Complexity of Counting

Lecture 22

#P: Toda's Theorem

Last Time

Last Time

- #P: counting problems of the form $\#R(x) = |\{w: R(x,w)=1\}|$, where R is a polynomial time relation

Last Time

- #P: counting problems of the form $\#R(x) = |\{w: R(x,w)=1\}|$, where R is a polynomial time relation
  - Can be hard: even #CYCLE is not in FP (unless P = NP)

Last Time

- #P: counting problems of the form $\#R(x) = |\{w: R(x,w)=1\}|$, where R is a polynomial time relation
  - Can be hard: even #CYCLE is not in FP (unless P = NP)
  - $\#P \subseteq FP^{PP}$ (and $PP \subseteq P^{\#P}$)

Last Time

- #P: counting problems of the form #R(x) = |{w: R(x,w)=1}|, where R is a polynomial time relation
  - Can be hard: even #CYCLE is not in FP (unless P = NP)
  - $\#P \subseteq FP^{PP}$ (and $PP \subseteq P^{\#P}$)
- #P complete problems

Last Time

- #P: counting problems of the form #R(x) = |{w: R(x,w)=1}|, where R is a polynomial time relation
  - Can be hard: even #CYCLE is not in FP (unless P = NP)
  - $\#P \subseteq FP^{PP}$ (and $PP \subseteq P^{\#P}$)
- #P complete problems
  - #SAT

Last Time

- #P: counting problems of the form $\#R(x) = |\{w: R(x,w)=1\}|$, where R is a polynomial time relation
  - Can be hard: even #CYCLE is not in FP (unless P = NP)
  - $\#P \subseteq FP^{PP}$ (and $PP \subseteq P^{\#P}$)
- #P complete problems
  - #SAT
  - Permanent

Last Time

- #P: counting problems of the form $\#R(x) = |\{w: R(x,w)=1\}|$, where R is a polynomial time relation
  - Can be hard: even #CYCLE is not in FP (unless P = NP)
  - $\#P \subseteq FP^{PP}$ (and $PP \subseteq P^{\#P}$)
- #P complete problems
  - #SAT
  - Permanent
- Next: Toda's Theorem: $PH \subseteq P^{\#P} = P^{PP}$

$\oplus P$

$\oplus$P

- $\oplus$P: parity of the number of witnesses

⊕P

- ⊕P: parity of the number of witnesses
  - e.g. ⊕SAT. Least significant bit of #SAT.

⊕P

- ⊕P: parity of the number of witnesses
  - e.g. ⊕SAT. Least significant bit of #SAT.
  - May not be as powerful as PP (or #P)

$\oplus$P

- $\oplus$P: parity of the number of witnesses
  - e.g. $\oplus$SAT. Least significant bit of #SAT.
  - May not be as powerful as PP (or #P)
  - $\oplus P \subseteq P$ may not imply NP = P

$\oplus$P

- $\oplus$P: parity of the number of witnesses
  - e.g. $\oplus$SAT. Least significant bit of #SAT.
  - May not be as powerful as PP (or #P)
  - $\oplus P \subseteq P$ may not imply $NP = P$
  - But it does imply **$NP \subseteq RP$** (even if only $\oplus P \subseteq RP$)

$\oplus$P

- $\oplus$P: parity of the number of witnesses
  - e.g. $\oplus$SAT. Least significant bit of #SAT.
  - May not be as powerful as PP (or #P)
  - $\oplus P \subseteq P$ may not imply NP = P
  - But it does imply **NP $\subseteq$ RP** (even if only $\oplus P \subseteq RP$)
- Randomized reduction of NP to $\oplus$P

⊕P

- $\oplus$P: parity of the number of witnesses
  - e.g. $\oplus$SAT. Least significant bit of #SAT.
  - May not be as powerful as PP (or #P)
  - $\oplus P \subseteq P$ may not imply NP = P
  - But it does imply **$NP \subseteq RP$** (even if only $\oplus P \subseteq RP$)
- Randomized reduction of NP to $\oplus$P
  - i.e., $\oplus$P oracle is quite useful to randomized algorithms

$$\oplus P \subseteq RP \Rightarrow NP = RP$$

$\oplus P \subseteq RP \Rightarrow NP=RP$

- Randomized reduction of NP to $\oplus$P

$\oplus P \subseteq RP \Rightarrow NP=RP$

- Randomized reduction of NP to $\oplus$P
  - A probabilistic polynomial time algorithm A such that

$\oplus P \subseteq RP \Rightarrow NP=RP$

- Randomized reduction of NP to $\oplus$P
  - A probabilistic polynomial time algorithm A such that
    - $\varphi \notin SAT \Rightarrow Pr[A(\varphi) \in \oplus SAT] = 0$

$\oplus P \subseteq RP \Rightarrow NP{=}RP$

- Randomized reduction of NP to $\oplus P$
  - A probabilistic polynomial time algorithm A such that
    - $\varphi \notin SAT \Rightarrow \Pr[A(\varphi) \in \oplus SAT] = 0$
      - In fact $A(\varphi)$ will have no satisfying assignment

$\oplus P \subseteq RP \Rightarrow NP=RP$

- Randomized reduction of NP to $\oplus$P
  - A probabilistic polynomial time algorithm A such that
    - $\varphi \notin SAT \Rightarrow Pr[A(\varphi) \in \oplus SAT] = 0$
      - In fact $A(\varphi)$ will have no satisfying assignment
    - $\varphi \in SAT \Rightarrow Pr[A(\varphi) \in \oplus SAT] \geq \varepsilon(n)$

$\oplus P \subseteq RP \Rightarrow NP=RP$

- Randomized reduction of NP to $\oplus$P
  - A probabilistic polynomial time algorithm A such that
    - $\varphi \notin SAT \Rightarrow \Pr[A(\varphi) \in \oplus SAT] = 0$
      - In fact $A(\varphi)$ will have no satisfying assignment
    - $\varphi \in SAT \Rightarrow \Pr[A(\varphi) \in \oplus SAT] \geq \varepsilon(n)$
      - With prob. $\geq \varepsilon(n)$, $A(\varphi)$ will have exactly one satisfying assignment

