Polynomial time Reductions

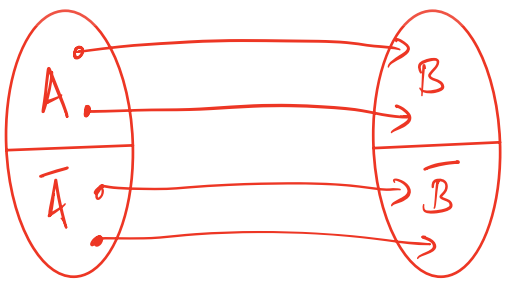
We had seen mapping reductions earlier. This is similar, but with a constraint on the machine that computes the reduction.

Def 7.28: $f : \Sigma^* \to \Sigma^*$ is a **polynomial time computable function** if there is some polynomial time DTM M that takes input w, writes $f(w)$ on the tape and halts.

Def 7.29: Language A is **polynomial time reducible** to language B, denoted $A \leq_p B$, if $\exists$ poly time computable function f such that $\forall\, w \in \Sigma^*$,

$$w \in A \iff f(w) \in B$$

f is called the polynomial time reduction from A to B.



Goal: This gives one way to decide A efficiently.

(1) Transform A to B : Compute $f(w)$

(2) Decide B.

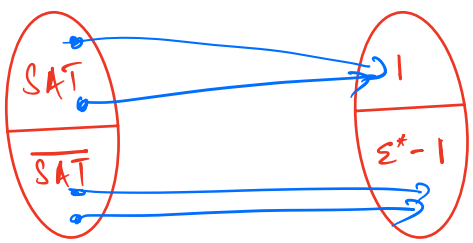
We will see that $A \leq_p B$ and $B \in P \Rightarrow A \in P$.

This restriction is important.

We cannot replace by $A \leq_m B$

The restriction that f should be polynomial time computable is essential. If there are no bounds on the time it takes to compute f, we can take exponential time and solve 3-COLORING or SAT.

$$f(\phi) = \begin{cases} 1 & \text{if } \phi \text{ is satisfiable} \\ 0 & \text{otherwise} \end{cases}$$



We can get easy reduction from SAT to $\{1\}$ if we don't impose restrictions on the power of the reduction function.

Theorem 7.31: $A \leq_p B$ and $B \in P \Rightarrow A \in P$.

Proof: Suppose M is the polynomial time decider for B. We can construct a decider for A as follows:

Assume f is the poly time reduction.

On input w:

(1) Compute $f(w)$. $\rightarrow n^{k_1}$

(2) Run M on $f(w)$.

Accept iff M accepts $f(w)$. } $n^{k_1 k_2}$.

Correctness: Straightforward.

Time: Both steps (1) and (2) are poly time.

Suppose $f(w)$ takes n^{k_1} time ($n = |w|$)

Suppose M runs in $|f(w)|^{k_2}$ time.

Since time taken for the reduction is n^{k_1}, we have $|f(w)| \leq n^{k_1}$.

M takes $(n^{k_1})^{k_2}$ time $= n^{k_1 k_2}$.

So the A-decider takes $n^{k_1 k_2}$ time.

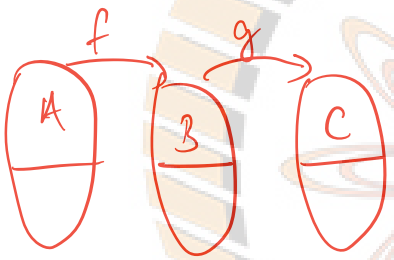
Other results

(1) $A \leq_p B$ and $B \in NP \Rightarrow A \in NP$

(2) $A \leq_p B$ and $A \notin P \Rightarrow B \notin P$

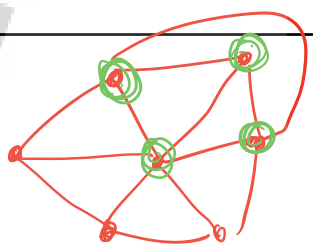
(3) $A \leq_p B$ and $B \leq_p C \Rightarrow A \leq_p C$

(4) $A \leq_p B \Rightarrow \overline{A} \leq_p \overline{B}$.



$g \cdot f$ gives a reduction from A to C.

Theorem 7.32: $3\text{-SAT} \leq_p \text{CLIQUE}$.



$3\text{-SAT} = \{\langle \phi \rangle \mid \phi$ is a 3-CNF formula, ϕ is satisfiable$\}$

$\text{CLIQUE} = \{\langle G, k \rangle \mid G$ is an undirected graph with a k-clique$\}$

Suppose ϕ has n variables m clauses.

$\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \ldots \wedge (a_m \vee b_m \vee c_m)$

where each of a_i, b_i, c_i are a literal $(x_j / \bar{x}_j)$.

