Theory of computation **Fall 2015**

Lecture 21: Problem Reduction

Faculty: K.R. Chowdhary *: Professor of CS*

21.1 Problem Reduction

Let there are problems A and B, with A not harder than B, is represented as $A \leq B$. Now, if A is *undecidable*, then B is also undecidable. Similarly, if B is decidable, then A is also decidable. Thus, given a hardness relation between two problems, and given one is hard or soft, we can conclude the hardness or softness of other problem.

The relation $A \leq B$ means that problem A can be solved using the solution to B. To be precise, to get solution of A for the instance w, what we need to do is to modify w some how to obtain w' and then obtain solution to B for the instance w', so that the solution of B for w' turns out to be the solution of A for w also. In this case, "$A \leq B$" means that problem "A is reducible to problem B". Further, the correspondence between w and w' is a function f such that $w' = f(w)$.

Definition 21.1 *Let A and B are Yes/No problems, and $A \leq B$ if there is a function $f : \Sigma^* \to \Sigma^*$, such that,*

$$A(w) = Yes \Leftrightarrow B(f(w)) = Yes,$$

for all $w \in \Sigma^*$. The function f is called reduction of A to B. When A and B are membership problems of language, language A is said to be reducible to language B, where languages are thought to be membership problems.

Whenever A and B are languages or problems, if reduction $A \leq B$ is computable by a TM, A is said to be mapping reducible to B, denoted by $A \leq_{mred} B$. Similarly, if the reduction is computable by a TM in polynomial time, we say $A \leq_P B$. Since, all the polynomial problems are computable, $A \leq_P B$ is simply written as $A \leq B$. In this reduction by TM using a reduction function $f : \Sigma^* \to \Sigma^*$, is polynomial if there exists integers k and m such that for $w \in \Sigma^*$, $f(w)$ is computed in $k|w|^m$ steps.

There is yet another concept of reducibility, called *Turing reducibility*. The Turing reducibility is defined in terms of an *Oracle*. An Oracle is a device, that is equipped to a TM: when the TM asks an Oracle whether any string w is member of a certain language, say B, the oracle answers it Yes/No to the TM depending on whether $w \in B$ or $w \notin B$. Such a TM which has capability to query an oracle about membership, is called "Oracle TM", and denoted by M^B. To be specific, a Oracle TM queries an oracle by placing a string on a special oracle tape and then obtain Yes/No from Oracle in a single computation step.

A language A is Turing reducible to B if there exists an oracle TM with an oracle of B that decides A. This is denoted by $A \leq_{Turing} B$. This Turing reducibility can be thought of as a generalization of mapping reducibility.

If $A \leq B$ holds for languages A and B, then,

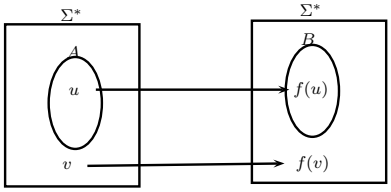

Figure 21.1: Reduction concept explained by sets.

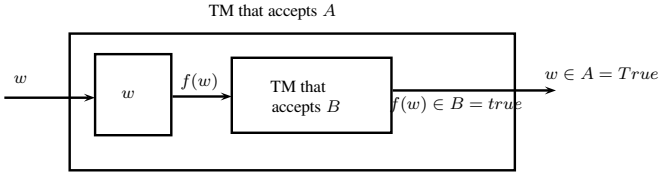

Figure 21.2: Reduction concept explained by Turing machine.

$$w \in A \Leftrightarrow f(w) \in B,$$

where f is reduction (see fig. 21.1).

The figure 21.2 illustrates how to construct a Turing machine that decides the membership problem for language A by using the Turing machine that decides the membership problem for language B.

Theorem 21.2 *(a) If $A \leq B$ and $B \leq C$ are mappings for problem A, B, C, then show that $A \leq C$. (Note $A \leq B$ is called mapping reducibility). (b) Also, if $A \leq_P B$ and $B \leq_P C$, then show that $A \leq_P C$.*

Proof. (a) Let $A \leq B$ and $B \leq C$ with reduction function f, g respectively. The $g(f(w))$ is reduction for $A \leq C$. This is possible as follows: For any input w to A,

$$\begin{aligned} A(w) = yes &\Leftrightarrow B(f(w)) = Yes, \ for A \leq B \\ &\Leftrightarrow c(g(f(w))) = Yes, \ for\ B \leq C \end{aligned}$$

Also, if f and g are computed by Turing machines M_f and M_g, respectively, then $g(f(w))$ can be computed by first converting input w to $f(w)$ by M_f then feeding $f(w)$ as input to M_g to obtain $g(f(w))$ as output. Let the TM M_{gf} computes this reduction of $A \leq C$.

(b) For polynomial time reducibility, suppose f and g are computed by TM M_f and M_g in time an^k and bn^l, respectively, where the input to A is $|w| = n$ and a, b, k, l are integer constants.

Then, since the length of an output is obviously upper bounded by the steps of M_f, we have $|f(w)| \leq an^k$. So the steps of M_g are upper bounded by $b|f(w)| \leq b(an^k)^l = ba^l n^{kl}$. Thus, M_{gf} computes the deduction $g(f(w))$ in at most $an^k + ba^l n^{kl}$ steps or time, which is in polynomial. This completes the proof.

Theorem 21.3 *(a) If language B is decidable and $A \leq B$ holds, then A is also decidable. (b) If B is computable in polynomial time, and $A \leq B$ holds, then A is computable in polynomial time.*

Proof: (a) Let the reduction $A \leq B$ is done by function f and corresponding Turing machine is denoted by M_f. Let TM to compute B be denoted by M_B.

Having computed B, the A is computed in similarly in (a) and (b), by TM M_A as follows:

1. *Input w,*
2. *run TM M_f on input w and yield $f(w)$ as output,*
3. *run TM M_B on input $f(w)$ and get its output as output of M_A.*

As per the definition, TM M_A decides A. Hence, if f and B are computable in polynomial time, then A is also computable in polynomial time. This proves the theorem.

Theorem 21.4 *If $A \leq_P B$ and $B \in P$, then $A \in P$.*

proof: *Let M be polynomial time algorithm deciding B, and f be polynomial time reduction from A to B. We describe polynomial time algorithm for M' for A as follows:*

M' = Input w, step 1. compute $f(w)$ on TM R (reducer for f), step 2. Run M on input $f(w)$. Therefore, M' is polynomial because each of above steps are polynomial (Note: Composition of two polynomial functions is polynomial).