Randomized Complexity Classes

Note Title 4/23/2013

model: probabilistic TM
deterministic TM w/ an additional read-only input tape containing coin flips.

BPP: Bounded-Error Probabilistic Poly-time.
$L \in BPP$ if $\exists$ p.p.t. TM M → probabilistic poly-time
$x \in L \Rightarrow Prob_y[M(x,y) \text{ accept}] \geq 2/3$
$x \notin L \Rightarrow Prob_y[M(x,y) \text{ rejects}] \geq 2/3$

RP: $x \in L \Rightarrow Prob_y[M(x,y) \text{ accepts}] \geq 1/2$
$x \notin L \Rightarrow Prob_y[M(x,y) \text{ rejects}] = 1$

co-RP
1
1/2.

ZPP: (zero error prob poly-time) ZPP = RP ∩ co-RP.
$Prob_y[M(x,y) \text{ outputs "fail"}] \leq 1/2$
otherwise it outputs the correct answer.
↳ or runs in expected poly time and always produces the right answer.

These classes may better capture "efficiently computable" better than P.

→ the 1/2 in the defn of ZPP, RP, co-RP can be replaced with any 1/poly(n).

→ the 2/3 in the defn of BPP can be replaced with any $1/2 + 1/poly(n)$.
(via Error reduction).

Suppose we have L + ppt M
$x \in L \Rightarrow Pr\ M \text{ accepts} \geq \epsilon$
$x \notin L \Rightarrow Pr\ M \text{ rejects} = 1$

M_2': Simulate M k/ϵ times, each time with independent coin flips.
- accept if any simulation accepts
- O.W. reject.

if $x \in L$ prob a given simulation "bad" $\le (1-\epsilon)$
prob all simulations bad: $(1-\epsilon)^{k/\epsilon} \sim e^{-k}$
Prob M accepts $\ge 1 - e^{-k}$

for $\epsilon = 1/\text{poly}(n)$
$\frac{n}{\epsilon} = \text{poly}(n)$
$\Rightarrow$ prob of error e^{-n}

if $x \notin L$ Prob M' rejects $= 1$.

Error reduction for BPP:
$x \in L$ Pr M accepts $\ge 1/2 + \epsilon$
$x \notin L$ Pr M rejects $\ge 1/2 + \epsilon$

Simulate M k/ϵ^2 times with independent coin flips.
take the majority answer.

X_i = random variable $= 1$ if i^{th} answer is correct
0 otherwise.

$\Pr[X_i = 0] \le 1/2 - \epsilon$ $\Pr[X_i = 1] \ge 1/2 + \epsilon$
$E[X_i] = 1/2 + \epsilon$.
X_i's are mutually independent
$X = \sum_i X_i$ $\mu = E[X] = (1/2 + \epsilon)\, k/\epsilon^2$ $m = k/\epsilon^2$
Chernoff's Inequality says $\Pr[X \le m/2] \le 2^{-\Omega(\epsilon^2 m)}$
$2^{-\Omega(k)}$

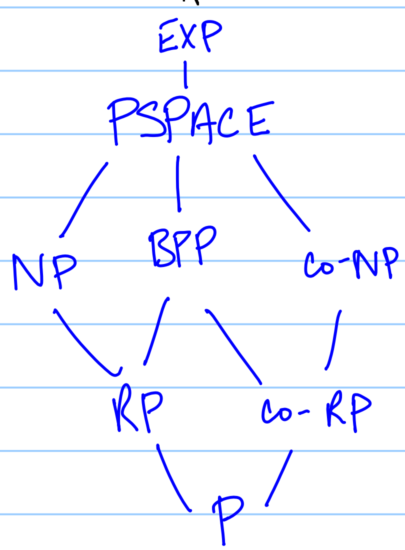
As long as $\epsilon > 1/\text{poly}(n)$ and k $O(\text{poly } n)$,
the running time is polynomial and error exp small //

RP, co-RP, BPP, ZPP are all contained in P
(you can always just ignore the random string).

