PRGs for SPACE BDD COMPUTATION

Agenda
1. Nisan's PRG
2. NZ PRG, if time.

NISAN'S PRG

- Before, we had the (lofty) goal of derandomizing BPP, and we needed to make some assumption.
- Today, we will lower our sights to derandomizing BPL

Thm (Nisan) Let $S: \mathbb{N} \to \mathbb{N}$. (Think $S(n) = \log(n)$), let $\ell = s(n)\log(n)$.
There is a map $G: \{0,1\}^\ell \to \{0,1\}^n$, so that

- G is computable in $O(\ell)$ space, $\text{poly}(n)$ time
- G is a $2^{-S(n)}$ – PRG against SPACE$(S(n))$:

$$\forall\ g \text{ which use SPACE}(S(n)),\ g: \{0,1\}^n \to \{0,1\}.$$
$$\left| \mathbb{P}\{g(U_n) = 1\} - \mathbb{P}\{g(G(U_\ell)) = 1\} \right| \leq 1/2^{S(n)}$$

So if $S(n) = \log(n)$, we have PRGs that "fool" L.

Cor Any language in BPL can be decided in space $O(\log^2(n))$ and time $n^{O(\log(n))}$.

Pf If A is a BPL algorithm, using $\text{poly}(n)$ random bits, then run $A(x;\cdot)$ w/ randomness $G(r)$, where r is $O(\log^2(n))$ random bits.
Space is $\log^2(n)$ ($\log^2(n)$ for G, and $+\log(n)$ for A), time is $n^{O(\log(n))}$ (running time of A)

So $BPL \subseteq SPACE(n^2)$.

That may not be so impressive: Savitch's thm says

$$RL \subseteq NL \subseteq \left(\text{space } \log^2(n) \text{ and time } \text{poly}(n)\right)$$

(RL ⊆ NL: obvious; NL ⊆ (space $\log^2(n)$ and time poly(n)): Savitch; poly(n) is better than $n^{O(\log(n))}$)

BUT

(a) This actually gives us a recipe for derandomization — (using the BPL alg A in a black box way)

(b) Turns out you can use these techniques to show — and — $BPL \subseteq (\text{space } \log^2(n), \text{time } \text{poly}(n))$; $BPL \subseteq \text{SPACE}(\log^{1.5}(n))$

OPEN QUESTION: $BPL = L$?? We think so.

Pf of Nisan's Thm.

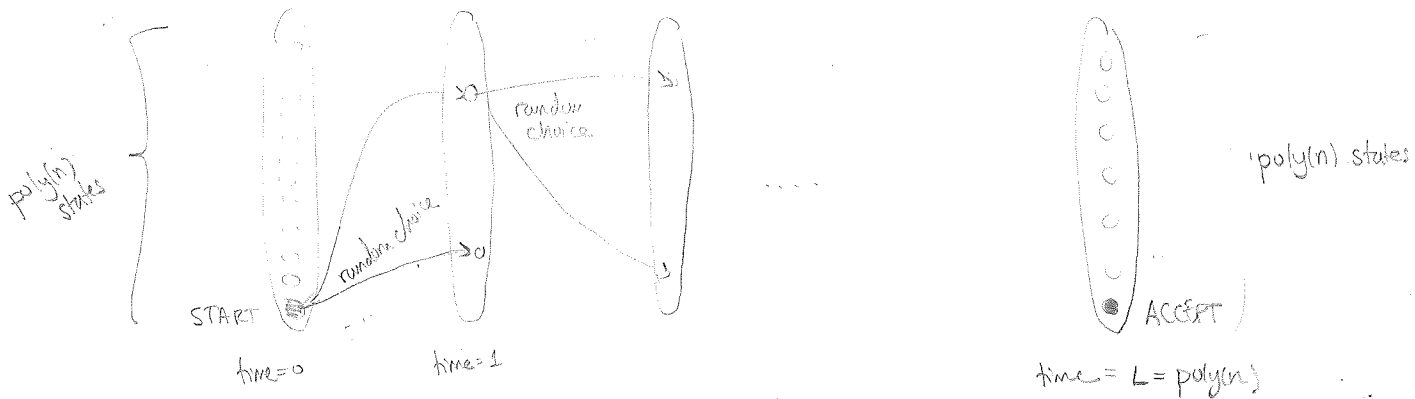
SET $S = \log(n)$

← This proof due to [Impagliazzo, Nisan, Wigderson 1995] (original proof uses hash fns to build an extractor in a less black-box way)

Tools.

(1) Extractors: recall $EXT : \{0,1\}^{n+d} \to \{0,1\}^{m}$ is (k,ϵ)-extractor if $\forall$ k-srces X, $\Delta(EXT(X, U_d), U_m) \le \epsilon$

(2) Configuration graph. Let's think of a randomized computation as progressing in LAYERS.



