



One Way Functions

Chapter 2, Lane-Ogi, Group B



Yang Feng
Anis Zaman
Ethan Johnson
Amin Mosayyebzadeh

Agenda

- Definitions
- Chapter 2 of Lane and Ogi's The Complexity Theory Companion
 - Theorem 2.5, has 2 claims
- Quiz in last 15 min of the class.



Hint Hint: Try to understand the definitions



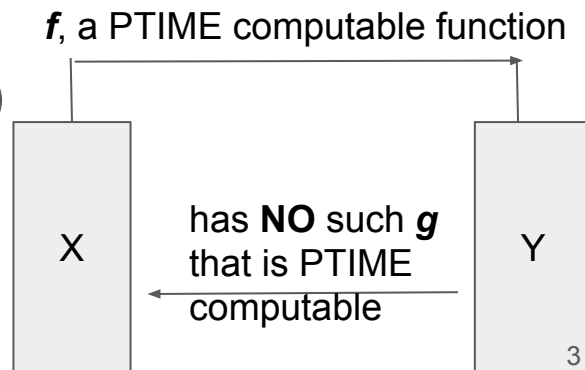
One Way Function (OWF)?

- Function that is easy to compute but **HARD TO INVERT**
- No known One Way Function, (OWF), yet to be found!!
- But there are candidates!!

Requirements to be a One Way Function, f :



- f can be computed in Polynomial Time (PTIME)
- f **can't** be inverted in PTIME
- f is honest



What is Honesty?



Definition 2.1: Honesty

A function f is honest if the following holds:

$(\exists \text{ polynomial } q)$

$(\forall y \in \text{range}(f))$

$(\exists x)$

$[|x| \leq q(|y|) \wedge f(x) = y]$



f can shrink its input
by no more than
polynomial

Why Honesty

Why Honesty?

- **f being honest means:** For each range element $y = f(x)$, there is an x that is at most polynomially longer than y (i.e., for which $f(x)$ is not more than “polynomially” shorter than x)
- Intuitively: function cannot drastically “shrink” its input
- Better reflects intuitive notion of non invertibility - no “length tricks”

$$f(x) = \mathbf{1}[\log\log\log(\max\{|x|, 4\})]$$

* simple example on board: $f(x) = \lceil \log|x| \rceil$

Polynomial Invertibility

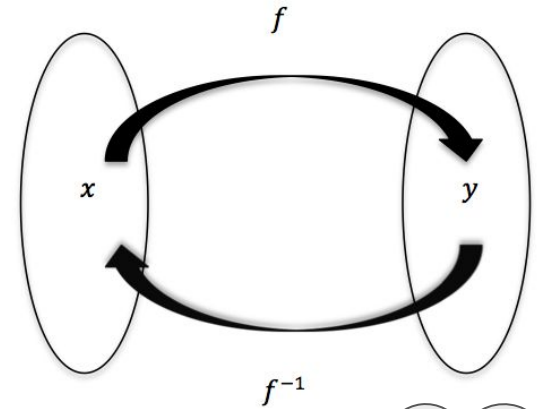


Definition 2.2:

A function (possibly non total) f is PTIME invertible if there is a possible (possibly non total) PTIME computable function g such that:

$$(\forall y \in \text{range}(f))$$

$$[y \in \text{domain}(g) \wedge g(y) \in \text{domain}(f) \wedge f(g(y)) = y]$$



simply means f
can be reversed
engineered in
somewhat
similar amount
of time

Last Definition...

Definition 2.4: One to One

A function f is one to one if:


$$(\forall y \in \Sigma^*) [|\{x \mid f(x) = y\}| \leq 1]$$

Simple High School algebra:

$$(\forall x_1, x_2 \in \Sigma^*) [f(x_1) = f(x_2) \Rightarrow (x_1 = x_2)]$$

Now what?

Now we know what One Way Function (OWF) is!!!

A function f is one way if : 

- f is polynomial-time computable,
- f is not polynomial time invertible, and
- f is honest

Theorem 2.5

1. One-way functions exist if and only if $P \neq NP$
2. One-to-one one way functions exist if and only if $P \neq UP$

We shall prove this theorem for the rest of the class session.

Note: Proof of 2 is simple modification of part 1. **SO PAY CLOSE ATTENTION** while we prove 1

Things to prove:

1. One-way functions exist if and only if $P \neq NP$
 - a. *if: $P \neq NP \Rightarrow$ One Way Function Exists*
 - b. *only if: One Way Function Exists $\Rightarrow P \neq NP$*

2. One-to-one one way functions exist if and only if $P \neq UP$
 - a. *if : $P \neq UP \Rightarrow$ one-to-one one way functions exist*
 - b. *only if : One-to-one one way functions exist $\Rightarrow P \neq UP$*

1(a) if: $P \neq NP \Rightarrow$ One Way Function Exists

- Assume $P \neq NP$
- Let $A \in NP - P$
- \exists a *NPTM*, N such that $L(N) = A$



Goal: Find a one way
function ***f***

1(a) if: $P \neq NP \Rightarrow$ One Way Function Exists ...

Quest for the magical f :

- Assume some standard nice pairing function $\langle \cdot, \cdot \rangle$:
 - PTIME computable and invertible
 - a bijection between $\Sigma^* \times \Sigma^*$ and Σ^*
- Consider a function f :

$$f(\langle x, w \rangle) = \begin{cases} 0x, & \text{if } w \text{ is an accepting path for } N(x) \\ 1x, & \text{otherwise} \end{cases}$$

A process to uniquely encode 2 things into one. Example

3	9	13	
2	5	8	12
1	2	4	7
0	0	1	3

14
10 11 12 13

Remember:

$L(N) = A$ and

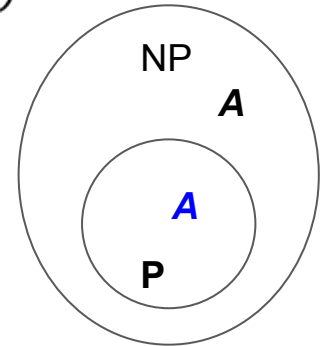
$A \in NP - P$

1(a) if: $P \neq NP \Rightarrow$ One Way Function Exists ...

$$f(\langle x, w \rangle) = \begin{cases} 0x, & \text{if } w \text{ is an accepting path for } N(x) \\ 1x, & \text{otherwise} \end{cases}$$

BIG PICTURE!

- Will show f is polynomial time computable, honest, hard to invert
- For the sake of contradiction assume f is easily invertible in PTIME
- Establish $A \in P$
- Lead to a contradiction $A \in P$ & $A \in NP - P$
- $P = NP$



Remember:

$L(N) = A$ and

$A \in NP - P$

1(a) if: $P \neq NP \Rightarrow$ One Way Function Exists ...

Quest for the magical f :

- f that takes paired values $\langle x, w \rangle$ as input
- Our Claim:
$$f(\langle x, w \rangle) = \begin{cases} 0x, & \text{if } w \text{ is an accepting path for } N(x) \\ 1x, & \text{otherwise} \end{cases}$$
 - f is polynomial time computable
 - f is honest

Remember:

$L(N) = A$ and

$A \in NP - P$

1(a) if: $P \neq NP \Rightarrow$ One Way Function Exists ...

- **f** is computable in PTIME? $f(\langle x, w \rangle) = \begin{cases} 0x, & \text{if } w \text{ is an accepting path for } N(x) \\ 1x, & \text{otherwise} \end{cases}$
 - checking whether **w** is an accepting path by running **N** on input **x** is clearly polynomial
- **f** is honest?
 - If **w** is an accepting path, no path in **N** can be longer than some polynomial $p(|x|)$. If **w** is not an accepting path, $|w|$ might not be polynomial. But honesty only requires that *some* short preimage for $1x$ exist (easy to come up with one).
 - Also, PTIME computability and invertibility of pairing function “**prevents f from destroying the honesty condition**”

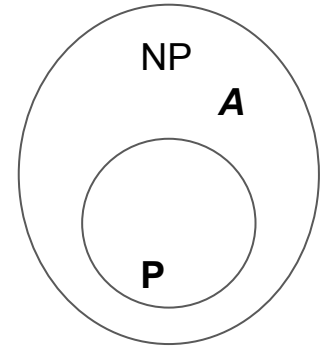
1(a) if: $P \neq NP \Rightarrow$ One Way Function Exists ...

- We are almost there!
- Showed that our f is
 - **computable in polynomial time** ✓
 - **honest** ✓
 - **hard to invert**

1(a) if: $P \neq NP \Rightarrow$ One Way Function Exists ...

BIG PICTURE!

- Showed f is polynomial time computable, honest ✓
- **Need to show hard to invert**
- For the sake of contradiction assume f is “easily” invertible in PTIME
- Gives us $A \in P$
- Will construct a DPTM M , s.t $L(M) = A$
- Lead to a contradiction $A \in P$ & $A \in NP-P$
- $P = NP$



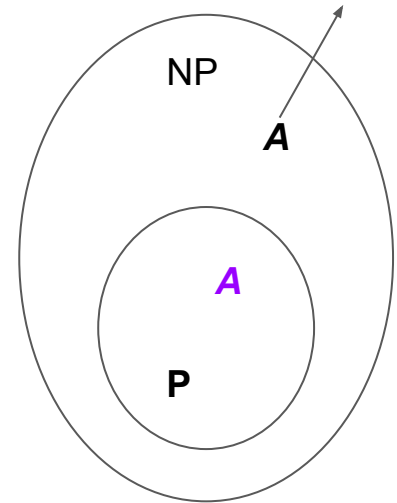
“easily” means
in **polynomial
time**

1(a) if: $P \neq NP \Rightarrow$ One Way Function Exists ...