They are also all contained in PSPACE:
exhaustively try all y and count the # of accepting computations. $\Pr[\text{accept}] = \dfrac{\#\, y \text{ s.t. } M(x,y) = \text{acc}}{\#\text{ all possible } y}$.

Also $RP \subset NP$ (and $\text{co-}RP \subseteq \text{co-}NP$)

An NTM can guess y then compute $M(x,y)$
$x \notin L$ $\quad \forall y \quad M(x,y) = \text{reject}$
$x \in L$ $\quad$ For at least half of the y's $M(x,y) = \text{accept}$



RP requires most y to be correct
NP only needs one y to be correct.

How powerful is BPP?

We have an example of a problem in BPP that we only know how to solve in EXP.

Not known if BPP = EXP (or even NEXP)
Strong hints that BPP ≠ EXP however.

Is there a deterministic simulation of BPP that does better than brute-force search?

Yes, if we allow non-uniformity.

Theorem

BPP $\subset$ P/poly (Adleman)

Take $L \in$ BPP

Error reduction gives TM M s.t.

if $x \in L$ $|x| = n$ $\Pr_y[M(x,y) \text{ accepts}] \geq 1 - (1/2)^{n^2}$

if $x \notin L$ $|x| = n$ $\Pr_y[M(x,y) \text{ rejects}] \geq 1 - (1/2)^{n^2}$

y is "bad" for x if $M(x,y)$ gives the wrong answer.

Fix x $\Pr_y[y \text{ is bad for } x] \leq (1/2)^{n^2}$

$\Pr_y[y \text{ is bad for some } x] \leq 2^n (1/2)^{n^2} < 1$

↳ summing up over all x.

$\exists y$ for which y is good for all inputs x of length n.

thus y is the hint for inputs of length n.

(hard code y into C_n).

$\Rightarrow$ if BPP = EXP then EXP $\subset$ P/poly.

If randomness is all powerful then non-uniformity gives an exponential advantage

Does BPP have complete problems?

Determining if a TM M is an NTM is easy

Determining if a TM M is in BPTIME is undecidable since it requires that every string is accepted w/ probability $\leq 1/3$ or $\geq 2/3$.

A natural candidate for a BPP-complete language would be $(M, x, 1^t)$: M accepts x w.p. $\geq 2/3$ in time t. This problem is BPP-hard.
However: is it in BPP. A BPP machine can't just simulate M on input x because it could be that M accepts w/ prob $1/2$ on input x.

However if BPP = P (conjectured to be true) then it does have complete problems because P does.

Next: try to de-randomize BPP by pseudo-random generators.
↳ simulate BPP in subexponential time or better.

Pseudo-random Generator (PRG)

$$\boxed{\text{Seed}} \longrightarrow \boxed{G} \longrightarrow \boxed{\text{output string}}$$

t bits   m bits.

G must be efficiently computable
Stretches t into m bits.
"fools" small circuits. For all C of size $\leq s$:

$$\left| \Pr_y [C(y) = 1] - \Pr_z [C(G(z)) = 1] \right| \leq \epsilon.$$

Simulating BPP w/ a PRG:
Recall: $L \in BPP \Rightarrow \exists$ p.p.t TM M

$x \in L \Rightarrow \Pr_y [M(x,y) \text{ accept}] \geq 2/3$

$x \notin L \Rightarrow \Pr_y [M(x,y) \text{ rejects}] \geq 2/3$

Convert M into a ckt $C(x,y)$

Simplification: pad y s.t. $|C| = |y| = m$.

Hardwire x into circuit to get $C_x(y)$

$$\Pr_y [C_x(y) = 1] \geq 2/3 \qquad \text{"yes"}$$

$$\Pr_y [C_x(y) = 1] \leq 2/3 \qquad \text{"no"}$$

PRG: output length: m

Seed length: $t \ll m$

error $\epsilon < 1/6$

fooling size $s = m$.