$\oplus P \subseteq RP \Rightarrow NP=RP$

- Randomized reduction of NP to $\oplus$P
  - A probabilistic polynomial time algorithm A such that
    - $\varphi \notin SAT \Rightarrow \Pr[A(\varphi) \in \oplus SAT] = 0$
      - In fact $A(\varphi)$ will have no satisfying assignment
    - $\varphi \in SAT \Rightarrow \Pr[A(\varphi) \in \oplus SAT] \geq \varepsilon(n)$
      - With prob. $\geq \varepsilon(n)$, $A(\varphi)$ will have exactly one satisfying assignment
- If an RP algorithm for $\oplus$SAT, then an RP algorithm for SAT

$\oplus P \subseteq RP \Rightarrow NP=RP$

$\oplus P \subseteq RP \Rightarrow NP=RP$

- Randomized reduction of SAT to Unique-SAT: A probabilistic polynomial time algorithm A such that

$\oplus$P $\subseteq$ RP $\Rightarrow$ NP=RP

- Randomized reduction of SAT to Unique-SAT: A probabilistic polynomial time algorithm A such that
  - If $\varphi \in$ SAT, with prob. $\geq \varepsilon(n)$, A_φ will have exactly one satisfying assignment. Else A_φ will have none.

$\oplus P \subseteq RP \Rightarrow NP=RP$

- Randomized reduction of SAT to Unique-SAT: A probabilistic polynomial time algorithm A such that
  - If $\varphi \in$ SAT, with prob. $\geq \varepsilon(n)$, A_φ will have exactly one satisfying assignment. Else A_φ will have none.
  - Add a filter which will pass exactly one witness (if any): $A_\varphi(w)$ = $\varphi(w)$ and filter(w)

Hashing for unique preimage

Hashing for unique preimage

- Let $S \subseteq X$ be a set of size m. Consider a pair-wise independent hash-function family H, from X to R, such that $|S|/|R| \in [1/4,1/2]$.

Hashing for unique preimage

- Let $S \subseteq X$ be a set of size m. Consider a pair-wise independent hash-function family H, from X to R, such that $|S|/|R| \in [1/4,1/2]$.
  - $Pr_h[h(x)=0] = 1/|R| =: p$, and $Pr_h[h(x)=h(y)=0] = p^2$. $|S|p \in [1/4,1/2]$.

Hashing for unique preimage

- Let $S \subseteq X$ be a set of size m. Consider a pair-wise independent hash-function family H, from X to R, such that $|S|/|R| \in [1/4,1/2]$.
  - $Pr_h[h(x)=0] = 1/|R| =: p$, and $Pr_h[h(x)=h(y)=0] = p^2$. $|S|p \in [1/4,1/2]$.
  - Let $N := |\{x \in S | h(x)=0\}|$. $Pr_h[N=1] = Pr_h[N \geq 1] - Pr_h[N \geq 2]$

Hashing for unique preimage

- Let $S \subseteq X$ be a set of size m. Consider a pair-wise independent hash-function family H, from X to R, such that $|S|/|R| \in [1/4,1/2]$.
  - $\Pr_h[h(x)=0] = 1/|R| =: p$, and $\Pr_h[h(x)=h(y)=0] = p^2$. $|S|p \in [1/4,1/2]$.
  - Let $N := |\{x \in S | h(x)=0\}|$. $\Pr_h[N=1] = \Pr_h[N \geq 1] - \Pr_h[N \geq 2]$
- By inclusion-exclusion: $\Pr_h[N \geq 1] \geq \Sigma_x \Pr_h[h(x)=0] - \Sigma_{x>y} \Pr_h[h(x)=h(y)=0]$

Hashing for unique preimage

- Let $S \subseteq X$ be a set of size m. Consider a pair-wise independent hash-function family H, from X to R, such that $|S|/|R| \in [1/4,1/2]$.
  - $Pr_h[h(x)=0] = 1/|R| =: p$, and $Pr_h[h(x)=h(y)=0] = p^2$. $|S|p \in [1/4,1/2]$.
  - Let $N := |\{x \in S | h(x)=0\}|$. $Pr_h[N=1] = Pr_h[N \geq 1] - Pr_h[N \geq 2]$
- By inclusion-exclusion: $Pr_h[N \geq 1] \geq \underbrace{\Sigma_x Pr_h[h(x)=0]}_{|S|p} - \Sigma_{x>y} Pr_h[h(x)=h(y)=0]$

Hashing for unique preimage

- Let $S \subseteq X$ be a set of size m. Consider a pair-wise independent hash-function family H, from X to R, such that $|S|/|R| \in [1/4,1/2]$.
  - $Pr_h[h(x)=0] = 1/|R| =: p$, and $Pr_h[h(x)=h(y)=0] = p^2$. $|S|p \in [1/4,1/2]$.
  - Let $N := |\{x \in S | h(x)=0\}|$. $Pr_h[N=1] = Pr_h[N \geq 1] - Pr_h[N \geq 2]$
- By inclusion-exclusion: $Pr_h[N \geq 1] \geq \underbrace{\Sigma_x Pr_h[h(x)=0]}_{|S|p} - \underbrace{\Sigma_{x>y} Pr_h[h(x)=h(y)=0]}_{(|S| \text{ choose } 2)p}$

Hashing for unique preimage

- Let $S \subseteq X$ be a set of size m. Consider a pair-wise independent hash-function family H, from X to R, such that $|S|/|R| \in [1/4, 1/2]$.
  - $\Pr_h[h(x)=0] = 1/|R| =: p$, and $\Pr_h[h(x)=h(y)=0] = p^2$. $|S|p \in [1/4, 1/2]$.
  - Let $N := |\{x \in S | h(x)=0\}|$. $\Pr_h[N=1] = \Pr_h[N \geq 1] - \Pr_h[N \geq 2]$
- By inclusion-exclusion: $\Pr_h[N \geq 1] \geq \underbrace{\Sigma_x \Pr_h[h(x)=0]}_{|S|p} - \underbrace{\Sigma_{x>y} \Pr_h[h(x)=h(y)=0]}_{(|S| \text{ choose } 2)p}$
- By Union-bound: $\Pr_h[N \geq 2] \leq \Sigma_{x>y} \Pr_h[h(x)=h(y)=0]$