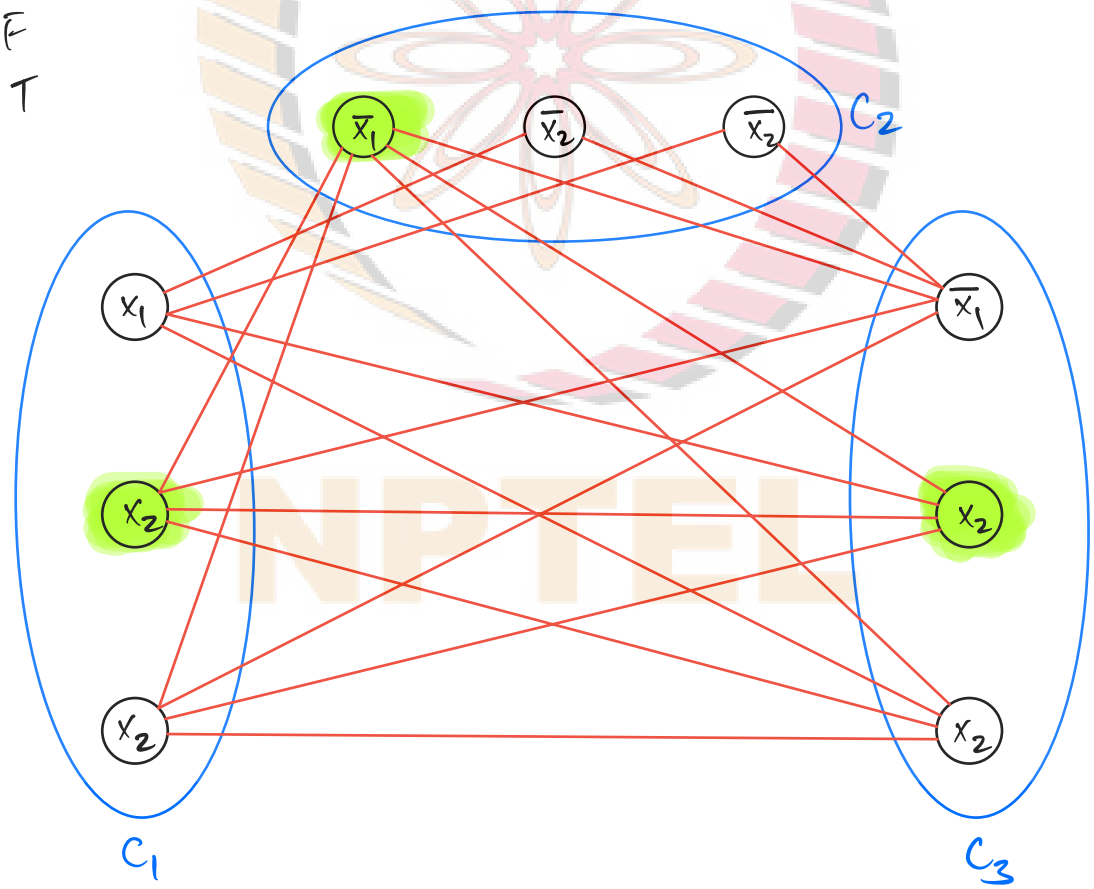
Given ϕ, we need to construct a CLIQUE instance $\langle G, k \rangle$ such that

$$\phi \in 3\text{-SAT} \iff \langle G, k \rangle \in \text{CLIQUE}$$

We will demonstrate the reduction through an example.

Consider $\phi = (x_1 \vee x_2 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_2)$.

$x_1 = F$
$x_2 = T$



G has $3m$ vertices, where m is the number of clauses. Each clause corresponds to a set of 3 vertices.

Edges: We add edges between any two vertices, except when (1) they are from the same clause and (2) they are labelled x_i and $\overline{x_i}$ for the same i.

The construction is straightforward given ϕ, and takes only $O(m^2)$ time.

Now we need to show the following:

ϕ is satisfiable $\iff$ G has m clique.

$\phi \in 3\text{-SAT} \iff \langle G, m \rangle \in CLIQUE$

($\Rightarrow$) $\phi \in 3\text{-SAT} \Rightarrow$ $\exists$ an assignment which sets each clause to true.

$\Rightarrow$ Every clause has at least one true literal.

$\Rightarrow$ Choose one true literal from each clause. These correspond to m vertices.

$\Rightarrow$ These m vertices will be adjacent to one another.

(Since we cannot have both x_i and $\bar{x}_i$ set to True in a satisfying assignment)

These m vertices form a clique. So

$\langle G, m \rangle \in$ CLIQUE.

($\Leftarrow$) $\langle G, m \rangle \in$ CLIQUE $\Rightarrow$ G has m clique.

$\Rightarrow$ The clique contains exactly one vertex from each clause.

$\Rightarrow$ We cannot have x_i & $\bar{x}_i$ both in the clique, for the same i. (Since by construction, there are no edges between them)

So we have m non-contradicting literals, one from each clause. We can assign true to all of them.

Each clause is true $\Rightarrow \phi$ is satisfied.

$\Rightarrow \phi \in$ 3-SAT.

The above assignment may leave some variables unassigned. It does not matter what we assign to these variables.

Note: The above reduction uses tools / construction to convert one problem into another. These tools are called gadgets.