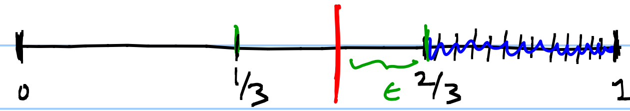
Compute $\Pr_z [C_x(G(z)) = 1]$ exactly

evaluate $C_x(G(z))$ for every $z \in \{0,1\}^t$

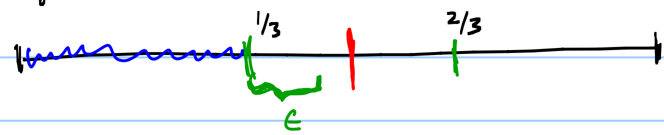
running time $\big(O(m) + (\text{time for } G) \big) 2^t$

This can distinguish between the two cases.

$x \in L$ $\quad \Pr_y [C_x(y) = 1] \geq 2/3 \quad \Pr_z [C_x(G(z)) = 1] \geq 2/3 - \epsilon > 1/2.$ $\quad < 1/6$



$x \notin L$ $\quad \Pr_y [C_x(y) = 1] \leq 1/3 \quad \Pr_z [C_x(G(z)) = 1] \leq 1/3 + \epsilon < 1/2$ $\quad < 1/6$



Blum-Micali-Yao PRG:

Initial goal: for all $1 > \delta > 0$ we will build a family of PRGs $\{G_m\}$ with

Output length $= m$ fooling size $s = m$

Seed length $= t = m^\delta$ running time: m^c

error $= \epsilon < 1/6$

implies $BPP \subset \bigcap_{\delta > 0} TIME(2^{n^\delta}) \subsetneq EXP$

Why? Simulation runs in time:

$$O\left((m + m^c) 2^{m^\delta} \right) = O\left(2^{m^{2\delta}} \right) = O(2^{n^{2c\delta}})$$

(Note: in order to get $BPP \subseteq P$, need $t = O(\log m)$)

Will require some kind of complexity assumption.
(PRGs of this type imply the existence of one-way fcns).

Definition: One Way Function (OWF)

function family $f = \{f_n\}$. $f_n \; \{0,1\}^n \to \{0,1\}^n$

f_n computable in polynomial time

for every polynomial size circuit $\{C_n\}$

$$Pr_x \left[C_n(f_n(x)) \in f_n^{-1}(f_n(x)) \right] \leq \epsilon(n)$$

$\epsilon(n) = o(n^{-c})$ for all c. note this requires hardness on average which is stronger than worst-case hardness.

It is generally believed that one-way functions exist: (integer multiplication, discrete log, etc...) widely used in cryptography.

Definition: One Way Permutation: OWF f_n which is one-to-one.

Can simplify $\Pr_x [C_n(f_n(x)) \in f_n^{-1}(f_n(x))] \le \epsilon(n)$

to $\Pr_y [C_n(y) = f_n^{-1}(y)] \le \epsilon(n)$.

Here's an attempt at a PRG from an OWF:

$t = m^{\delta}$ — Computable in time

$y_0 \in \{0,1\}^t$ — $kt^c < mt^{c-1} \Rightarrow$

$y_i = f_t(y_{i-1})$ — $m \, m^{\delta(c-1)} \le m^c$.

$G(y_0) = y_{k-1} \, y_{k-2} \cdots y_0$

$k = m/t$.

The output is "unpredictable"

no poly size ckt C can output y_{i-1} given $y_{k-1} \cdots y_i$ w/ non-negligible success prob.

if C could, then given y_i compute $y_{k-1} \cdots y_{i+1}$

use $y_{k-1}, \ldots, y_i$ to get y_{i-1}

this would be a ckt to invert f:

$f_t^{-1}(y_i) = y_{i-1}$

↳ the 1-1 assumption makes f^{-1} unique.