Well actually be kind to our logspace machine: between random choices, it can do ANY COMPUTATION IT WANTS.

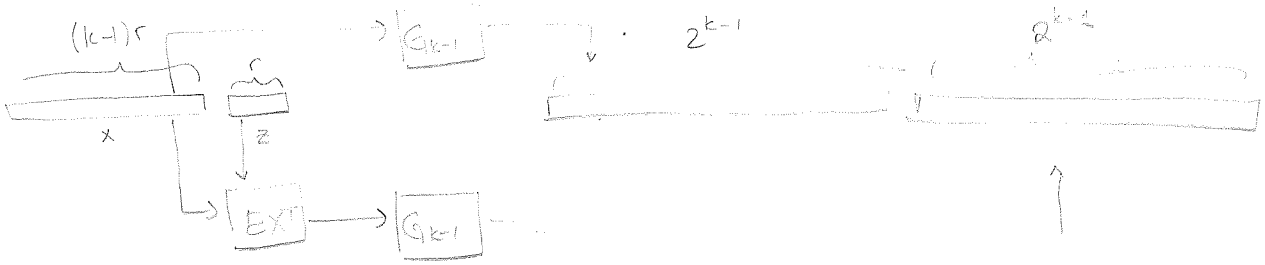
Idea: Say we have a PRG $G_{k-1} : \{0,1\}^{(k-1)r} \longrightarrow \{0,1\}^{2^{k-1}}$

and an extractor $EXT : \{0,1\}^{(k-1)r} \times \{0,1\}^{r} \longrightarrow \{0,1\}^{(k-1)r}$

Consider $G_k : \{0,1\}^{kr} \longrightarrow \{0,1\}^{2^k}$ given by

$$G_k(x, z) = G_{k-1}(x) \circ G_{k-1}(EXT(x, z))$$

(concatenation)



Why might this work?

The reason for concern is that, given $G_{k-1}(x)$, the output $EXT(x,z)$ is not uniformly random, and so the second half might depend in some bad way on the first.

However, if the algorithm using these random bits has limited space, it can't remember all of $G_{k-1}(x)$. And, the distribution on X, conditional on what the alg can remember, is a weak source.

Lemma "Recycling random bits"

Let $f : \{0,1\}^n \to \{0,1\}^s$ be any function ($f(x)$ will be all the adv knows about x).

Let $EXT : \{0,1\}^n \times \{0,1\}^r \to \{0,1\}^m$ be a

$(n - (s+1) - \log(1/\varepsilon), \varepsilon/2)$ - extractor.

← if s is really small, adv doesn't know much about x, and we can get away w/ a pretty lossy extractor (which only works on nice srcs)

Then $\Delta\left(f(X) \circ U_m, f(X) \circ EXT(X, U_t) \right) < \varepsilon$

when $X \sim U_n$ (and the other uniform dists that appear are independent).

Proof For $v \in \{0,1\}^s$ (output of f), let X_v be uniform on $f^{-1}(v)$.

Then

$$\Delta\left((f(X) \circ U_m), f(X) \circ EXT(X, U_r) \right)$$

$$= \frac{1}{2} \sum_{v,w} \left| \mathbb{P}\{ f(X) = v \wedge U_m = w \} - \mathbb{P}\{ f(X) = v \wedge EXT(X, U_r) = w \} \right|$$

$$= \frac{1}{2} \sum_{v,w} \mathbb{P}\{f(X) = v\} \left| \mathbb{P}\{U_m = w\} - \mathbb{P}\{ EXT(X, U_r) = w \mid f(X) = v \} \right|$$

$$= \sum_v \mathbb{P}\{f(X) = v\} \cdot \Delta(U_m, EXT(X_v, U_r)).$$

Let $V = \{v : \mathbb{P}\{f(X) = v\} \geq \varepsilon / 2^{s+1}\}$. So if $v \in V$, $|supp(X_v)| \geq \frac{2^n \varepsilon}{2^{s+1}} = 2^{n-(s+1)-\log(1/\varepsilon)}$.

$$\leq \sum_{v \in V} \mathbb{P}\{f(X) = v\} \cdot \frac{\varepsilon}{2} + \sum_{v \notin V} \frac{\varepsilon}{2^{s+1}}$$

$$\leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon \quad \blacksquare$$

Now, let's use this to make that idea work.