Hashing for unique preimage

- Let $S \subseteq X$ be a set of size m. Consider a pair-wise independent hash-function family H, from X to R, such that $|S|/|R| \in [1/4,1/2]$.
  - $Pr_h[h(x)=0] = 1/|R| =: p$, and $Pr_h[h(x)=h(y)=0] = p^2$. $|S|p \in [1/4,1/2]$.
  - Let $N := |\{x \in S | h(x)=0\}|$. $Pr_h[N=1] = Pr_h[N \geq 1] - Pr_h[N \geq 2]$
- By inclusion-exclusion: $Pr_h[N \geq 1] \geq \underbrace{\Sigma_x Pr_h[h(x)=0]}_{|S|p} - \underbrace{\Sigma_{x>y} Pr_h[h(x)=h(y)=0]}_{(|S| \text{ choose } 2)p}$
- By Union-bound: $Pr_h[N \geq 2] \leq \Sigma_{x>y} Pr_h[h(x)=h(y)=0] \rightarrow (|S| \text{ choose } 2)p$

Hashing for unique preimage

- Let $S \subseteq X$ be a set of size m. Consider a pair-wise independent hash-function family H, from X to R, such that $|S|/|R| \in [1/4,1/2]$.
  - $Pr_h[h(x)=0] = 1/|R| =: p$, and $Pr_h[h(x)=h(y)=0] = p^2$. $|S|p \in [1/4,1/2]$.
  - Let $N := |\{x \in S | h(x)=0\}|$. $Pr_h[N=1] = Pr_h[N \geq 1] - Pr_h[N \geq 2]$
- By inclusion-exclusion: $Pr_h[N \geq 1] \geq \underbrace{\Sigma_x Pr_h[h(x)=0]}_{|S|p} - \underbrace{\Sigma_{x>y} Pr_h[h(x)=h(y)=0]}_{(|S| \text{ choose } 2)p}$
- By Union-bound: $Pr_h[N \geq 2] \leq \Sigma_{x>y} Pr_h[h(x)=h(y)=0] \rightarrow (|S| \text{ choose } 2)p$
  - $Pr_h[N=1] \geq |S|\, p - 2\,(|S| \text{ choose } 2)\, p^2 \geq |S|p - (|S|p)^2 \geq 3/16$

$\oplus P \subseteq RP \Rightarrow NP{=}RP$

- Randomized reduction of SAT to Unique-SAT: A probabilistic polynomial time algorithm A such that
  - If $\varphi \in$ SAT, with prob. $\geq \varepsilon(n)$, A_φ will have exactly one satisfying assignment. Else A_φ will have none.
  - Add a filter which will pass exactly one witness (if any): $A_\varphi(w)$ = $\varphi(w)$ and filter(w)

$\oplus P \subseteq RP \Rightarrow NP=RP$

- Randomized reduction of SAT to Unique-SAT: A probabilistic polynomial time algorithm A such that
  - If $\varphi \in$ SAT, with prob. $\geq \varepsilon(n)$, A_φ will have exactly one satisfying assignment. Else A_φ will have none.
  - Add a filter which will pass exactly one witness (if any): $A_\varphi(w)$ = $\varphi(w)$ and filter(w)
    - filter(w): a Boolean formula saying h(w)=0. (If using auxiliary variables, i.e., $\exists z$ filter(w,z), use a parsimonious reduction.)

$\oplus P \subseteq RP \Rightarrow NP=RP$

- Randomized reduction of SAT to Unique-SAT: A probabilistic polynomial time algorithm A such that
  - If $\varphi \in SAT$, with prob. $\geq \varepsilon(n)$, A_φ will have exactly one satisfying assignment. Else A_φ will have none.
  - Add a filter which will pass exactly one witness (if any): $A_\varphi(w)$ = $\varphi(w)$ and filter(w)
    - filter(w): a Boolean formula saying h(w)=0. (If using auxiliary variables, i.e., $\exists z$ filter(w,z), use a parsimonious reduction.)
    - If witness n-bit long ($|X|=\{0,1\}^n$), pick $R=\{0,1\}^k$, with k random in the range [1,n]

Reducing PH to $P^{\#P}$

Reducing PH to $P^{\#P}$

- Two steps

Reducing PH to $P^{\#P}$

- Two steps
  - Randomized reduction of PH to $P^{\oplus P}$

Reducing PH to $P^{\#P}$

- Two steps
  - Randomized reduction of PH to $P^{\oplus P}$
  - Converting the probabilistic guarantee to a deterministic #P statement

Quantifier Gallery!

Quantifier Gallery!

$\exists$

For at least one

Quantifier Gallery!

$\exists$
For at least one

$\forall$
For all

Quantifier Gallery!

$\exists$

For at least one

$\forall$

For all

$\exists_r$

For at least r fraction

Quantifier Gallery!

$\exists$

For at least one

$\forall$

For all

$\exists_r$

For at least r fraction

$\exists!$

For exactly one

Quantifier Gallery!

$\exists$
For at least one

$\forall$
For all

$\exists_r$
For at least r fraction

$\exists!$
For exactly one

$\oplus$
For an odd number of

QBF to $\oplus$BF

QBF to $\oplus$BF

- We have a randomized reduction: φ to A_φ such that

QBF to $\oplus$BF

- We have a randomized reduction: φ to A_φ such that
  - $\underline{\exists_w\ \varphi(w)} \Rightarrow \underline{\oplus_w\ A_\varphi(w)}$ with prob. $\geq \varepsilon(n)$

QBF to ⊕BF

- We have a randomized reduction: φ to A_φ such that
  - $\underline{\exists_w\ \varphi(w)} \Rightarrow \underline{\oplus_w\ A_\varphi(w)}$ with prob. $\geq \varepsilon(n)$
  - $\underline{\forall_w \text{ not } \varphi(w)} \Rightarrow$ not $\underline{\oplus_w\ A_\varphi(w)}$ (with prob. = 1)

QBF to $\oplus$BF

- We have a randomized reduction: φ to A_φ such that
  - $\underline{\exists_w\ \varphi(w)} \Rightarrow \underline{\oplus_w\ A_\varphi(w)}$ with prob. $\geq \varepsilon(n)$
  - $\underline{\forall_w \text{ not } \varphi(w)} \Rightarrow \text{not } \underline{\oplus_w\ A_\varphi(w)}$ (with prob. = 1)
  - i.e., with prob $\geq \varepsilon(n)$, we have $\underline{\exists_w\ \varphi(w)} \Leftrightarrow \underline{\oplus_w\ A_\varphi(w)}$ (and hence also $\underline{\forall_w \text{ not } \varphi(w)} \Leftrightarrow \text{not } \underline{\oplus_w\ A_\varphi(w)}$)

