P and NP, intuitively speaking

(a) P is the class of problems for which a solution can always be found within polynomial time.

(b) NP is the class of problems for which we can check, within polynomial time, whether a potential solution is indeed a solution.

Part (b) can be made precise as follows.

Let Σ be an alphabet. A relation $R \subseteq \Sigma^* \times \Sigma^*$ is called <u>polynomially decidable</u> if there is a TM which decides the language

$$\{ vw : (v,w) \in R \}$$

in polynomial time.

We call R <u>polynomially balanced</u> if

$$(v,w) \in R \implies |w| \leq |v|^k \text{ for some } k \geq 1.$$

<u>Prop. 9.1</u> Let $L \subseteq \Sigma^*$ be a language. Then, $L \in NP \iff$ there is a polynomially decidable and polynomially balanced relation $R \subseteq \Sigma^* \times \Sigma^*$ such that

$$L = \{ v : \exists w \in \Sigma^*, (v,w) \in R \}.$$

(In the language of (6), a 'potential solution' for input v is a pair (v,w).)

coNP

Def. $L \in coNP \iff$

$$\bar{L} = \{w : w \notin L\} \in NP$$

Example The problem of deciding, given a propositional formula φ, whether $\neg\varphi$ is satisfiable is in NP, in fact it is NP-complete.

Since φ is valid $\iff \neg\varphi$ is not satisfiable, the problem VALIDITY of deciding whether a propositional formula is valid is in coNP, even coNP-complete.

The following results are straightforward consequences of the definitions:

Prop. 10.1 If L is NP-complete, then its complement $\bar{L}$ is coNP-complete.

Prop 10.2 If a coNP-complete problem belongs to NP, then coNP = NP.

Primality

PRIMES is the problem of deciding whether a given (positive) integer is a prime.
In 2004 a proof was published showing that PRIMES $\in P$.

Nevertheless it is instructive to understand the (easier) proof of the (following) older and weaker result:

Thm. PRIMES $\in NP \cap coNP$.

For this we need a number theoretic result:

Thm. 10.1 An integer $p > 1$ is prime $\iff$ there is $1 < r < p$ such that $r^{p-1} = 1, \bmod p$, and $r^{\frac{p-1}{q}} \neq 1, \bmod p$, for all prime divisors q of $p-1$.

Here are the ideas behind the proof of $PRIMES \in NP \cap coNP$.

We can check within polynomial time whether a potential nontrivial factorization of a number p is indeed a factorization of p. Hence $PRIMES \in coNP$.

Now I sketch the proof that $PRIMES \in NP$, where Thm. 10.1 is used.

Recall Prop. 9.1. It is sufficient to find a polyn. decidable and polyn. balanced relation R such that p is prime $\iff$ $(p, w) \in R$ for some w.

Numbers are represented as binary strings, so the representation of $n \in \mathbb{N}$ has length $\log_2 n$.

Multiplication and division can be carried out in time $O(\ell^2)$ where ℓ is the length of the binary repres. of the largest of the two numbers.

$r^{p-1} \pmod p$ can be calculated by first calculating $r^2, r^{2^2}, r^{2^3}, \ldots, r^{2^\ell}$, modulo p, where $\ell = \lceil \log_2 p \rceil$; this requires ℓ multiplications.

Then, by considering the binary representation of $2^\ell - (p-1)$ and the fact that $r^{p-1} = \dfrac{r^{2^\ell}}{r^{2^\ell-(p-1)}}$,

we can calculate r^{p-1}, mod p, with ℓ more divisions, and then see if $r^{p-1} = 1$.

For all this the time needed is
$O(\ell^3) = O(\log_2^3 p)$.

Similarly, for every prime divisor q of $p-1$, we can check, in time $O(\ell^3)$ whether $r^{\frac{p-1}{q}} \neq 1, \bmod p$; and there are at most ℓ $(= \lceil \log_2 p \rceil)$ prime divisors of $p-1$.

If p is a prime let

$C(p) = (r, C(q_1), \ldots, C(q_k))$

where $q_1, \ldots, q_k$ are all prime divisors of $p-1$ and $1 < r < p$ satisfies Thm. 10.1. This definition is inductive but terminates because $q_1, \ldots, q_k$ are smaller than p.

Now define $(p, w) \in R \iff w = C(p)$.

