

# Introduction to Computational Complexity

Lance Fortnow

Notes(1-14-2008) by Ramanathan Narayanan

Assignment 1 Posted. Due on Wednesday Jan 23, 2008

- $f : \Sigma^* \rightarrow \Sigma^*$   
 $f$  is computable if there is a Turing Machine  $M$  such that  $\forall x, M(x)$  outputs  $f(x)$   
examples:  $f(x) = x, f(x) = xx, f(x) = \text{factors of } m$  where  $x$  is binary representation of  $m$
- $A \leq B, A, B \subseteq \Sigma^* \Rightarrow A$  is reducible to  $B$  if there is a computable function  $f$  such that
  - $x \in A \Rightarrow f(x) \in B$
  - $x \notin A \Rightarrow f(x) \notin B$
- Lemmas
  - If  $A \leq B$  and  $B$  is computable, then  $A$  is computable
  - If  $A \leq B$  and  $B$  is c.e, then  $A$  is c.e
  - If  $A \leq B$  and  $A$  is not computable, then  $B$  is not computable
  - If  $A \leq B$  and  $A$  is not c.e, then  $B$  is not c.e
- Proof of Lemma 1: Suppose  $A \leq B$  and  $B$  is computable, then there is a computable function  $f$  that reduces  $A$  to  $B$

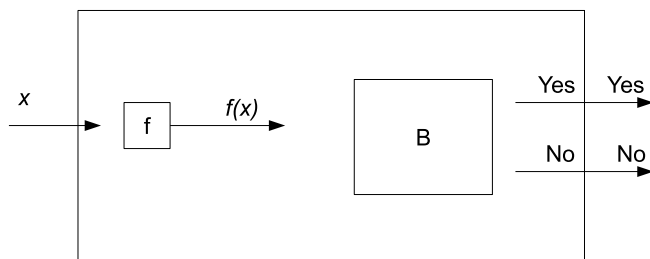
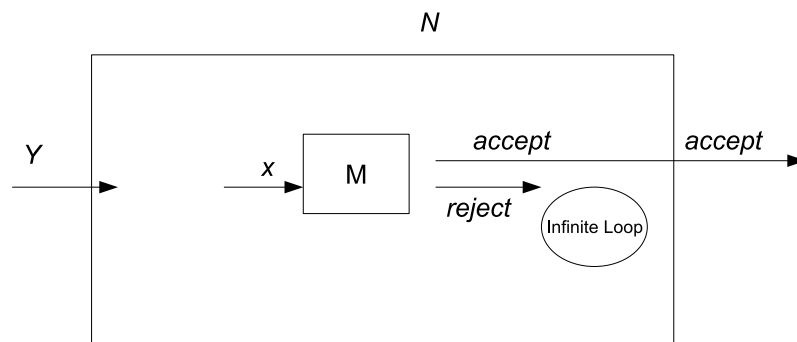


Figure 1. Using  $f$  to generate a T.M that computes  $A$

- Proof of Lemma 2:  
Identical to Proof of Lemma 1

- $L_d \leq L_u$ 
  - $L_d = \{ \langle M \rangle \mid M(\langle M \rangle) \text{ does not accept} \}$  is not c.e
  - $\overline{L_d} = \{ \langle M \rangle \mid M(\langle M \rangle) \text{ accepts} \}$  is not computable
  - $L_u = \{ (\langle M \rangle, x) \mid M(x) \text{ accepts} \}$
  - Proof:  $\overline{L_d} \leq L_u$ 
    - $f(\langle M \rangle) = (\langle M \rangle, \langle M \rangle)$
    - $\langle M \rangle \in \overline{L_d} \Rightarrow M(\langle M \rangle) \text{ accepts}$
    - $f(\langle M \rangle) = (\langle M \rangle, \langle M \rangle) \in L_u \Rightarrow M(\langle M \rangle) \text{ accepts}$
  - Corollary :  $L_u$  is not computable  
 Since if  $L_u$  is computable,  $\overline{L_d}$  is computable, which means that  $L_d$  is computable (Contradiction)  
 $\Rightarrow \overline{L_d}$  is c.e, since  $L_u$  is c.e
- Let  $C$  be a class, then  $\text{co-}C$  is  $\{ \overline{L} \mid L \in C \}$ 
  - co-computable = computable
  - c.e  $\cap$  co-ce = computable
- Halting Problem  
 $L_h = \{ \langle M \rangle \mid M(\epsilon) \text{ halts} \}$ 
  - Theorem:  $L_h$  is not computable ( $L_h$  is c.e)



**Figure 2. T.M N that accepts  $L_h$  using  $M_h$**

Proof:  $L_u \leq L_h$   
 $(\langle M \rangle, x) \rightarrow M_h$   
 If  $M(x)$  accepts, then  $N(\epsilon)$  accepts  
 If  $M(x)$  rejects or does not halt, then  $N(\epsilon)$  does not halt

- Peano Arithmetic, ZFC

- $Th_{ZFC} = \{ T | T \text{ is a theorem in ZFC} \}$   
 $Th_{ZFC}$  is c.e

Simply enumerate all proofs, accept if you can find a proof of  $T$

- $A = \{ \langle M \rangle \mid \text{There is a proof that "M}(\epsilon) \text{ does not halt"} \}$   
 $A$  is c.e again by enumerating proofs  
 $\overline{L_h}$  is not c.e, since  $L_h$  is c.e but not computable  
 $\Rightarrow$  There is some  $\langle N \rangle$  such that  $\langle N \rangle \in \overline{L_h} - A$  (Godel's Incompleteness)

- Complexity

- Consider a T.M  $M$  which has a read only Input tape and several work tapes
- Time complexity:  $t_m(x) =$  number of steps  $M(x)$  takes before it halts
- Space complexity:  $S_m(x) =$  maximum position used by  $M$  on any work tape