6.6 The Class NP

We say that the TURING machine M computes $f\colon L \longrightarrow \Sigma^*$ **nondeterministically** if for each $x \in L$ there is a $y \in \Sigma^*$ such that M, given the concatenation xy of x and y as input, reaches "halt" after finitely many steps with output $f(x)$.

Example Let $\Sigma = \mathbb{F}_2$ and $L = \{(n, a, x) \in \mathbb{N}^3 \mid n \geq 2, a, x \in \mathbb{M}_n\}$. Let $f = \log_a \bmod n$ be the discrete logarithm.

For a given x let y be the logarithm of x—it doesn't matter in the definition from where we get the logarithm, in any case it exists. All the TURING machine M has to do is to check whether $a^y = x$. Then it writes y to the tape and halts.

General idea A candidate y for the solution is provided, M only does a check.

Alternative idea An unbounded number of *parallel* TURING machines each checks a different $y \in \Sigma^*$.

The set **NP** ("nondeteministic polynomial time") is defined as the set of all functions for which there exists a TURING machine M and an integer $k \in \mathbb{N}$ with:

(i) M computes f nondeterministically,

(ii) $t_M(n) \leq n^k$ for almost all $n \in \mathbb{N}$.

We have the inclusions

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{EXPTIME}.$$

The first of these is trivial, the second is a theorem that we don't prove here.

The most important unsolved problem of theoretical computer science is the conjecture

$$\mathbf{P} \overset{?}{\neq} \mathbf{NP}.$$

Likewise unproven is the conjecture

$$\mathbf{NP} \overset{?}{\neq} \mathbf{EXPTIME}.$$

On the other hand the statement

$$\mathbf{P} \neq \mathbf{EXPTIME},$$

is proven, if only by constructing "artificial" problems. There is no known "natural" problem proven to be in the difference set.

By the way we cannot make cryptanalysis of a cipher more difficult than **NP**: Exhaustion—that is trying all keys with a known plaintext—is always possible, and the encryption function must be efficient, hence in **P**.

Examples

1. If f is the discrete logarithm as above, then $f \in \mathbf{NP}$.
2. Likewise factoring integers is in **NP**.
3. Also the knapsack problem is in **NP**.

We call the function f **NP-complete** if for each TURING machine M that computes f (deterministically!) and each function $g \in \mathbf{NP}$ there exists a TURING machine N that computes g and an integer $k \in \mathbb{N}$ such that

$$t_N(n) \leq t_M(n)^k \quad \text{for almost all } n \in \mathbb{N}.$$

In other words the complexity of N is at most polynomial in the complexity of M.

Interpretation **NP**-complete problems are the maximally complex ones among those in **NP**.

It is known that **NP**-*complete problems exist.* We refrain from proving this theorem here.

For instance the knapsack problem is **NP**-complete, as is the determination of zeroes of (polynomial) functions $p\colon \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$. Factoring integers is presumably not **NP**-complete.

Should $\mathbf{P} = \mathbf{NP}$ hold—nobody believes it—, then all functions in **NP** would be **NP**-complete. If not, the following drawing illustrates the relative situation of the complexity classes: