Nondeterministic Complexity Classes

Let \( t : \mathbb{N} \to \mathbb{N} \) be a function. We say that a nondeterministic program \( P \) is \( t(n) \)-time bounded, if for every \( n \in \mathbb{N} \), for every \( x \in \Sigma^n \), every path of \( P \) on \( x \) stops within \( t(n) \) steps.

Similarly a nondeterministic program \( P \) is \( t(n) \)-space bounded, if for every \( n \in \mathbb{N} \), for every \( x \in \Sigma^n \), every path of \( P \) on \( x \) uses at most \( s(n) \) amount of memory.

Now we have the following nondeterministic complexity classes.

The class \( \text{NTIME}(t(N)) \) is the collection of all languages that are accepted by a \( t(n) \)-time bounded nondeterministic program. Similarly, \( \text{NSPACE}(s(N)) \) is the collection of all languages that are accepted by a \( s(n) \)-space bounded nondeterministic program.

Now \( \text{NP} = \bigcup_{k>0} \text{NTIME}(n^k) \), \( \text{NL} = \bigcup_{c>0} \text{NSPACE}(c \log n) \), and \( \text{NPSPACE} = \bigcup_{k>0} \text{NSPACE}(n^k) \).

We now define another complexity class \( \text{PV} \). A language \( L \) belongs to the class \( \text{PV} \), if there is a polynomial \( p(.) \) and a polynomial-time computable function \( f \) such that

\[
\begin{align*}
x \in L &\Rightarrow \exists y \in \{0, 1\}^{\leq p(|x|)}, f(x, y) = 1, \\
x \notin L &\Rightarrow \forall y \in \{0, 1\}^{\leq p(|x|)}, f(x, y) = 0.
\end{align*}
\]

Thus if \( x \) is in \( L \), then there is a string \( y \) such that \( f(x, y) = 1 \). Note that there could be more than one \( y \) for which \( f(x, y) = 1 \). Such string \( y \) is called witness/proof of \( x \). The constraint that \( y \in \{0, 1\}^{\leq p(|x|)} \) can be interpreted as “\( y \) is a short witness/prooof”.

\( \text{PV} \) stands for Polynomial-time verifiable.

**Theorem** \( \text{NP} = \text{PV} \).

Before proving this theorem, let us first show that \( \text{SAT} \) is in \( \text{PV} \). Suppose, I claim that a Boolean formula \( \phi \) is satisfiable. How can I convince you that I am not lying? What “proof” I should produce? We want this “proof” should be “short” and you should be able to verify the correctness of my proof “quickly”.

Any satisfying assignment serves as a proof. Verification is checking whether the assignment indeed satisfies the formula. Observe that verification can be done “quickly” (in polynomial time).

Consider the following function. Let \( \phi \) be a Boolean formula and \( a \) be a truth assignment. \( f(\phi, a) = 1 \) if \( a \) is a satisfying assignment for \( \phi \). Else \( f(\phi, a) = 0 \).

If a formula \( \phi \) is satisfiable, then there is an assignment \( a \) such that \( f(\phi, a) = 1 \). Observe that \( |a| \leq |\phi| \). If a formula \( \phi \) is not satisfiable, then for every assignment \( a \), \( f(\phi, a) = 0 \). This shows that \( \text{SAT} \) is in \( \text{PV} \). Can you show that Hamiltonian is in \( \text{PV} \)?
Now we will show that $PV \subseteq NP$. Let $L$ be a language in $PV$. Thus there is a polynomial $p$ and a polynomial-time function $f$ such that 

$$x \in L \iff \exists y \in \Sigma^{\leq p(|x|)} f(x, y) = 1.$$ 

Let $q$ be a polynomial such that $f$ can be computed in time $q(n)$. Consider the following non-deterministic algorithm for $L$.

1. Input $x$.
2. Guess $y$ from $\Sigma^{\leq p(|x|)}$.
3. If $f(x, y) = 1$ ACCEPT, else REJECT.

Let $|x| = n$. If $x \in L$, then there is a string $y$ in $\Sigma^{\leq p(n)}$ such that $f(x, y) = 1$. In Step 2, some computation path will guess this $y$, and that computation path accepts $x$. So above program accepts $x$.

If $x \not\in L$, then for every $y \in \Sigma^{\leq p(n)}$, $f(x, y) = 0$. Any computation path of the above program accepts only when it guesses a string $y$ from $\Sigma^{\leq p(n)}$ such that $f(x, y) = 1$. Thus when $x \not\in L$, every computation path of the above program rejects $x$. So above program does not accept $x$.

Size of $\Sigma^{\leq p(n)}$ is $2^{p(n)+1}$. Thus step 2 takes $O(p(n))$ steps. The function $f$ is $q(n)$-time computable. We are evaluating the value of $f$ on $x$ and $y$. Length of $x$ is $n$ and length of $y$ is at most $p(n)$. Thus we are evaluating $f$ on an input whose size is atmost $n + p(n)$. This step takes $q(n+p(n))$ time. Thus the total time is $O(p(n) + q(n+p(n)))$. Since both $q$ and $p$ are polynomials, this time is bounded by a polynomial. Thus $L$ is in $PV$.

We will skip the other direction of the proof, $NP \subseteq PV$.

Completeness

A language $A$ is polynomial-time many-one reducible to $B$, $A \leq^p_\text{m} B$, if there is a total polynomial-time computable function $f$ such that $x \in A$ if and only if $f(x) \in B$.

If $A \leq^p_\text{m} B$ and $B \in P$, then $A \in P$.

A language $L$ is $NP$-complete if $L \in NP$ and for every $L' \in NP$, $L' \leq^p_\text{m} L$.

Most prominent NP-complete problem is $SAT$. If $SAT$ is in $P$ then, $NP = P$. Thus $P = NP$ if and only if $SAT \in P$.

In addition to $SAT$, thousands of problems that arise in practice turn out be NP-complete. Some other example of NP-complete problems: Hamiltonian, Vertex Cover, Clique, Traveling Sales Person.

Similarly, we can define $EXP$-completeness. A language $L$ is $EXP$-complete if $L \in EXP$ and for every $L' \in EXP$, $L' \leq^p_\text{m} L$.

Hierarchy Theorems

We will state time and hierarchy theorems without proof.

Let $t$ and $T$ be two functions from $\mathbb{N}$ to $\mathbb{N}$ such that $t(n) \log t(n) \in o(T(n))$. There is a language in $DTIME(T(n))$ that is not in $DTIME(t(n))$, and thus $DTIME(t(n)) \subset DTIME(T(n))$. 

2
For example, $\text{DTIME}(n) \subset \text{DTIME}(n^2)$, and $P \subset \text{EXP}$.

Let $s$ and $S$ be two functions from $\mathbb{N}$ to $\mathbb{N}$ such that $s(n) \in o(S(n))$. There is a language in $\text{DSPACE}(S(n))$ that is not in $\text{DSPACE}(s(n))$, and thus $\text{DSPACE}(s(n)) \subset \text{DSPACE}(S(n))$.

Using the time hierarchy theorem, we can show that EXP-complete languages are not in P. Let $L$ be a EXP-complete language. By the time-hierarchy theorem, there is a language $L'$ in EXP that is not in $P$. Since $L$ is EXP-complete $L'$ polynomial-time many-one reduces to $L$. So if $L$ belong to $P$, then $L'$ also belongs to $P$. Since $L'$ does not belong to $P$, it follows that $L$ doe snot belong to $P$. Thus EXP-complete languages do not belong to $P$. 