- Goal: Show f is hard to invert
- Assume f is invertible via a polynomially computable function g
- g allows us to accept A in PTIME
- We get $A \in P$
- We will show that $A \in P$ by constructing a

DPTM M such that $L(M) = A$

Remember:
 $L(N) = A$ and
 $A \in NP - P$

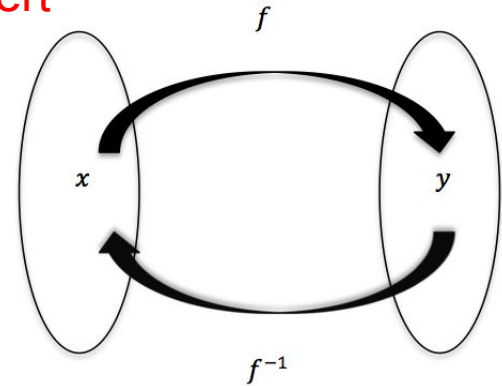


1(a) if: $P \neq NP \Rightarrow$ One Way Function Exists ...

- Construction ***M***:

$$f(\langle x, w \rangle) = \begin{cases} 0x, & \text{if } w \text{ is an accepting path for } N(x) \\ 1x, & \text{otherwise} \end{cases}$$

Goal: Show ***f*** is hard to invert

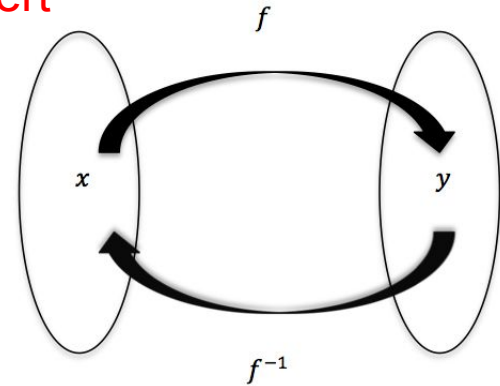


1(a) if: $P \neq NP \Rightarrow$ One Way Function Exists ...

$$f(\langle x, w \rangle) = \begin{cases} 0x, & \text{if } w \text{ is an accepting path for } N(x) \\ 1x, & \text{otherwise} \end{cases}$$

- Construction ***M***:
- On input $x \in \Sigma^*$, check if $0x \in \text{domain of } g$
- if not, **REJECT!**

Goal: Show ***f*** is hard to invert



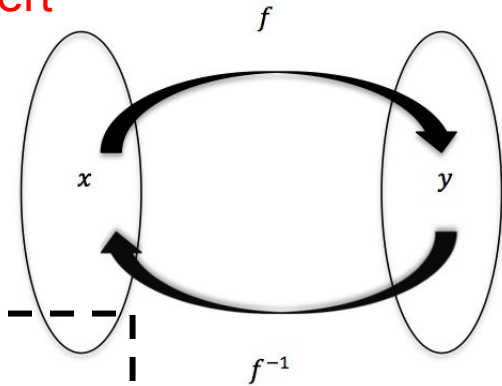
1(a) if: $P \neq NP \Rightarrow$ One Way Function Exists ...

$$f(\langle x, w \rangle) = \begin{cases} 0x, & \text{if } w \text{ is an accepting path for } N(x) \\ 1x, & \text{otherwise} \end{cases}$$

- Construction ***M***:
- On input $x \in \Sigma^*$, check if $0x \in \text{domain of } g$
- if not, **REJECT!**
- if yes:
 - compute $g(0x)$, which returns $\langle x, w \rangle$
 - test if w is an accepting path in $N(x)$
 - if yes, **ACCEPT!**
 - Otherwise, **REJECT!**

PTIME

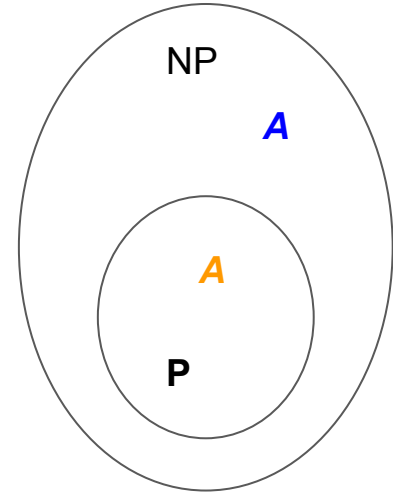
Goal: Show ***f*** is hard to invert



Mr. RABBIT
will use this
construction in 2(a)

1(a) if: $P \neq NP \Rightarrow$ One Way Function Exists ...

- M accepts A in deterministic polynomial time
- Under our assumption $P \neq NP$
- Just showed a DPTM for M and $L(M) = A$
- Showed $A \in P$
- Assumed $A \in NP - P$
- Contradiction!



What actually happened?

- We constructed DPTM M assuming an inverse of f , g existed, which is polynomial time computable
- But g does not exist i.e. no polynomial time computable inverse of f exists
- f^{-1} must not be polynomially computable, **f^{-1} is HARD to invert**
- f must not be polynomially invertible
- Vola!!! f is now a One Way Function ✓

Where are we so far?

1. One-way functions exist if and only if $P \neq NP$
 - a. *if: $P \neq NP \Rightarrow$ One Way Function Exists \checkmark*
 - b. *only if: One Way Function Exists $\Rightarrow P \neq NP$*

2. One-to-one one way functions exist if and only if $P \neq UP$
 - a. *if : $P \neq UP \Rightarrow$ one-to-one one way functions exist*
 - b. *only if : One-to-one one way functions exist $\Rightarrow P \neq UP$*

1(b) only if: One Way Function Exists $\Rightarrow P \neq NP$

- Reverse our previous strategy: given a one-way function f , **assume $P=NP$** , lead to contradiction.
- Let p be f 's honesty polynomial
- Think of this language:

$$L = \{ \langle z, pre \rangle \mid (\exists y)[|y| + |pre| \leq p(|z|) \wedge f(pre.y) = z] \}$$

- What does this language “mean”?
 - *Prefixes of the inverse of z , i.e. $f^{-1}(z)$, that are sufficiently short (for honesty)*

1(b) only if: One Way Function Exists $\Rightarrow P \neq NP$

$$L = \{ \langle z, pre \rangle \mid (\exists y)[|y| + |pre| \leq p(|z|) \wedge f(pre.y) = z] \}$$

- Clearly L is NP: guess a string y , then check if $f(pre.y) = z$.
- **Since we assumed $P=NP$, L is also P!**
- We'll use this fact to invert f "easily" (in P-time) - contradicting that f is a one-way function.

1(b) only if: One Way Function Exists $\Rightarrow P \neq NP$

- Goal: given z , find its inverse with respect to f in polynomial time
 - (find x such that $f(x) = z$)
- Since (we assumed) $L \in P$, there is a DPTM accepting L .
- If $\langle z, pre \rangle$ is in L , pre is a prefix of z 's inverse
- We can check “easily” (P-time) whether something is in L !

What can we do with this?

1(b) only if: One Way Function Exists $\Rightarrow P \neq NP$

Complexity rabbit says...



Search **ALL** the prefixes!

1(b) only if: One Way Function Exists $\Rightarrow P \neq NP$

Searching all the prefixes:

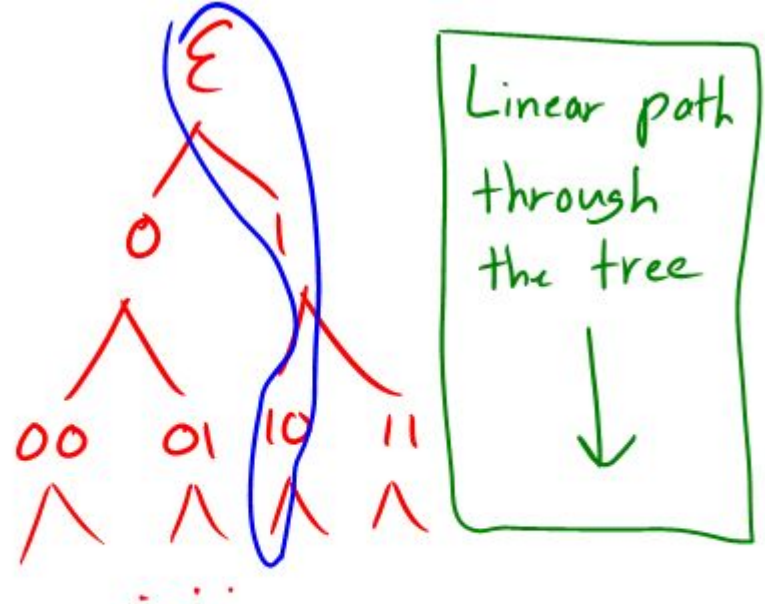
- Check if $f(\epsilon) = z$; if so we are done (ϵ is an inverse), if not go to next step.
- Is 0 a prefix of a suitably short inverse?
 - If NO, then 1 must be a prefix!
 - Either way, we determine the first bit.
- Are we done yet? (is this prefix the *whole* inverse?)
 - Check if $f(pre) = z$. If yes, we're done! Otherwise, we need to find out the next bit...
- (Let b be the bit we've already figured out.) Is $b0$ a prefix of a suitably short inverse?
 - If no, then $b1$ must be a prefix...
 - Now we have the second bit (c), check if $f(bc) = z$...

