

Polynomial-time Reductions

We want to solve a decision problem A in polynomial time.

Suppose we know how to solve a different decision problem B in polynomial time.

We can solve A using B if we have a **polynomial-time** procedure R that transforms **any** instance x of A into some instance $R(x)$ of B such that the answer for x is “yes” if and only the answer for $R(x)$ is “yes”.

B is at least as hard as A .

Suppose A is a “hard” problem, say, it has no polynomial time algorithm.

Then the above reduction can be used to show that B has no polynomial time algorithm as well.

Formal Definition of Reduction

Definition 1. [Polynomial time Reduction] We say a language L_1 is reducible to L_2 if there is a function R from strings to strings computable by a deterministic TM in polynomial time such that for all inputs x the following is true: $x \in L_1$ iff $R(x) \in L_2$. R is called a **reduction** from L_1 to L_2 .

Theorem 1. If R is a reduction from language L_1 to L_2 and R' is a reduction from L_2 to L_3 , then the composition $R \cdot R'$ is a reduction from L_1 to L_3 .

Reducibility is transitive: gives an “ordering” to the problems based on their hardness.

Example 1

Theorem 2. *CIRCUIT SAT can be reduced to SAT.*

Proof: Given a circuit C we will construct a Boolean expression $R(C)$ such that $R(C)$ is satisfiable iff C is satisfiable.

The variables of $R(C)$ contain all the variables of C plus a variable g for each gate g of C . For each gate of C we generate clauses of $R(C)$:

(a) g is a variable gate corresponding to variable x :
 $(\neg g \vee x)$ and $(g \vee \neg x)$

(b) g is a True gate: (g)

(c) g is a False gate: $(\neg g)$

(d) g is a not gate and h feeds it: $g \equiv \neg h$

(e) g is a OR gate with h and h' feeding it:
 $g \equiv (h \vee h')$

(f) g is a AND gate with h and h' feeding it:
 $g \equiv (h \wedge h')$

(g) g is a output gate: (g)

Example 2

Theorem 3. *HAMILTON PATH can be reduced to SAT.*

Given a graph G we will construct a Boolean expression $R(G)$ such that $R(G)$ is satisfiable iff G has a Hamilton path.

Let G has n nodes, $1, 2, \dots, n$.

$R(G)$ has n^2 Boolean variables x_{ij} , $1 \leq i, j \leq n$.

x_{ij} : "node j is the i th node in the Hamilton path".

$R(G)$ in CNF form has clauses of type:

(a) $\bigvee_{i=1}^n x_{ij}$ for each j .

(b) $(\neg x_{ij} \vee \neg x_{kj})$ for all j and $i \neq k$.

(c) $\bigvee_{j=1}^n x_{ij}$ for each i .

(d) $(\neg x_{ij} \vee \neg x_{ik})$ for all i , and $i \neq k$.

(e) $(\neg x_{ki} \vee \neg x_{k+1,j})$ for each pair (i, j) not in G and for $k = 1, \dots, n - 1$.

The reduction can be done in polynomial time.

Completeness

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

Definition 2. *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 3. *A class C' is closed under reductions if, whenever L is reducible to L' and $L' \in C'$, then also $L \in C'$.*

Theorem 4. *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 5. *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 $C = C'$.*