

Completeness

In the partial order enforced by reducibility we are interested in the **maximal** elements of the partial order.

Definition 1. *Let C be a complexity class and let L be a language in C . We say that L is C -complete if any language $L' \in C$ can be reduced to L .*

Definition 2. *A class C' is closed under reductions if, whenever L is reducible to L' and $L' \in C'$, then also $L \in C'$.*

Theorem 1. *P , NP , $PSPACE$, and EXP are closed under reductions.*

If a complete problem belongs to a weaker class $C' \subseteq C$ then the whole class coincides with C' as long as C' is closed under reductions.

Theorem 2. *If two classes C and C' are closed under reductions, and there is a language L which is complete for both C and C' , then $C = C'$.*

NP-Complete Problems

The “hardest” problems in NP.

A problem B is NP-complete if it satisfies two conditions:

1. B is in NP and
2. B is **NP-Hard**: every A in NP is polynomial time reducible to B .

P vs. NP

Theorem 3. *If any NP-complete problem is polynomial-time solvable then $P = NP$. Equivalently, if any problem in NP is not polynomial time solvable, then no NP-complete problem is polynomial-time solvable.*

Proof. Suppose $A \in P$ and A is NP-complete.

Let B be any NP problem.

We can reduce B to A in polynomial time and hence we can solve B in polynomial time. \square

A NP-complete Problem

Theorem 4. [Cook-Levin Theorem] *SAT is NP-complete.*

Proof: The problem is in NP. We show that all languages in NP reduces to CIRCUIT SAT (which reduces to SAT).

Let $L \in NP$. We give a reduction R which for each string x constructs a circuit $R(x)$ such that $x \in L$ iff $R(x)$ is satisfiable.

Since $L \in NP$ there is a NTM $M = (K, \Sigma, \Delta, s)$ that decides L in time n^k . We can assume that M has a single string and its degree of non-determinism is 2.

Let the sequence of nondeterministic choices be a bitstring: $(c_1, \dots, c_{|x|^{k-1}}) \in \{0, 1\}^{|x|^{k-1}}$.

Fix a particular sequence of choices $c = (c_0, \dots, c_{|x|^{k-1}})$: we define a computation table $T(M, x, c)$ corresponding to this (deterministic) computation.

Computation Table for a (Deterministic) Machine

Given a (deterministic, single-string) TM $M = (K, \Sigma, \delta, s)$ deciding language L , we define a **computation table** $T(M, x)$ on input x :

(1) A table of size $|x|^k \times |x|^k$, where $|x|^k$ is the time bound.

Rows are time steps (from 0 to $|x|^k - 1$) and columns are positions in the machine's string, i.e., (i, j) th entry represents the contents of position j of the string of M at time i .

If at time i the cursor scans the j th position and the state is q , then the (i, j) th entry contains the symbol σ_q (or yes/no).

(2) W.l.o.g, assume that x halts after at most $|x|^k - 2$ steps.

The string is padded with blanks to the right so that its total length is $|x|^k$.

W.l.o.g, assume that the cursor starts at the first symbol of the input (and not at \triangleright) and never visits the leftmost \triangleright .

If the machine halts before its time bound n^k then all subsequent rows will be identical.

Lemma 1. *M accepts x iff the computation table of M on input x is accepting (i.e., $T_{|x|^k-1,j} = \text{yes}$ for some j).*

Using a Boolean Circuit to Capture Deterministic Computation

T_{ij} depends only on the entries $T_{i-1,j-1}$, $T_{i-1,j}$, and $T_{i-1,j+1}$.

Let Γ denote the set of all symbols that can appear on the table.

Encode each symbol $\sigma \in \Gamma$ as a vector (s_1, \dots, s_m) where $(s_1, \dots, s_m) \in \{0, 1\}$ and $m = \lceil \log |\Gamma| \rceil$.

Thus the computation table can be thought of as a table of binary entries S_{ijl} with $0 \leq i \leq n^k - 1$, $0 \leq j \leq n^k - 1$ and $1 \leq l \leq m$.

Each entry $S_{i,j,l}$ depends only on $3m$ entries of the previous row.

There are boolean functions F_1, \dots, F_m with $3m$ inputs each such that for all $i, j > 0$

$$S_{ijl}$$

$$= F_l(S_{i-1,j-1,1}, \dots, S_{i-1,j-1,m}, S_{i-1,j,1}, \dots, S_{i-1,j+1,m})$$

Thus there is a Boolean circuit C with $3m$ inputs and m outputs that computes the binary encoding of T_{ij} given the encodings of $T_{i-1,j-1}$, $T_{i-1,j}$ and $T_{i-1,j+1}$, for all i and for all j .

Circuit C depends only on M and has a fixed, constant size independent of the length of x .

To capture the entire computation table $T(M, x)$ we use $(|x|^k - 1)(|x|^k - 2)$ copies of C .

The input gates of the overall circuit correspond to the first row and the first and last column.

The output is the first output of the circuit $C_{|x|^k-1,1}$ (assuming that M ends with yes/no in its second string and the first bit says whether it is yes/no).