* special note on multiple preimages

1(b) only if: One Way Function Exists $\Rightarrow P \neq NP$

Can think of this as a search as a tree:

- At each step, we discover one more bit of the inverse.
- We *will* make progress with each step: if the next bit isn't 0, it must be 1. No exponential expansion!
- Hence prefix search is *linear* in the length of the inverse!
- The honest polynomial bounds the length of the inverse



Suppose the inverse we're looking for is **11010**. With each bit uncovered, we narrow the search space.

Recap

- We started with a one way function f
- We supposed $P = NP$
- We examined a language L that lets us check if a string is a prefix of $f^{-1}(z)$
- Since $P = NP$, $L \in P$
- We used L to search for the inverse in polynomial time
- Thus f can be inverted in polynomial time. **Contradiction!**
- Hence our assumption was wrong: $P \neq NP$



Where are we so far?

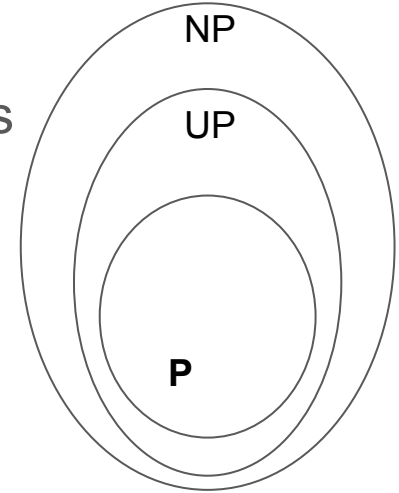
1. One-way function exists if and only if $P \neq NP$
 - a. *if: $P \neq NP \Rightarrow$ One Way Function Exists \checkmark*
 - b. *only if: One Way Function Exists $\Rightarrow P \neq NP \checkmark$*

2. One-to-one one way function exists if and only if $P \neq UP$
 - a. *if : $P \neq UP \Rightarrow$ one-to-one one way functions exist*
 - b. *only if : One-to-one one way functions exist $\Rightarrow P \neq UP$*

What is UP?

- A complexity class like (NP, P) that has unique witness
- $L \in UP$ if:
 - NP machine N accepts $x \in L$
 - For all such x , the computation of $N(x)$ has

at most 1 accepting path



$UP = \{ L \mid \exists \text{ NPTM, } N \text{ such that } L = L(N), \text{ and } \forall x \in L, N(x) \text{ has at most } 1 \text{ accepting path} \}$

2(a) if: $P \neq UP \Rightarrow$ one-to-one one way functions exist

- Let $A \in UP - P$
- \exists NPTM, N such that $A = L(N)$
- Consider function f :

$$f(\langle x, w \rangle) = \begin{cases} 0x, & \text{if } w \text{ is an accepting path for } N(x) \\ 1 \langle x, w \rangle, & \text{otherwise} \end{cases}$$



Goal: want to show f is 1-to-1 one way function

Where are we now?

1. One-way function exists if and only if $P \neq NP$
 - a. *if: $P \neq NP \Rightarrow$ One Way Function Exists ✓*
 - b. *only if: One Way Function Exists $\Rightarrow P \neq NP$ ✓*

2. One-to-one one way function exists if and only if $P \neq UP$
 - a. *if : $P \neq UP \Rightarrow$ one-to-one one way functions exist ✓*
 - b. *only if : One-to-one one way functions exist $\Rightarrow P \neq UP$*

2(b): *only if* : 1-to-1 one way functions exist $\Rightarrow P \neq UP$

- No changes from 1(b)
 - Replace “one way function” with “1-to-1 one way function” and “NP” with “UP” - same argument holds
- Only difference: there is only one path in the prefix search tree that will lead us to an inverse.
 - We were ignoring the extras anyway (see special note)
- **1-to-1 one way functions exist $\Rightarrow P \neq UP!!!$**

Big Picture!!

1. Got introduced to One Way Function (1-to-1 as well)!
2. Existence of One Way Function is tied to whether $P=NP$
3. For 1-to-1 One Way Function, it is tied to a more strongly regulated version of NP i.e. UP^{***}
4. Next class we will expand on $***$ to cover a constant bounded version of UP

One way Functions

Chapter 2.2 Hem-ogi

Group B:

Yang Feng

Anis Zaman

Ethan Johnson

Amin Mosayyebzadeh

Reminder

- one-to-one one-way functions are characterized by $P \neq UP$
- one-way functions is characterized by $P \neq NP$

- $P \neq UP \Rightarrow P \neq NP$,
 - the converse has never been established



I am the Rabbit,
but I have just
eaten a carrot

Note

- one-to-one function f is completely unambiguous in terms of inversion
 - each element of $\text{range}(f)$ has exactly one inverse

- "constant-to-one" functions are called bounded-ambiguity functions. They are often referred as " $O(1)$ -to-one" functions



Definition 2.6: Bounded-Ambiguity Functions

1. For each $k \geq 1$, we say that

a. a (possibly non-total) function f is k -to-one if $(\forall y \in \text{range}(f)) [|\{x \mid f(x) = y\}| \leq k]$

compare it with one-to-one function:
 $(\forall y \in \text{range}(f)) [|\{x \mid f(x) = y\}| \leq 1]$



2. We say that

a. a (possibly non-total) function f is of bounded-ambiguity if there is a $k \geq 1$ such that f is k -to-one.

Big picture

- We are going to prove that
 - Unambiguous one-way functions exist \Leftrightarrow Bounded-ambiguity one-way functions exist

Theorem 2.7

one-to-one one-way functions exist \Leftrightarrow constant-to-one one-way functions exist

Proof

- “Only if” direction is easy:
 - All one-to-one functions are constant-to-one functions, so the “only if” direction holds
- We will show the “if” direction

Definition 2.8

A language L is in $UP_{\leq k}$, $k \geq 1$, if there is an NPTM N such that

1. $(\forall x \in L)$ [$N(x)$ has at least one and at most k accepting paths]

and

2. $(\forall x \in \bar{L})$ [$N(x)$ has no accepting paths]

Recall

- Part 2 of Theorem 2.5
 - one-to-one one-way functions exist $\Leftrightarrow P \neq UP$

Fact 2.8

- For each $k \geq 2$, k -to-one one-way functions exist $\Leftrightarrow P \neq UP_{\leq k}$

It can be proved by exactly
analogous proof of Theorem 2.5



Proving the “if” direction

- We know that:
 - $P \neq UP \Leftrightarrow$ one-to-one one-way functions exist
 - $P \neq UP_{\leq k} \Leftrightarrow$ k-to-one one-way functions exist (from previous slide)
- If we show that $P \neq UP \Leftrightarrow P \neq UP_{\leq k}$, then
 - one-to-one one-way functions exist \Leftrightarrow k-to-one one-way functions exist
- We will use induction that:
 - for all $k \in \{1, 2, 3, \dots\}$, $P = UP \Rightarrow P = UP_{\leq k}$

Proving the “if” direction

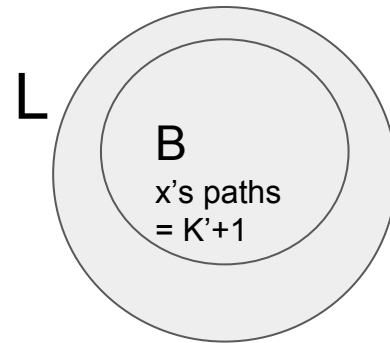
- We know that:
 - $P \neq UP \Leftrightarrow$ one-to-one one-way functions exist
 - $P \neq UP_{\leq k} \Leftrightarrow$ k-to-one one-way functions exist (from previous slide)
- If we show that $P \neq UP \Leftrightarrow P \neq UP_{\leq k}$, then
 - one-to-one one-way functions exist \Leftrightarrow k-to-one one-way functions exist
- We will use induction that:
 - for all $k \in \{1, 2, 3, \dots\}$, $P = UP \Rightarrow P = UP_{\leq k}$

Induction

- holds for $k = 1$:
 - $P = UP \Rightarrow P = UP_{\leq 1}$
- Assume:
 - $P = UP \Rightarrow P = UP_{\leq k'}$
- prove:
 - $P = UP \Rightarrow P = UP_{\leq k'+1}$

$$P \neq UP \Leftrightarrow P \neq UP_{\leq k}$$

Proving $P = UP \Rightarrow P = UP_{\leq k'+1}$
 (Assuming $P = UP \Rightarrow P = UP_{\leq k'}$)