QBF to $\oplus$BF

- We have a randomized reduction: φ to A_φ such that
  - $\exists_w\ \varphi(w) \Rightarrow \oplus_w\ A_\varphi(w)$ with prob. $\geq \varepsilon(n)$
  - $\forall_w$ not $\varphi(w) \Rightarrow$ not $\oplus_w\ A_\varphi(w)$ (with prob. = 1)
  - i.e., with prob $\geq \varepsilon(n)$, we have $\exists_w\ \varphi(w) \Leftrightarrow \oplus_w\ A_\varphi(w)$ (and hence also $\forall_w$ not $\varphi(w) \Leftrightarrow$ not $\oplus_w\ A_\varphi(w)$)
- Reduction works even if $\varphi(w)$ is a partially quantified Boolean formula

QBF to $\oplus$BF

- We have a randomized reduction: φ to A_φ such that
  - $\underline{\exists_w \varphi(w)} \Rightarrow \underline{\oplus_w A_\varphi(w)}$ with prob. $\geq \varepsilon(n)$
  - $\underline{\forall_w \text{ not } \varphi(w)} \Rightarrow$ not $\underline{\oplus_w A_\varphi(w)}$ (with prob. = 1)
  - i.e., with prob $\geq \varepsilon(n)$, we have $\underline{\exists_w \varphi(w)} \Leftrightarrow \underline{\oplus_w A_\varphi(w)}$ (and hence also $\underline{\forall_w \text{ not } \varphi(w)} \Leftrightarrow \text{not } \underline{\oplus_w A_\varphi(w)}$)
- Reduction works even if $\varphi(w)$ is a partially quantified Boolean formula
  - Can all $\exists/\forall$ be removed, by repeating, so that only $\oplus$ remain?

Some # arithmetic

Some # arithmetic

- Given two boolean formulas $\varphi(x)$ and $\psi(y)$, define

Some # arithmetic

- Given two boolean formulas $\varphi(x)$ and $\psi(y)$, define
    - $F_{\varphi.\psi}(x,y)$: $\varphi(x)$ and $\psi(y)$

Some # arithmetic

- Given two boolean formulas $\varphi(x)$ and $\psi(y)$, define
  - $F_{\varphi.\psi}(x,y)$: $\varphi(x)$ and $\psi(y)$
    - $\#F_{\varphi.\psi} = \#\varphi \, . \, \#\psi$

Some # arithmetic

- Given two boolean formulas $\varphi(x)$ and $\psi(y)$, define
  - $F_{\varphi.\psi}(x,y)$: $\varphi(x)$ and $\psi(y)$
    - $\#F_{\varphi.\psi} = \#\varphi \,.\, \#\psi$
  - $F_{\varphi+\psi}(x,y,z)$: ($z=0,y=0$ and $\varphi(x)$) or ($z=1,x=0$ and $\psi(y)$)

Some # arithmetic

- Given two boolean formulas $\varphi(x)$ and $\psi(y)$, define
  - $F_{\varphi.\psi}(x,y)$: $\varphi(x)$ and $\psi(y)$
    - $\#F_{\varphi.\psi} = \#\varphi \,.\, \#\psi$
  - $F_{\varphi+\psi}(x,y,z)$: ($z=0,y=0$ and $\varphi(x)$) or ($z=1,x=0$ and $\psi(y)$)
    - $\#F_{\varphi+\psi} = \#\varphi + \#\psi$

Some # arithmetic

- Given two boolean formulas $\varphi(x)$ and $\psi(y)$, define
  - $F_{\varphi.\psi}(x,y)$: $\varphi(x)$ and $\psi(y)$
    - $\#F_{\varphi.\psi} = \#\varphi \,.\, \#\psi$
  - $F_{\varphi+\psi}(x,y,z)$: $(z=0, y=0$ and $\varphi(x))$ or $(z=1, x=0$ and $\psi(y))$
    - $\#F_{\varphi+\psi} = \#\varphi + \#\psi$
    - $F_{\varphi+1} :=$ $(z=0$ and $\varphi(x))$ or $(z=1$ and $x=0)$. $\#F_{\varphi+1} = \#\varphi + 1$

Some # arithmetic

- Given two boolean formulas $\varphi(x)$ and $\psi(y)$, define
  - $F_{\varphi.\psi}(x,y)$: $\varphi(x)$ and $\psi(y)$
    - $\#F_{\varphi.\psi} = \#\varphi \, . \, \#\psi$
  - $F_{\varphi+\psi}(x,y,z)$: ($z=0, y=0$ and $\varphi(x)$) or ($z=1, x=0$ and $\psi(y)$)
    - $\#F_{\varphi+\psi} = \#\varphi + \#\psi$
    - $F_{\varphi+1}$:= ($z=0$ and $\varphi(x)$) or ($z=1$ and $x=0$). $\#F_{\varphi+1} = \#\varphi + 1$
  - Works even if φ, ψ are partially quantified boolean formulas

Some $\oplus$ arithmetic

Some ⊕ arithmetic

- Boolean combinations of QBFs with ⊕ quantifiers

Some ⊕ arithmetic

- Boolean combinations of QBFs with ⊕ quantifiers
  - $\underline{\oplus_x\ \varphi(x)}$ and $\underline{\oplus_y\ \psi(y)}$ $\Leftrightarrow$ $\underline{\oplus_{x,y}\ F_{\varphi.\psi}(x,y)}$, i.e. $\underline{\oplus_{x,y}\ \varphi(x) \text{ and } \psi(y)}$

Some ⊕ arithmetic

- Boolean combinations of QBFs with ⊕ quantifiers
  - $\oplus_x \varphi(x)$ and $\oplus_y \psi(y) \Leftrightarrow \oplus_{x,y} F_{\varphi.\psi}(x,y)$, i.e. $\oplus_{x,y} \varphi(x)$ and $\psi(y)$
  - not $\oplus_x \varphi(x) \Leftrightarrow \oplus_{x,z} F_{\varphi+1}(x,z)$. i.e. $\oplus_{x,z} (z=1,x=0)$ or $(z=0,\varphi(x))$

