Chapter 5
The Witness Reduction Technique

Luke Dalessandro Rahul Krishna

December 6, 2006

Outline of Part I

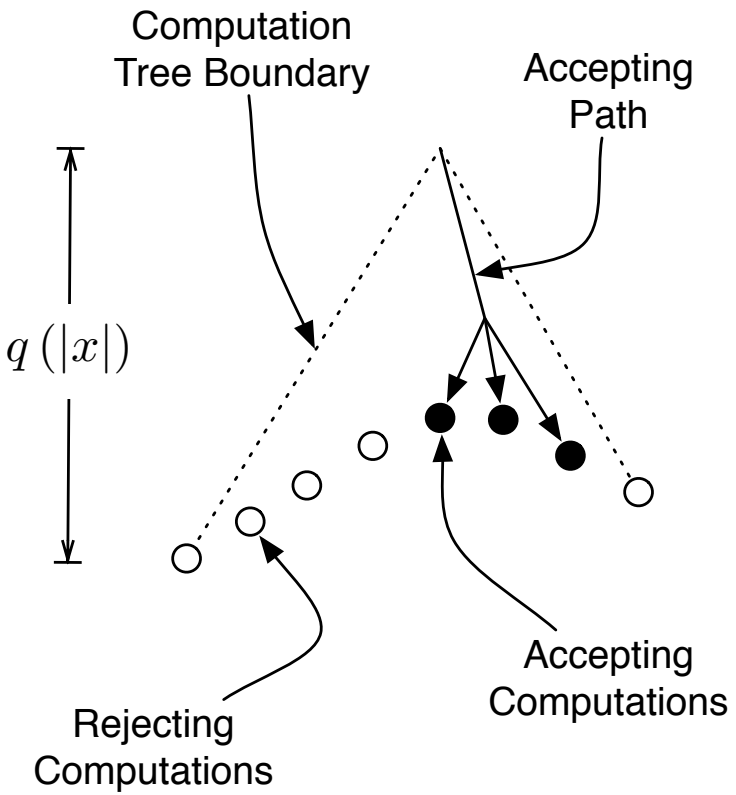
1. Notes On Our NP Computation Model
   - NP Machines
2. Complexity Soup
   - NP
   - UP
   - PP
   - $\oplus$P
   - #P

Outline of Part II

Part I

Background Material

Our previous NP machine model (informally)



- Polynomially bounded runtime
  - $q(|x|)$ here
- Non-deterministic transition function
  - Branching factor based on machine constants
  - Limited by # of states, tape alphabet, tape configuration
- Accepting state implies halting

Figure: Computation Tree

Adjusted NP machine model (informally)

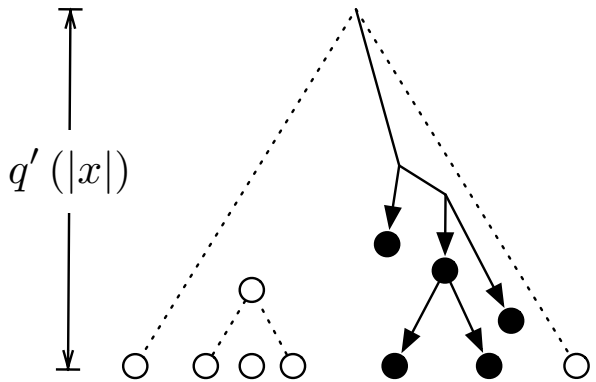


Figure: Adjusted Tree

- Want a complete balanced binary tree
- Binary by restricting δ function branching factor to 2
  - Increases tree size but is independent from input
- Balanced and complete by extending all computation paths to $q'(|x|)$
  - Pre-compute q' and decrement as we compute
  - Detect accept/reject and continue with dummy states if needed
- Restrict alphabet to $\{0, 1\}$ w.l.o.g. (we've done this before)

Review of NP

Definition

A language L is in NP if there exists a polynomial-time computable predicate R and a polynomial q such that for all x,

$$L = \left\{ x \middle| \left(\exists y : |y| \leq q(|x|)\right) [R(x, y)] \right\}$$

NP computation

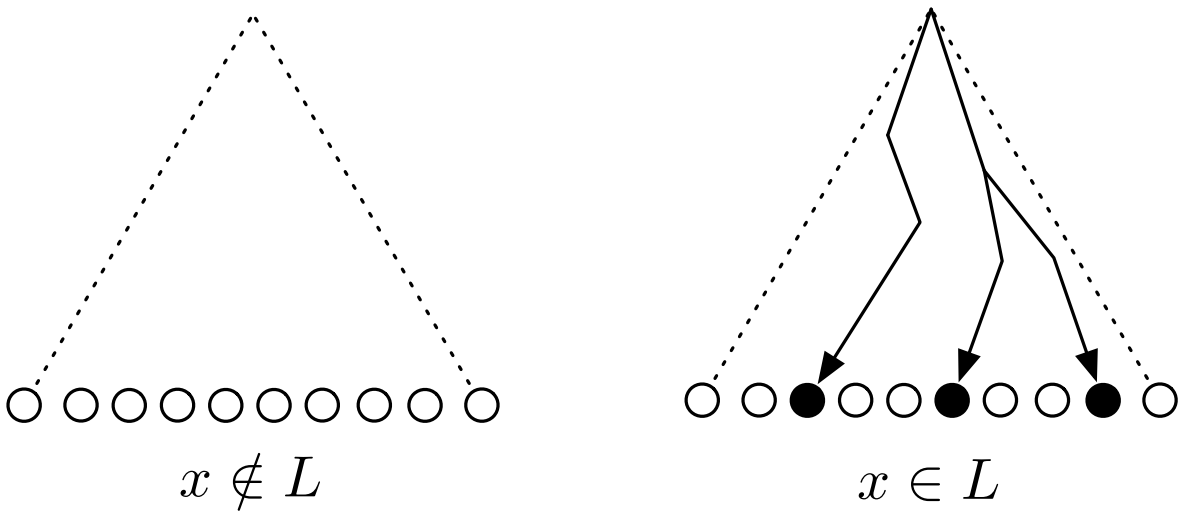


Figure: Example NP Computation Trees

- Languages in NP are characterized by NP machines that have at least one accepting path for $x \in L$, and have no accepting paths for $x \notin L$.