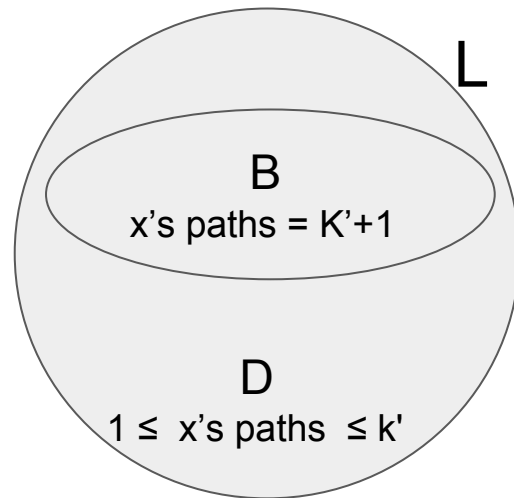
- Assume $P = UP$
- Let L be an arbitrary member of $UP_{\leq k'+1}$.
- Let N be an NPTM –having at most $k' + 1$ accepting paths on each input– that accepts L (recall Definition 2.8)
- Consider the set
 - $B = \{ x \mid N(x) \text{ has exactly } k' + 1 \text{ accepting paths} \}$
 - Clearly, $B \in UP$, via the machine that on each input x guesses each lexicographically ordered $(k'+1)$ -tuple of distinct computation paths and that accepts on such a path exactly if each of the $k' + 1$ guessed paths is an accepting path on input x .
 - by our $P = UP$ assumption, $B \in P$

$K'+1$ or K' , that is the question

So, we are excluding
elements with $k'+1$ paths
(Set B)



- since $B \in P$, the set
 - $D = \{x \mid x \notin B \wedge x \in L(N)\}$ is in $UP_{\leq k'}$



Proving $D \in UP_{\leq k'}$

- We construct a TM M such that:
- We first deterministically check whether x is in B
 - Using some P algorithm for B .
 - Under our current assumptions, $B \in P$. So some such algorithm exists.
- If $x \in B$ we reject
- If $x \notin B$ we directly simulate $N(x)$.
 - This simulation will have at most k' accepting paths
 - $x \notin B$ precludes there being exactly $k'+1$ paths
 - N 's choice precludes there being more than $k' + 1$ paths
- Since $D \in UP_{\leq k'}$, we conclude
 - from our assumption that $P = UP$,
 - from our inductive hypothesis (which was $P = UP \Rightarrow P = UP_{\leq k'}$)

$\Rightarrow D \in P$.

Prove

- Since P is closed under union, $B \cup D \in P$.
- However
 - $L = B \cup D \Rightarrow L \in P$
 - L is an arbitrary member of $UP_{\leq k'+1}$

$$\Rightarrow P = UP \Rightarrow P = UP_{\leq k'+1}$$



One Way Functions

Chapter 2, Lane-Ogi, Group B



Yang Feng
Anis Zaman
Ethan Johnson
Amin Mosayyebzadeh

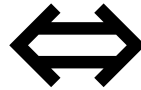
Two-argument (denoted 2-ary) one-way functions

$$f(x, x') = y$$

- ❖ Strong
- ❖ Total
- ❖ Commutative
- ❖ Associative



One-Way Functions Exist



One-Way Functions Exist

2-ary function honesty

$$f(\boxed{x}, \boxed{x'}) = y$$
$$\boxed{q(y)}$$

Definition 2.10:

We say a 2-ary function $f: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ is honest if

$$(\exists \textit{ polynomial } q)(\forall y \in \textit{range}(f))(\exists x, x')[|x| + |x'| \leq q(|y|) \wedge f(x, x') = y].$$

This definition only requires that each element of $\textit{range}(f)$ have one appropriate pair (x, x') .

2-ary function invertible

$$\begin{aligned} f(x, x') &= y \\ g(y) &= (x, x') \end{aligned}$$

Definition 2.11:

We say a 2-ary function $f: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ is polynomial-time invertible if there is a polynomial-time computable function g such that, for each $y \in \text{range}(f)$,

$$\begin{aligned} &y \in \text{domain}(g) \wedge \\ &(first(g(y)), second(g(y))) \in \text{domain}(f) \wedge \\ &f(first(g(y)), second(g(y))) = y, \end{aligned}$$

where the projection functions $first(z)$ and $second(z)$ denote, respectively, the first and second components of the unique ordered pair of strings that when paired give z .

2-ary one-way function

We say a 2-ary function $f: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ is one-way if

- f is polynomial-time computable,
- f is not polynomial time invertible, and
- f is honest

Are this familiar?

2-ary function s-honesty

We say a 2-ary function $f: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ is s-honest if

$$1. (\exists \text{ polynomial } q) (\forall y, a : (\exists b)[f(a, b) = y])$$

$$(\exists \underline{b'})[|b'| \leq q(|y| + |a|) \wedge f(a, b') = y].$$

$$2. (\exists \text{ polynomial } q) (\forall y, b : (\exists a)[f(a, b) = y])$$

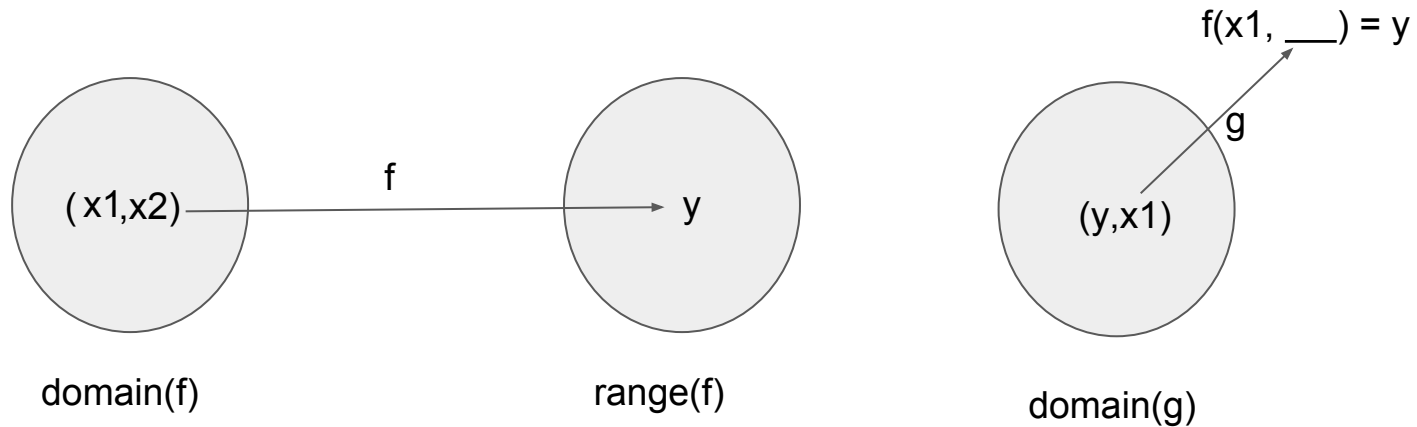
$$(\exists a')[|a'| \leq q(|y| + |b|) \wedge f(a', b) = y].$$

How to understand it? See next page.

2-ary function invertible

$$f(x_1, x_2) = y$$

$$g(y, x_1) = \text{something similar to } x_2$$



2-ary function strongly noninvertible

We say a 2-ary function $f: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ is strongly noninvertible if it is s-honest and yet neither of the following conditions holds.

- 1. There is a (possibly nontotal) polynomial-time computable function $g: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ such that $(\forall y \in \text{range}(f))(\forall x_1, x_2 : (x_1, x_2) \in \text{domain}(f) \wedge f(x_1, x_2) = y)[(y, x_1) \in \text{domain}(g) \wedge f(x_1, g(y, x_1)) = y]$.*
- 2. There is a (possibly nontotal) polynomial-time computable function $g: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ such that $(\forall y \in \text{range}(f))(\forall x_1, x_2 : (x_1, x_2) \in \text{domain}(f) \wedge f(x_1, x_2) = y)[(y, x_2) \in \text{domain}(g) \wedge f(g(y, x_2), x_2) = y]$.*

2-ary function associative and commutative

We say a 2-ary function $f: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ is associative if

$$(\forall x, y, z)[f(f(x, y), z) = f(x, f(y, z))].$$

We say a 2-ary function $f: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ is commutative if

$$(\forall x, y)[f(x, y) = f(y, x)].$$

multiplication of integers?

concatenation of strings?

Proposition 2.17

The following are equivalent.

1. One-way functions exist.
2. 2-ary one-way functions exist.
3. $P \neq NP$.

(2) \Rightarrow (1)

Let $\langle \cdot, \cdot \rangle$ be a pairing function used before

Let $f : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ be any 2-ary one-way function:

$$g(z) = f(\text{first}(z), \text{second}(z))$$

where, $\text{first}(z)$ and $\text{second}(z)$ denotes the first and second component of the pair mapped to z .

(1) \Rightarrow (2)

Let $h : \Sigma^* \rightarrow \Sigma^*$ be a one-way function:

$$h'(x,y) = \langle h(x), h(y) \rangle$$



One Way Functions

Chapter 2, Lane-Ogi, Group B



Yang Feng
Anis Zaman
Ethan Johnson
Amin Mosayyebzadeh

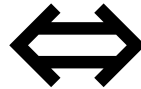
Two-argument (denoted 2-ary) one-way functions

$$f(x, x') = y$$

- ❖ Strong
- ❖ Total
- ❖ Commutative
- ❖ Associative



One-Way Functions Exist



One-Way Functions Exist

2-ary function honesty

$$f(\boxed{x}, \boxed{x'}) = y$$
$$\boxed{q(y)}$$

Definition 2.10:

We say a 2-ary function $f: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ is honest if

$$(\exists \textit{ polynomial } q)(\forall y \in \textit{range}(f))(\exists x, x')[|x| + |x'| \leq q(|y|) \wedge f(x, x') = y].$$

This definition only requires that each element of $\textit{range}(f)$ have one appropriate pair (x, x') .

