Polynomial reductions & NP-completeness

Def: A function $f: \Sigma^* \to \Sigma^*$ is *ptime computable* if there is a transducer M that computes f and $\forall w \in \Sigma^*$, M runs in time $\text{Poly}(|w|)$.
(Class of such functions is sometimes denoted FP).

Note: For $f \in FP$,
$|f(w)| = \text{Poly}(|w|)$.

Prop: $f, g \in FP \Rightarrow f \circ g \in FP$.

Proof: To compute $f \circ g$:
"On input w:
1. Let $x := g(w)$ | $\text{Poly}(|w|)$
2. Let $y := f(x)$ | $\text{Poly}(|x|)$ $= \text{Poly}(\text{Poly}(|w|))$
3. Output y" | $= \text{Poly}(|w|)$

This is ptime.

Def: A *polynomial reduction* is an m-reduction that is ptime computable.

An analogy

Resource unbounded	Time-bounded (efficient)
decidable	"efficiently decidable" [i.e., "in the class P"]
T-rec.	"efficiently verifiable" [i.e., in the class NP]
m-reduction	polynomial reduction

Def: $A, B \subseteq \Sigma^*$. A is *ptime m-reducible* (*p-reducible*) to B ($A \leq_p B$) if $A \leq_m B$ via a polynomial reduction.

Thm: Let $A, B \subseteq \Sigma^*$ with $A \leq_p B$. Then:
1. If $B \in P$, then $A \in P$.
2. If $B \in NP$, then $A \in NP$.

Proof: Fix a p-reduction f such that $A \leq_p B$ via f [$w \in A \iff f(w) \in B$].
For (1): Assume $B \in P$. Then let
$D :=$ "On input w:
1. Let $x := f(w)$ | $\text{Poly}(|w|)$
2. If $x \in B$ then accept, else reject" | $\text{Poly}(|x|)$ $= \text{Poly}(\text{Poly}(|w|))$ $= \text{Poly}(|w|)$

$\therefore$ D takes time $\text{Poly}(|w|)$ [and decides A]
$\therefore A \in P$. //(1)

For (2). Let V be a TM verifying membership in B, i.e., $\forall x$ $\exists y$ $V(x \# y)$ runs in time $\text{Poly}(|x|)$ and $\forall x$
$x \in B \iff \exists y, V(x \# y)$ accepts.

Let V' be the following TM:
$V' :=$ "On input $w \# y$ ...":
1. Let $x := f(w)$ | actually computable by $\text{Poly}(|x|)$
2. Run $V(x \# y)$ [& do what V does]

$V'(w \# y)$ runs in time $\text{Poly}(|w|)$
[because $V(x \# y_{\text{truncate}})$ runs in time $\text{Poly}(|x|)$].

$\forall w$, letting $x := f(w)$.
$w \in A \iff x \in B$
$\iff \exists y_{|y| = \text{Poly}(|x|)}$ [$V(x \# y)$ accepts]
$\iff \exists y_{|y| = \text{Poly}(|w|)}$ [$V'(w \# y)$ accepts]
(same y)

$\therefore V'$ verifies membership in A
$\therefore A \in NP$ //(2) ☐

Def: A language A is complete (under m-reductions) if
1. A is T-rec.
2. $\forall$ T-rec. B, $B \leq_m A$.

Prop: If A is complete, then $\forall B$, B is T-rec $\iff B \leq_m A$
FE ($\Rightarrow$) def of completeness
[$\Leftarrow$) because $\leq_m$ preserves T-recognizability]

Prop: A_{TM} is complete.

Proof: (1) A_{TM} is T-rec. (via universal TM U).

(2) Let B be any T-rec language, and let M be a TM recognizing B ($B = L(M)$). Define

$f :=$ "On input w:

1. Form the pair $\langle M, w \rangle$ [M is fixed & hard-coded in f]
2. Output $\langle M, w \rangle$"

f is computable, and for all w,

$w \in B \iff M$ accepts w

$\iff \langle M, w \rangle \in A_{TM}$

$\iff f(w) \in A_{TM}$

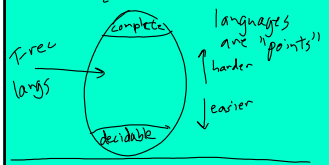
$\therefore B \leq_m A_{TM}$. ∎

Also, if L_1 is complete & L_2 is T-rec & $L_1 \leq_m L_2$, then L_2 is complete.

Pf: Because $\leq_m$ is transitive //

Cor: E_{TM} is complete

A_ε " "



Def: Let A be a language.

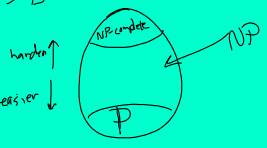
A is NP-hard if $B \leq_p A$ for every $B \in NP$.

A is NP-complete if

(1) A is in NP and

(2) A is NP-hard.

Fact: $\leq_p$ is transitive, so A NP-hard & $A \leq_p B$ $\Rightarrow B$ is NP-hard.



Open: $P = NP$?

Wide belief (conventional wisdom): $P \neq NP$.

SATISFIABILITY (SAT):

Instance: A Boolean formula φ [formed from constants 0, 1 & boolean vars $x_1, \ldots, x_n$ & $\wedge, \vee, \neg$]

Question: Is φ satisfiable? [I.e., is there a setting of $x_1, \ldots, x_n$ that makes φ true?]

Prop: $SAT \in NP$.

[Proof of satisfiability is an actual assignment to $x_1, \ldots, x_n$ that makes φ true (a satisfying assgt). The verifier evaluates $\varphi(\ldots)$ under this assgt & accepts if result is 1.]