Uniform Derandomization

Simulation of BPP, RP and AM under Uniform Assumptions

A. Antonopoulos (N.T.U.A.)

Computation and Reasoning Laboratory

Uniform Derandomization of BPP

Theorem (IW98)

If **EXP** $\neq$ **BPP**, *then, for every* $\delta > 0$, *every* **BPP** *algorithm can be simulated deterministically in time* $2^{n^{\delta}}$ *so that, for infinitely many* n*'s, this simulation is correct on at least* $1 - \frac{1}{n}$ *fraction of all inputs of size* n.

- That's the first (universal) Derandomization result, which implies the non-trivial derandomization of **BPP**, under a fair (but open) assumption!

But:

1. The simulation works only for infinitely many input lengths (i.o. complexity)
2. May fail on a negligible fraction of inputs even of these lengths!

Proof Outline

1. **Hard Function**: We will use a "Σ_2^p-hard" Boolean Function f with some desired properties (PERMANENT in our case).
2. **The Generator**: We'll construct a PRG G using the above function, similar to the NW-construction.
3. **Derandomization**: We will fix a (probabilistic) algorithm for an $L \in$ **BPP**, and for all inputs we will run it deterministically over all outputs of G, and take the majority vote!
   If this algorithm fails to be in subexponential time, then we'll have an efficient distinguisher!
4. **Removing the Oracle:** If the above holds we have:
   - An efficient algorithm for f_n given an oracle.
   - We can "use" our construction as a **BPP** algorithm for f, by removing its oracles!

   And, thus, we have a contradiction, which proves our theorem!

The Hard Function

Theorem (BFNW93)

If $\mathbf{EXP} \not\subseteq \mathbf{P}_{/\mathbf{poly}}$*, then* $\mathbf{BPP} \subseteq \mathbf{SUBEXP}$ *for infinitely many input lengths.*

- So, we -*fairly*- assume that $\mathbf{EXP} \subseteq \mathbf{P}_{/\mathbf{poly}}$.
- Then, $\mathbf{EXP} = \Sigma_2^p$
- $\mathbf{EXP} \subseteq \mathbf{PH} \subseteq \mathbf{P}^{\mathtt{PERMANENT}}$
- Then $\mathtt{PERMANENT}$ is $\mathbf{EXP}$-complete!
- We construct a PRG (like the NW-construction) using $\mathtt{PERMANENT}$ as hard function...
- Why $\mathtt{PERMANENT}$? ? ?
- $\mathtt{PERMANENT}$ is:
  1. **Random Self-Reducible**
  2. **Downward Self-Reducible**

The Hard Function

- **Formal Definitions:**

Definition (Construction Problems)

Let $f : \{0,1\}^* \to \{0,1\}^*$ and $\varepsilon : \mathbb{N} \to [0,1]$.
Construction problem $C_n^{f,\epsilon}$ contains all circuits C with n inputs satisfying:

$$Pr_{x \in U^{\{0,1\}^n}}[C(x) = f(x)] \geq \epsilon(n)$$

- **Circuits Computing** f: $C^f = C^{f,1}$
- **Distinguishers**: Let $m : \mathbb{N} \to \mathbb{N}$, $G = \{G_n : \{0,1\}^{m(n)} \to \{0,1\}^n\}$. $D^{G,\varepsilon}$ contains all circuits D with n inputs satisfying

$$\left| \mathbf{Pr}_{y \in \{0,1\}^{m(n)}} [D(G(y)) = 1] - \mathbf{Pr}_{x \in \{0,1\}^n} [D(x) = 1] \right| \geq \varepsilon$$

The Hard Function

Definition

An *efficient construction* of B from A is a probabilistic polynomial-time algorithm that $\forall n \forall \alpha$, outputs a member of B_n with probability at least $1-\alpha$. If such a construction exists, we denote it by $A \to B$. When we allow to the construction to make also queries to an oracle O, we denote it $A \to^O B$.

- The relation "$\to$" is transitive.