Some ⊕ arithmetic

- Boolean combinations of QBFs with ⊕ quantifiers
  - $\oplus_x\ \varphi(x)$ and $\oplus_y\ \psi(y) \Leftrightarrow \oplus_{x,y}\ F_{\varphi.\psi}(x,y)$, i.e. $\oplus_{x,y}\ \varphi(x)$ and $\psi(y)$
  - not $\oplus_x\ \varphi(x) \Leftrightarrow \oplus_{x,z}\ F_{\varphi+1}(x,z)$. i.e. $\oplus_{x,z}\ (z{=}1,x{=}0)$ or $(z{=}0,\varphi(x))$
- $\oplus_x\ (\oplus_y\ \varphi(x,y)) \Leftrightarrow \oplus_{x,y}\ \varphi(x,y)$

Some $\oplus$ arithmetic

- Boolean combinations of QBFs with $\oplus$ quantifiers
  - $\oplus_x\ \varphi(x)$ and $\oplus_y\ \psi(y) \Leftrightarrow \oplus_{x,y}\ F_{\varphi.\psi}(x,y)$, i.e. $\oplus_{x,y}\ \varphi(x)$ and $\psi(y)$
  - not $\oplus_x\ \varphi(x) \Leftrightarrow \oplus_{x,z}\ F_{\varphi+1}(x,z)$. i.e. $\oplus_{x,z}\ (z=1,x=0)$ or $(z=0,\varphi(x))$
- $\oplus_x\ (\oplus_y\ \varphi(x,y)) \Leftrightarrow \oplus_{x,y}\ \varphi(x,y)$
- $(\oplus,\exists,\forall)$-QBF can be converted to the form $\oplus_z\ F(z)$, where F is a $(\exists,\forall)$-QBF, increasing the size by at most a constant factor, and not changing number of $\exists,\forall$

QBF to $\oplus$BF

QBF to $\oplus$BF

- Recall: with prob $\geq \varepsilon(n)$, we have $\exists_w\ \varphi(w) \Leftrightarrow \oplus_w\ A_\varphi(w)$ (and $\forall_w$ not $\varphi(w) \Leftrightarrow$ not $\oplus_w\ A_\varphi(w)$)

QBF to $\oplus$BF

- Recall: with prob $\geq \varepsilon(n)$, we have $\exists_w\ \varphi(w) \Leftrightarrow \oplus_w\ A_\varphi(w)$ (and $\forall_w$ not $\varphi(w) \Leftrightarrow$ not $\oplus_w\ A_\varphi(w)$)

- Boosting the probability: $\varepsilon(n)$ to $1-\delta(n)$

QBF to ⊕BF

- Recall: with prob $\geq \varepsilon(n)$, we have $\exists_w\ \varphi(w) \Leftrightarrow \oplus_w\ A_\varphi(w)$ (and $\forall_w$ not $\varphi(w) \Leftrightarrow$ not $\oplus_w\ A_\varphi(w)$)
- Boosting the probability: $\varepsilon(n)$ to $1-\delta(n)$
  - $\oplus_w\ A^1{}_\varphi(w)$ or $\oplus_w\ A^2{}_\varphi(w)$ or ... or $\oplus_w\ A^t{}_\varphi(w)$

QBF to $\oplus$BF

- Recall: with prob $\geq \varepsilon(n)$, we have $\exists_w\ \varphi(w) \Leftrightarrow \oplus_w\ A_\varphi(w)$ (and $\forall_w$ not $\varphi(w) \Leftrightarrow$ not $\oplus_w\ A_\varphi(w)$)
- Boosting the probability: $\varepsilon(n)$ to $1-\delta(n)$
  - $\oplus_w\ A^1_\varphi(w)$ or $\oplus_w\ A^2_\varphi(w)$ or ... or $\oplus_w\ A^t_\varphi(w)$
    - Can rewrite in the form $\oplus_z\ B_\varphi(z)$ where B_φ has no $\oplus$

QBF to $\oplus$BF

- Recall: with prob $\geq \varepsilon(n)$, we have $\exists_w\ \varphi(w) \Leftrightarrow \oplus_w\ A_\varphi(w)$ (and $\forall_w$ not $\varphi(w) \Leftrightarrow$ not $\oplus_w\ A_\varphi(w)$)
- Boosting the probability: $\varepsilon(n)$ to $1-\delta(n)$
  - $\oplus_w\ A^1_\varphi(w)$ or $\oplus_w\ A^2_\varphi(w)$ or ... or $\oplus_w\ A^t_\varphi(w)$
    - Can rewrite in the form $\oplus_z\ B_\varphi(z)$ where B_φ has no $\oplus$
    - In prenex form $\oplus_z\ B_\varphi(z)$ has one less $\exists/\forall$ than $\exists_w\ \varphi(w)$

QBF to $\oplus$BF

- Recall: with prob $\geq \varepsilon(n)$, we have $\exists_w\ \varphi(w) \Leftrightarrow \oplus_w A_\varphi(w)$ (and $\forall_w$ not $\varphi(w) \Leftrightarrow$ not $\oplus_w A_\varphi(w)$)

- Boosting the probability: $\varepsilon(n)$ to $1-\delta(n)$
  - $\oplus_w A^1_\varphi(w)$ or $\oplus_w A^2_\varphi(w)$ or ... or $\oplus_w A^t_\varphi(w)$
    - Can rewrite in the form $\oplus_z B_\varphi(z)$ where B_φ has no $\oplus$
    - In prenex form $\oplus_z B_\varphi(z)$ has one less $\exists/\forall$ than $\exists_w\ \varphi(w)$
- If we start from $\oplus_x \exists_w\ \varphi(w,x)$ we get equivalent (with probability $1-\delta(n)$) $\oplus_x \oplus_z B_\varphi(z,x)$

QBF to $\oplus$BF

- Recall: with prob $\geq \varepsilon(n)$, we have $\exists_w\ \varphi(w) \Leftrightarrow \oplus_w\ A_\varphi(w)$ (and $\forall_w$ not $\varphi(w) \Leftrightarrow$ not $\oplus_w\ A_\varphi(w)$)

