the answer is '*yes*', **reject**; else **accept**.

If there is some y_2 that satisfies $\overline{\phi}(y_2)$ ($\overline{R}(x, y_1, y_2)$), then it is not the case that $R(x, y_1, y_2)$ is *true* for all y_2.

Now we show that $\mathbf{NP}^{SAT} \subseteq \Sigma_2^p$.

Let $L \in \mathbf{NP}^{SAT}$. There is a polynomial time bounded OTM N with the oracle set SAT so that $x \in L$ if and only if there is a sequence of non-deterministic choices $c_1, \cdots, c_m$, there is a sequence of queries $\phi_1, \cdots \phi_k$ and the corresponding answers $a_1, \cdots, a_k$; and the corresponding computation of N reaches an *accept* state, where

$$a_i = \begin{cases} 1 & \text{if } \exists u_i \phi_i(u_i) = 1 \\ 0 & \text{if } \forall v_i \phi_i(v_i) = 0. \end{cases} \quad i = 1, \cdots, k.$$

So

$$x \in L \text{ if and only if } \exists c_1 \cdots \exists c_m \exists u_1 \cdots \exists u_k \forall v_1 \cdots \forall v_k \ N \text{ accepts } x.$$

So $L \in \Sigma_2^p$.

0.4 Complete Languages of Σ_n^P

Definition 3. A language L is said to be Σ_n^P-*complete* if

- $L \in \Sigma_n^P$, and
- Every $L' \in \Sigma_n^P$ is Karp reducible to L i.e. there is a polynomial time computable function $f : \mathbb{N}_0 \to \mathbb{N}_0$, such that for all $x \in \{0,1\}^*$, $x \in L'$ if and only if $f(x) \in L$.

The definition of **PH**-*complete* and Π_n^P-*complete* are similar. We shall show that there are complete problems for Σ_n^P and Π_n^P. Note the following points.

- It is believed that there is no **PH**-*complete* problem due to the following simple reason.

  If L is **PH**-*complete*, then L must belong to Σ_n^P, for some n. But then for every $L' \in \mathbf{PH}$, $L' \leq_m^p L \in \Sigma_n^P$ implies $L' \in \Sigma_n^P$, so $\mathbf{PH} \subseteq \Sigma_n^P$.

- It is clear that $\mathbf{PH} \subseteq \mathbf{PSPACE}$ because a language $L \in \Sigma_n^P$ can be decided in polynomial space - polynomial time TM M computes on $< x, u_1, \cdots, u_n >$ where $|u_i| = q(|x|)$, q is a polynomial, $i = 1, 2, \cdots, n$.

- But it is believed that **PH** is not same as **PSPACE**, as it was pointed out that **PH** may not have a complete problem. But **PSPACE** has a complete problem.

Every class Σ_n^P has its own satisfiability problem that is complete for the class. We define the satisfiability problem $\Sigma_n SAT$ for the class Σ_n^P as follows.

$$\Sigma_n SAT = \{\phi(u_1, \cdots, u_n) : \exists u_1 \forall u_2 \cdots Q_n u_n \phi(u_1, \cdots, u_n) \text{ is } true\},$$

where $Q_n = \exists$ if n is odd, it is '$\forall$' otherwise. Each u_i, $1 \leq i \leq n$ is a vector of Boolean variables[3]. It is not difficult to see that this set is in Σ_n^P. We can have a polynomial (over the first input) time bounded Turing machine M such that every Boolean vector u_i, $1 \leq i \leq n$ has a valuation V_i which may be viewed as a string over $\{0,1\}$ and $\exists V_1 \forall V_2 \cdots Q_n V_n$, the machine M accepts $< \phi, V_1, \cdots, V_n >$ if and only if $\exists u_1 \forall u_2 \cdots Q_n u_n \phi(u_1, \cdots, u_n)$ is *true*.

To show that it is Σ_n^P-*hard*, we consider a $L \in \Sigma_n^P$. So we have a polynomial ($q(n)$) time bounded Turing machine M and a polynomial $p : \mathbb{N}_0 \to \mathbb{N}_0$ so that for all $x \in \{0,1\}^*$, $x \in L$ if and only if

$$\exists w_1 \in \{0,1\}^{p(|x|)} \forall w_2 \in \{0,1\}^{p(|x|)} \cdots Q w_n \in \{0,1\}^{p(|x|)},$$

such that M accepts $< x, w_1, \cdots, w_n >$.