Definition (Random Self-Reducibility)

Solving the problem on any input x can be reduced to solving it on a sequence of random inputs $y_1, y_2, \ldots,$ where each y_i is uniformly distributed among all inputs. In our formalism:

$$C^{f,1-n^{-c}} \to C^f$$

The Hard Function

Definition (Downward Self-Reducibility)

A language L is *downward self-reducible* if there is a polynomial-time algorithm R, such that:

$$\forall n \forall x \in \{0,1\}^n : R^{L_{n-1}}(x) = L(x)$$

where by L_k we denote an oracle that solves L on inputs of size at most k. With Turing Reductions:

$$L_n \leq_T^p L_{n-1}$$

The Pseudorandom Generator

- As we saw, a PRG is a function $G : \{0,1\}^{\ell} \to \{0,1\}^{m}$ with certain properties.
- It is easy to construct a generator $G : \{0,1\}^{\ell} \to \{0,1\}^{\ell+1}$, by concatenating a bit: $G(z) = z \circ f(z)$
  where $f : \{0,1\}^{\ell} \to \{0,1\}$ s.t. $H(f) \geq s$.
- We argue that this generator is $(s-3, 1/s)$-pseudorandom!

Theorem

If there is a circuit D, $|D| = s$, such that:

$$\left|\mathbf{Pr}_x\left[D(x \circ f(x)) = 1\right] - \mathbf{Pr}_{x,b}\left[D(x \circ b) = 1\right]\right| > \varepsilon$$

Then there is a circuit A, $|A| = s + 3$, such that:

$$\mathbf{Pr}_x\left[A(x) = f(x)\right] > \frac{1}{2} + \varepsilon$$

The Pseudorandom Generator

- By applying many times the same idea, we can construct a PRG that *doubles* the length of its output:
- But, for our purpose, we need a generator with output **exponentially larger** than the input!
- NW idea is to take the seed's pieces **partly depedent** (nondisjoint), while we want to "take care" of their intersections. The idea is simple & smart:

Definition

A family $\mathcal{S} = \{S_1, \ldots, S_m\}$ of subsets of $[\ell]$ is an (ℓ, n, d)-design, if $|S_j| = n, \forall j$ and $|S_j \cap S_k| \leq d, \forall j \neq k$ $(d < n < \ell)$.

- The following Lemma implies that we can constuct efficiently such designs:

The Pseudorandom Generator

Lemma

For every integer n and fraction $\gamma > 0$, there is a $(\ell, n, \log m)$-design $\{S_1, \dots, S_m\}$ over $[\ell]$, where $\ell = \mathcal{O}(n/\gamma)$ and $m = 2^{\gamma n}$. Such a design can be constructed in $\mathcal{O}(2^\ell \ell m^2)$ steps.

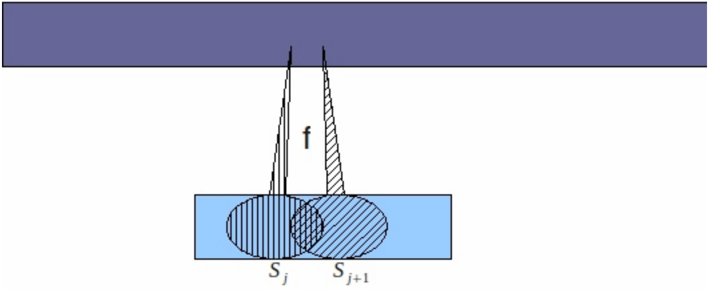
- Now, we can formally define the NW-generator:

Definition

Let $\mathcal{S} = \{S_1, \dots, S_m\}$ a (ℓ, n, d)-design and $f : \{0,1\}^n \to \{0,1\}$. The NW-generator is the function $NW_{\mathcal{S}}^f : \{0,1\}^\ell \to \{0,1\}^m$ that maps every $z \in \{0,1\}^\ell$ to

$$NW_{\mathcal{S}}^f(z) = f(z_{|S_1}) \circ f(z_{|S_2}) \circ \cdots \circ f(z_{|S_m})$$

The Pseudorandom Generator



Theorem

If $\mathcal{S} = \{S_1, \ldots, S_m\}$ a (ℓ, n, d)-design with $m = 2^{d/10}$ and $f : \{0,1\}^n \to \{0,1\}$ satisfies $H(f) > 2^{2d}$, then the distribution $NW^f_{\mathcal{S}}(U_\ell)$ is $(\frac{H(f)}{10}, \frac{1}{10})$-pseudorandom.