Using induction one can verify that R is polyn. decidable, polyn. balanced and that

p is prime $\iff (p, w) \in R$ for some w. $\square$

Function problems

Let $L \in NP$. So there is a polyn. decidable and polyn. balanced relation R such that

$$L = \{v : \exists w, (v, w) \in R\}.$$

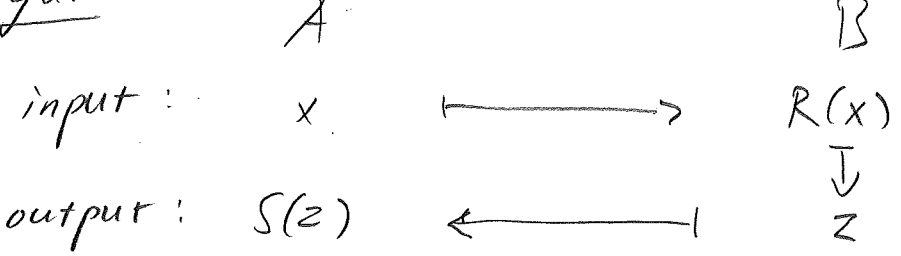
Define "the function problem associated with L", denoted FL, to be the problem of computing w such that $(v, w) \in R$ for any given v, if such w exists; otherwise the output is "no".

Example FSAT is the problem of computing a truth assignment which satisfies φ, if such exists.

Def. The function problem A reduces to the function problem B if there are string functions R and S computable in logarithmic space such that:

- x is an instance of $A \Rightarrow R(x)$ is an instance of B, and
- z is a correct output of $R(x)$ $\Rightarrow S(z)$ is a correct output of x (if x is an instance of A).

Figure:



Def. $FNP = \{FL : L \in NP\}$

$FP = \{FL : L \in P\}$.

Def. Let $A \in FNP$. We say that A is FNP-complete if every $B \in FNP$ reduces to A.

Example FSAT is FNP-complete.

Thm 10.2 $FP = FNP \iff P = NP$.

See Example 10.3, which considers FSAT and SAT, for an explanation.

Hence, function problems of the form FL where L is a language don't offer much new.

But there are interesting function problems (in the sense that the solution can be something more complex than "yes" or "no") which cannot be obtained as FL for some non trivial L.

Example Let FACTORING be the problem of producing a prime decomposition $p_1^{k_1} \cdots p_m^{k_m}$ for any given integer $n \geq 2$.

If we require that the output must also include $C(p_1), \ldots, C(p_k)$, as in the proof that PRIMES $\in$ NP, then we get the problem FACTORING', which is in FNP.

Random algorithms for primality.

Some number theory:

The "Legendre symbol" is defined as

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{if } a \equiv 0 \pmod p \\ 1 & \text{if } a \not\equiv 0 \pmod p \text{ and} \\ & \quad x^2 \equiv a \pmod p \text{ for some } x. \\ -1 & \text{if no such } x \text{ exists.} \end{cases}$$

for any positive $a \in \mathbb{N}$ and odd prime p.

For any odd $n \in \mathbb{N}$, the "Jacobi symbol" is

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{\alpha_1} \left(\frac{a}{p_2}\right)^{\alpha} \cdot \ldots \cdot \left(\frac{a}{p_k}\right)^{\alpha_k},$$

where $n = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_k^{\alpha_k}$ is the prime decomposition of n.

(By convention, $\left(\frac{a}{1}\right) = 1$.)

Given odd n consider the congr.

(1) $a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) \pmod{n}$.

(2) If n is prime then (1) is true for all a.

(3) If n is not prime then for at least half of all $a \in \{1, \ldots, n-1\}$ (1) fails.

the Solovay-Strassen primality test:

Input: n and k (which determines the accuracy of the test).

Repeat k times:
Choose a randomly from $\{1, \ldots, n-1\}$
If (1) fails, stop and reply "not prime".
Reply "probably prime".

In binary representation the length of n is $\lceil \log_2 n \rceil = \ell$ and $\left(\frac{a}{n}\right)$ can be computed in time $O(\ell^2)$. As we have seen before, $a^{\frac{n-1}{2}}$, mod n, can be computed in time $O(\ell^3)$, so the running time of the algorithm is $O(k\ell^3)$, where ℓ is the length of the input n (in binary). The probability of failure is 2^{-k}.