Following the Cook-Levin's theorem, the computation of a polynomial time bounded Turing machine can be encoded as a polynomial size Boolean formula ϕ which has $(n+2)$ tuples of Boolean variables. The first tuple X corresponds to the input x. Similarly there are Boolean vectors $Y_1, \cdots, Y_n$, corresponding to n witness strings $w_1, \cdots, w_n$. Finally there is a $O(q(|x|)^2)$ length Boolean vector Z corresponding to the $q(|x|)$ time bounded computation of M on $< x, w_1, \cdots, w_n >$. The variables of Z are associated with each cell of the computation tableau of M.

By definition $x \in L$ if and only if $\exists w_1 \in \{0,1\}^{p(|x|)} \forall w_2 \in \{0,1\}^{p(|x|)} \cdots Q w_n \in \{0,1\}^{p(|x|)}$ such that M accepts $< x, w_1, \cdots, w_n >$ if and only if there is a valuation of Y_1, for all valuations of $Y_2, \cdots$, there is a value for Y_n (if n is odd) such that ϕ is satisfiable.

This is equivalent to saying that $x \in L$ if and only if $\exists Y_1 \forall Y_2 \cdots \exists Y_n \phi$ is satisfiable. This formula is in $\Sigma_n SAT$.

Note that $\Sigma_n SAT$ and $\Pi_n SAT$ are subset of QBF. So $\Sigma_n^P, \Pi_n SAT \subseteq$ **PSPACE** for all $n = 1, 2, \cdots$. So **PH** $\subseteq$ **PSPACE**.

0.5 Properties of PH

People believe that **P** $\neq$ **NP**, **NP** $\neq$ **coNP**. In **PH** it is conjectured that for all $n \geq 1$, Σ_n^P is a proper subset of Σ_{n+1}^P.

The *polynomial hierarchy* will collapse to the height k i.e. $\Sigma_k^P =$ **PH**, if there is some positive integer k so that $\Sigma_k^P = \Sigma_{k+1}^P$.

Proposition 1.

1. For every $n \geq 1$, if $\Sigma_n^P = \Pi_n^P$, then $\Sigma_n^P =$ **PH**.
2. If **P** = **NP**, then **P** = **PH**.

Lemma 1. If $\Sigma_n^P = \Pi_n^P$ and $\Sigma_n^P = \Sigma_{n+1}^P$, then $\Sigma_{n+1}^P = \Pi_{n+1}^P$.
Proof:(*lemma*) $\Sigma_{n+1}^P = \Sigma_n^P = \Pi_n^P \subseteq \Pi_{n+1}^P$. We prove the other direction. Let $L \in \Pi_{n+1}^P$; this implies $\overline{L} \in \Sigma_{n+1}^P \subseteq \Pi_{n+1}^P \Rightarrow L \in \Sigma_{n+1}^P$. QED.
Proof:(*proposition*)

[3] $\exists u_i \equiv \exists u_{i1} \exists u_{i2} \cdots \exists u_{ik}$, where u_{ij}, $1 \leq j \leq k$, are boolean variables. Similar is the case for the universal quantifier.

1. The proof is by mathematical induction starting from some $n \geq 1$.
   Basis: $\Sigma_n^P = \Pi_n^P$. This amounts to say that Σ_n^P is closed under complementation.

   Induction: We prove that $\Sigma_{n+1}^P \subseteq \Sigma_n^P$ to show that $\Sigma_{n+1}^P = \Sigma_n^P$, as the other direction of inclusion is known. This fact uses the previous lemma to establish
   $$\Pi_n^P = \Sigma_n^P = \Sigma_{n+1}^P = \Pi_{n+1}^P.$$
   So the *polynomial hierarchy* (**PH**) collapses to the n^{th} level, i.e. $\mathbf{PH} = \Sigma_n^P$.

   proof:
   Let $L \in \Sigma_{n+1}^P$. So there is a polynomial time bounded Turing machine M and a polynomial $p : \mathbb{N}_0 \to \mathbb{N}_0$ such that, for all $x \in \{0,1\}^*$,

   $x \in L$ if and only if $\exists u_1 \in \{0,1\}^{p(|x|)} \forall u_2 \in \{0,1\}^{p(|x|)} \cdots Q_{n+1} u_{n+1} \in \{0,1\}^{p(|x|)}$

   such that M accepts $< x, u_1, \cdots u_{n+1} >$, where $Q_{n+1} = \exists$ or $\forall$ depending on whether n is even or odd.