Define $G_k : \{0,1\}^{kr} \to \{0,1\}^{2^k}$ as before:

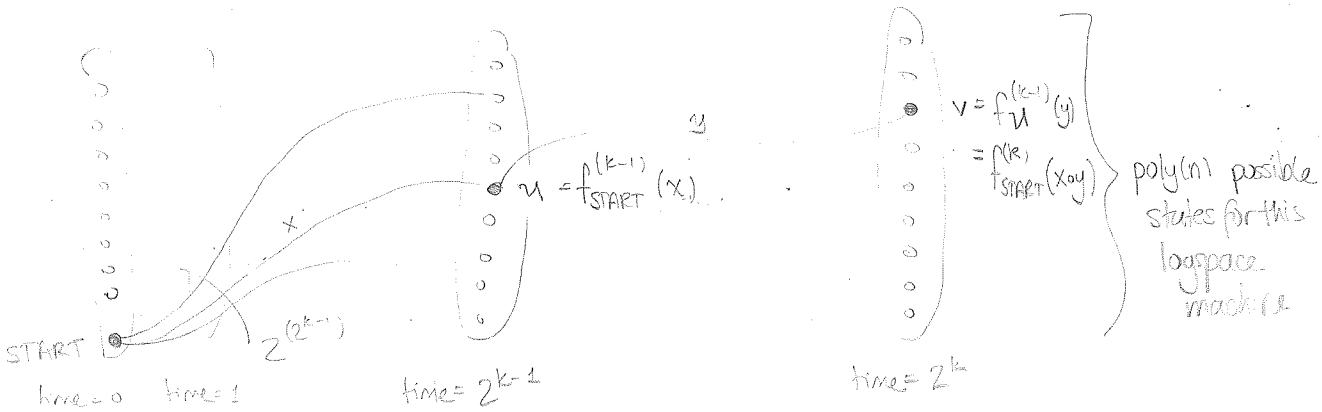
$$G_1(z) = z_1, z_2 \qquad \text{(first 2 bits of } z\text{)}$$

$$G_k(x, z) = G_{k-1}(x) \circ G_{k-1}(\mathrm{EXT}(x, z))$$

($x \in \{0,1\}^{(k-1)r}$, $z \in \{0,1\}^r$)

We'll argue by induction on k. Some notation:

Imagine we are 2^k steps deep into the computation, and we have



Let $f_w^{(k-1)}(x)$ be the state you get if you run M for 2^{k-1} steps from state w, with random string $x \in \{0,1\}^{2^{k-1}}$

Fix $\varepsilon > 0$, TBD.

INDUCTIVE HYPOTHESIS:

$$\forall w, \quad \Delta\left(f_w^{(k-1)}(U_{2^{k-1}}),\ f_w^{(k-1)}(G_{k-1}(U_{(k-1)r}))\right) \leq 3^{k-1}\varepsilon$$

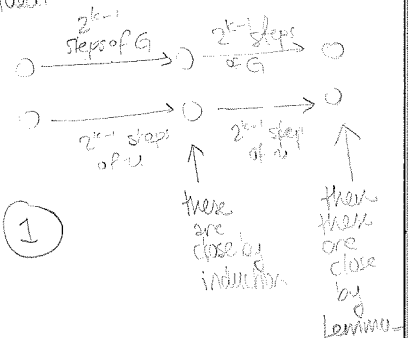
Base case is easy, since $G_1(U_r) = U_{2^1}$ ↘

Now let's look at the next step. WTS, $\forall w$,

$$\Delta\left(f_w^{(k)}(U_{2^k}) \;,\; f_w^{(k)}(G_k(U_{kr})) \right) \le 3^k\varepsilon \qquad (*)$$

We'll use the triangle inequality to step through a few distributions.

Idea:



$$f_w^{(k)}(U_{2^k}) = f_w^{(k)}\left(\underbrace{U_{2^{k-1}} \circ U_{2^{k-1}}}_{\mathcal{D}_1} \right) \quad (1)$$

$$f_w^{(k)}\left(U_{2^{k-1}} \circ \underbrace{G_{k-1}(U_{(k-1)r})}_{\mathcal{D}_2} \right) \quad (2)$$

independent U's

$$f_w^{(k)}\left(\underbrace{G_{k-1}(U'_{(k-1)r}) \circ G_{k-1}(U_{(k-1)r})}_{\mathcal{D}_4} \right) \quad (3)$$

$\mathcal{D}_3$

$$f_w^{(k)}(G_k(U_{kr})) = f_w^{(k)}\left(G_{k-1}(U'_{(k-1)r}) \circ G_{k-1}\left(\mathrm{EXT}(U'_{(k-1)r}, U_r)\right) \right)$$

U_r: independent

$U'_{(k-1)r}$: these are identical

We'll show that each of these steps is small.