- Boosting the probability: $\varepsilon(n)$ to $1-\delta(n)$
  - $\oplus_w\ A^1{}_\varphi(w)$ or $\oplus_w\ A^2{}_\varphi(w)$ or ... or $\oplus_w\ A^t{}_\varphi(w)$
    - Can rewrite in the form $\oplus_z\ B_\varphi(z)$ where B_φ has no $\oplus$
    - In prenex form $\oplus_z\ B_\varphi(z)$ has one less $\exists/\forall$ than $\exists_w\ \varphi(w)$
- If we start from $\oplus_x\exists_w\ \varphi(w,x)$ we get equivalent (with probability $1-\delta(n)$) $\oplus_x\oplus_z\ B_\varphi(z,x)$
    - By repeating, QBF can be converted to the form $\oplus_z\ F(z)$ where F is unquantified, equivalent with prob. close to 1

Reducing PH to $P^{\#P}$

Reducing PH to $P^{\#P}$

- Two steps

Reducing PH to $P^{\#P}$

- Two steps
  - Randomized reduction of PH to $P^{\oplus P}$

Reducing PH to $P^{\#P}$

- Two steps
  - Randomized reduction of PH to $P^{\oplus P}$
    - TQBF instance ψ to $\oplus$SAT instance φ_ψ

Reducing PH to $P^{\#P}$

- Two steps
  - Randomized reduction of PH to $P^{\oplus P}$
    - TQBF instance ψ to $\oplus$SAT instance φ_ψ
    - $\psi \Rightarrow \oplus\varphi_\psi$ w.p. > 2/3; $\neg\psi \Rightarrow \neg\oplus\varphi_\psi$ (w.p. 1)

Reducing PH to $P^{\#P}$

- Two steps
  - Randomized reduction of PH to $P^{\oplus P}$
    - TQBF instance ψ to $\oplus$SAT instance φ_ψ
    - $\psi \Rightarrow \oplus\varphi_\psi$ w.p. > 2/3; $\neg\psi \Rightarrow \neg\oplus\varphi_\psi$ (w.p. 1)
  - Converting the probabilistic guarantee to a deterministic #P calculation

Reducing PH to $P^{\#P}$

- Two steps
  - Randomized reduction of PH to $P^{\oplus P}$
    - TQBF instance ψ to $\oplus$SAT instance φ_ψ
    - $\psi \Rightarrow \oplus\varphi_\psi$ w.p. > 2/3; $\neg\psi \Rightarrow \neg\oplus\varphi_\psi$ (w.p. 1)
  - Converting the probabilistic guarantee to a deterministic #P calculation
    - ψ s.t. $\neg\oplus\varphi_\psi \Rightarrow \#\theta_\psi = 0 \pmod N$

Reducing PH to $P^{\#P}$

- Two steps
  - Randomized reduction of PH to $P^{\oplus P}$
    - TQBF instance ψ to $\oplus$SAT instance φ_ψ
    - $\psi \Rightarrow \oplus\varphi_\psi$ w.p. > 2/3; $\neg\psi \Rightarrow \neg\oplus\varphi_\psi$ (w.p. 1)
  - Converting the probabilistic guarantee to a deterministic #P calculation
    - ψ s.t. $\neg\oplus\varphi_\psi \Rightarrow \#\theta_\psi = 0 \pmod{N}$
    - ψ s.t. $\oplus\varphi_\psi$ w.p. > 2/3 $\Rightarrow \#\theta_\psi \neq 0 \pmod{N}$

Reduction to #P

Reduction to #P

- Converting the probabilistic guarantee to a deterministic #P calculation

Reduction to #P

- Converting the probabilistic guarantee to a deterministic #P calculation
  - ψ s.t. $\neg\oplus\varphi_\psi \Rightarrow \#\theta_\psi = 0$ (mod N)

Reduction to #P

- Converting the probabilistic guarantee to a deterministic #P calculation
  - ψ s.t. $\neg\oplus\varphi_\psi \Rightarrow \#\theta_\psi = 0$ (mod N)
  - ψ s.t. $\oplus\varphi_\psi$ w.p. > 2/3 $\Rightarrow \#\theta_\psi \neq 0$ (mod N)

Reduction to #P

- Converting the probabilistic guarantee to a deterministic #P calculation
  - ψ s.t. $\neg\oplus\varphi_\psi \Rightarrow \#\theta_\psi = 0$ (mod N)
  - ψ s.t. $\oplus\varphi_\psi$ w.p. > 2/3 $\Rightarrow \#\theta_\psi \neq 0$ (mod N)
- Attempt 1: let $\varphi_\psi{}^r$ be the formula generated using random tape r. To determine if ψ is such that number of random tapes r for which $\oplus\varphi_\psi{}^r$ holds is 0 or > $(2/3)2^m$

Reduction to #P

- Converting the probabilistic guarantee to a deterministic #P calculation
  - ψ s.t. $\neg\oplus\varphi_\psi \Rightarrow \#\theta_\psi = 0$ (mod N)
  - ψ s.t. $\oplus\varphi_\psi$ w.p. > 2/3 $\Rightarrow \#\theta_\psi \neq 0$ (mod N)
- Attempt 1: let $\varphi_\psi{}^r$ be the formula generated using random tape r. To determine if ψ is such that number of random tapes r for which $\oplus\varphi_\psi{}^r$ holds is 0 or > $(2/3)2^m$
  - Enough to compute $\#_r \oplus\varphi_\psi{}^r$

Reduction to #P

- Converting the probabilistic guarantee to a deterministic #P calculation
  - ψ s.t. $\neg\oplus\varphi_\psi \Rightarrow \#\theta_\psi = 0$ (mod N)
  - ψ s.t. $\oplus\varphi_\psi$ w.p. > 2/3 $\Rightarrow \#\theta_\psi \neq 0$ (mod N)
- Attempt 1: let φ_ψ^r be the formula generated using random tape r. To determine if ψ is such that number of random tapes r for which $\oplus\varphi_\psi^r$ holds is 0 or > $(2/3)2^m$
  - Enough to compute $\#_r \oplus\varphi_\psi^r$
  - But $\oplus\varphi_\psi^r$ may not be in P (though $\varphi_\psi^r(x)$ is in P)