Review of UP

Definition

A language L is in UP if there is a polynomial-time predicate P and a polynomial q such that for all x,

$$\|\{y||y| \leq q(|x|) \wedge P(x,y)\}\| = \begin{cases} 0 & \text{if } x \notin L \\ 1 & \text{if } x \in L \end{cases}$$

UP computation

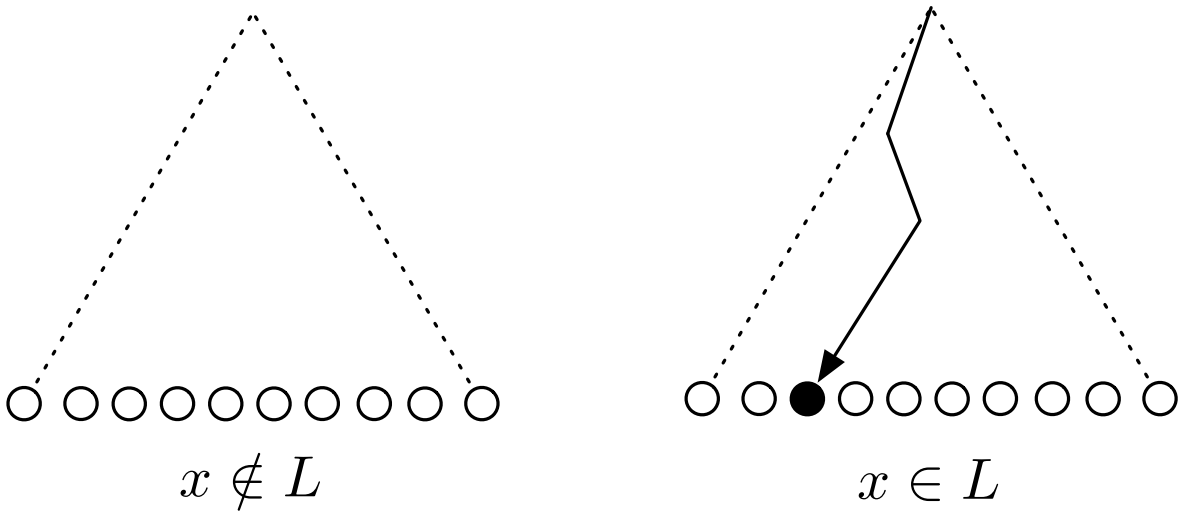


Figure: Example UP Computation Trees

- Languages in UP are characterized by NP machines that have exactly one accepting path for $x \in L$ and no accepting paths for $xd \notin L$.

Probabilistic-Polynomial, PP

Definition

A language L is in PP if there exists a polynomial q and a polynomial-time predicate R such that for all x,

$$x \in L \Leftrightarrow \left\| \left\{ y \middle| |y| = q(|x|) \wedge R(x,y) \right\} \right\| \geq 2^{q(|x|)-1}$$

PP computation

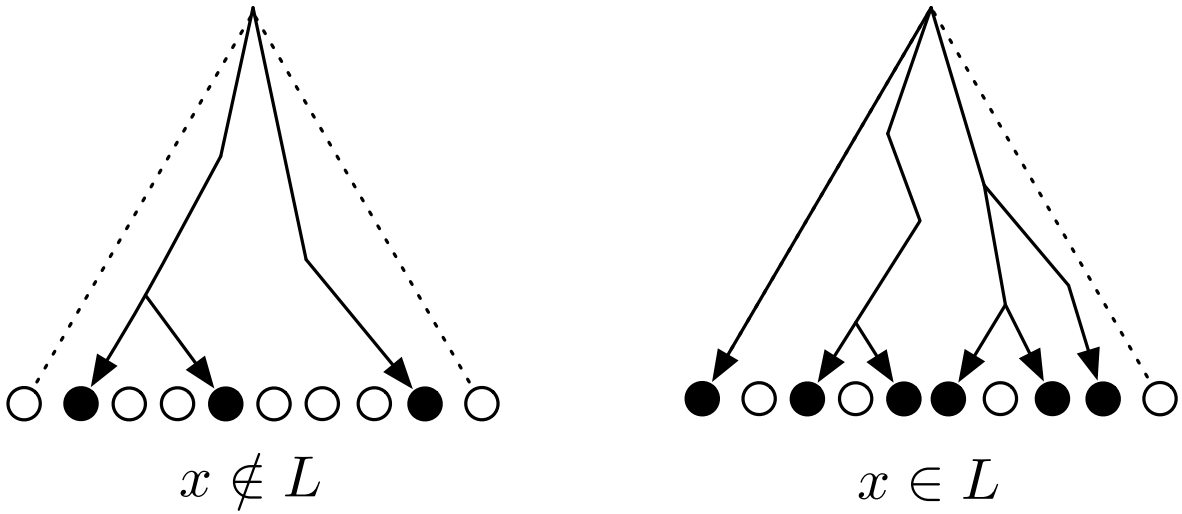


Figure: Example PP Computation Trees

- Languages in PP are characterized by NP machines that accept along at least half of their computation paths for $x \in L$, and reject on at least half of their paths for $x \notin L$.

Parity-P, ⊕P

Definition

A language L is in ⊕P if there is a polynomial time predicate P and a polynomial q such that for all x,

$$x \in L \Leftrightarrow \left\| \{ y \big| |y| \leq q(|x|) \wedge P(x,y) \} \right\| \not\equiv 0 \pmod 2$$

- Languages in the class ⊕P are characterized by NP machines that have an odd number of accepting paths for $x \in L$.
- We will talk more about ⊕P on Wednesday.

Sharp-P, #P

Definition

A function f is in #P if there is a polynomial time predicate P and a polynomial q such that for all x,

$$\left\|\left\{y \middle| |y| \leq q(|x|) \wedge P(x,y)\right\}\right\| = f(x)$$

#P continued

- Note that #P is a class of functions rather than a class of languages
- Each #P function is defined by a NP machine
- Each NP machine defines a #P function

#P continued

Example

Let L be a UP language. Consider the NPTM N that accepts L, and that for each $x \in L$ has exactly one accepting path, and 0 accepting paths for $x \notin L$. This N defines the #P function f such that

$$f(x) = \begin{cases} 0 & \text{if } x \notin L \\ 1 & \text{if } x \in L \end{cases}$$

Class relationships

	NP	UP	PP
$x \in L$	≥ 1	1	$\geq \frac{2^{q(\lvert x \rvert)}}{2}$
$x \notin L$	0	0	$< \frac{2^{q(\lvert x \rvert)}}{2}$

Table: Number of accepting paths for NP machines characterized by each class

Part II

Chapter 5

Mapping strings to natural numbers

- When considering closure properties, #P functions, and NPTMs, it is convenient to use strings and natural numbers interchangeably.
- There exists a natural bijection between strings and natural numbers.
  - The lexicographically first string in $\Sigma^\star$ is mapped to 0
  - The lexicographically second string in $\Sigma^\star$ is mapped to 1
  - etc
- We'll use this bijection implicitly whenever necessary in the following discussion.

Closure properties

Definition

Unless otherwise stated, an operation is a mapping from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$.

Definition

Let σ be an operation and let $\mathcal{F}$ be a class of functions from $\mathbb{N}$ to $\mathbb{N}$. We say that $\mathcal{F}$ is closed under (the operation) σ if

$$(\forall f_1 \in \mathcal{F})(\forall f_2 \in \mathcal{F})[h_{f_1,f_2} \in \mathcal{F}]$$

where $h_{f_1,f_2}(n) = \sigma(f_1(n), f_2(n))$.

