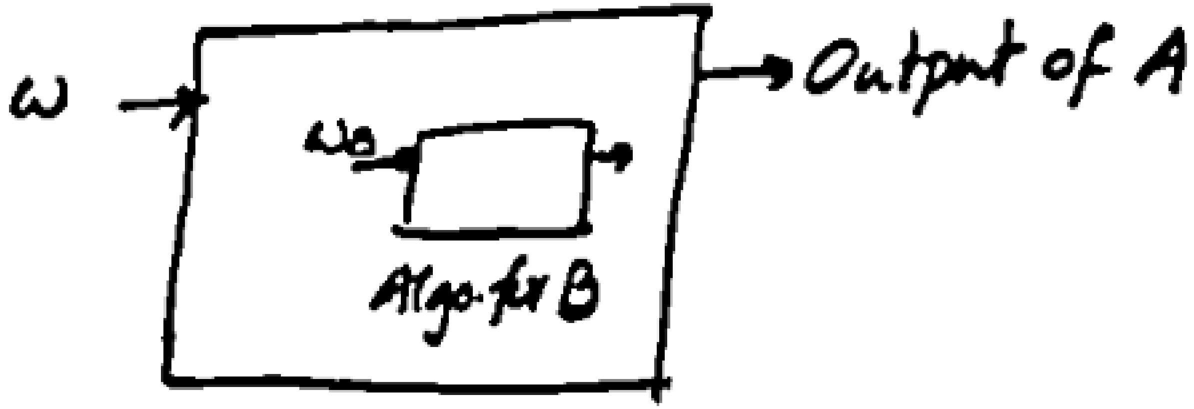
· Polynomial-time reduction - contd.

- Formally, A problem A (or a 'language' A) is polynomial time reducible to problem B, denoted as $A \leq_p B$ if we can design a polynomial time algorithm for solving A assuming there is a poly. time algo for B.



Algorithm for A

- Literally, it means solving A in poly. time "reduces to" solving B in poly. time.

or, if $A \leq_p B$ and $B \in P$, then $A \in P$.

- Some more problems in NP and some more polynomial reductions.

- 3-SAT:

Similar to SATISFIABILITY problem (SAT), but boolean formula is given in conjunctive normal form (product of sums) and each clause has 3 literals.

Example:

$$\phi: (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_3 \vee \bar{x}_4 \vee x_5)$$
$$\wedge (\bar{x}_1 \vee x_6 \vee x_7) \wedge (x_2 \vee x_4 \vee \bar{x}_6).$$

Question remains same:

Is ϕ satisfiable?

(Easy to see it is in NP).

- Is this problem simpler than the general satisfiability problem?
- As far as having a polynomial-time solution is concerned, the answer is NO.

i.e., $SAT \leq_p 3\text{-}SAT$.

- How to give a proof?

- By converting any boolean formula ϕ (in CNF, or product-of-sums form) to a formula ϕ' in 3-SAT form.

- 3-literal clauses can be kept as it is.

How to convert other clauses?

1. One-literal clause: First convert to a 2-literal clause.

$$x_1 \Rightarrow (x_1 + x_2)(x_1 + \bar{x}_2)$$

We used one additional variable.

(even if it was $\bar{x}_1$, it works).

(SUDEEP)

- When replacing one clause with 2 or more other clauses, it must be ensured that
  - the new set of clauses are satisfiable if and only if the original clause is satisfiable.

2. Two literal clause to 3 literals:

$(l_1 + l_2)$ can be replaced with

$$(l_1 + l_2 + z_1) \wedge (l_1 + l_2 + \bar{z}_1).$$

- Same idea as before. If $(l_1 + l_2)$ is true, this is also true (irrespective of the value of z_1); if $(l_1 + l_2)$ is False, so is this.

(SUDEEP)

- Case 3: What to do when there are four literals in a clause?
  - Idea: $l_1+l_2+l_3+l_4$ is false means all the four literals are false.
- Replace it with $(l_1+l_2+z_1)(l_3+l_4+\bar{z}_1)$
  - Argument is different from cases 1&2.

- Here, z_1's value matters.
- But we can still say that if the original clause was true, this product is <u>satisfiable</u>.
- If l_1 or l_2 is true, make z_1 = false. Else:
- If l_3 or l_4 is true, make z_1 = true.

• Also, if the original clause is false, i.e., when l_1, l_2, l_3 and l_4 are false, no value of z_1 will make this true.

• i.e.,

$l_1 + l_2 + l_3 + l_4$ is true (or satisfiable)

$\Longleftrightarrow (l_1 + l_2 + z_1)(l_3 + l_4 + \bar{z}_1)$ is satisfiable.

- Applying this idea twice or more, we can handle the clauses that contain five or more literals.
- $l_1 + l_2 + l_3 + l_4 + l_5$ is first replaced by

$$(l_1 + l_2 + z_1)(\bar{z}_1 + l_3 + l_4 + l_5);$$

In one more step, we get

$(l_1 + l_2 + z_1)(\bar{z}_1 + l_3 + z_2)(\bar{z}_2 + l_4 + l_5)$.

- Note that when l_1 to l_5 are all false; this becomes $z_1(\bar{z}_1 + z_2)\bar{z}_2$.

- this can never be true.

- In general, $(l_1 + l_2 + \dots l_k)$ becomes $(l_1 + l_2 + z_1)(\bar{z}_1 + l_3 + z_2) \dots\dots (\overline{z_{k-3}} + l_{k-1} + l_{k-2})$.