- **THEOREM**
  $P \subseteq NP \subseteq EXP$

  PROOF

  - $P \subseteq NP$

    Proving that $P \subseteq NP$ means having that if $\mathcal{L} \in P$, then $\mathcal{L}$ can be written as follows

    $$\mathcal{L} = \{x \in \{0,1\}^* \mid \exists y \in \{0,1\}^{p(|x|)} . M(x,y) = 1\}$$

    where p is a polynomial and M is polytime computable. We can indeed choose $p(n) = 0$ and M as a polytime (deterministic) machine deciding $\mathcal{L}$, which exists by hypothesis. This way, indeed

    $$\mathcal{L} = \{x \in \{0,1\}^* \mid M(x) = 1\} =$$
    $$= \{x \in \{0,1\}^* \mid \exists y \in \{0,1\}^0 . M(x,y) = 1\}$$

  - $NP \subseteq EXP$

    Suppose $\mathcal{L} \in NP$. We need to prove that $\mathcal{L} \in EXP$. Since $\mathcal{L} \in NP$, there are a machine M and a polynomial p such that

    $$\mathcal{L} = \{x \in \{0,1\}^* \mid \exists y \in \{0,1\}^{p(|x|)} . M(x,y) = 1\}$$

    We can decide $\mathcal{L}$ in exponential time as follows

INPUT: $x \in \{0,2\}^*$

FOREACH $y \in \{0,2\}^{p(|x|)}$ DO

- SIMULATE $\mathcal{M}(x,y)$ UNTIL IT PRODUCES A RESULT b
- IF $b=1$ THEN RETURN 1.

RETURN 0

$|\{0,2\}^n| = 2^n$

The fact that this algorithm is correct is self-evident. But how about its complexity? We need to be sure that is exponential.

- The FOR loop is executed at most a number of times equal $2^{p(|x|)}$ [THIS IS THE CARDINALITY OF $\{0,2\}^{p(|x|)}$]
- Each iteration of the for loop takes time $q(|x|)$ where q is a polynomial because $\mathcal{M}$ works in polynomial time and both its inputs are polynomially bounded. The runtime of the algorithm is bounded by

$$q(|x|) \cdot 2^{p(|x|)} \leq 2^{\lg(q(|x|))} \cdot 2^{p(|x|)}$$
$$\leq 2^{\lg(q(|x|)) + p(|x|)}$$
$$\leq 2^{q(|x|) + p(|x|)}$$

THIS EXPRESSION IS UPPER BOUNDED BY 2^{n^c} WHERE c IS "BIG ENOUGH"

THEOREM (NDTM)

$$NP = \bigcup_{c \in \mathbb{N}} NDTIME(n^c)$$

(DTM)

PROOF

The main idea is that nondeterministic choices can be seen as a certificate

2) Suppose that $L \in \bigcup_{c \in \mathbb{N}} NDTIME(n^c)$. There is a NDTM N and a polynomial p such that N decides L and N needs at most $p(|x|)$ steps to handle any input $x \in \{0,1\}^*$. We want to prove that $L \in NP$ and thus we need to come up with a verifier and a length for certificates. The latter is just the p alone! The certificate y for any x, in other words is the $p(|x|)$-long binary strings

mode of the sequence of choices N performs on input x. The fact that this can be verified in polynomial time, comes from the fact that N can be simulated by a DETERMINISTIC Turing machine if the latter knows how to resolve the nondeterministic choices:

$M(x,y)$ = SIMULATION OF $N(x)$ WHERE THE NONDET. CHOICES ARE RESOLVED AS FOR y.

M is nothing more than the verifier you need, i.e.

$$L = \{x \in \{0,1\}^* \mid \exists y \in \{0,1\}^{p(|x|)} . M(x,y) = 1\}$$

$\subseteq$) Suppose now that $L \in NP$. This means that

$$L = \{x \in \{0,1\}^* \mid \exists y \in \{0,1\}^{p(|x|)} . M(x,y) = 1\}$$

where M is polytime. Then we can construct a NDTM N which decides L:

- First of all, N on input x, takes advantage of its nondeterministic features and builds any possible $y \in \{0,2\}^{p(|x|)}$ nondeterministically. It can do that in time $p(|x|)$
- This way, all possible values of y are produced, each of them in a distinct nondeterministic branch
- N, then, having both x and a candidate certificate y, it can simulate M on input (x, y), without using its nondeterministic features, and if M returns 1, N halts in q_{ACCEPT}, otherwise it rejects the input x.

- The fact that N, designed this way, is correct, comes easily from the correctness of M.

The overall runtime of N is at most $p(|x|) + q(|x|)$, where q is a polynomial bounding the runtime of M

IT IS STILL A POLYNOMIAL, SO N WORKS IN POLYNOMIAL TIME, TOO!

THEOREM

1. $\leq_p$ IS A PRE-ORDER
2. If a language $\mathcal{L}$ is NP-HARD and $\mathcal{L} \in P$, then $P = NP$
3. If a language $\mathcal{L}$ is NP-COMPLETE, then $\mathcal{L} \in P$ iff $P = NP$.

PROOF.

1 → EXERCISE

2. Suppose $\mathcal{L}$ is NP-HARD and in P. Let us prove that $P = NP$. The only thing we need to prove is that $NP \subseteq P$. Let H be any language in NP. Since L is NP-HARD, then by definition $H \leq_p \mathcal{L}$. We can then thus solve any instance of H in polynomial time simply by going through $\mathcal{L}$. $\Rightarrow H \in P$ ✓

3. Suppose L is NP-complete.
We want to prove that $L \in P$ iff $P = NP$.

$\Rightarrow$) If $L \in P$ and L is NP-complete, then any language H in NP is such that $H \leq_p L$ and since $L \in P$, H is itself in P. Or a consequence, $P = NP$

$\Leftarrow$) Suppose that $P = NP$. We want to prove that L is in P. But L by hypothesis is NP-complete, thus in $NP = P$, then $L \in P$. ✓