Closure property example for #P

Theorem

#P is closed under addition

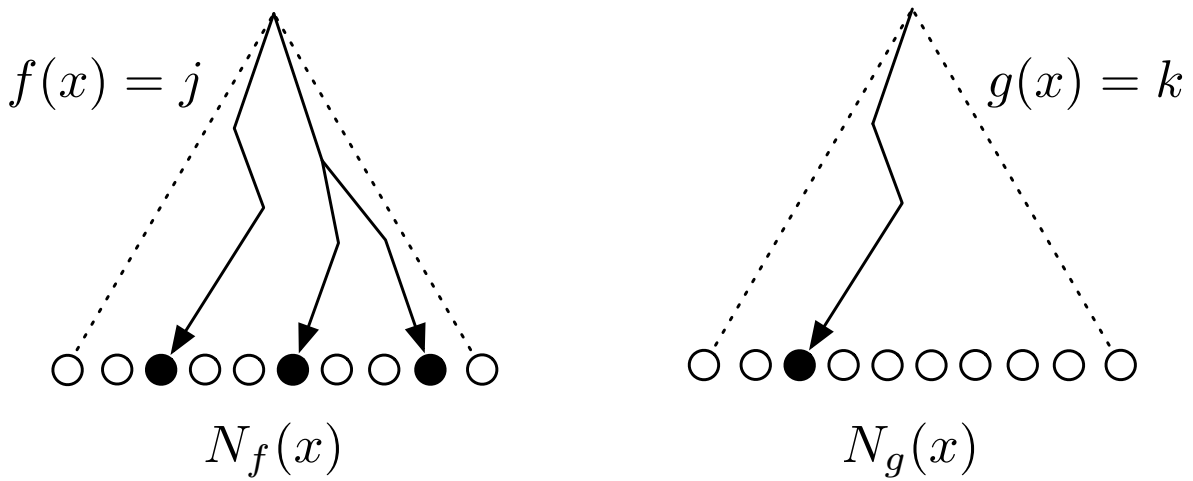
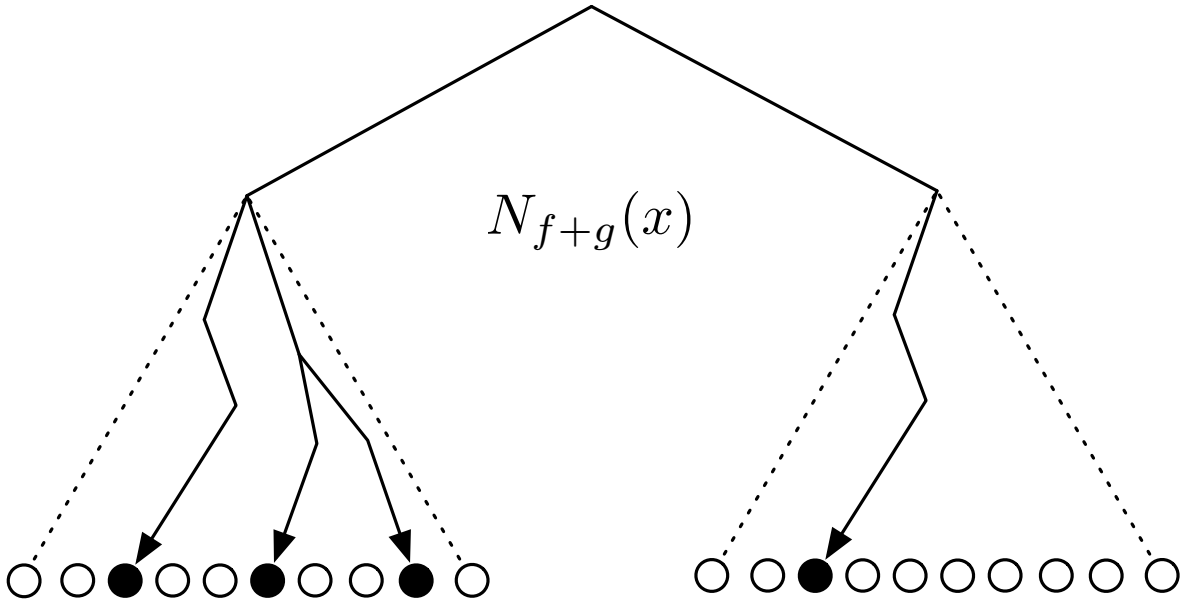


Figure: NP machines witnessing f and g

Closure example continued



$$h(x) = f(x) + g(x) = j + k$$

Figure: NP machine witnessing $f + g$

Non-obvious properties

- What if it is not obvious how to prove or disprove a closure property?
- Is #P closed under proper subtraction?
  - Proper subtraction $m \ominus n = max(m - n, 0)$
  - TM construction doesn't work
  - Maybe proof by contradiction?
- Assume the class is closed under the property and look for consequences

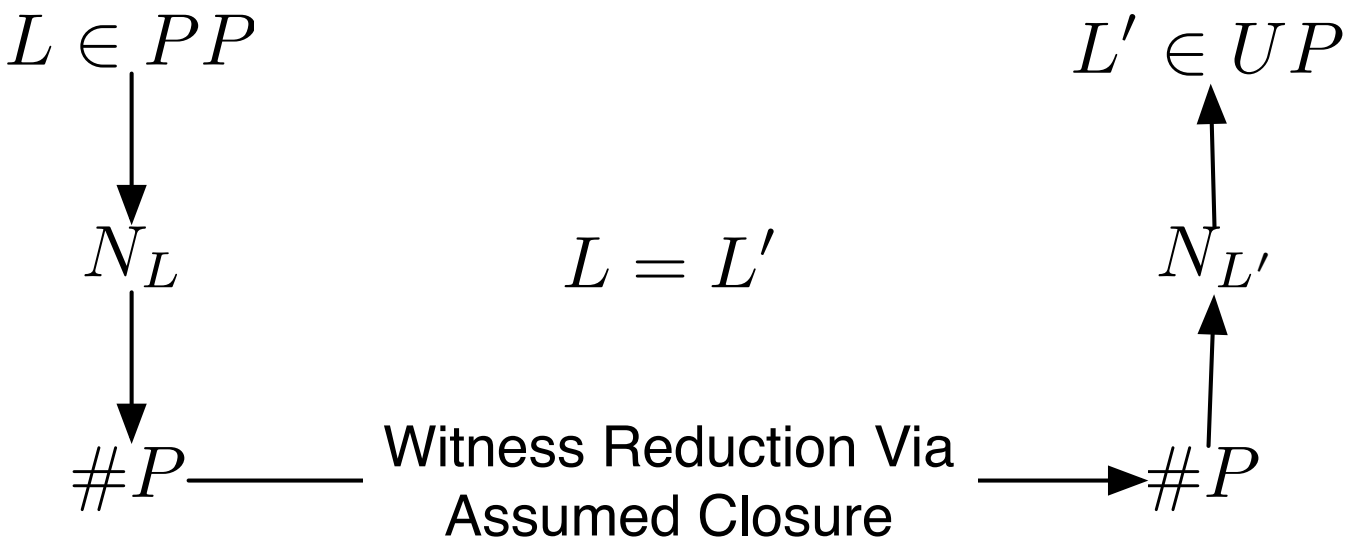
The Witness Reduction Technique

- The Witness Reduction Technique exactly follows this second proposal
- Use an assumed $\#P$ closure property that reduces the number of witnesses of its associated machine to show complexity class collapse.

The witness reduction algorithm

1. Take a set in a large complexity class (e.g. PP), take the machine for the set, and examine the #P function that the machine defines
2. Use an assumed witness-reducing closure to create a new #P function
3. Examine a machine for this new #P function, preferably one that defines the language in a smaller class (e.g. UP)

The witness reduction algorithm continued



Theorem 5.6

Theorem

The following statements are equivalent:

1. $\#P$ *is closed under proper subtraction.*
2. $\#P$ *is closed under every polynomial-time computable operation.*
3. $UP = PP$

$2 \Rightarrow 1$

Assume #P is closed under every polynomial-time computable operation

Show #P is closed under proper subtraction

Proof

This implication is trivial as proper subtraction is a polynomial-time computable operation.