The Pseudorandom Generator

- We can prove the above theorem using the corresponding (to the toy-generator) lemma:

Lemma

Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function and $\mathcal{S} = \{S_1, \ldots, S_m\}$ is a $(\ell, n, \log m)$-design. Suppose that $D : \{0,1\}^m \to \{0,1\}$ is such that:

$$\left| \mathbf{Pr}_r \left[D(r) = 1 \right] - \mathbf{Pr}_z \left[D(NW^f_{\mathcal{S}}(z)) = 1 \right] \right| > \varepsilon$$

Then, there exists a circuit C of size $\mathcal{O}(m^2)$ such that:

$$\left| \mathbf{Pr}_x \left[D(C(x)) = f(x) \right] - 1/2 \right| \geq \frac{\varepsilon}{m}$$

The Pseudorandom Generator

- Our Main Lemma is the following:

Lemma

$$D^{G_d, 1/5} \rightarrow^{f_n} C^{f}$$

The Derandomization

- Our main goal is to show that:
  If this simulation fails on all input lengths for a given δ, then $\exists d \forall n$, using an oracle for f_n, we can construct a distinguisher for G_d.
- For each n, we will construct a G_n from $n^c \to n^d$ bits, computable in pol-time with an f_n oracle.
- Given a Distinguisher for the output of the generator, we will construct a circuit computing f.
- We simulate a **BPP** algorithm as follows:
  - Let k be the input size.
  - BPP algorithm uses k^{c_1} random bits!
  - Set: $d = 2cc_1/\delta$ and $n = k^{\delta/2c}$
  - Compute the range of G_n, a set of $n^d = k^{c_1}$ bit strings, in time $\mathcal{O}\left(2^{n^c}\right) = \mathcal{O}\left(2^{k^\delta}\right)$.

The Derandomization

Lemma

If this algorithm fails to be in Subexponential Time 2^{n^δ}, then we will have an efficient Distinguisher. That is, for some $c, \delta > 0$, $D^{G_d^f, 1/5}$ is efficiently constructible with oracle access to f_n.

Proof Sketch:

- Assume that the above algorithm is incorrect with probability $1/k^d$ according to some sampleable distribution μ_k
- Given n, we set $k = n^{2c/\delta}$ and sample $x_1, \ldots, x_r$ $(r = k^{\mathcal{O}(1)})$ according to μ_k.
- With high probability, the algorithm fails for one of $x_1, \ldots, x_r$!
- Let D_i view its input as random sequence, and simulate the **BPP** on x_i using that random sequence.

The Derandomization

Proof Sketch: (cont.)

- $D_1, \ldots, D_r$ are produced in PPT, so, with high probability D_i distinguishes outputs of G_n from truly random strings.
- We use the f_n oracle to find which D_i is actually a distinguisher.
- Assuming that error probability of **BPP** algorithm is $1/10$, the distinguisher is in $D^{G_d^f, 1/5}$. $\square$

Removing the Oracle

- We saw that, if the conclusion of our Theorem fails, then we have a PPT algorithm which $\forall n$, constructs a circuit for f_n using an oracle.
- Recall that f is R.S.R and D.S.R.
- This algorithm can be turned into a **BPP** algorithm for f!

Lemma

If f is D.S.R. and C^f is efficiently constructible using oracle f_n, then $f \in$ **BPP**.

Proof Sketch:

- We recursively construct circuits $C_1 \in C_1^f, \ldots, C_n \in C_n^f$
- Say we have computed C_i.

Removing the Oracle

Proof Sketch: (cont.)