2-ary function invertible

$$\begin{aligned} f(x, x') &= y \\ g(y) &= (x, x') \end{aligned}$$

Definition 2.11:

We say a 2-ary function $f: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ is polynomial-time invertible if there is a polynomial-time computable function g such that, for each $y \in \text{range}(f)$,

$$\begin{aligned} &y \in \text{domain}(g) \wedge \\ &(first(g(y)), second(g(y))) \in \text{domain}(f) \wedge \\ &f(first(g(y)), second(g(y))) = y, \end{aligned}$$

where the projection functions $first(z)$ and $second(z)$ denote, respectively, the first and second components of the unique ordered pair of strings that when paired give z .

2-ary one-way function

We say a 2-ary function $f: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ is one-way if

- f is polynomial-time computable,
- f is not polynomial time invertible, and
- f is honest

Are this familiar?

A new version of honesty
that will help us define
"strong invertibility"

2-ary function s-honesty

We say a 2-ary function $f: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ is s-honest if

$$1. (\exists \text{ polynomial } q) (\forall y, a : (\exists b)[f(a, b) = y])$$

$$(\exists \underline{b'})[|b'| \leq q(|y| + |a|) \wedge f(a, b') = y].$$

$$2. (\exists \text{ polynomial } q) (\forall y, b : (\exists a)[f(a, b) = y])$$

Intuitively:

$$(\exists a')[|a'| \leq q(|y| + |b|) \wedge f(a', b) = y].$$

- f doesn't shrink its output drastically relative to *either* of its inputs *individually*
- Not the same as regular honesty - a dishonest function can still be s-honest if it shrinks output drastically relative to *both* of its inputs together

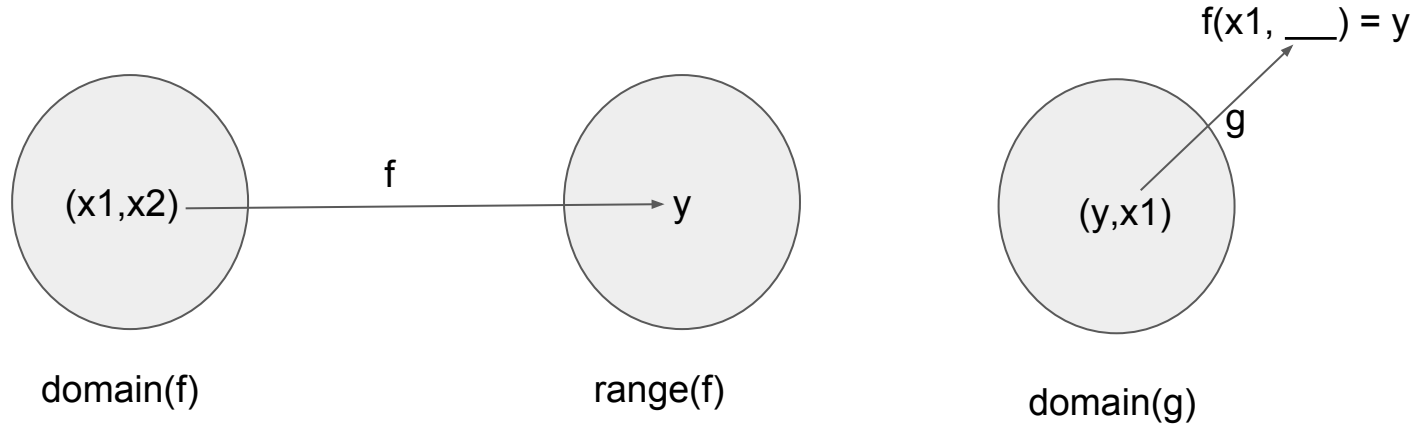
○ Example:

$$f(a, b) = \begin{cases} \lceil \log \log \log (\max(|a|, 4)) \rceil & \text{when } |a|=|b| \\ \text{undefined} & \text{otherwise} \end{cases}$$

2-ary function *strong* noninvertibility

$$f(x_1, x_2) = y$$

$$g(y, x_1) = \text{something similar to } x_2$$



Intuitively: given a and y where $f(a,b)=y$,
we cannot easily recover b

2-ary function strongly noninvertible

We say a 2-ary function $f: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ is strongly noninvertible if it is honest and yet neither of the following conditions holds.

1. *There is a (possibly nontotal) polynomial-time computable function $g: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ such that $(\forall y \in \text{range}(f))(\forall x_1, x_2: (x_1, x_2) \in \text{domain}(f) \wedge f(x_1, x_2) = y)[(y, x_1) \in \text{domain}(g) \wedge f(x_1, g(y, x_1)) = y]$.*
2. *There is a (possibly nontotal) polynomial-time computable function $g: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ such that $(\forall y \in \text{range}(f))(\forall x_1, x_2: (x_1, x_2) \in \text{domain}(f) \wedge f(x_1, x_2) = y)[(y, x_2) \in \text{domain}(g) \wedge f(g(y, x_2), x_2) = y]$.*

2-ary function associative and commutative

We say a 2-ary function $f: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ is associative if

$$(\forall x, y, z)[f(f(x, y), z) = f(x, f(y, z))].$$

We say a 2-ary function $f: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ is commutative if

$$(\forall x, y)[f(x, y) = f(y, x)].$$

multiplication of integers?

concatenation of strings?

Proposition 2.17

The following are equivalent.

1. One-way functions exist.
2. 2-ary one-way functions exist.
3. $P \neq NP$.

(2) \Rightarrow (1)

Let $\langle \cdot, \cdot \rangle$ be a pairing function used before

Let $f : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ be any 2-ary one-way function; then

$$g(z) = f(\text{first}(z), \text{second}(z))$$

where $\text{first}(z)$ and $\text{second}(z)$ denotes the first and second component of the pair mapped to z , is also one-way.

(1) \Rightarrow (2)

Let $h : \Sigma^* \rightarrow \Sigma^*$ be a one-way function; then

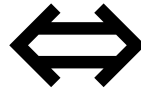
$$h'(x,y) = \langle h(x), h(y) \rangle$$

is clearly one-way.

- ❖ Strong
- ❖ Total
- ❖ Commutative
- ❖ Associative



One-Way Functions Exist



One-Way Functions Exist

Only if direction \Rightarrow EASY

S. T. C. A. One Way Function exist

\Rightarrow

2-ary one-way functions exist

By Proposition 2.17

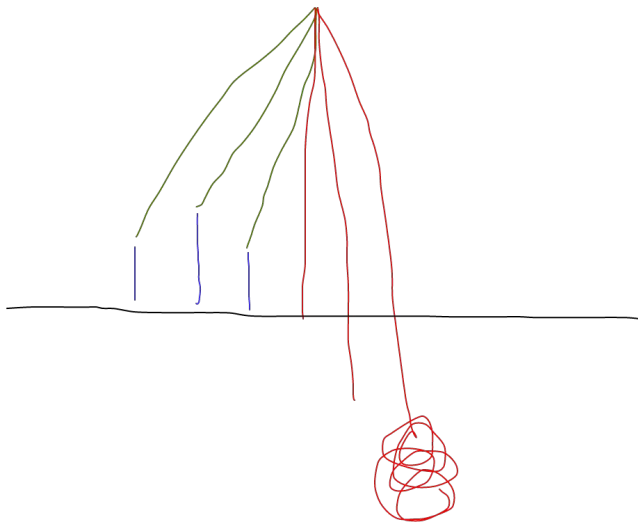
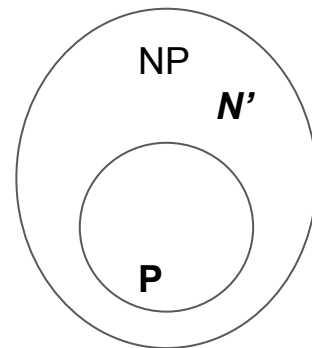
\Rightarrow

One-way functions exist

If direction \Leftarrow

Assume $P \neq NP$, \exists NPTM N' such that $L(N') \in NP-P$

\exists NPTM N such that $L(N) = L(N')$. WLOG, assume that on each input x , each computation path of $N(x)$ has exactly $p(|x|)$ bits. We also require $p(n) > n$.



If direction \Leftarrow

- We will assume One-Way Functions exist.
- We will construct a one-way function that is strongly noninvertible, total, commutative, and associative.

If direction \Leftarrow

Witness: A witness for x is an accepting path of $N(x)$.

Let $W(x)$ be the set of all witnesses for $x \in L(N)$.

- $p(n) > n \implies$ A string can never be a witness for its own membership.
- Since $L(N) \in NP$, each witness for x has length polynomially related to $|x|$.

