WE SAY THAT TWO ENCODINGS e_1, e_2 OF A SET Polynomially RELATED IFF THERE EXIST FUNCTIONS

$$f_{12}, f_{21} : \{0,1\}^* \to \{0,1\}^*$$

SUCH THAT $f_{12}(e_1(\alpha)) = e_2(\alpha)$ AND $f_{21}(e_2(\alpha)) = e_1(\alpha)$ FOR ALL INSTANCES α 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 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

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

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 <u>LANGUAGE ACCEPTED BY A</u> 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

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

WE SAY $L \subseteq \{0,1\}^*$ IS ACCEPTED BY A 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.

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

THEOREM:

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

Proof on p. 977 (Thm 34.2).

A <u>Verification Algorithm</u> is an algorithm that takes two input strings x, y. We call y a <u>certificate</u>. We say A <u>verifies</u> x if there exists a certificate y such that

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

The language verified by A is

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

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

<u>Ex</u> Consider the language (i.e. problem)

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

<u>Notation</u>: $\langle G \rangle$ = bit string encoding graph G.

DEFN: A HAMILTONIAN CYCLE C IN A GRAPH G IS A CYCLE S.t. $V(C) = V(G)$, 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 LANGUAGE HAM-CYCLE.

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

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

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 certificates is polynomially bounded by the size of x.

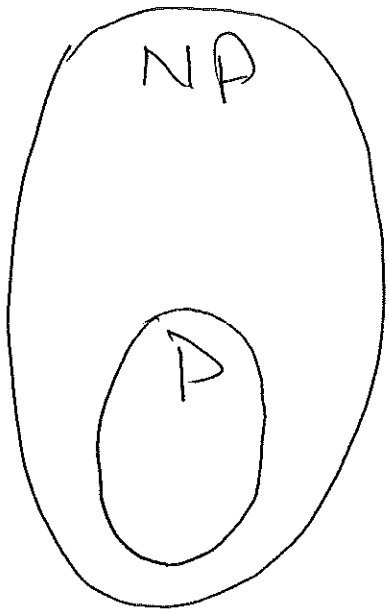
Obviously $P \subseteq NP$, i.e. if we can decide a language in poly-time we can certainly verify it. (just ignore the certificate.)

Ex $PATH = \{ \langle G, u, v, k \rangle \mid G \text{ contains a } u\text{-}v \text{ path of len at most } k \}$

is in P, and also in NP.

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

HAM-CYCLE $\in$ NP AS WE SAW, BUT NO ONE HAS SHOWN THAT HAM CYCLE $\in$ P, I.E. NO POLY-TIME DECISION ALGORITHM IS KNOWN TO EXIST.



QUESTION: IS P = NP OR P ≠ NP.

DEFN: WE SAY L_1 IS POLY-TIME REDUCIBLE TO L_2 IFF THERE EXISTS A POLY-TIME COMPUTABLE FUNCTION

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

SUCH $x \in L_1$ IFF $f(x) \in L_2$

We write

$$L_1 \leq_p L_2$$

(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 thm last time

Def: $L \subseteq \{0,1\}^*$ 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 problem in NP is NOT Poly Time solvable, then none is.

Pf:

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

$$L' \leq_p L$$

By Prob. 2.

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

2nd stmt is obvious pos.

///