Reduction to #P

Reduction to #P

- Attempt 2: If $\oplus_x \varphi_\psi{}^r = \#_x \varphi_\psi{}^r$ then enough to compute the number of (x,r) such that $\varphi_\psi{}^r(x)$

Reduction to #P

- Attempt 2: If $\oplus_x \varphi_\psi^r = \#_x \varphi_\psi^r$ then enough to compute the number of (x,r) such that $\varphi_\psi^r(x)$
    - But $\oplus\varphi$ is $\#\varphi$ mod 2

Reduction to #P

- Attempt 2: If $\oplus_x \varphi_\psi{}^r = \#_x \varphi_\psi{}^r$ then enough to compute the number of (x,r) such that $\varphi_\psi{}^r(x)$
  - But $\oplus\varphi$ is $\#\varphi$ mod 2
- Plan: Create $\varphi' = T(\varphi)$, such that

Reduction to #P

- Attempt 2: If $\oplus_x \varphi_\psi{}^r = \#_x \varphi_\psi{}^r$ then enough to compute the number of (x,r) such that $\varphi_\psi{}^r(x)$
    - But $\oplus\varphi$ is $\#\varphi$ mod 2
- Plan: Create $\varphi' = T(\varphi)$, such that
    - For each r, $\neg\oplus_x\varphi \Rightarrow \#_x\varphi' = 0 \bmod N$

Reduction to #P

- Attempt 2: If $\oplus_x \varphi_\psi^r = \#_x \varphi_\psi^r$ then enough to compute the number of (x,r) such that $\varphi_\psi^r(x)$
  - But $\oplus\varphi$ is $\#\varphi$ mod 2
- Plan: Create $\varphi' = T(\varphi)$, such that
  - For each r, $\neg\oplus_x\varphi \Rightarrow \#_x\varphi' = 0 \bmod N$
  - For each r, $\oplus_x\varphi \Rightarrow \#_x\varphi' = -1 \bmod N$

Reduction to #P

- Attempt 2: If $\oplus_x \varphi_\psi{}^r = \#_x \varphi_\psi{}^r$ then enough to compute the number of (x,r) such that $\varphi_\psi{}^r(x)$
  - But $\oplus\varphi$ is $\#\varphi$ mod 2
- Plan: Create $\varphi' = T(\varphi)$, such that
  - For each r, $\neg\oplus_x\varphi \Rightarrow \#_x\varphi' = 0 \bmod N$
  - For each r, $\oplus_x\varphi \Rightarrow \#_x\varphi' = -1 \bmod N$
  - $N > 2^m$ so that for $(2/3).2^m < R \leq 2^m$ we have $R.(-1) \neq 0 \bmod N$

Reduction to #P

- Attempt 2: If $\oplus_x \varphi_\psi^r = \#_x \varphi_\psi^r$ then enough to compute the number of (x,r) such that $\varphi_\psi^r(x)$
  - But $\oplus\varphi$ is $\#\varphi$ mod 2
- Plan: Create $\varphi' = T(\varphi)$, such that
  - For each r, $\neg\oplus_x\varphi \Rightarrow \#_x\varphi' = 0 \bmod N$
  - For each r, $\oplus_x\varphi \Rightarrow \#_x\varphi' = -1 \bmod N$
  - $N > 2^m$ so that for $(2/3).2^m < R \leq 2^m$ we have $R.(-1) \neq 0 \bmod N$
  - Let $\theta_\psi(x,r) = T(\varphi_\psi^r)(x)$. Use $\#\theta_\psi$ mod N to check if w.h.p. $\oplus\varphi$

Reduction to #P

Reduction to #P

- Remains to do: Given φ, create φ' such that for $N=2^{2^k}$, where $k = O(\log m)$

Reduction to #P

- Remains to do: Given φ, create φ' such that for $N=2^{2^{\wedge}k}$, where $k = O(\log m)$
  - $\neg\oplus\varphi \Rightarrow \#\varphi' = 0 \bmod N$

Reduction to #P

- Remains to do: Given φ, create φ' such that for $N=2^{2^k}$, where $k = O(\log m)$
  - $\neg\oplus\varphi \Rightarrow \#\varphi' = 0 \bmod N$
  - $\oplus\varphi \Rightarrow \#\varphi' = -1 \bmod N$

Reduction to #P

- Remains to do: Given φ, create φ' such that for $N=2^{2^k}$, where $k = O(\log m)$
  - $\neg\oplus\varphi \Rightarrow \#\varphi' = 0 \bmod N$
  - $\oplus\varphi \Rightarrow \#\varphi' = -1 \bmod N$
- Initially true for $N = 2$ (2^{2^i}, $i=0$)

Reduction to #P

- Remains to do: Given φ, create φ' such that for $N=2^{2\hat{}k}$, where $k = O(\log m)$
  - $\neg\oplus\varphi \Rightarrow \#\varphi' = 0 \bmod N$
  - $\oplus\varphi \Rightarrow \#\varphi' = -1 \bmod N$
- Initially true for $N = 2$ ($2^{2\hat{}i}$, $i=0$)
  - $\varphi_{i+1} = F_{4(\varphi i)\hat{}3\ +\ 3(\varphi i)\hat{}4}$ so that $\#\varphi_{i+1} = 4(\#\varphi_i)^3 + 3(\#\varphi_i)^4$

Reduction to #P

- Remains to do: Given φ, create φ' such that for $N=2^{2\hat{}k}$, where $k = O(\log m)$
  - $\neg\oplus\varphi \Rightarrow \#\varphi' = 0 \bmod N$
  - $\oplus\varphi \Rightarrow \#\varphi' = -1 \bmod N$
- Initially true for $N = 2$ ($2^{2\hat{}i}$, $i=0$)
  - $\varphi_{i+1} = F_{4(\varphi i)\hat{}3\, +\, 3(\varphi i)\hat{}4}$ so that $\#\varphi_{i+1} = 4(\#\varphi_i)^3 + 3(\#\varphi_i)^4$
    - $\#\varphi_i = -1 \bmod 2^{2\hat{}i}$ implies $\varphi_{i+1} = -1 \bmod 2^{2\hat{}(i+1)}$ (for $i\geq 0$)