2 Problems:

① Although it's hard to compute y_{i-1} from y_i, it may be possible to compute one or more bits of y_{i-1} which could be used to distinguish G's output from the uniform distribution over $\{0,1\}^m$

② This notion of "unpredictability" is not necessarily enough to meet the fooling requirement:

$$\left| \Pr_y [C(y)=1] - \Pr_z [C(G(z)) = 1] \right| \le \epsilon.$$

Hard Bits

If $\{f_n\}$ is a one-way permutation, we know that no poly-size circuit can compute $f_n^{-1}(y)$ from y w/ non-negligible success prob:

$$\Pr_y [C_n(y) = f_n^{-1}(y)] \le \epsilon(n)$$

We want to identify a single bit position j for which:

no poly-size ckt can compute $(f_n^{-1}(y))_j$ from y w/ non-negligible advantage over a coin flip.

$$\Pr_y [C_n(y) = (f_n^{-1}(y))_j] \le \tfrac{1}{2} + \epsilon(n)$$

For some specific functions we know a bit position j, but would like a more general:

$$h_n : \{0,1\}^n \to \{0,1\}$$

rather than just a bit position j.

Definition: hard bit for $g = \{g_n\}$ is a family $h = \{h_n\}$ $h_n : \{0,1\}^n \to \{0,1\}$ such that if circuit family $\{C_n\}$ of size $s(n)$ achieves:

$$\Pr_y [C_n(y) - h_n(g_n(x))] \ge \tfrac{1}{2} + \epsilon(n)$$

then there is a ckt family $\{C'_n\}$ of size $s'(n)$ that achieves $\Pr_y [C'_n(y) = g_n(y)] \ge \epsilon'(n)$

$\epsilon'(n) = (\epsilon(n)/n)^{O(1)}$ $\quad s'(n) = (s(n)\, n / \epsilon(n))^{O(1)}$

In order to get a generic hard bit, we need to modify our one-way permutation.

Define $f_n' : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{2n}$

$$f_n'(x,y) = (f_n(x), y).$$

① f is a permutation iff f' is a permutation.
② f is a one-way perm iff f' is a one-way perm.

Goldreich-Levin function:

$$GL_{2n} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$$

$$GL_{2n}(x,y) = \bigoplus_{i : y_i = 1} x_i \quad \text{(inner product over } GF_2 \text{ of } x \cdot y\text{).}$$

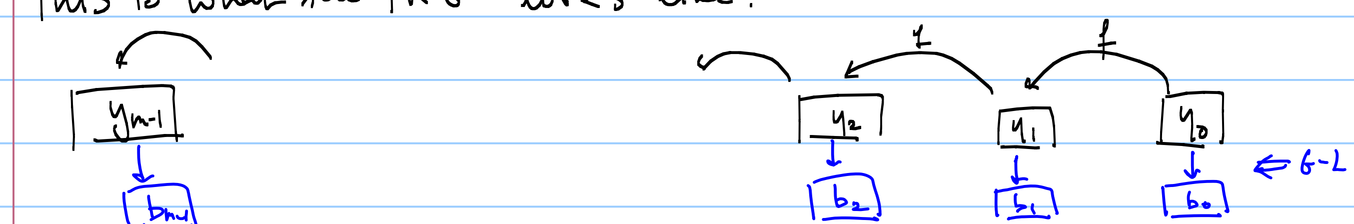
↳ y selects a subset of x's bits for parity.

Theorem: (G-L) for every function f, GL is a hard bit for f'.

We won't prove this here, but let's discuss how it will be used.

We can assume now that if we have a one-way function (call it $\{f_n\}$), that it has a hard bit for f^{-1}. (Use the modified one-way fcn and the G-L function for the hard bit).

This is what the PRG looks like.



y_0 is chosen uniformly from $\{0,1\}^t$
this yields a distribution over $(b_{m-1} b_{m-2} \cdots b_0)$ m-bit strings.