$1 \Rightarrow 3$

Assume #P is closed under proper subtraction

Show UP=PP (equivalently UP$\subseteq$PP and PP$\subseteq$UP)

Outline

1. Show UP$\subseteq$PP directly
2. Show PP$\subseteq$coNP via witness reduction
3. Show coNP$\subseteq$UP via witness reduction

UP$\subseteq$PP

- This condition holds independent of the assumption.
- Let L be a UP language. Let N be the NPTM that accepts L.
- From the definition of UP
  - $\exists$ polynomial q such that $q(|x|)$ is the depth of N's computation tree
  - For $x \in$L the number of accepting paths of $N(x)$ is 1
  - For $x \notin$L the number of accepting paths of $N(x)$ is 0

UP$\subseteq$PP continued

- Let N' be a NPTM with the same q as N, and that accept on all paths except one
- Consider NPTM N_{PP} whose first step on input x is to non-deterministically choose to simulate N or N'
  1. N_{PP} has $2^{q(|x|)+1}$ total computation paths
  2. For $x \in$ L, N contributes 1 accepting path and N' contributes $2^{q(|x|)} - 1$ accepting paths for a total of $2^{q(|x|)}$ accepting paths
  3. For $x \notin$ L, there are only N''s $2^{q(|x|)} - 1$ accepting paths
- N_{PP} demonstrates that L$\in$PP since
  1. For $x \in$L exactly half of the paths of N_{PP} accept
  2. For $x \notin$L strictly less than half accept

$1 \Rightarrow 3$

Assume #P is closed under proper subtraction

Show UP=PP (equivalently UP$\subseteq$PP and PP$\subseteq$UP)

Outline

1. Show UP$\subseteq$PP directly
2. Show PP$\subseteq$coNP via witness reduction
3. Show coNP$\subseteq$UP via witness reduction

PP$\subseteq$coNP

- Let L be a PP language. From the definition of PP we have a polynomial q and a polynomial-time predicate R such that

$$x \in L \Leftrightarrow \left\| \{ y \big| |y| = q(|x|) \wedge R(x,y) \} \right\| \geq 2^{q(|x|)-1}$$

- Let $q'(x) = q(n) + 1$ and for $b \in \{0,1\}$, $R'(x, yb) = R(x,y)$ and require that for all n $q(n) \geq 1$
- Consider the NPTM that on input x guesses each y such that $|y| = q(|x|)$ and tests $R(x,y)$.

PP$\subseteq$coNP continued

- Consider the #P function f defined by this NPTM
  - $x \in \mathrm{L} \Rightarrow f(x) \geq 2^{q(|x|)-1}$
  - $x \notin \mathrm{L} \Rightarrow f(x) < 2^{q(|x|)-1}$
- Consider the #P function $g(x) = 2^{q(|x|)-1} - 1$
- Under the assumption that #P is closed under proper subtraction, we have #P function h such that
  - $h(x) = f(x) \ominus g(x)$
- Substitution yields
  - $h(x) \geq 1$ if $x \in \mathrm{L}$
  - $h(x) = 0$ if $x \notin \mathrm{L}$

PP⊆coNP continued

- There exists a NPTM $N(x)$ for which $h(x)$ computes the number of accepting paths.
- Based on the values of $h(x)$, N is an NP machine, thus L=L(N) and PP⊆NP
- Since PP=coPP, we have that PP⊆coNP

$1 \Rightarrow 3$

Assume #P is closed under proper subtraction

Show UP=PP (equivalently UP$\subseteq$PP and PP$\subseteq$UP)

Outline

1. Show UP$\subseteq$PP directly
2. Show PP$\subseteq$coNP via witness reduction
3. Show coNP$\subseteq$UP via witness reduction

coNP$\subseteq$UP

- Let L be an arbitrary coNP language.
- There exists a NPTM N that accepts $\overline{\mathrm{L}}$
- N defines #P function f such that
  - $x \in \mathrm{L} \Rightarrow f(x) = 0$
  - $x \notin \mathrm{L} \Rightarrow f(x) \geq 1$
- Consider the constant #P function $g(x) = 1$

coNP$\subseteq$UP continued

- Since #P is closed under $\ominus$ there exists a #P function h where
  - $h(x) = g(x) \ominus f(x)$
- Substitution yields
  - $h(x) = 1$ if $x \in$L
  - $h(x) = 0$ if $x \notin$L
- By the same reasoning as before, $h(x)$ has an associated UP machine, thus our arbitrary coNP language is also in UP

1⇒3 complete

1⇒3

We have shown that UP⊆PP and that PP⊆coNP⊆UP, thus we have shown that If #P is closed under proper subtraction then UP=PP.

3⇒2

Assume UP=PP

Show #P is closed under every polynomial-time computable operation

Proof Strategy

Given that f and g are arbitrary #P functions and that op is an arbitrary polynomial-time operation, and given the assumption that UP=PP, we must show that $h(x) = op(f(x), g(x))$ is also a #P function.

3⇒2

- Our first goal is to actually compute the values for $f(x)$ and $g(x)$ for arbitrary input x
- We use the following two sets for this computation
  - $B_f = \{\langle x, n\rangle | f(x) \geq n\} \in$PP
  - $B_g = \{\langle x, n\rangle | g(x) \geq n\} \in$PP
- However we need the precise values for $f(x)$ and $g(x)$ which we can get using the set

$$V = \{\langle x, n_1, n_2\rangle | \quad \langle x, n_1\rangle \in B_f \wedge \langle x, n_1 + 1\rangle \notin B_f \wedge \langle x, n_2\rangle \in B_g \wedge \langle x, n_2 + 1\rangle \notin B_g\}$$

3⇒2 continued

- V decides $n_1 = f(x) \wedge n_2 = g(x)$ by testing adjacent ns to find the transition points in B_f and B_g
- Let $\oplus$ indicate disjoint union