Reduction to #P

- Remains to do: Given φ, create φ' such that for $N=2^{2^k}$, where $k = O(\log m)$
  - $\neg\oplus\varphi \Rightarrow \#\varphi' = 0 \bmod N$
  - $\oplus\varphi \Rightarrow \#\varphi' = -1 \bmod N$
- Initially true for $N = 2$ (2^{2^i}, $i=0$)
  - $\varphi_{i+1} = F_{4(\varphi i)^3 + 3(\varphi i)^4}$ so that $\#\varphi_{i+1} = 4(\#\varphi_i)^3 + 3(\#\varphi_i)^4$
    - $\#\varphi_i = -1 \bmod 2^{2^i}$ implies $\varphi_{i+1} = -1 \bmod 2^{2^{(i+1)}}$ (for $i \geq 0$)
    - Clearly $\#\varphi_i = 0 \bmod 2^{2^i}$ implies $\varphi_{i+1} = 0 \bmod 2^{2^{(i+1)}}$

$PH \subseteq P^{\#P}$

$PH \subseteq P^{\#P}$

- Summary:

$PH \subseteq P^{\#P}$

- Summary:
  - First, randomized reduction of PH to $P^{\oplus P}$

$PH \subseteq P^{\#P}$

- Summary:
  - First, randomized reduction of PH to $P^{\oplus P}$
    - TQBF instance ψ to $\oplus$SAT instance φ_ψ

$PH \subseteq P^{\#P}$

- Summary:
  - First, randomized reduction of PH to $P^{\oplus P}$
    - TQBF instance ψ to $\oplus$SAT instance φ_ψ
    - $\psi \Rightarrow \oplus\varphi_\psi$ w.p. > 2/3; $\neg\psi \Rightarrow \neg\oplus\varphi_\psi$ (w.p. 1)

$PH \subseteq P^{\#P}$

- Summary:
  - First, randomized reduction of PH to $P^{\oplus P}$
    - TQBF instance ψ to $\oplus$SAT instance φ_ψ
    - $\psi \Rightarrow \oplus\varphi_\psi$ w.p. > 2/3; $\neg\psi \Rightarrow \neg\oplus\varphi_\psi$ (w.p. 1)
  - Converting the probabilistic guarantee to a deterministic #P calculation

$PH \subseteq P^{\#P}$

- Summary:
  - First, randomized reduction of PH to $P^{\oplus P}$
    - TQBF instance ψ to $\oplus$SAT instance φ_ψ
    - $\psi \Rightarrow \oplus\varphi_\psi$ w.p. > 2/3; $\neg\psi \Rightarrow \neg\oplus\varphi_\psi$ (w.p. 1)
  - Converting the probabilistic guarantee to a deterministic #P calculation
    - ψ s.t. $\neg\oplus\varphi_\psi \Rightarrow \#\theta_\psi = 0 \pmod N$

$PH \subseteq P^{\#P}$

- Summary:
  - First, randomized reduction of PH to $P^{\oplus P}$
    - TQBF instance ψ to $\oplus$SAT instance φ_ψ
    - $\psi \Rightarrow \oplus\varphi_\psi$ w.p. > 2/3; $\neg\psi \Rightarrow \neg\oplus\varphi_\psi$ (w.p. 1)
  - Converting the probabilistic guarantee to a deterministic #P calculation
    - ψ s.t. $\neg\oplus\varphi_\psi \Rightarrow \#\theta_\psi = 0 \pmod{N}$
    - ψ s.t. $\oplus\varphi_\psi$ w.p. > 2/3 $\Rightarrow \#\theta_\psi \neq 0 \pmod{N}$

Approximation for #P

Approximation for #P

- α-approximation of f: estimate f(x) within a factor α

Approximation for #P

- α-approximation of f: estimate f(x) within a factor α
- Randomized approximation ("PAC"): answer is within a factor α with probability at least $1-\delta$

Approximation for #P

- α-approximation of f: estimate f(x) within a factor α
- Randomized approximation ("PAC"): answer is within a factor α with probability at least $1-\delta$
- #CYCLE is hard to even approximate unless P=NP

Approximation for #P

- α-approximation of f: estimate f(x) within a factor α
- Randomized approximation ("PAC"): answer is within a factor α with probability at least $1-\delta$
- #CYCLE is hard to even approximate unless P=NP
  - If P=NP, every problem in #P can be "well approximated"

Approximation for #P

- α-approximation of f: estimate f(x) within a factor α
- Randomized approximation ("PAC"): answer is within a factor α with probability at least 1-δ
- #CYCLE is hard to even approximate unless P=NP
  - If P=NP, every problem in #P can be "well approximated"
- Permanent has an FPRAS

Approximation for #P

- α-approximation of f: estimate f(x) within a factor α
- Randomized approximation ("PAC"): answer is within a factor α with probability at least $1-\delta$
- #CYCLE is hard to even approximate unless P=NP
    - If P=NP, every problem in #P can be "well approximated"
- Permanent has an FPRAS
    - For any ε, $\delta > 0$, α-approximation for $\alpha = 1-\varepsilon$ in time poly(n, log $1/\varepsilon$, log $1/\delta$)

Approximation for #P

- α-approximation of f: estimate f(x) within a factor α
- Randomized approximation ("PAC"): answer is within a factor α with probability at least $1-\delta$
- #CYCLE is hard to even approximate unless P=NP
  - If P=NP, every problem in #P can be "well approximated"
- Permanent has an FPRAS
  - For any $\varepsilon, \delta > 0$, α-approximation for $\alpha = 1-\varepsilon$ in time poly(n, log $1/\varepsilon$, log $1/\delta$)
  - Technique: Monte Carlo Markov Chain (MCMC)

Approximation for #P

- α-approximation of f: estimate f(x) within a factor α
- Randomized approximation ("PAC"): answer is within a factor α with probability at least $1-\delta$
- #CYCLE is hard to even approximate unless P=NP
  - If P=NP, every problem in #P can be "well approximated"
- Permanent has an FPRAS
  - For any $\varepsilon, \delta > 0$, α-approximation for $\alpha = 1-\varepsilon$ in time poly(n, log $1/\varepsilon$, log $1/\delta$)
  - Technique: Monte Carlo Markov Chain (MCMC)
    - Very useful for sampling. Turns out counting ≈ sampling!