- We run the (efficient) construction algorithm for C^f_{i+1} with oracle f_{i+1} (with error $\alpha = 1/n^2$), simulating queries to f_{i+1} by R^{C_i}
- $|C_i| \leq$(Time taken by the construction)$\leq p(n)$, indepedent of the size of C_i.
- So, the time for each stage (including evaluating oracle calls) is polynomial in n.
- Also, the probability that $C_n \notin C^f_n$ is at most $\alpha \cdot n = 1/n$, so the error is bounded. $\square$

So, what have we done?

How can we formalize Computational Indistinguishability?

- Leibniz: "*Indistinguishable things are identical.*"
- As we saw, in Complexity Theory we consider as **equal**, objects we cannot "separate" with any efficient procedure.
- We can formalize this as:

Definition

A refuter is a (length-preserving) Turing Machine R, such that $R(1^n) \in \{0,1\}^n$.

Definition

Two languages $L, M \subseteq \{0,1\}^*$ are $t(n)$*-indistinguishable*, denoted as $L \overset{t(n)}{=} M$, if for every deterministic $t(n)$-time refuter R we have $R(1^n) \notin L \triangle M$ for all but finitely many n's.

How can we formalize Computational Indistinguishability?

- A **refuter** is an *adversasy*, who, given specific computing power, tries to distinguish a language (or a Boolean function) from another.
  If it fails, we consider the two languages equal.
- Refuters can be deterministic, non-deterministic, or probabilistic. In the case of non-determinism, refuter's each nondeterministic branch, on input 1^n, either produces a string in $\{0,1\}^n$, or is marked with *reject*.
- We say that L and M are **P**-*indistinguishable*, denoted as $L \overset{\mathbf{P}}{=} M$, if $L \overset{p(n)}{=} M$ for every polynomial $p(n)$.
- Similarly for other classes (e.g. $L \overset{\mathbf{EXP}}{=} M$, $L \overset{\mathbf{SUBEXP}}{=} M$ etc).
- This setting can work also for **infinitely many** input sizes (*i.o. complexity*).

How can we formalize Computational Indistinguishability?

- We can define the appropriate family of complexity classes:

Definition

For a complexity class $\mathcal{C}$ of languages over $\{0,1\}$, we can define the complexity class:

$$\mathrm{pseudo}_P\mathcal{C} = \{L \subseteq \{0,1\}^* | \exists M \in \mathcal{C} \textit{ such that } L \overset{\mathbf{P}}{=} M\}$$

- The refuters above are required to fail almost everywhere at producing a certain string ($\in L \triangle M$).
  This requirement can be relaxed in **i.o. complexity** setting.
- We can prove a "*Time Hierarchy Theorem*" for the pseudo setting:

How can we formalize Computational Indistinguishability?

Theorem

Let $t_2(n)$ be a constructible function, and let $t_1(n) \log t_1(n) \in o(t_2(n))$. Then, for infinitely many input sizes:

$$DTIME(t_2(n)) \not\subseteq \mathtt{pseudo}DTIME(t_1(n))$$

Variations of Refuters

Definition (Bounded-error probabilistic refuters)

Let $t(n)$ be a time bound. Two languages $L, M \subseteq \{0,1\}^*$ are *bounded-error* probabilistically $t(n)$-*indistinguishable*, denoted as $L \overset{\mathbf{BP}-t(n)}{=} M$, if for every probabilistic $t(n)$-time refuter R we have:

$$\mathbf{Pr}\left[R(1^n) \notin L \triangle M\right] \geq 1 - n^{-c}$$

for every $c \in \mathbb{N}$, and for all but finitely many n's. Analogously:

$$\mathrm{pseudo}_{BPP}\mathcal{C} = \{L \subseteq \{0,1\}^* | \exists M \in \mathcal{C} \textit{ such that } L \overset{\mathbf{BP}-p(n)}{=} M\}$$

for every polynomial $p(n)$.