If direction \Leftarrow

Let t be any fixed string such that $t \notin L(N)$. Our magic function is:

$$f(u, v) = \begin{cases} \langle x, \text{lexmin}(w_1, w_2) \rangle & \text{if } u = \langle x, w_1 \rangle \wedge v = \langle x, w_2 \rangle \wedge \{w_1, w_2\} \subseteq W(x) \\ \langle x, x \rangle & \text{if } (\exists w \in W(x))[\{u, v\} = \{\langle x, x \rangle, \langle x, w \rangle\}] \\ \langle t, t1 \rangle & \text{otherwise,} \end{cases}$$

Intuitively:

- u and v are interpreted as pairs $\langle x, x \rangle$ or $\langle x, w \rangle$ - w is a witness for x
- These two pairs represent zero, one, or two (not necessarily unique) witnesses for x
- If input is of “wrong form” (x 's don't match, or neither pair has a witness), output the “garbage dump” $\langle t, t1 \rangle$. Otherwise, output the input with one less witness instance.
 - Removing information - expensive (NP) to find witnesses

If direction \Leftarrow

Let us verify that f is a strongly noninvertible, total, commutative, associative, 2-ary one-way function.

$$f(u, v) = \begin{cases} \langle x, \text{lexmin}(w_1, w_2) \rangle & \text{if } u = \langle x, w_1 \rangle \wedge v = \langle x, w_2 \rangle \wedge \{w_1, w_2\} \subseteq W(x) \\ \langle x, x \rangle & \text{if } (\exists w \in W(x))[\{u, v\} = \{\langle x, x \rangle, \langle x, w \rangle\}] \\ \langle t, t1 \rangle & \text{otherwise,} \end{cases}$$

Total and polynomial-time computable?

If direction \Leftarrow

Honest?

$$f(u, v) = \begin{cases} \langle x, \text{lexmin}(w_1, w_2) \rangle & \text{if } u = \langle x, w_1 \rangle \wedge v = \langle x, w_2 \rangle \wedge \{w_1, w_2\} \subseteq W(x) \\ \langle x, x \rangle & \text{if } (\exists w \in W(x))[\{u, v\} = \{\langle x, x \rangle, \langle x, w \rangle\}] \\ \langle t, t1 \rangle & \text{otherwise,} \end{cases}$$

- First two cases easy since we chose N such that all paths for $N(x)$ are $p(|x|)$ bits
- Third case (“garbage dump”) seems obviously dishonest (due to fixed length), but since we have only *one* such case, we can choose our honesty polynomial large enough to allow the smallest string that maps to $\langle t, t1 \rangle$.

If direction \Leftarrow

Commutative?

$$f(u, v) = \begin{cases} \langle x, \text{lexmin}(w_1, w_2) \rangle & \text{if } u = \langle x, w_1 \rangle \wedge v = \langle x, w_2 \rangle \wedge \{w_1, w_2\} \subseteq W(x) \\ \langle x, x \rangle & \text{if } (\exists w \in W(x))[\{u, v\} = \{\langle x, x \rangle, \langle x, w \rangle\}] \\ \langle t, t1 \rangle & \text{otherwise,} \end{cases}$$

If direction \Leftarrow

$$1. (\exists \text{ polynomial } q) (\forall y, a : (\exists b)[f(a, b) = y]) \\ (\exists b')[|b'| \leq q(|y| + |a|) \wedge f(a, b') = y].$$

s-honest?

$$2. (\exists \text{ polynomial } q) (\forall y, b : (\exists a)[f(a, b) = y]) \\ (\exists a')[|a'| \leq q(|y| + |b|) \wedge f(a', b) = y].$$

$$f(u, v) = \begin{cases} \langle x, \text{lexmin}(w_1, w_2) \rangle & \text{if } u = \langle x, w_1 \rangle \wedge v = \langle x, w_2 \rangle \wedge \{w_1, w_2\} \subseteq W(x) \\ \langle x, x \rangle & \text{if } (\exists w \in W(x))[\{u, v\} = \{\langle x, x \rangle, \langle x, w \rangle\}] \\ \langle t, t1 \rangle & \text{otherwise,} \end{cases}$$

If direction \Leftarrow

strongly noninvertible?

$$f(u, v) = \begin{cases} \langle x, \text{lexmin}(w_1, w_2) \rangle & \text{if } u = \langle x, w_1 \rangle \wedge v = \langle x, w_2 \rangle \wedge \{w_1, w_2\} \subseteq W(x) \\ \langle x, x \rangle & \text{if } (\exists w \in W(x))[\{u, v\} = \{\langle x, x \rangle, \langle x, w \rangle\}] \\ \langle t, t1 \rangle & \text{otherwise,} \end{cases}$$

If direction \Leftarrow

strongly noninvertible:

Intuitively: given a and y where $f(a,b)=y$, we cannot easily recover b

Approach:

- Assume f is not strongly noninvertible
- Will lead to a **contradiction** $L(N) \in P$ (\Rightarrow assumption is wrong)

Assumed $L(N) \in NP-P$

strongly noninvertible?

$$f(u, v) = \begin{cases} \langle x, \text{lexmin}(w_1, w_2) \rangle & \text{if } u = \langle x, w_1 \rangle \wedge v = \langle x, w_2 \rangle \wedge \{w_1, w_2\} \subseteq W(x) \\ \langle x, x \rangle & \text{if } (\exists w \in W(x))[\{u, v\} = \{\langle x, x \rangle, \langle x, w \rangle\}] \\ \langle t, t1 \rangle & \text{otherwise,} \end{cases}$$

If direction \Leftarrow

- Assume f is not strongly noninvertible via function g in PTIME
- Since f is s-honest, at least one of the 2 conditions of strong noninvertibility holds from **Definition 2.14**

1. *There is a (possibly nontotal) polynomial-time computable function $g : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ such that $(\forall y \in \text{range}(f))(\forall x_1, x_2 : (x_1, x_2) \in \text{domain}(f) \wedge f(x_1, x_2) = y)[(y, x_1) \in \text{domain}(g) \wedge f(x_1, g(y, x_1)) = y]$.*
2. *There is a (possibly nontotal) polynomial-time computable function $g : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ such that $(\forall y \in \text{range}(f))(\forall x_1, x_2 : (x_1, x_2) \in \text{domain}(f) \wedge f(x_1, x_2) = y)[(y, x_2) \in \text{domain}(g) \wedge f(g(y, x_2), x_1) = y]$.*

If direction \Leftarrow

$$f(u, v) = \begin{cases} \langle x, \text{lexmin}(w_1, w_2) \rangle & \text{if } u = \langle x, w_1 \rangle \wedge v = \langle x, w_2 \rangle \wedge \{w_1, w_2\} \subseteq W(x) \\ \langle x, x \rangle & \text{if } (\exists w \in W(x))[\{u, v\} = \{\langle x, x \rangle, \langle x, w \rangle\}] \\ \langle t, t1 \rangle & \text{otherwise,} \end{cases}$$

- Assume f is not strongly noninvertible via function g in PTIME
 - Given an output and one argument, the other argument can be computed in polynomial-time
- Thus, at least one of the 2 conditions of strong non invertibility holds from **Definition 2.14**
 - WLOG suppose that Case 2 of definition (see previous slide) holds
- If $x \in L(N)$,

$g(\underline{\langle x, x \rangle}, \underline{\langle x, x \rangle})$ outputs $\langle x, w \rangle$ where $w \in W(x)$

If direction \Leftarrow

$$f(u, v) = \begin{cases} \langle x, \text{lexmin}(w_1, w_2) \rangle & \text{if } u = \langle x, w_1 \rangle \wedge v = \langle x, w_2 \rangle \wedge \{w_1, w_2\} \subseteq W(x) \\ \langle x, x \rangle & \text{if } (\exists w \in W(x))[\{u, v\} = \{\langle x, x \rangle, \langle x, w \rangle\}] \\ \langle t, t1 \rangle & \text{otherwise,} \end{cases}$$

- If $x \in L(N)$,
 - $g(\underline{\langle x, x \rangle}, \underline{\langle x, x \rangle})$ outputs $\langle x, w \rangle$ where $w \in W(x)$
- Going to show a DPTIME algorithm to check an input $x \in L(N)$
 - On input x , compute $g(\underline{\langle x, x \rangle}, \underline{\langle x, x \rangle})$
 - **Reject**, if output is not of the form $\langle x, w \rangle$
 - Otherwise simulate $N(x)$ using w as computation path
 - **Accept** if $N(x)$ accepts
 - **Reject** otherwise

if $x \notin L(N)$, $g(\underline{\langle x, x \rangle}, \underline{\langle x, x \rangle})$ outputs anything, but since testing membership is PTIME, we can not be fooled!!

If direction \Leftarrow

- Showed a DPTIME algorithm to test membership in $L(N)$
- So $L(N) \in P$
- But we assumed $L(N) \in NP - P$
- **Contradiction!!**
- Our **assumption** that f is not strongly noninvertible **is wrong**
- So f is strongly non invertible
- Done!!!