Note that because f_t is a permutation, the distribution is the same if we pick y_i at random and compute

$$\boxed{y_{m-1}} \longleftarrow \cdots \quad \boxed{y_{i+1}} \longleftarrow \boxed{y_i} \xrightarrow{f^{-1}} \boxed{y_{i-1}} \xrightarrow{f^{-1}} \boxed{y_{i-2}} \quad \cdots \xrightarrow{f^{-1}} \boxed{y_0}$$

This may be difficult to compute but it is well defined and produces the same distribution as if we start at y_0.

We know that there are no poly-sized circuits that can predict b_{i-1} from y_i w/ non-negligible bias away from a random bit:

$$Pr_{y_i}[C(y_i) = b_{i-1}] \leq \frac{1}{2} + \epsilon.$$

Given y_i, we can use f and G-L to produce
$b_{m-1} b_{m-2} \cdots b_i$

If y_i is chosen at random, this will be the same induced distribution as if $b_{m-1} \cdots b_0$ is produced starting at y_0 and then tossing out $b_{i-1} \cdots b_0$.

There is no poly size circuit that can take $b_{m-1} \cdots b_i$ and predict b_{i-1} w/ probability better than a random bit when $b_{m-1} \cdots b_i$ is chosen according to this induced distribution.

Now we need to relate this notion of predictability to distinguishability.

Distinguishers and Predicters:

Distribution D on $\{0,1\}^n$

Distinguisher

D ϵ-passes statistical tests of size s if for all circuits of size s:

$$\Pr_{y \in U_n}[C(y)=1] - \Pr_{y \leftarrow D}[C(y)=1] \leq \epsilon$$

A circuit violating this is called an efficient distinguisher.

D ϵ-passes prediction tests of size s if for all circuits of size s:

$$\Pr_{y \leftarrow D}[C(y_1, \dots y_{i-1}) = y_i] \leq \frac{1}{2} + \epsilon$$

Circuit violating this is called a predicter.

Having a predicter seems stronger than having a distinguisher.
We have that our distribution has no predicter but we need to be able to say that there is no distinguisher.

Yao showed that these are essentially the same.

Theorem (Yao): if a distribution D over $\{0,1\}^n$ $(\epsilon/n2)$-passes all prediction tests of size s, then it ϵ-passes all statistical tests of size $s' = s - O(n)$.

Proof by contradiction:

given an δ' distinguisher C:

$$\Pr_{y \leftarrow U_n}[C(y)=1] - \Pr_{y \leftarrow D}[C(y)=1] > \epsilon$$

we will show that there is a predictor P: (for some i)

$$\Pr_{y \leftarrow D}[P(y_1 \cdots y_{i-1}) = y_i] > \tfrac{1}{2} + \epsilon/n2$$

Consider hybrid distributions between D and U_n:

$D_0 = U_n$ $\qquad D_i \qquad D_n = D$

$$\underbrace{b_1\, b_2 \cdots b_i}_{\text{induced by } D}\ \underbrace{y_{i+1} \cdots y_n}_{\text{uniform}}$$

generate $b_1 \cdots b_n$
toss out $b_{i+1} \cdots b_n$

Let $p_i = \Pr_{y \leftarrow D_i}[C(y)=1]$

$p_0 = \Pr_{y \leftarrow U_n}[C(y)=1] \qquad p_n = \Pr_{y \leftarrow D}[C(y)=1]$

by assumption $|p_n - p_0| > \epsilon$.

$$\epsilon < |p_n - p_0| \leq \sum_{i=1}^{n} |p_i - p_{i-1}|$$

$\Rightarrow \exists\, i$ s.t. $|p_i - p_{i-1}| > \epsilon/n$. (assume w.l.o.g $p_i > p_{i-1}$ otherwise can just toggle the output).

Let $D_i^* = D_i$ except flip the i^{th} bit.

$p_i' = \Pr_{y \leftarrow D_i^*}[C(y)=1]$.