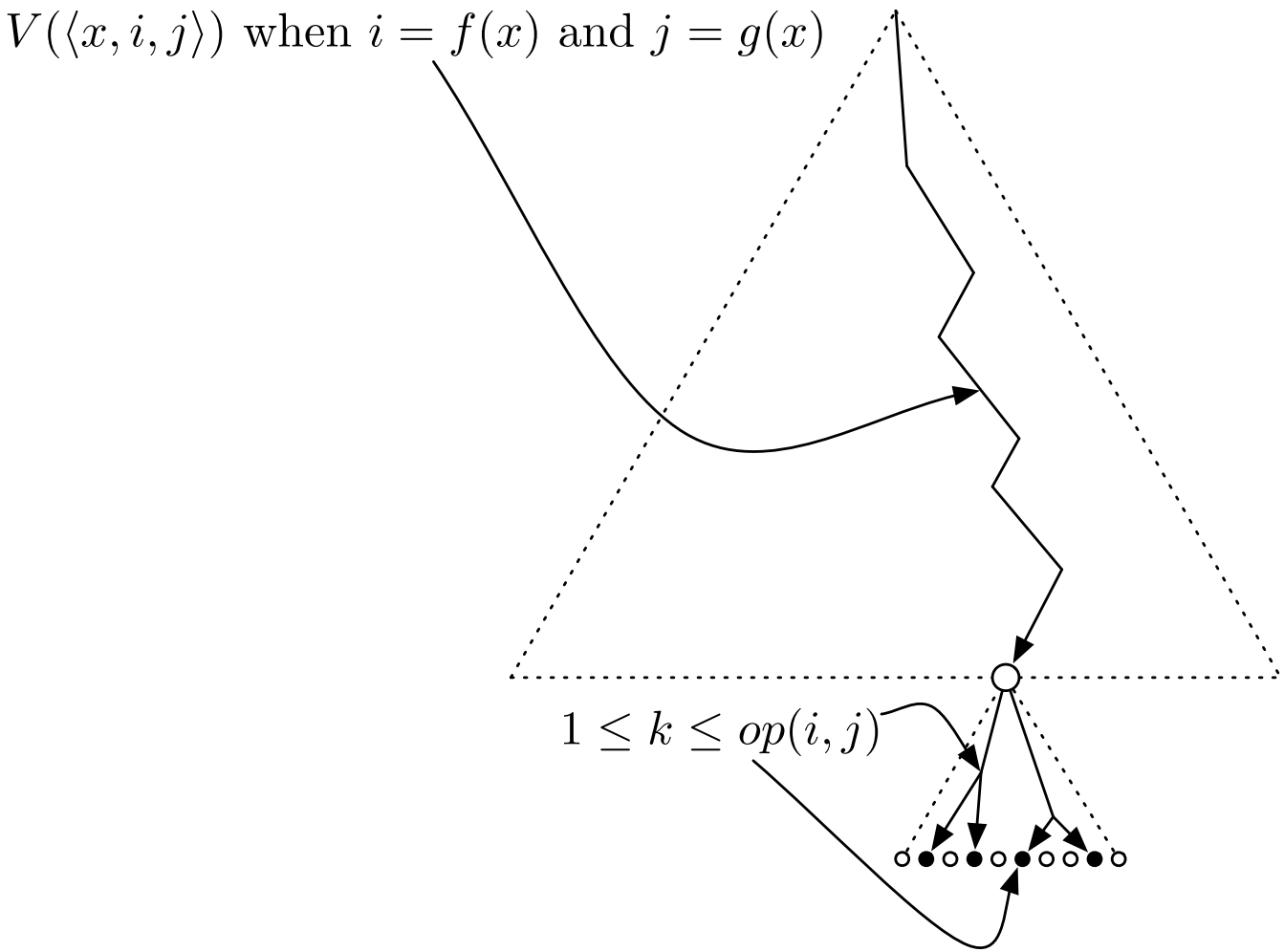
$$V \leq^p_{4\text{-}tt} (B_f \oplus B_g) \text{ and } B_f \oplus B_g \in \text{PP}$$

- Theorem 9.17 shows us that PP is closed under $\leq^p_{btt}$ and disjoint union so we conclude that $V \in PP$
- From our assumption that UP=PP we conclude that $V \in$UP

3⇒2 continued

- With V in UP, and able to test if $f(x) = n_1$ and $g(x) = n_2$, we examine the following NPTM, N that will show $h(x) = op(f(x), g(x))$ and $h(x) \in \#\mathrm{P}$
- f and g are $\#\mathrm{P}$ functions so there is some polynomial q such that $max\{f(x), g(x)\} \leq 2^{q(|x|)}$
- N, on input x
  1. Nondeterministically choose an integer i, $0 \leq i \leq 2^{q(|x|)}$
  2. Nondeterministically choose an integer j, $0 \leq j \leq 2^{q(|x|)}$
  3. Guesses a computation path of V on input $\langle x, i, j \rangle$. If this path accepts, nondeterministically guess an integer k, $1 \leq k \leq op(i, j)$ and accept.

3⇒2 continued



3⇒2 continued

- For all $i \neq f(x)$ and $j \neq g(x)$, $V(\langle x, i, j \rangle)$ rejects (recall $V \in$UP)
- For the correct i and j, $N(x)$ accepts along precisely $op(i, j)$ paths
- The #P function defined by this machine is $h(x) = op(f(x), g(x))$ thus #P is closed under our arbitrary op

Theorem 5.7

Theorem

The following statements are equivalent:

1. $UP = PP$.
2. $UP = NP = coNP = PH = \oplus P = PP = PP \cup PP^{PP} \cup PP^{PP^{PP}} \cup \ldots$

- To prove this, we need other results.
- We prove each of these results one by one.
- We use $\mathrm{UP} = \mathrm{PP}$ as the initial assumption.
- We use results for each stage as assumptions for the next stage.

UP $\subseteq$ NP

Proposition

$UP \subseteq NP$

Proof.

Let $L \in \mathrm{UP}$. Let N be the NPTM deciding L.

1. $x \in L \implies$ exactly one accepting path in N
2. $x \notin L \implies$ no accepting paths in N

Clearly, $L \in \mathrm{NP}$. □

$\mathrm{NP} \subseteq \mathrm{PP}$

Proposition

$NP \subseteq PP$.

Construction

1. Let $L \in \mathrm{NP}$ and let NPTM N decide L.
2. Construct NPTM N' that has two subtrees at its root
3. Left subtree is exactly the same as N.
4. Right subtree is of the same depth as N and has exactly one rejecting path.
5. $x \in L \implies$ no. of accepting paths in $N' \geq \frac{1}{2}(\#paths_{N'})$
6. $x \notin L \implies$ no. of accepting paths in $N' < \frac{1}{2}(\#paths_{N'})$

NP $\subseteq$ PP (Example)

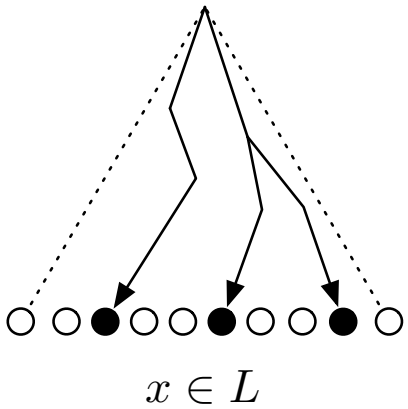


Figure: Computation Tree of NPTM N

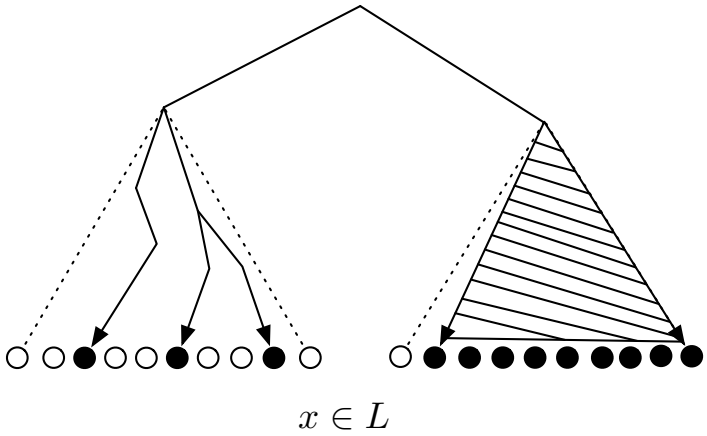


Figure: Computation Tree of NPTM N'

NP $\subseteq$ PP (Example)