   We define the language L' as follows.

   $(x, u_1) \in L'$ if and only if $\forall u_2 \in \{0,1\}^{p(|x|)} \cdots Q_{n+1} u_{n+1} \in \{0,1\}^{p(|x|)}$

   such that M' (a modification of M) accepts $<< x, u_1 >, u_2, \cdots u_{n+1} >$, where $Q_{n+1} = \exists$ or $\forall$ depending on whether n is even or odd. Clearly $L' \in \Pi_n^P$. But then $\Pi_n^P = \Sigma_n^P$. So there is a polynomial time bounded Turing machine M'' and a polynomial $q : \mathbb{N}_0 \to \mathbb{N}_0$ such that,

   $(x, u_1) \in L'$ if and only if $\exists v_1 \in \{0,1\}^{q(|x|)} \forall v_2 \in \{0,1\}^{q(|x|)} \cdots Q_n v_n \in \{0,1\}^{q(|x|)}$

   such that M'' accepts $<< x, u_1 >, v_1 \cdots, v_n >$, where $Q_n = \exists$ or $\forall$ depending on whether n is odd or even.

   So we redefine the language L as

   $x \in L$ if and only if $\exists < u_1, v_1 > \in \{0,1\}^{r(|x|)} \forall v_2 \in \{0,1\}^{r(|x|)} \cdots Q_n v_n \in \{0,1\}^{r(|x|)}$

   such that M''' (a modification of M'') accepts $< x, < u_1, v_1 >, v_2, \cdots, v_n >$, where r may be taken as $p + q$. So $L \in \Sigma_n^P$.

2. This is a special case when $n = 1$. $\mathbf{P} = \mathbf{NP}$ is equivalent to $\Sigma_1^P = \Pi_1^P$. So under this condition $\mathbf{P} = \mathbf{PH}$.

QED.

0.6 Alternation

Using an NTM one can solve a problem that requires *guessing* an accepting computation path. This is equivalent to deterministic verification of a certificate. If the NTM is polynomial time bounded, the verifier is also so. The length of the certificate is also of polynomial bounded.
But there are problems for which there is no obvious (known) short certificate. An *alternating Turing machine (ATM)* was proposed as a generalization of NTM to capture such (and higher order) computation problems.

An ATM has two transition functions δ_0 and δ_1. From every configuration there are two possible next configurations[4]. Each state of an ATM, except the final states q_A and q_R, is labeled either with '$\vee$'(or) or with '$\wedge$' (and). An NTM is an ATM where all states are labeled with '$\vee$'.

The configuration graph of an ATM is similar to that of an NTM. But in case of an NTM a configuration C in the graph is said to be accepting (labeled with 'A') if there is path from C to a configuration with an accepting state q_A. In other words, one of the successor configuration of C is labeled with 'A'. A string $x \in \Sigma^*$ is accepted by an NTM if the start configuration of $G_{M,x}$ is labeled with 'A'.

In case of an ATM we inductively define (label) accepting configurations as follows.

1. *Basis:* All configurations with q_A are labeled with 'A' and all configurations with q_R are labeled with 'R'.

2. *Induction:* If C is a configuration with a state labeled '$\vee$', and C_1 and C_2 are its successor configurations, then C is labeled with 'A' (*accepting*), if any one of C_1 or C_2 is labeled with 'A' (*accepting*).

   If the label of the state of C is '$\wedge$', and C_1 and C_2 are its successor configurations, then C is labeled with 'A' (*accepting*), if both C_1 and C_2 are labeled with 'A' (*accepting*).

An alternating Turing machine M accepts a string $x \in \Sigma^*$, if the start configuration of $G_{M,x}$ can be labeled with 'A' (*accepting*).

