

WE SAY THAT TWO ENCODINGS e_1, e_2 OF A ARE 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 ASSUME FROM NOW ON THAT ALL DECISION PROBLEMS COME WITH A CHICE 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

$$e_1(Q) \in P \text{ IFF } e_2(Q) \in P.$$

PROOF IN BOOK (LEMMA 34.1, P. 975)

thus we can talk of a problem
 Q being polynomial time solvable
 without reference to any
 particular encoding.

A language L is a subset

$$L \subseteq \{0,1\}^* = \{\text{bit strings}\}$$

we write ϵ for the empty
 string.

Let A be an algorithm whose
 input is any $x \in \{0,1\}^*$ and
 whose output $A(x)$ is 0, or 1,
 or no output (i.e. A does not halt on x .)

we say

A accepts x if $A(x) = 1$

A rejects x if $A(x) = 0$

The language accepted by A
 is the set

$$L = \{x \in \{0,1\}^* \mid A(x) = 1\}$$

NOTE $x \in \{0,1\}^* - L$ DOES NOT
 IMPLY $A(x) = 0$, SINCE IT IS POSSIBLE
 A DOES NOT HALT ON INPUT x .

WE SAY A LANGUAGE L IS
DECIDED BY AN ALGORITHM A IFF

$A(x) = 1$ IFF $x \in L$
 AND $A(x) = 0$ IFF $x \in \{0,1\}^* - L$.

WE SAY $L \subseteq \{0,1\}^*$ IS ACCEPTED ^{BY}
IN POLYNOMIAL TIME IFF THERE
 EXISTS CONSTANT k SUCH THAT

$\forall x \in L$: A RETURNS WITH 1
 IN TIME $O(n^k)$
 WHERE $n = |x|$.

WE SAY L IS DECIDED IN POLYNOMIAL
TIME BY A IFF THERE EXISTS
 k S.T.

$\forall x \in L$: A RETURNS WITH 1 IN TIME $O(n^k)$
 AND $\forall x \notin L$: A RETURNS WITH 0 IN TIME $O(n^k)$
 WHERE $n = |x|$.

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 .

$$P = \left\{ L \subseteq \{0,1\}^* \mid \text{THERE EXISTS A POLY-TIME ALGORITHM THAT ACCEPTS } L \right\}$$

Theorem!

$$P = \left\{ L \subseteq \{0,1\}^* \mid \text{THERE EXISTS A POLY-TIME ALGORITHM THAT DECIDES } L \right\}$$

PROOF ON P. 977 (THEM 34.2).

A Verification Algorithm is an algorithm that takes two input strings x, y . We call y a Certificate. We say A Verifies x if there exists a certificate y such that

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

The language verified by A is

$$L = \{ x \in \{0, 1\}^* \mid \exists y \in \{0, 1\}^* \text{ st } A(x, y) = 1 \}.$$

We A verifies L iff $\forall x \in L$
 $\exists y$ which can be used to prove
 that $x \in L$. Moreover iff $x \notin L$
 then no such y can exist.

Ex Consider the language (i.e. Problem)

$$\text{HAM-CYCLE} = \{ \langle G \rangle \mid G \text{ is a Hamiltonian graph} \}.$$

NOTATION: $\langle G \rangle =$ Bit strings encoding graph G .

Defn: A Hamiltonian Cycle C in a graph G is a cycle st. $V(C) = V(G)$, i.e. C visits every vertex of G .

G is called Hamiltonian iff it contains a Hamiltonian Cycle.

Notice it is not hard to write an algorithm that verifies the presence of a Hamiltonian Cycle.

Given a list strings $\langle C \rangle$ (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

$$A(\langle G \rangle, \langle C \rangle) = 1$$

iff C is a Hamiltonian Cycle in G . Furthermore, this can be done in polynomial time.

Now we define

$$NP = \{ L \subseteq \{0,1\}^* \mid \text{there exists a Poly-time Algorithm that verifies } L \}$$

more precisely, we have $L \in NP$ iff there exists a Poly-time Algorithm $A(\cdot, \cdot)$ and a constant c such that

$$L = \{ x \mid \exists y \text{ with } |y| = O(|x|^c) \text{ s.t. } A(x, y) = 1 \}$$

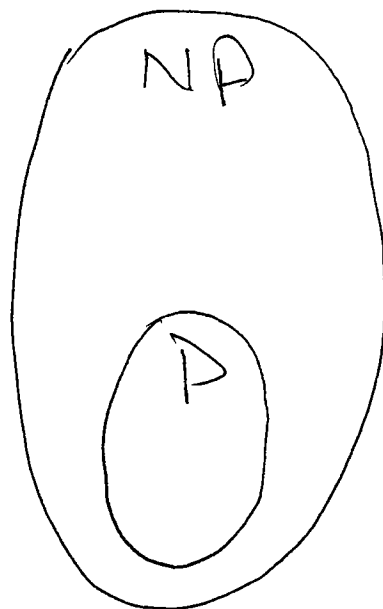
note: to say A is Poly-time here requires that the size of the certificate is polynomially bounded by the size of x .

obviously $P \subseteq NP$, i.e. if we can decide a language in Poly-time we can certainly verify it. (just ignore the certificate.)

Ex $PATH = \{ \langle G, u, v, k \rangle \mid G \text{ contains a } u-v \text{ path of len at most } k \}$ is in P , and also in NP .

Ex. $\text{HAM-CYCLE} = \{ \langle G \rangle \mid G \text{ is Hamiltonian} \}$

$\text{HAM-CYCLE} \in \text{NP}$ 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 $P = \text{NP}$ or $P \neq \text{NP}$.

DEFN: We say L_1 is Poly-Time
Reducible to L_2 iff there exist
 a Poly-Time Computable Function

$$f: \{0,1\}^* \rightarrow \{0,1\}^*$$

st. $x \in L_1$ iff $f(x) \in L_2$

was written

$$L_1 \leq_p L_2$$

(Note also $x \notin L_1$ iff $f(x) \notin L_2$.)

Lemma:

IF $L_1 \leq_p L_2$, THEN $L_2 \in P$ implies $L_1 \in P$

pf: This is really just a formal version of the theorem 1.8.1

was written

Defn: $L \subseteq \{0,1\}^*$ is NP-Complete iff

(1) $L \in NP$

(2) $L' \leq_p L$ for all $L' \in NP$

Thm: if any NP-Comp Prob is Poly-Time Solvable, THEN $P=NP$. Likewise if any Prob in NP is NOT Poly Time Solvable, THEN NONE is.

Prf:

Since $L \in P$ and $L \in NP$
 For any $L' \in NP$ we have

$$L' \leq_p L \quad \text{by}$$

By Red. 2.

Thus by lemma $L' \in P$ which
 proves 1st stmt.

2nd stmt is contr. pos.

///