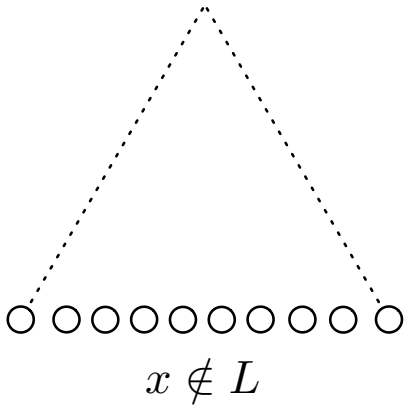


Figure: Computation Tree of NPTM N

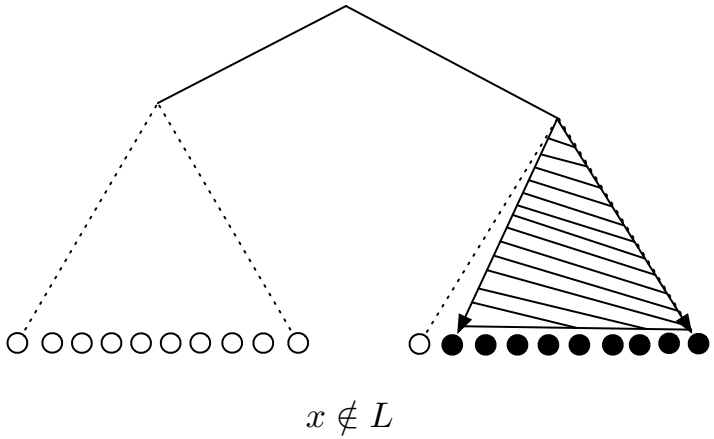


Figure: Computation Tree of NPTM N'

$\mathrm{UP} = \mathrm{NP} = \mathrm{PP}$

Proposition

If $UP = PP$, then $UP = NP = PP$

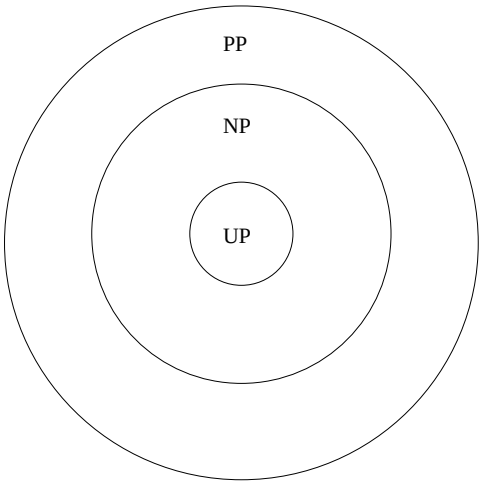


Figure: Known relationship between UP , NP , PP

Known Facts & Assumptions

- $\mathrm{UP} \subseteq \mathrm{NP} \subseteq \mathrm{PP}$.
- $\mathrm{UP} = \mathrm{PP}$

Clearly, given the assumptions, $\mathrm{UP} = \mathrm{NP} = \mathrm{PP}$

Status

$$\mathrm{UP}=\mathrm{PP}=\mathrm{NP}=\mathrm{coNP}=\mathrm{PH}=\oplus\mathrm{P}=\mathrm{PP}\cup\mathrm{PP}^{\mathrm{PP}}\cup\mathrm{PP}^{\mathrm{PP}^{\mathrm{PP}}}\cup\ldots$$

PP is closed under complementation

Proposition

PP is closed under complementation

Construction

Construction: Outline

1. Let $L \in \mathrm{PP}$ and let NPTM N decide L.
2. Construct NPTM N' that is equivalent to N and has the rightmost path as a rejecting path
3. Construct NPTM N'' by adding another level to N' by adding 2 child nodes to each of the leaf nodes.
4. For the leaf node of the rightmost path, one child is accepting and the other is rejecting
5. For accepting leaf nodes, both children are rejecting.
6. For rejecting leaf nodes (other than the rightmost leaf node), both children are accepting

Construction: Details

We can construct NPTM N' that is equivalent to N and has the rightmost path as a rejecting path by

1. Construct NPTM N' that has two subtrees at its root
2. Left subtree is exactly the same as N.
3. Exactly half the paths of right subtree are accepting and the remaining half are rejecting.
4. $x \in L \implies$ no. of accepting paths in $N' \geq \frac{1}{2}(\#paths_{N'})$
5. $x \notin L \implies$ no. of accepting paths in $N' < \frac{1}{2}(\#paths_{N'})$

Example: Construction of N'

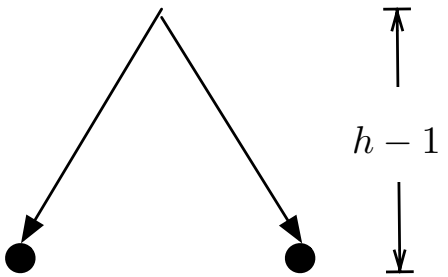


Figure: NPTM N

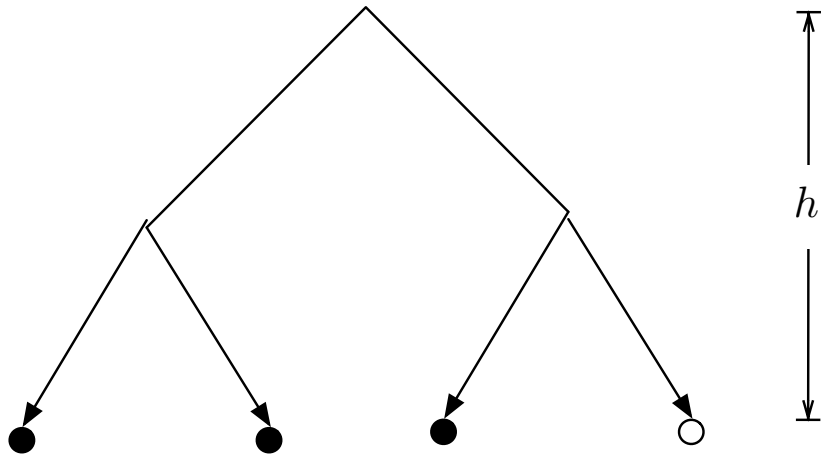


Figure: NPTM N'

Example: Construction of N''

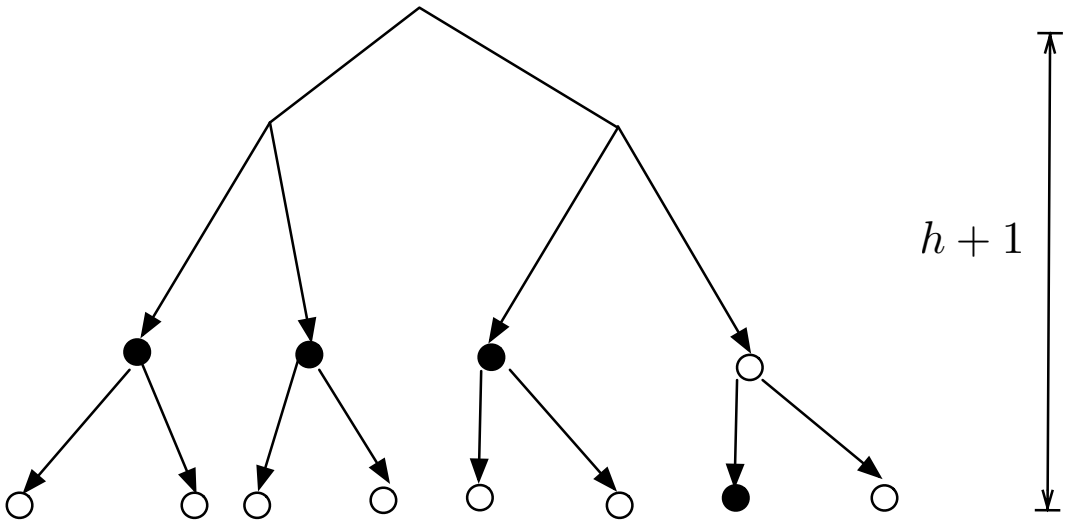


Figure: NPTM N''

