We say that two encodings $e_1$, $e_2$ of a set polynomially related iff there exist functions $f_{12}, f_{21} : \{0,1\}^* \rightarrow \{0,1\}^*$ such that $f_{12}(e_1(x)) = e_2(x)$ and $f_{21}(e_2(x)) = e_1(x)$ for all instances $x$ of the problem $A$. And further both $f_{12}$ and $f_{21}$ are computable in polynomial time.

We observe from new one that all decision problems come with a class of polynomially related "good" encodings.

Lemma:

Let $Q$ be a decision problem with two polynomially related encodings $e_1$ and $e_2$. Write $e_1(Q)$ for the corresponding concrete problem, and similarly $e_2(Q)$. Then

$$e_1(Q) \in P \iff e_2(Q) \in P.$$

Proof in book (Lemma 34.1, P. 975)
Thus we can talk of a problem \( A \) being \textit{polynomial time solvable} without reference to any particular encoding.

Let \( L \) be a subset of \( \{0,1\}^* \) = \{bit strings\}

We write \( \varepsilon \) for the empty string.

Let \( A \) be an algorithm whose input is any \( x \in \{0,1\}^* \) and whose output \( A(x) \in 0, 1, \) or \( \varepsilon \) (i.e., \( A \) does not halt on \( x \)).

We say
\[ A \text{ accepts } x \text{ if } A(x) = 1 \]
\[ A \text{ rejects } x \text{ if } A(x) = 0 \]

The language \textit{accepted by} \( A \) is the set
\[ L = \{ x \in \{0,1\}^* \mid A(x) = 1 \} \]
Note: $x \in \{0, 1\}^*$ implies $A(x) = 1$, i.e., $x \in L$, and $A(x) = 0$ if $x \in \{0, 1\}^* - L$.

We say a language $L$ is decided by an algorithm $A$ if
\[
A(x) = 1 \quad \text{iff} \quad x \in L \\
\text{and} \quad A(x) = 0 \quad \text{iff} \quad x \in \{0, 1\}^* - L.
\]

We say $L \subseteq \{0, 1\}^*$ is accepted by $A$ in polynomial time if there exists a constant $K$ such that
\[
\forall x \in L : A \text{ returns with 1 in time } O(n^K) \\
\text{whenever } n = |x|.
\]

We say $L$ is decided in polynomial time by $A$ if there exists $K$ such that
\[
\forall x \in L : A \text{ returns with 1 in time } O(n^K) \\
\text{and} \quad \forall x \notin L : A \text{ returns with 0 in time } O(n^K) \\
\text{whenever } n = |x|.
\]
Notice to accept a language, an algorithm needs only worry about $x \in L$; whereas to decide a language, it must correctly accept or reject every $x \in \{0, 1\}^*$. We now give an alternate definition of the complexity class $P$.

$$P = \{ L \subseteq \{0, 1\}^* \mid \text{there exists a poly-time algorithm that accepts } L \}$$

**Theorem:**

$$P = \{ L \subseteq \{0, 1\}^* \mid \text{there exists a poly-time algorithm that decides } L \}$$

Proof on p. 977 (Theorem 34.2).
A verification algorithm is an algorithm that takes two input strings $x, y$. We call $y$ a certificate. We say $x$ verifies $y$ if there exists a certificate $y$ such that

$$A(x, y) = 1.$$ 

The language $L$ verified by $A$ is

$$L = \{ x \in \{0, 1\}^* | \exists y \in \{0, 1\}^* s.t. A(x, y) = 1 \}.$$ 

i.e. $A$ verifies $x$ if $x \in L$ and there exists $y$ such that $x \in L$. Moreover, if $x \notin L$ then no such $y$ can exist.

Ex: Consider the language (i.e. problem)

$$\text{HAM-CYCLE} = \{ G \in \{0, 1\}^* | G \text{ is a Hamiltonian graph} \}.$$ 

Notation: $\langle G \rangle = \text{bit string encoding } G$. 

Define: A Hamiltonian cycle \( C \)

in a graph \( G \) is a cycle \( C \)

such that \( V(C) = V(G) \), i.e., \( C \)

visits every vertex of \( G \).

\( C \) is called Hamiltonian if it contains a Hamiltonian cycle.

Notice it is not used to write an algorithm that verifies the language Hamiltonian.

Given a bit string \( \langle C \rangle \) (i.e., encoding of a seq. of vertices), just check that

1. \( C \) is a cycle
2. \( C \) includes all vertices

Thus, with some effort we have

\[ A(\langle C \rangle, \langle C \rangle) = 1 \]

iff \( C \) is a Hamiltonian cycle in \( G \). Furthermore, this can be done in polynomial time.
Now we define

\[ NP = \{ L : \exists \text{ poly-time algorithm } \text{ that verifies } L \} \]

more precisely, we have \( L \in NP \) iff there exists a poly-time algorithm \( A(., .) \) and a constant \( c \) such that

\[ L = \{ x : \exists y \text{ with } |y| = O(1|x|^c) \text{ st } A(x,y) = 1 \} \]

Note: To say \( A \) is poly-time here requires that the size of the certificate is polynomially bounded by the size of \( x \).

Obviously \( \emptyset \in NP \), i.e. if we can describe a language in poly-time, we can certainly verify it (just ignore the certificate).

Example: \( \text{PATH} = \{ \langle G, u, v, k \rangle \mid \text{G contains a u-v path of length at most } k \} \) is in \( P \), and also in \( NP \).
Ex. \( \text{HAM-CYCLE} = \{G \mid G \text{ is Hamiltonian} \} \)

\( \text{HAM-CYCLE} \in \text{NP} \) as we saw, but no one has shown that \( \text{HAM-CYCLE} \in \text{P} \), i.e. no poly-time decision algorithm is known to exist.

Question: is \( \text{P} = \text{NP} \) or \( \text{P} \neq \text{NP} \).

Def: We say \( L_1 \) is \text{poly-time reducible} to \( L_2 \) if there exist a poly-time computable function

\[ f : \{0,1\}^* \rightarrow \{0,1\}^* \]

such \( x \in L_1 \) if and only if \( f(x) \in L_2 \).
We write
\[ L_1 \leq_p L_2 \]
(Note: Also $x \notin L_1$ if $f(x) \notin L_2$.)

**Lemma:**
If $L_1 \leq_p L_2$, then $L_2 \in \text{P} \implies L_1 \in \text{P}.$

**II:** This is really just a formal version of the time hierarchy theorem.

**Demo:** $L \subset \{0,1\}^*$ is $\text{NP}-
\text{Complete}$ if
(1) $L \in \text{NP}$
(2) $L' \leq_p L$ for all $L' \in \text{NP}$

Theorem: If any $\text{NP}-\text{Complete}$ problem is poly-time solvable, then $P = \text{NP}.$ Likewise, if any problem in $\text{NP}$ is not poly-time solvable, then $\text{NP} \neq \text{P}$.
Thus by lemma IEP and LEP coincide.

By Prop 2.

For any LEP and LIP.

\[ P_i \]