Definition 4. Let $f : \mathbb{N}_0 \to \mathbb{N}_0$ be a time constructible function. We say that an *alternating Turing machine (ATM)*, M runs for $f(n)$ time, if for all input $x \in \{0,1\}^*$, such that $|x| = n$, the depth of the configuration graph $G_{M,x}$ is at most $f(|x|)$.

If a language $L \subseteq \{0,1\}^*$ is decided by an *alternating Turing machine* M, that halts on all input within $f(|x|)$ steps, then $L \in \mathbf{ATIME}(f(n))$. The start state q_0x of $G_{M,x}$ is labeled 'A' if and only if $x \in L$. Similarly we define the class $\mathbf{ASPACE}(f(n))$. We also have

$$\begin{aligned} \mathbf{AP} &= \bigcup_{c \geq 1} \mathbf{ATIME}(n^c), \\ \mathbf{APSPACE} &= \bigcup_{c \geq 1} \mathbf{ASPACE}(n^c), \\ \mathbf{AL} &= \mathbf{ASPACE}(\log n), \end{aligned}$$

Example 3. We know that $TAUT = \{< \phi >:$ Boolean formula ϕ is a tautology$\}$ is in **coNP**[5]. Following is an ATM algorithm for TAUT.
M: input $< \phi >$

[4]It is possible to have ATM with more than two possible next configurations. But that can be transformed to two-way alternation.

[5]SAT is in **NP**. So $\overline{SAT}$, the collection of unsatisfiable formulas, is in **coNP**. Negation of unsatisfiable formula is a tautology.

1. Start state is universally quantified and branches on all possible assignments of the variables of ϕ[6]

2. Evaluate ϕ for every assignment.

3. If ϕ evaluates to 1 in all branches, then *accept*.

The depth of the tree is polynomial in the length ϕ. So TAUT $\in$ **AP**.
Example 4. NOT-EQIV = $\{< \phi, \psi >$: the boolean formulas ϕ and ψ are not equivalent$\}$. NOT-EQUIV is in **NP**.
M: input $< \phi >$

(a) Guess (nondeterministic) an assignment $x = (x_1, \cdots, x_n)$ so that $\phi(x) \neq \psi(x)$.

(b) Return 1 if $\phi(x) \neq \psi(x)$; else return 0.

Example 5.

$$NONMIN - FORMULA = \{< \phi >: \ \phi \text{ is not a minimal formula}\}.$$

NONMIN-FORMULA is in $\mathbf{NP}^{SAT}$.
We know that NOT-EQUIV $\in$ **NP**. There is a polynomial bounded function f so that $NOT - EQUIV \leq_m^p SAT$.
M: input $< \phi >$

(a) Guess (nondeterministic) a shorter formula ψ.

(b) Compute $f(\phi, \psi) = \alpha$ (say).

(c) Ask the oracle whether $\alpha \in SAT$.

(d) If the answer is '*yes*', return 1; else return 0.

Example 6. Prove that

$$MIN - FORMULA = \{< \phi >: \ \phi \text{ is a minimal formula}\}$$

is in **AP**.
M: input $< \phi >$

1. Start state is universally quantified and branches on all formulas ψ, shorter than ϕ (there are finite number of them).

2. Existentially select an assignment to the variables.

3. Evaluate both ϕ and ψ.

4. If for every universal branch, there is an existential branch where where the values of ϕ and ψ are different, then *accept*; else *reject*.

[6]Note that it can be decomposed into two way branches, each configuration labeled with '$\wedge$'.

Proposition 2. Let $f : \mathbb{N}_0 \to \mathbb{N}_0$ where $f(n) \geq n$. $\mathbf{ATIME}(f(n)) \subseteq \mathbf{SPACE}(f(n))$.

Proof: A deterministic $O(f(n))$ space bounded Turing machine S can do a depth-first search (DFS) on the configuration graph of $O(f(n))$ time bounded ATM M to decide about the labels (*accept* or *reject*) of the configurations of M.