If direction \Leftarrow

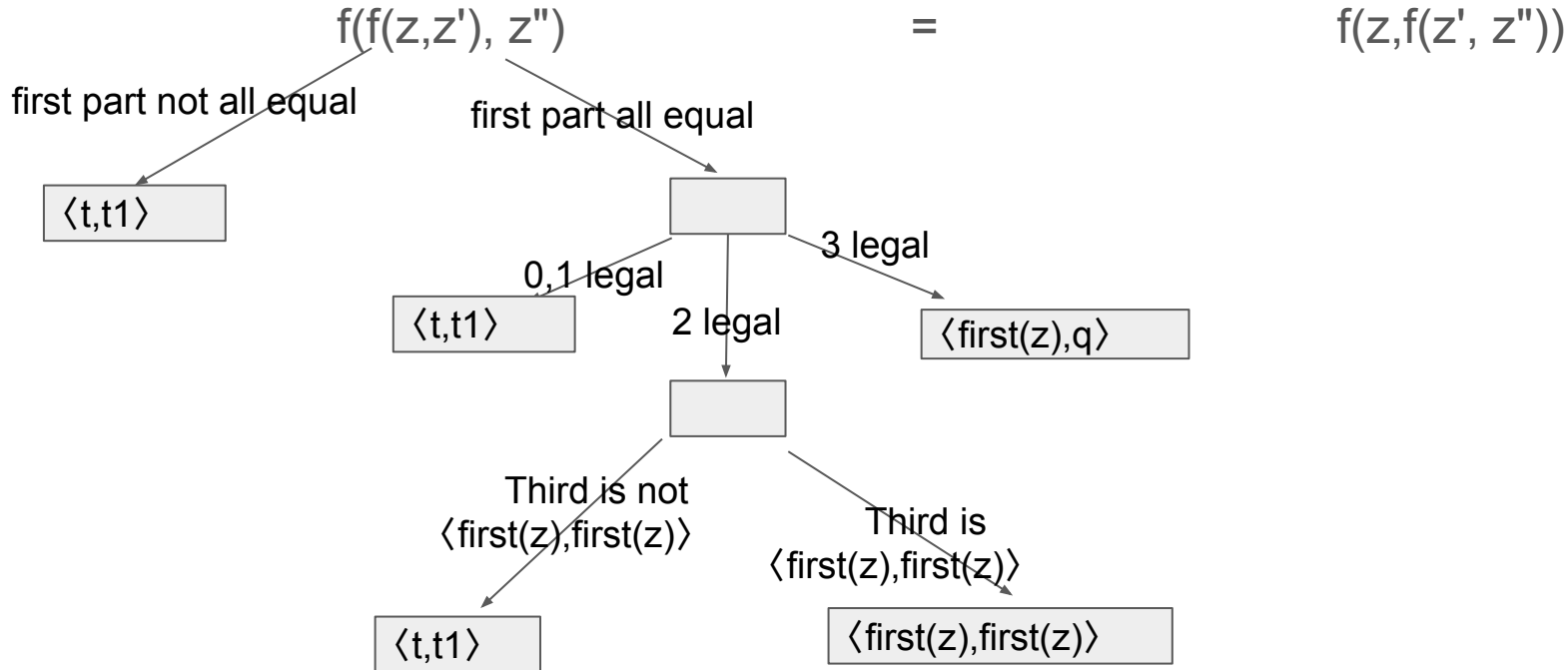
The only one left: **associative?**

$$f(f(z, z'), z'') = f(z, f(z', z''))?$$

$$f(u, v) = \begin{cases} \langle x, \text{lexmin}(w_1, w_2) \rangle & \text{if } u = \langle x, w_1 \rangle \wedge v = \langle x, w_2 \rangle \wedge \{w_1, w_2\} \subseteq W(x) \\ \langle x, x \rangle & \text{if } (\exists w \in W(x))[\{u, v\} = \{\langle x, x \rangle, \langle x, w \rangle\}] \\ \langle t, t1 \rangle & \text{otherwise,} \end{cases}$$

If direction \Leftarrow

We say a string a is legal if $(\exists x)(\exists w)[w \in W(x) \wedge a = \langle x, w \rangle]$.



If direction \Leftarrow

$$z = \langle x, w \rangle$$

$$z' = \langle x', w' \rangle$$

$$z'' = \langle x'', w'' \rangle$$

$$f(f(z, z'), z'') = f(z, f(z', z''))$$

$$\text{if } x \neq x' \quad f(z, z') = \langle t, t1 \rangle$$

$$f(\langle x, w \rangle, \langle x', w' \rangle)$$

$$\text{if } x \neq x'' \quad f(\langle x, w \rangle, \langle x', w' \rangle)$$

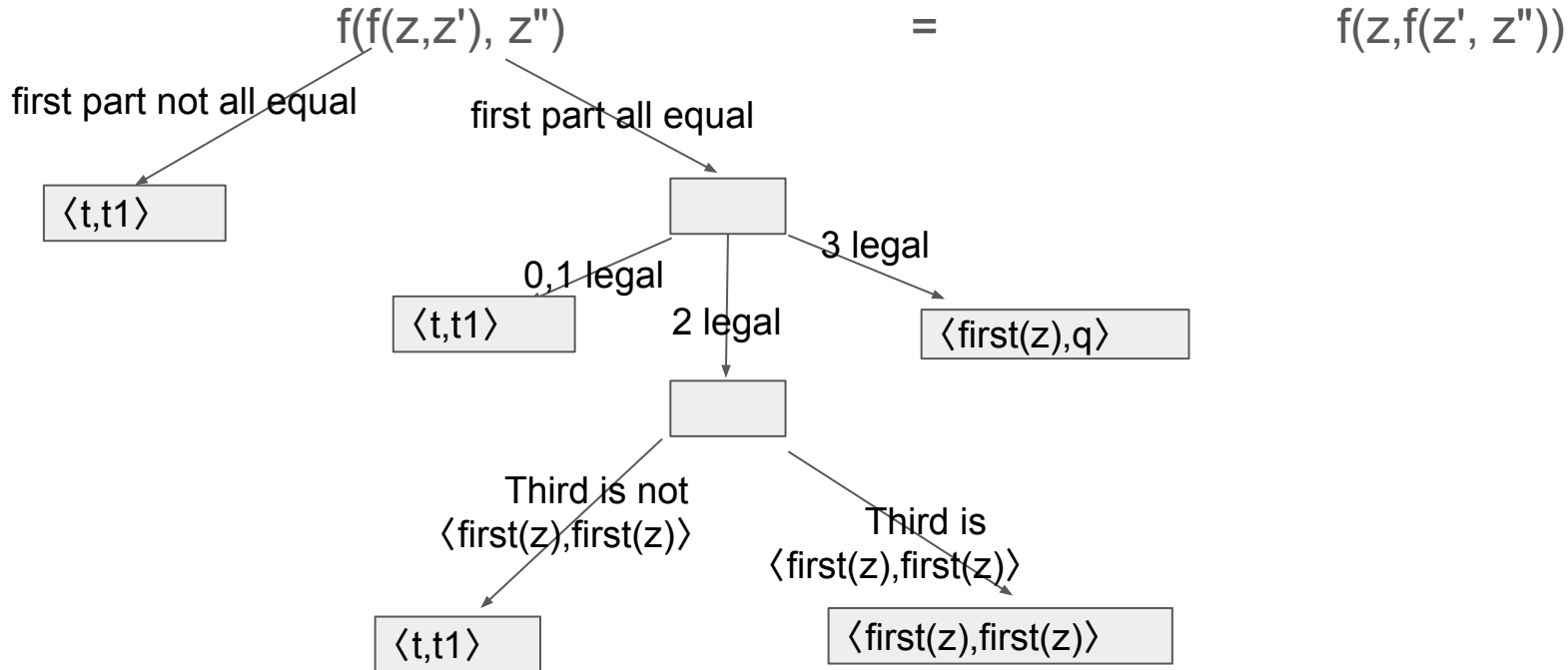
$$f(\langle x, w \rangle, \langle x', w' \rangle)$$

$$\text{if } x' \neq x'' \quad f(\langle x, w \rangle, \langle x', w' \rangle)$$

$$f(z, z') = \langle t, t1 \rangle$$

If direction \Leftarrow

We say a string a is legal if $(\exists x)(\exists w)[w \in W(x) \wedge a = \langle x, w \rangle]$.