Variations of Refuters

Definition (Zero-error probabilistic refuters)

Let $t(n)$ be a time bound. Two languages $L, M \subseteq \{0,1\}^*$ are *zero-error* probabilistically $t(n)$-indistinguishable, denoted as $L \overset{\mathbf{ZP}-t(n)}{=} M$, if for every probabilistic refuter R which halts within time $t(n)$ with probability at least n^{-c}, for some $c \in \mathbb{N}$ and for all by finitely many n's we have:
$R(1^n) \notin L \triangle M$, for at least one legal computation of R on input 1^n which halts in time $t(n)$. Analogously:

$$\mathtt{pseudo}_{ZPP}\mathcal{C} = \{L \subseteq \{0,1\}^* | \exists M \in \mathcal{C} \text{ such that } L \overset{\mathbf{ZP}-p(n)}{=} M\}$$

for every polynomial $p(n)$.

Variations of Refuters

- The phrase "*for all by finitely many n's*" can be replaced by "*for infinitely many n's*" in the two above definitions.
- Refuters, as we defined them, are *uniform* adversaries.
- Using our new formalism, we can restate *IW98* Theorem (the Uniform **BPP** Derandomization Main Theorem) as follows:

Theorem (IW98)

If **BPP** $\neq$ **EXP**, then, for infinitely many input sizes:

$$\mathbf{BPP} \subseteq \mathrm{pseudo}_{BPP}\mathbf{SUBEXP}$$

- Recall that

$$\mathbf{SUBEXP} = \bigcap_{\varepsilon > 0} DTIME\left(2^{n^{\varepsilon}}\right)$$

.

Uniform Derandomization of RP

Theorem 1

If **ZPP** $\neq$ **EXP**, then, for infinitely many input sizes:

$$\mathbf{RP} \subseteq \mathrm{pseudo}_{ZPP}\mathbf{SUBEXP}$$

Theorem 2

At least one of the following holds:

1. **ZPP** $=$ **BPP**
2. **RP** $\subseteq \mathrm{pseudo}_{ZPP}$**SUBEXP** infinitely often

Uniform Derandomization of RP

Proof (of Theorem 1):

- Suppose that the conclusion does not hold.
- Then, Theorem 2$\Rightarrow$ **BPP** $=$ **ZPP**.
- On the other hand, we also have:
  $\mathbf{BPP} \not\subseteq \mathrm{pseudo}_{BPP}\mathbf{SUBEXP} \overset{IW98}{\Longrightarrow} \mathbf{BPP} = \mathbf{EXP} \Rightarrow$
  $\Rightarrow \mathbf{ZPP} = \mathbf{EXP}$ Contradiction!

$\square$

Uniform Derandomization of RP

Proof Sketch (of Theorem 2):

- Based on the notion of *Natural Proofs* [RR94]
- Conjecture:

  There are no natural predicates which are n^c-useful, for some $c \in \mathbb{N}$.

- So, we can use the truth tables of (non-uniformly) easy functions instead of random strings, and accept if *at least* one of them works.
- The resulting deterministic simulation run in subexponential time, since there are few easy functions.
- If the simulation **fails** in the uniform setting, we obtain a natural predicate which can be used as a hardness test.

- Both **BPP** and **RP** results, can be stated as **Gap Theorems**, providing an alternative interpretation:

Theorem (IW98)

Either:

1. **BPP** $=$ **EXP**
2. *For any* $\varepsilon > 0$, *every* **BPP** *algorithm can be simulated in deterministic time* $2^{n^{\varepsilon}}$, *so that this simulations seems correct to any Bounded-error probabilistic refuter infinitely often.*

Theorem (Kab00)

Either:

1. **ZPP** $=$ **EXP**
2. *For any* $\varepsilon > 0$, *every* **RP** *algorithm can be simulated in deterministic time* $2^{n^{\varepsilon}}$, *so that this simulations seems correct to any Zero-error probabilistic refuter infinitely often.*

Arthur-Merlin Games Definition

- King Arthur doesn't trust wizard Merlin, but he recognizes his "supernatural" abilities.
- Merlin wants to convince the King that a string x belongs to a certain language L.
- So, Merlin plays the role of the *prover*, and Arthur the role of the *verifier*.
- Also (unlike regular IPs), Merlin is able to read the whole history of the computation of Arthur on the given input, including the random coin tosses made by Arthur!
- So, **AM** is the class of languages L with an interactive proof, in which the verifier sends a random string, and the prover responding with a message, where the verifier's decision is obtained by applying a deterministic polynomial-time algorithm to the message.