It is necessary to stack $O(f(n))$ configurations for the DFS as the depth of $G_{M,x}$ is $O(f(n))$, where $|x| = n$. Each configuration takes $O(f(n))$ space. So apparently it is necessary to use $O(f(n)^2)$ space.

But then it is not necessary to keep the whole configuration but the history of non-deterministic moves taken up at every level. Sequence of configurations, one at a time, can be regenerated. So, only a constant amount of information is necessary to save at each depth of recursion, and the space requirement is $O(f(n))$. QED.

The conclusion is $\mathbf{AP} \subseteq \mathbf{PSPACE}$.

Proposition 3. Let $f : \mathbb{N}_0 \to \mathbb{N}_0$ where $f(n) \geq n$. $\mathbf{SPACE}(f(n)) \subseteq \mathbf{ATIME}(f(n)^2)$.

Proof: Given an $O(f(n))$ space bounded deterministic Turing machine M, we construct a $O(f(n)^2)$ time bounded alternating Turing machine A such that A simulates M.

The construction is similar to the construction in the Savitch's theorem. Given two configurations c_1 and c_2 of M and a number n, is c_2 reachable from c_1 in n steps?

An ATM first guesses the existence of a configuration c_i, midway between c_1 and c_2.

Then it will takes an universal branch to recursively test whether c_i is reachable from c_1 in $n/2$ steps, and also c_2 is reachable from c_i in $n/2$ steps.

An $f(n)(\geq n)$ time bounded TM can have at the most $2^{O(f(n))}$ configurations. An ATM A uses the recursive procedure to test whether M can reach an accepting configuration from the start configuration in $2^{df(n)}$ steps. At every stage of recursion the ATM takes $O(f(n))$ time to write the intermediate configuration c_i. The depth of recursion is $\log 2^{df(n)} = O(f(n))$. So the total time is $O(f^2(n))$. QED.

So polynomial space bounded TM can be simulated by a polynomial time bounded ATM i.e. $\mathbf{PSPACE} \subseteq \mathbf{AP}$ and finally we have $\mathbf{PSPACE} = \mathbf{AP}$.

Proposition 4. Let $f : \mathbb{N}_0 \to \mathbb{N}_0$ where $f(n) \geq \log n$. $\mathbf{ASPACE}(f(n)) \subseteq \mathbf{TIME}(2^{O(f(n))})$.

Proof: Let A be an $O(f(n))$ space bounded ATM. We construct an equivalent $2^{O(f(n))}$ time bounded TM D. The machine D on the input x, constructs the configuration graph of M. Each configuration is bounded by $k_A f(n)$ space, where k_A is a constant. The graph can be constructed in $2^{O(f(n))}$ time.

After the construction of the graph, D scans it and marks the configurations as *accepting*. It requires the traversal of the graph of $2^{O(f(n))}$ nodes. The time it takes is $2^{O(f(n))}$. So the total time of construction and labeling the graph takes $O\left(2^{O(f(n))}\right)$ time. QED. So $\mathbf{APSPACE} \subseteq \mathbf{EXP}$ and $\mathbf{AL} \subseteq \mathbf{P}$.

We can also prove the following proposition.

Proposition 5. Let $f : \mathbb{N}_0 \to \mathbb{N}_0$ where $f(n) \geq \log n$. $\mathrm{TIME}(2^{O(f(n))}) \subseteq \mathbf{ASPACE}(f(n))$. The conclusion is $\mathbf{ASPACE}(f(n)) = TIME(2^{O(f(n))})$. And the final conclusion is $\mathbf{AL} = \mathbf{P}$ and $\mathbf{APSPACE} = \mathbf{EXP}$.

So we have

References

[MS] *Theory of Computation* by Michael Sipser, (3rd. ed.), Pub. Cengage Learning, 2007, ISBN 978-81-315-2529-6.

[SABB] *Computational Complexity, A Modern Approach* by Sanjeev Arora & Boaz Barak, Pub. Cambridge University Press, 2009, ISBN 978-0-521-42426-4.

[CHP] Computational Complexity by *Christos H Papadimitriou*, Pub. Addision-Wesley, 1994, ISBN 0-201-53082-1.

[RPNS] *Lecture 5: Polynomial Hierarchy*, by Rafael Pass & Navin Sivakumar, February 3, 2009.