If direction \Leftarrow

$$z = \langle x, w \rangle$$

$$z' = \langle x, w' \rangle$$

$$z'' = \langle x, w'' \rangle$$

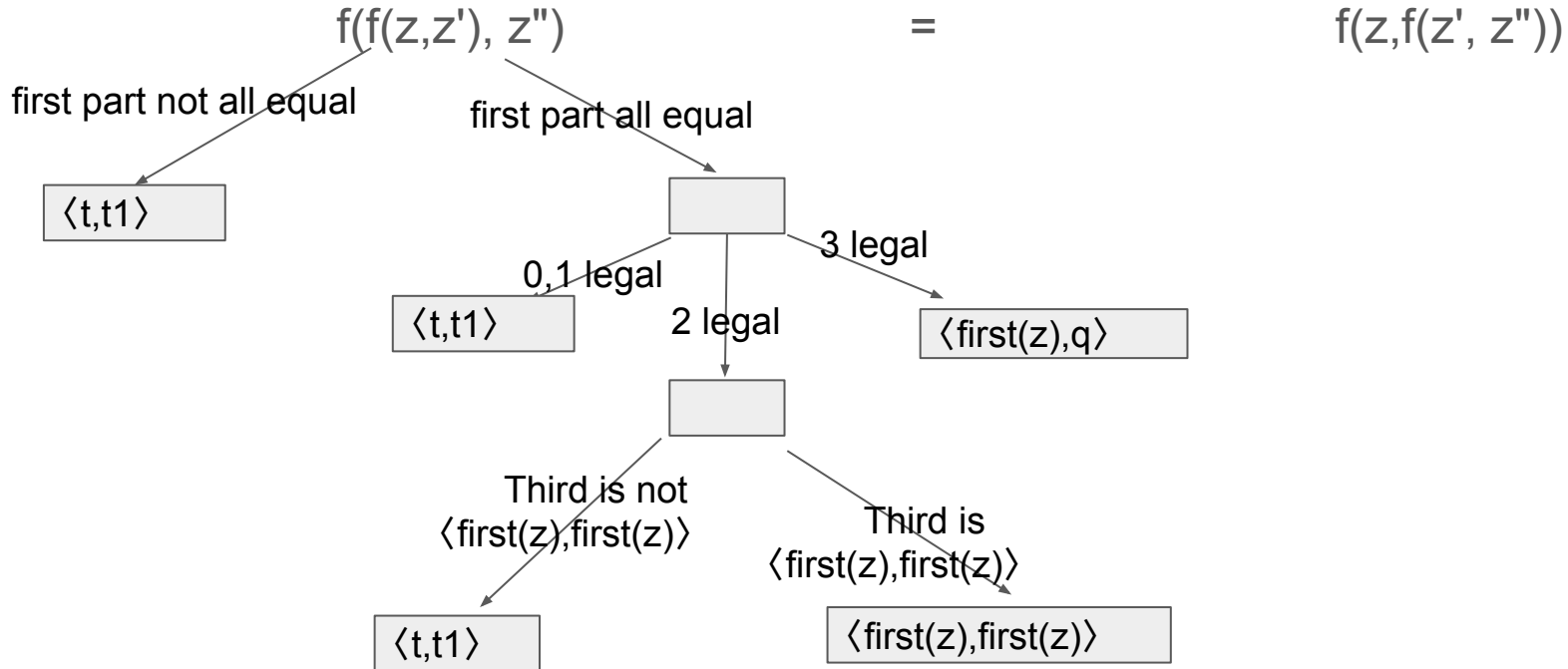
if w is witness, and w' , w'' are not,

$$f(z, z') = \langle x, x \rangle \text{ or } \langle t, t1 \rangle$$

$$f(\langle x, x \rangle, z'') = \langle t, t1 \rangle$$

If direction \Leftarrow

We say a string a is legal if $(\exists x)(\exists w)[w \in W(x) \wedge a = \langle x, w \rangle]$.



If direction \Leftarrow

$$z = \langle x, w \rangle$$

$$z' = \langle x, w' \rangle$$

$$z'' = \langle x, w'' \rangle$$

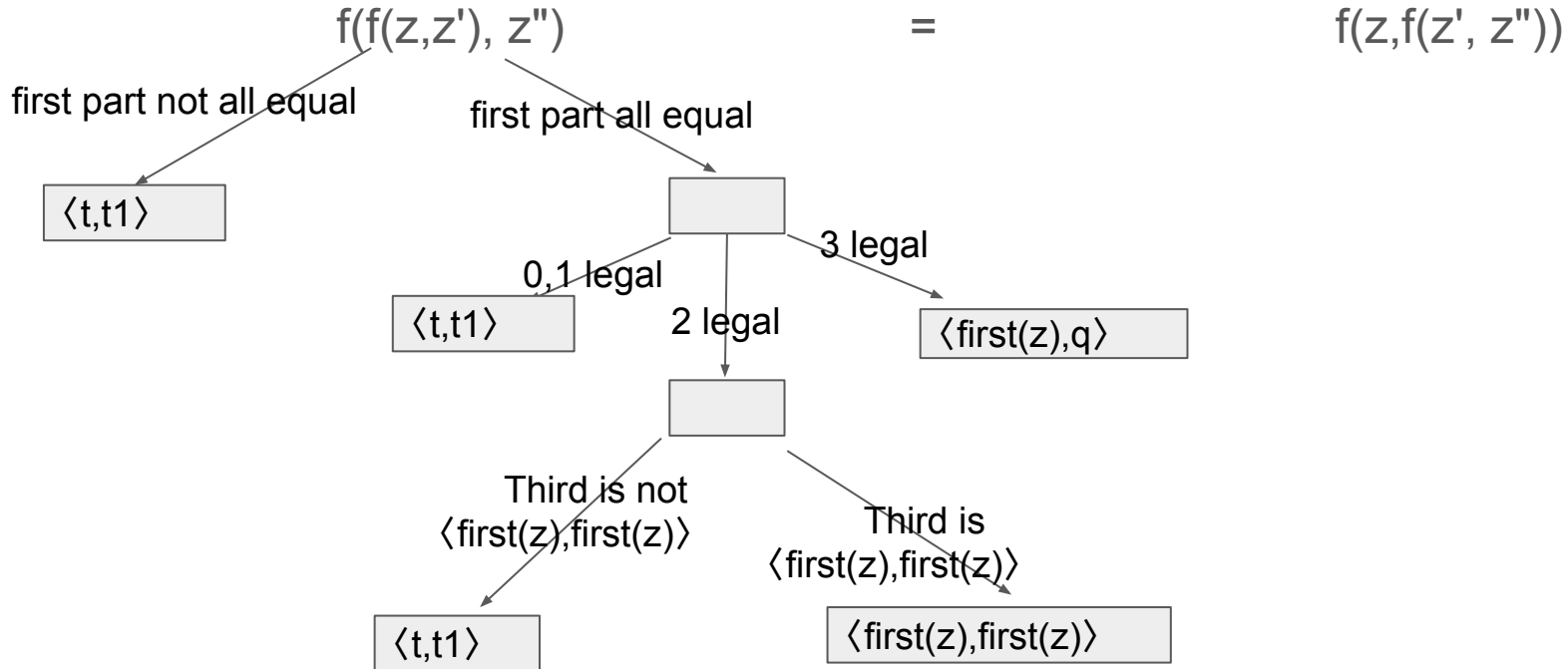
if w, w' are witnesses,

$$w'' = x, f(z, z') = \langle x, \min(w, w') \rangle \quad f(z'', \langle x, \min(w, w') \rangle) = \langle x, x \rangle$$

$$w'' \neq x, f(z', z'') = \langle t, t1 \rangle$$

If direction \Leftarrow

We say a string a is legal if $(\exists x)(\exists w)[w \in W(x) \wedge a = \langle x, w \rangle]$.



If direction \Leftarrow

$$z = \langle x, w \rangle$$

$$z' = \langle x, w' \rangle$$

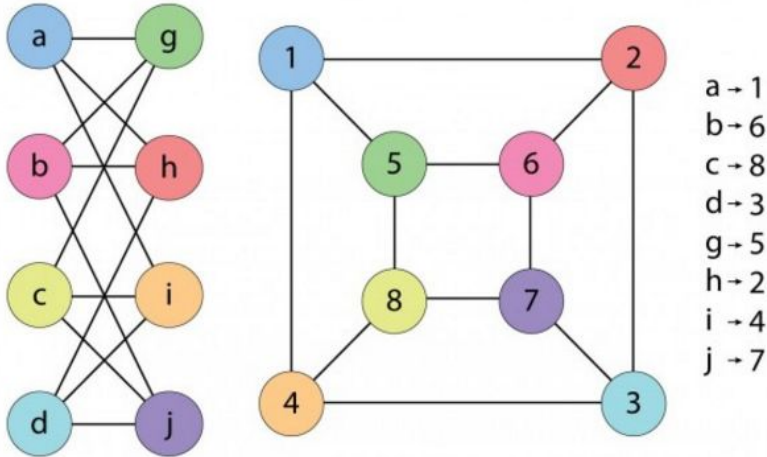
$$z'' = \langle x, w'' \rangle$$

if w, w', w'' are witnesses

$$f(z, z') = \langle x, \min(w, w') \rangle$$

$$f(x, \min(w, w'), z'') = \langle x, \min(w, w', w'') \rangle$$

LATEST NEWS



ADAPTED FROM: BOOYABAZOOKA WITH INFORMATION FROM GRAPHISOMORPHISM1.PNG ON W:EN:GRAPH ISOMORPHISM; WIKIMEDIA/CREATIVE COMMONS

In the graph isomorphism problem, the challenge is to determine whether two apparently different graphs can be rearranged to be identical, as these two can.

Mathematician claims breakthrough in complexity theory

Tweet 1,312 Share 22k G+ 468



Staff Writer

Email Adrian

By Adrian Cho | 10 November 2015 5:45 pm | 35 Comments

For days, rumors about the biggest advance in years in so-called complexity theory have been lighting up the Internet. That's only fitting, as the breakthrough involves comparing networks just like researchers' webs of online connections. László Babai, a mathematician and computer scientist at the University of Chicago in Illinois, has developed a mathematical recipe or "algorithm" that supposedly can take two networks—no matter how big and tangled—and tell whether they are, in fact, the same, in far fewer steps than the previous best algorithm. Computer scientists are abuzz, as the task had been

something of a poster child for hard-to-solve problems.

<http://news.sciencemag.org/math/2015/11/mathematician-claims-breakthrough-complexity-theory>