D_{i-1} : $b_1 \cdots b_{i-1}\ y_i\ y_{i+1} \cdots y_n$
D_i : $b_1 \cdots b_{i-1}\ b_i\ y_{i+1} \cdots y_n$
D_i^* : $b_1 \quad b_{i-1}\ \bar{b}_i\ y_{i+1} \cdots y_n$.

$p(x) = p(x_1 \cdots x_{i-1})\, 2^{-(n-i+1)}$
$p(x) = p(x_1 \cdots x_{i-1})\, p(b \mid x_1 \cdots x_{i-1})\, 2^{-n-i}$
$p(x) = p(x_1 \cdots x_{i-1})(1 - p(b \mid x_1 \cdots x_{i-1}))\, 2^{-n-i}$

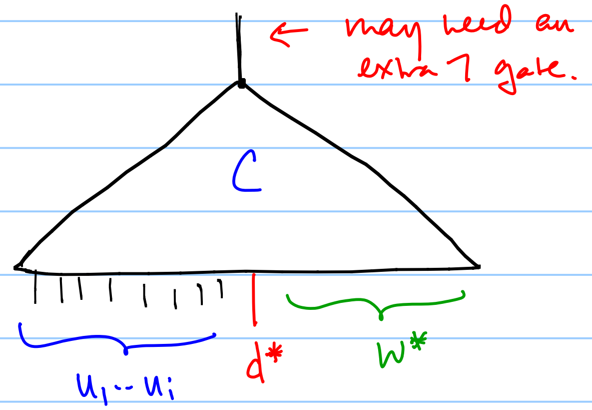
$$\Rightarrow \quad D_{i-1} = \frac{D_i + D_i^*}{2}.$$

$$P_{i-1} = \frac{P_i + P_i^*}{2}$$

Randomized predictor P' for i^{th} bit:

input: $b = b_1 \cdots b_{i-1}$ (generated by D)
flip a coin $d \in \{0,1\}$.
$w = w_{i+1} w_{i+2} \cdots w_n \leftarrow U_{n-i}$
evaluate $C(b\ d\ w)$
if $1 \rightarrow$ output d if $0 \rightarrow$ output $\neg d$.

Claim $\Pr_{b_1 \cdots b_{i-1} \leftarrow D} [P'(b_1 \cdots b_{i-1}) = b_i] > 1/2 + \epsilon/n$.



P' is a randomized procedure, we will choose a way to fix the random bits to preserve the probability of success. Call these settings d^* + w^*. P has these hardwired.

Size is $s' + O(n) = s$.

$$\Pr_{b_1\cdots b_{i-1}\leftarrow D;\ d,w\leftarrow U}\left[P'(b_1\cdots b_{i-1}) = b_i\right] =$$

$$\Pr\left[b_i = d \mid C(b,d,w)=1\right]\ \Pr\left[C(b,d,w)=1\right] + \quad \leftarrow p_{i-1}$$

$$\Pr\left[b_i = \neg d \mid C(b,d,w)=0\right]\ \Pr\left[C(b,d,w)=0\right] \quad \leftarrow (1-p_{i-1})$$

$$\Pr\left[b_i = d \mid C(b,d,w)=1\right] = \frac{\Pr\left[C(b,d,w)=1 \mid b_i=d\right]\ \overbrace{\Pr\left[b_i=d\right]}^{1/2}}{\underbrace{\Pr\left[C(b,d,w)=1\right]}_{p_{i-1}}}$$

$$\Pr\left[C(b,b_i,w)=1\right] = p_i$$

Similarly
$$\Pr\left[b_i=\neg d \mid C(b,d,w)=0\right] = \frac{\overbrace{\Pr\left[C(b,d,w)=0 \mid b_i=\neg d\right]}^{1-p_i'}\ \overbrace{\Pr\left[b_i=\neg d\right]}^{\frac{1}{2}}}{\underbrace{\Pr\left[C(b,d,w)=0\right]}_{(1-p_{i-1})}}$$