Arthur-Merlin Games Definition

- Also, the class **MA** consists of all languages L, where there's an interactive proof for L in which the prover first sending a message, and then the verifier is "tossing coins" and computing its decision by doing a deterministic polynomial-time computation involving the input, the message and the random output.

Definition

A language L is in **AM** if $\exists$ relation $M \in \mathbf{NP}$, and $m = poly(n)$, such that $\forall x \in \{0,1\}^n$:

$$x \in L \Rightarrow \mathbf{Pr}_{y \in \{0,1\}^m}[M(x,y) = 1] \geq \frac{3}{4}$$

$$x \notin L \Rightarrow \mathbf{Pr}_{y \in \{0,1\}^m}[M(x,y) = 1] < \frac{1}{4}$$

Arthur-Merlin Games Definition

- By $\mathbf{AM}[k]$ we denote the k-round interaction between Arthur (*Verifier*) and Merlin (*Prover*).
- By $\mathbf{AM\text{-}TIME}(t(n))$ we denote the Arthur-Merlin proof that takes $t(n)$ computational steps.
- $\mathbf{MA} \subseteq \mathbf{AM}$
- It should be clear that $\mathbf{MA}[1] = \mathbf{NP}$, $\mathbf{AM}[1] = \mathbf{BPP}$
- $\mathbf{AM} = \mathbf{BP}\cdot\mathbf{NP}$
- The Arthur-Merlin Hierarchy (AM proof systems wih bounded number of rounds):

  $$\mathbf{AM}[0] \subseteq \mathbf{AM}[1] \subseteq \cdots \subseteq \mathbf{AM}[k] \subseteq \mathbf{AM}[k+1] \subseteq \cdots$$

  collapses to the second level: $\mathbf{AM}[k] = \mathbf{AM}[2]$, for constants $k \geq 2$.
- $\mathbf{IP}[k] \subseteq \mathbf{AM}[k+2]$
- If $co\mathbf{NP} \subseteq \mathbf{AM}$, then: $\Sigma_2^p = \Pi_2^p = \mathbf{AM}$.

Uniform Derandomization of AM

Theorem (Lu00)

At least one of the following holds:

1. $\mathbf{AM} = \mathbf{NP}$
2. $\mathbf{NP} \subseteq \mathit{pseudo}_{NP}\mathbf{SUBEXP}$ *infinitely often.*

Uniform Derandomization of AM

Theorem

$$co\mathbf{NP} \cap \mathbf{AM} \subseteq \bigcap_{\varepsilon>0} pseudo_{NP}\mathbf{NTIME}\left(2^{n^{\varepsilon}}\right)$$

- Since GNI is in both **AM** and co**NP**, the above theorem implies that either GNI $\in$ **NP**, or it can be simulated in deterministic subexponential time, and the simulation is correct for the point of view of any *nondeterministic* polynomial-time refuter.

Corollary

$$\mathtt{GNI} \in \bigcap_{\varepsilon>0} \mathtt{pseudo}_{NP}\mathbf{NTIME}\left(2^{n^{\varepsilon}}\right)$$

- The above inclusions hold for infinitely many n's.

Other Results (focusing on Time vs. Space)

Theorem

Either:

1. $\mathbf{DTIME}(t(n)) \subseteq \bigcap_{\varepsilon>0} \mathbf{DSPACE}(t^{\varepsilon}(n))$ *infinitely often for any function* $t(n) = 2^{\Omega(n)}$*, or*
2. $\mathbf{P} = \mathbf{BPP}$ *and* $\mathbf{AM} = \mathbf{NP}$ *and* $\mathbf{PH} \subseteq \oplus\mathbf{P}$

Theorem

Either:

1. $\mathbf{DTIME}(t(n)) \subseteq \bigcap_{\varepsilon>0} \mathbf{DSPACE}\left(2^{\log^{\varepsilon} t(n)}\right)$ *infinitely often for any function* $t(n) = 2^{\Omega(n)}$*, or*
2. $\mathbf{BPP} \subseteq \mathbf{QuasiP}$ *and* $\mathbf{AM} \subseteq \mathbf{NQuasiP}$ *and* $\mathbf{PH} \subseteq \oplus\mathbf{QuasiP}$

Other Results (focusing on Time vs. Space)

Theorem

Either:

1. $\mathbf{DTIME}(t(n)) \subseteq \mathbf{DSPACE}(poly(\log t(n)))$ *infinitely often for any function* $t(n) = 2^{\Omega(n)}$, *or*
2. $\mathbf{BPP} \subseteq \mathbf{SUBEXP}$ *and* $\mathbf{AM} \subseteq \mathbf{NSUBEXP}$ *and* $\mathbf{PH} \subseteq \oplus\mathbf{SUBEXP}$

Recall that:

- $\mathbf{QuasiP} = \mathbf{DTIME}\left(2^{poly(\log n)}\right)$
- $\mathbf{NQuasiP} = \mathbf{NTIME}\left(2^{poly(\log n)}\right)$
- $\oplus\mathbf{QuasiP} = \oplus\mathbf{TIME}\left(2^{poly(\log n)}\right)$
- $\mathbf{SUBEXP} = \cap_{\varepsilon>0}\mathbf{DTIME}\left(2^{n^{\varepsilon}}\right)$
- $\mathbf{NSUBEXP} = \cap_{\varepsilon>0}\mathbf{NTIME}\left(2^{n^{\varepsilon}}\right)$
- $\oplus\mathbf{SUBEXP} = \cap_{\varepsilon>0} \oplus \mathbf{TIME}\left(2^{n^{\varepsilon}}\right)$

The High-End

Theorem

If $\mathbf{E} \not\subseteq \mathbf{AM\text{-}TIME}(2^{\epsilon n})$, *for some* $\epsilon > 0$, *then for all* $c > 0$, *and infinitely many input sizes, we have:*

$$\mathbf{AM} \subseteq \mathtt{pseudo}_{NTIME(n^c)}\mathbf{NP}$$

- "Gap Theorem" interpretation: Either **AM** is almost as powerful as **E**, or **AM** is no more powerful than **NP** from the point of view of any *non-deterministic* efficient observer!

The High-End

- Why be interested in $\mathbf{AM} \cap co\mathbf{AM}$:

$$\mathbf{PZK} \subseteq \mathbf{SZK} \subseteq \mathbf{AM} \cap co\mathbf{AM}$$

- We also have a similar gap-theorem for $\mathbf{AM} \cap co\mathbf{AM}$

Theorem

If $\mathbf{E} \not\subseteq \mathbf{AM\text{-}TIME}(2^{\epsilon n})$, *for some* $\epsilon > 0$, *then, for infinitely many input sizes:*

$$\mathbf{AM} \cap co\mathbf{AM} \subseteq \mathbf{NP} \cap co\mathbf{NP}$$

- Indeed, in the above Theorem we can (*non-trivially of course*) get rid of "*infinitely often*" setting, and go up to "*almost everywhere*" complexity:

The High-End

Theorem

If $\mathbf{E} \not\subseteq \mathbf{AM\text{-}TIME}(2^{\epsilon n})$*, for some* $\epsilon > 0$*, then, for all but finitely many input sizes:*

$$\mathbf{AM} \cap co\mathbf{AM} \subseteq \mathbf{NP} \cap co\mathbf{NP}$$

- The following theorem concerns Derandomization of **AM** under the assumption that there is a *hard function* in **NE**:

Theorem

If $\mathbf{NE} \cap co\mathbf{NE} \not\subseteq \mathbf{AM\text{-}TIME}(2^{\delta n})$*, for some* $\delta > 0$*, then, for infinitely many* n*'s, and for every* $c, \varepsilon > 0$*:*

$$\mathbf{AM} \subseteq \mathtt{pseudo}_{NTIME(n^c)}\mathbf{NTIME}\left(2^{n^{\varepsilon}}\right)$$