Correctness

Let $h-1$ represent the depth of the computation tree of N.
Let y represent the number of accepting paths in N'
We see that the number of accepting and rejecting paths in N'' is:

1. Number rejecting: $2y+1$
2. Number accepting: $2^{h+1}-2y-1$

Correctness(contd)

1. Case 1: $x \in L \implies y \geq 2^{h-1}$
   In this case, the number of accepting paths in $N'' \leq 2^h - 1$.
   $2^h - 1 < 2^h$.
2. Case 2: $x \notin L \implies y < 2^{h-1}$
   In this case, the number of accepting paths in $N'' \geq 2^h + 1$.
   Clearly, $2^h + 1 > 2^h$.

Hence, $\overline{L} \in \mathrm{PP}$.

$\mathrm{UP} = \mathrm{NP} = \mathrm{PP} = \mathrm{coNP}$

Proposition

If $NP = PP$, then $NP = coNP$

Known Facts & Assumptions

- $\mathrm{NP} = \mathrm{PP}$
- PP is closed under complementation

Proof

Proof.

$(\forall L), L \in \mathrm{PP} => \overline{L} \in \mathrm{PP}$
Since we have assumed that $\mathrm{NP} = \mathrm{PP}$, we have,
$\overline{L} \in \mathrm{PP} => \overline{L} \in \mathrm{NP} => L \in \mathrm{coNP}$
Therefore, $(\forall L), L \in \mathrm{PP} \implies L \in \mathrm{coNP}$.

Since, $\mathrm{PP} \subseteq \mathrm{coNP}$ and (since $\mathrm{NP} \subseteq \mathrm{PP}$) $\mathrm{coNP} \subseteq \mathrm{coPP} = \mathrm{PP}$, we have
$\mathrm{NP} = \mathrm{PP} = \mathrm{coNP}$

Status

$$\mathrm{UP}=\mathrm{PP}=\mathrm{NP}=\mathrm{coNP}=\mathrm{PH}=\oplus\mathrm{P}=\mathrm{PP}\cup\mathrm{PP}^{\mathrm{PP}}\cup\mathrm{PP}^{\mathrm{PP}^{\mathrm{PP}}}\cup\ldots$$

PH = NP

Theorem

If $NP = coNP$, then $PH = NP$.

Definition

$$\mathrm{PH} = \bigcup_i \Sigma_i^p = \mathrm{P} \cup \mathrm{NP} \cup \mathrm{NP}^{\mathrm{NP}} \cup \mathrm{NP}^{\mathrm{NP}^{\mathrm{NP}}} \cup \ldots$$

We first show that if $\mathrm{NP} = \mathrm{coNP}$, then $\mathrm{NP}^{\mathrm{NP}} = \mathrm{NP}$.

PH = NP (contd.)

Let $A \in \mathrm{NP}$. We can build an NPTM N'_A having the power of an oracle making use of NPTMs N_A that decides A, and $N_{\overline{A}}$ that decides $\overline{A}$ as follows:

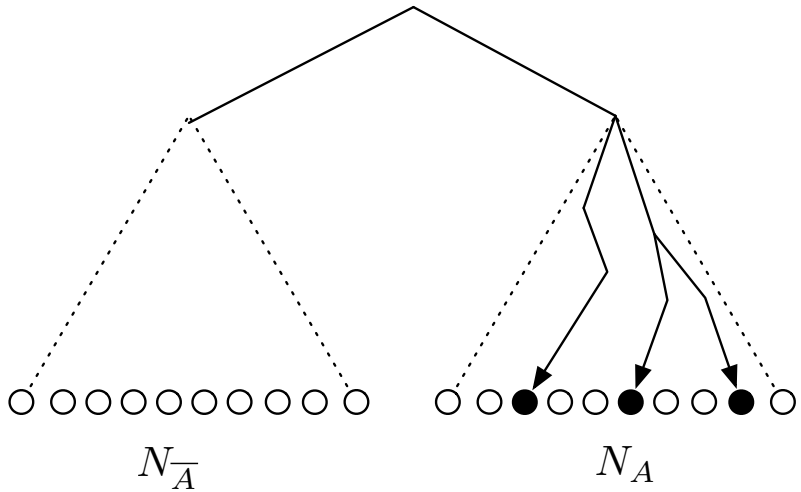


Figure: NPTM N'_A

Exactly one of N_A and $N_{\overline{A}}$ and must accept. The decision can be made in non-deterministic polynomial time.

$\mathrm{PH} = \mathrm{NP}$ (contd.)

- Building on this, we can show that $\mathrm{NP}^{\mathrm{NP} \cap \mathrm{coNP}} = \mathrm{NP}$
- And so, if $\mathrm{NP} = \mathrm{coNP}$, we have $\mathrm{NP}^{\mathrm{NP}} = \mathrm{NP}^{\mathrm{NP} \cap \mathrm{coNP}} = \mathrm{NP}$
- We can inductively reduce a stack of NPs of arbitrary height to NP .

For example,
$\mathrm{NP}^{\mathrm{NP}^{\mathrm{NP}^{\mathrm{NP}}}} = \mathrm{NP}^{\mathrm{NP}^{\mathrm{NP}}} = \mathrm{NP}^{\mathrm{NP}} = \mathrm{NP}$ Therefore, if $\mathrm{NP} = \mathrm{coNP}$, $\mathrm{PH} = \mathrm{NP}$.

Status

$$\mathrm{UP}=\mathrm{PP}=\mathrm{NP}=\mathrm{coNP}=\mathrm{PH}=\oplus\mathrm{P}=\mathrm{PP}\cup\mathrm{PP}^{\mathrm{PP}}\cup\mathrm{PP}^{\mathrm{PP}^{\mathrm{PP}}}\cup\ldots$$

$\mathrm{P}^{\mathrm{UP}} = \mathrm{UP}$

Proposition

If $PH = UP$, $P^{UP} = UP$

Proof.

Since $\mathrm{P}^{\mathrm{NP}} \subseteq \mathrm{PH}$ and $\mathrm{UP} \subseteq \mathrm{NP}$, we have $\mathrm{P}^{\mathrm{UP}} \subseteq \mathrm{PH}$.
So, $\mathrm{PH} = \mathrm{UP} \implies \mathrm{P}^{\mathrm{UP}} \subseteq \mathrm{PH} = \mathrm{UP}$
Clearly, $\mathrm{UP} \subseteq \mathrm{P}^{\mathrm{UP}}$.
Thus under our hypothesis, $\mathrm{P}^{\mathrm{UP}} = \mathrm{UP}$.

$\mathrm{PP}^{\oplus \mathrm{P}} \subseteq \mathrm{P}^{\mathrm{PP}}$

Here, we need to make use of Lemma 4.14 from the Hemaspaandra-Ogihara text. We state it below without proof.