So:
$$\Pr_{b_1\cdots b_{i-1}\leftarrow D;\ d,w\leftarrow U}\left[P'(b_1\cdots b_{i-1}) = b_i\right] =$$

$$\frac{p_i\cdot\cancel{(p_{i-1})}}{2\cancel{(p_{i-1})}} + \frac{(1-p_i')\cancel{(1-p_{i-1})}}{2\cancel{(1-p_{i-1})}} = \frac{1}{2} + \frac{1}{2}(p_i - p_i')$$

$$= \frac{1}{2} + \frac{1}{2}(p_i - p_{i-1})$$

$$> \frac{1}{2} + \frac{\epsilon}{2n}. \quad //$$

Generator $G^\delta = \{G^\delta_m\}$

$t = m^\delta \quad y_0 = \{0,1\}^t \quad y_i = f_t(y_{i-1}) \quad b_i = h_t(y_i)$

$G^\delta_m(y_0) = b_{m-1}\, b_m \cdots b_0$

Theorem: (BMY) For every $\delta > 0$ there is a constant c s.t. for all d, e G^δ is a PRG with

$$\left.\begin{array}{l} \text{error } \epsilon < 1/m^d \\ \text{fooling size } S = m^e \\ \text{running time } m^c \end{array}\right\}$$

this is stronger than what we need: only need $\epsilon < 1/6$, $S = m$.

Proof: Time to compute $G^\delta_m(y_0)$ is $mt^c < m^{c+1}$

assume G^δ does not $(1/m^d)$-pass a statistical test $C = \{C_m\}$ of size m^e

$$\left| \Pr_{y \leftarrow U_m}[C(y) = 1] - \Pr_{z \leftarrow D}[C(z) = 1] \right| > 1/m^d$$

We can transform this into a predictor P of size $m^e + O(m)$:

$$\Pr_{z \leftarrow D}[P(b_{m-1}, \ldots, b_{m-i}) = b_{m-i+1}] > 1/2 + 1/m^{d+1}$$

We will use this to devise a procedure to compute $h_t(f_t^{-1}(y))$

set $y \leftarrow y_{m-i}$ $\qquad b_{m-i} = h_t(y_{m-i})$

Compute y_j for $j = m-i+1, \ldots, m-1$ as above.
$b_j = h_t(y_j)$.

Evaluate $P(\underline{b_{m-1}, \ldots, b_{m-i}})$

$\hookrightarrow$ distributed according to the prefix of the generator.

$$\Pr_y \left[P(b_{m-1}, \ldots, b_{m-i}) = b_{m-i+1} \right] > \frac{1}{2} + 1/m^{d+1}$$

initial y chosen uniformly

b_{m-i+1} is $h_t(y_{m-i+1}) = h_t(f^{-1}(y_{m-i})) = h_t(f^{-1}(y))$

This is a family of circuits that computes $h_t(f^{-1}(y))$ from y with success greater than $1/2 + 1/poly(m)$.

$\Rightarrow$ Contradiction. //

To get BPP = P need $t = O(\log m)$
(need to run over all seeds of length t $\rightarrow 2^t$).

BMY building block one-way $f: \{0,1\}^t \rightarrow \{0,1\}^t$ required to fool circuits of size m^e for all e.
But with these settings, f can be inverted by brute force!

BMY generator:
one generator fooling <u>all</u> poly-sized circuits
for all polynomials.
One-way permutation is a hard function.
implies hard function in NP ∩ co-NP.
Computing $f^{-1}(x)$ is hard
but can show $f(y)=x$: y is witness

Nisan-Wigderson generator:
for each poly-size bound, a different generator.
hard function can be in $E = \bigcup_k DTIME(2^{kn})$

this allows them to get $t = O(\log m)$.

Hardness assumption still average case.
Can be made worst case using error-correcting codes.