And also, for every $\varepsilon > 0$*:*

The Low-End

Theorem

There exists a language A complete for **E** *(resp.* **EXP)***, such that for every time-constructible function t: $m < t(m) < 2^m$, either:*

1. *A has an Arthur-Merlin protocol running in time $t(m)$*
2. *for any language $L \in$* **AM** *there is a nondeterministic machine M that runs in time $2^{\mathcal{O}(m)}$ (resp. $2^{m^{\mathcal{O}(1)}}$) on inputs of length $n = t(m)^{\Theta(1/(\log m - \log\log t(m))^2)}$ (resp. $n = t(m)^{\Theta(1/(\log m)^2)}$) such that for any refuter R running in time $t(m)$ when producing strings of length n, there are infinitely many n's on which L and $L(M)$ are $t(m)$-indistinguishable.*

The Low-End

Theorem

There exists a language A complete for **E** *(resp.* **EXP***), such that for every time-constructible function t: $m < t(m) < 2^m$, either:*

1. *A has an Arthur-Merlin protocol running in time $t(m)$*
2. *for any language $L \in \mathbf{AM} \cap \mathrm{co}\mathbf{AM}$ there is a nondeterministic machine M that runs in time $2^{\mathcal{O}(m)}$ (resp. $2^{m^{\mathcal{O}(1)}}$) on inputs of length $n = t(m)^{\Theta(1/(\log m - \log\log t(m))^2)}$ (resp. $n = t(m)^{\Theta(1/(\log m)^2)}$) such that for any refuter R running in time $t(m)$ when producing strings of length n, there are infinitely many n's on which L and $L(M)$ are $t(m)$-indistinguishable.*

- And, like the theorem of the previous section, we can extract the "infinitely often" setting and have a more general result.

Further Reading 1

- Andrew C. Yao, *Theory and application of trapdoor functions*, SFCS82: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, pages 80-91, 1982.
- Manuel Blum and Silvio Micali, *How to generate cryptographically strong sequences of pseudo-random bits*, SIAM J. Comput., 13:850-864, November 1984
- L.Babai, L.Fortnow, N.Nisan, and A.Wigderson, *BPP has subexponential time simulations unless EXPTIME has publishable proofs*, Comput. Complex., 3(4):307-318, 1993
- Noam Nisan and Avi Wigderson. *Hardness vs Randomness*, J. Comput. Syst. Sci., 49(2):149-167, 1994
- Russell Impagliazzo and Avi Wigderson, *P=BPP unless E has sub-exponential circuits: Derandomizing the XOR lemma, In Proceedings of the 29th STOC*, pages 220-229. ACM Press, 1997.

Further Reading 2

- Russell Impagliazzo and Avi Wigderson.*Randomness vs Time: De-Randomization under a Uniform Assumption*, *In FOCS*, pages 734- 743, 1998.
- Valentine Kabanets. *Easiness assumptions and hardness tests: Trading time for zero error*, In Proceedings of the 15th Annual IEEE Conference on Computational Complexity, COCO 00, pages 150-, Washington, DC, USA, 2000. IEEE Computer Society
- Chi-Jen Lu. *Derandomizing Arthur-Merlin Games under Uniform Assumptions*, Computational Complexity, 10:247-259, 2000.
- Christopher Umans, *Pseudo-random generators for all hardness*, Journal of Computer and System Science, pages 419-440, 2003.
- Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. *Uniform hardness versus randomness tradeoffs for Arthur-Merlin games*, Comput.Complex., 12:85-130, September 2003.

Further Reading 3

- Ronen Shaltiel and Christopher Umans, *Low-end uniform hardness versus randomness tradeoffs for* **AM**, SIAM J. Comput., 39(3):1006-1037, 2009.
- Ronen Shaltiel, *Typically-correct derandomization*, *SIGACT News*, 41(2):57-72, 2010.

Thank You!