Lemma

$PP^{\oplus P} \subseteq P^{PP}$

$\mathrm{UP} = \oplus\mathrm{P} = \mathrm{PP} = \mathrm{PP}^{\oplus\mathrm{P}}$

Proposition

If $UP = PP$ and $P^{UP} = UP$, then $UP = \oplus P = PP = PP^{\oplus P}$

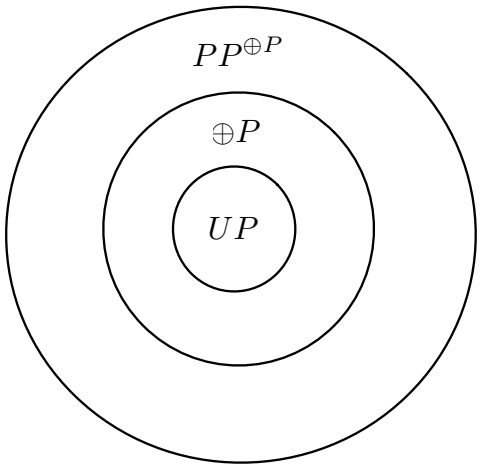


Figure: Known relationship between UP , PP , $\mathrm{PP}^{\oplus\mathrm{P}}$

Proof.

$\mathrm{P}^{\mathrm{PP}} = \mathrm{P}^{\mathrm{UP}} = \mathrm{UP}$.

From Lemma 4.14, $\mathrm{PP}^{\oplus\mathrm{P}} \subseteq \mathrm{P}^{\mathrm{PP}} = \mathrm{UP}$.

Clearly, $\mathrm{UP} \subseteq \oplus\mathrm{P} \subseteq \mathrm{PP}^{\oplus\mathrm{P}}$.

Therefore, $\mathrm{UP} = \oplus\mathrm{P} = \mathrm{PP}^{\oplus\mathrm{P}} = \mathrm{PP}$. □

Status

$$\mathrm{UP}=\mathrm{PP}=\mathrm{NP}=\mathrm{coNP}=\mathrm{PH}=\oplus\mathrm{P}=\mathrm{PP}\cup\mathrm{PP}^{\mathrm{PP}}\cup\mathrm{PP}^{\mathrm{PP}^{\mathrm{PP}}}\cup\ldots$$

$\mathrm{PP} \cup \mathrm{PP}^{\mathrm{PP}} \cup \mathrm{PP}^{\mathrm{PP}^{\mathrm{PP}}} \cup \ldots = \mathrm{PP}$

Assumptions

- $\oplus \mathrm{P} = \mathrm{PP}$
- $\mathrm{PP}^{\oplus \mathrm{P}} = \mathrm{PP}$

From the above assumptions we can write,
$\mathrm{PP}^{\mathrm{PP}} = \mathrm{PP}^{\oplus \mathrm{P}} = \mathrm{PP}$

We can inductively reduce a stack of PPs of arbitrary height to PP .

For example, $\mathrm{PP}^{\mathrm{PP}^{\mathrm{PP}^{\mathrm{PP}}}} = \mathrm{PP}^{\mathrm{PP}^{\mathrm{PP}}} = \mathrm{PP}^{\mathrm{PP}} = \mathrm{PP}$

Therefore, $\mathrm{PP} \cup \mathrm{PP}^{\mathrm{PP}} \cup \mathrm{PP}^{\mathrm{PP}^{\mathrm{PP}}} \cup \ldots = \mathrm{PP}$

Theorem 5.7 Proved

$$\mathrm{UP}{=}\mathrm{PP}{=}\mathrm{NP}{=}\mathrm{coNP}{=}\mathrm{PH}{=}\oplus\mathrm{P}{=}\mathrm{PP} \cup \mathrm{PP}^{\mathrm{PP}} \cup \mathrm{PP}^{\mathrm{PP}^{\mathrm{PP}}} \cup \ldots$$

Integer Division

Definition

Let $\mathcal{F}$ be a class of functions from $\mathbb{N}$ to $\mathbb{N}$. We say that $\mathcal{F}$ is closed under integer division ($\oslash$) if

$$(\forall f_1 \in \mathcal{F})(\forall f_2 \in \mathcal{F} : (\forall n)[f_2(n) > 0])[f_1 \oslash f2 \in \mathcal{F}],$$

where the 0 above is the integer zero (i.e., the integer represented by the empty string).

Theorem 5.9

Theorem

The following statements are equivalent:

1. $\#P$ *is closed under integer division.*
2. $\#P$ *is closed under every polynomial-time computable operation.*
3. $UP = PP$.

We will not prove 3 $\Rightarrow$ 2 since it was already proved in Theorem 5.6.

$2 \Rightarrow 1$

Assume $\#\mathrm{P}$ is closed under every polynomial-time computable operation

Show $\#\mathrm{P}$ is closed under Integer Division

Proof

This implication is trivial as integer division is a polynomial-time computable operation.

$1 \Rightarrow 3$

Assume $\#\mathrm{P}$ is closed under integer division

Show $\mathrm{UP} = \mathrm{PP}$

We know that $\mathrm{UP} \subseteq \mathrm{PP}$ without any assumption. Thus, we only prove $\mathrm{PP} \subseteq \mathrm{UP}$ given our assumption.

Let $L \in \mathrm{PP}$. There exists NPTM N and integer $k \geq 1$ such that,

1. $(\forall x), N(x)$ has exactly $2^{|x|^k}$ computation paths, each containing exactly $|x|^k$ choices
2. $x \in L \iff N(x)$ has at least $2^{|x|^k - 1}$ accepting paths
3. $(\forall x), N(x)$ has at least one rejecting path

Proof for 1 $\Rightarrow$ 3

- Let f be the $\#\mathrm{P}$ function for NPTM N which decides language $L \in \mathrm{PP}$.
- Define the $\#\mathrm{P}$ function g as, $g(x) = 2^{|x|^k - 1}$.

By our assumption, $h(x) = f(x) \oslash g(x)$ must be a $\#\mathrm{P}$ function.

- if $x \in L$, $\quad h(x) = \left\lfloor \frac{2^{|x|^k-1} \leq f(x) < 2^{|x|^k}}{2^{|x|^k-1}} \right\rfloor = 1$
- if $x \notin L$, $\quad h(x) = \left\lfloor \frac{0 \leq f(x) < 2^{|x|^k-1}}{2^{|x|^k-1}} \right\rfloor = 0$

The NPTM corresponding to h is a UP machine for L.
Hence $L \in UP$.

Intermediate Closure Properties

- If $\#\mathrm{P}$ is closed under proper subtraction and integer division, then $\#\mathrm{P}$ is also closed under all polynomial-time computable operations and $\mathrm{UP} = \mathrm{PP}$.
- Are there any operations that $\#\mathrm{P}$ is not know to be closed under, and does not have the property if $\#\mathrm{P}$ is closed under these operations if and only if $\#\mathrm{P}$ is closed under all polynomial-time computable operations.
- Analogy with sets that are in NP but are not known to be either NP-complete or in P.
- Examples of intermediate closure properties are taking minimums, maximums, proper decrement and integer division by 2.

Conclusions

- *We've shown that the following statements are equivalent:*
  1. $\#P$ *is closed under proper subtraction*
  2. $\#P$ *is closed under integer division.*
  3. $\#P$ *is closed under every polynomial-time computable operation.*
  4. $UP = PP$.
- We discussed the consequences of $\mathrm{UP} = \mathrm{PP}$