(1) $\Delta\left(f_w^k(\mathcal{D}_1), f_w^k(\mathcal{D}_2)\right)$

$$= \frac{1}{2}\sum_{\text{states } v} \left| \mathbb{P}_{\mathcal{D}_1}(w \to v) - \mathbb{P}_{\mathcal{D}_2}(w \to v) \right|$$

$$= \frac{1}{2}\sum_{\text{states } v} \left| \sum_{\text{states } u} \mathbb{P}_{U_{2^{k-1}}}(w \to u) \cdot \mathbb{P}_{U_{2^{k-1}}}(u \to v) - \mathbb{P}_{U_{2^{k-1}}}(w \to u) \cdot \mathbb{P}_{G_{k-1}(U_{(k-1)r})}^{(u \to v)} \right|$$

$$= \frac{1}{2}\sum_{\text{states } v} \left| \sum_{\text{states } u} \mathbb{P}_{U_{2^{k-1}}}(w \to u) \left(\mathbb{P}_{U_{2^{k-1}}}(u \to v) - \mathbb{P}_{G_{k-1}(U_{(k-1)r})}^{(u \to v)} \right) \right|$$

$$\le \sum_{\text{States } u} \underbrace{\mathbb{P}_{U_{2^{k-1}}}(w \to u)}_{\text{add to 1}} \; \frac{1}{2}\sum_v \left| \mathbb{P}_{U_{2^{k-1}}}(u \to v) - \mathbb{P}_{G_{k-1}(U_{(k-1)r})}^{(u \to v)} \right|$$

$\le 3^{k-1}\varepsilon$ for all u, by induction

$$\le 3^{k-1}\varepsilon$$

Now the difference (2) follows in the same way, $\leq 3^{k-1}\varepsilon$.

Finally (3):

$$f_W^k\left(G_{k-1}(U'_{(k-1)r}) \circ G_{k-1}(U_{(k-1)r})\right)$$

vs.

$$f_W^k\left(G_{k-1}(U'_{(k-1)r}) \circ G_{k-1}(\mathrm{EXT}_{k-1}(U_{(k-1)r}, U_r))\right)$$

(don't write again)

Apply the lemma with

$$f_W^k \circ G_{k-1} : \{0,1\}^{(k-1)r} \longrightarrow \{0,1\}^{\Theta(s)}$$ ← this will be log(#states per buyer) — let's say it's $100\log(n)$

Since EXT is a $\left((k-1)r - (100\log(n)+1) - \log(1/\varepsilon), \varepsilon\right)$-extractor, we have

$$\Delta\Big(f(G_{k-1}(U'_{(k-1)r})) \circ U_{(k-1)r},\ f(G_{k-1}(U'_{(k-1)r})) \circ \mathrm{EXT}(U'_{(k-1)r}, U_r)\Big) \leq \varepsilon$$

NOTE: For any deterministic g, $\Delta(g(X), g(Y)) \leq \Delta(X,Y)$:

$$\sum_x \left|\mathbb{P}(g(X)=x) - \mathbb{P}(g(Y)=x)\right| = \sum_x \left|\sum_{z:\,g(z)=x} \mathbb{P}(X=z) - \mathbb{P}(Y=z)\right|$$

$$\leq \sum_{x,\,z:\,g(z)=x} \left|\mathbb{P}(X=z) - \mathbb{P}(Y=z)\right| = \Delta(X,Y).$$

so apply G_{k-1} to the second bunch of bits,

$$\Delta\Big(f(G_{k-1}(U'_{(k-1)r})) \circ G_{k-1}(U_{(k-1)r}),\ f(G_{k-1}(U'_{(k-1)r})) \circ G_{k-1}(\mathrm{EXT}(U_{(k-1)r}, U_r))\Big) \leq \varepsilon$$

so, we proved (*) with $\varepsilon_k = 2\cdot 3^{k-1}\varepsilon + \varepsilon \leq 3^k\varepsilon$, as desired.

Now let's see what we've managed, parameter-wise.

Say $EXT_{k-1} : \{0,1\}^{(k-1)r} \times \{0,1\}^{r} \longrightarrow \{0,1\}^{(k-1)r}$ is a

$$\left(\underbrace{(k-1)r - (100\log(n)+1) - \log(1/\varepsilon)}_{\text{let's say } (k-1)r - 200\log(n)} \;,\; \varepsilon \right) \text{-extractor,}$$

and say $\log(1/\varepsilon) \leq \log(n)$

We can get these with seed length $r = O(\log(n))$, as we saw last time.

[For example, our expander construction, using an expander of degree 2^r and $2^{r(k-1)}$ vertices will get this for us.]

Say the total computation has length $= \text{poly}(n)$, so choose $k = \lg(L)$, and $\varepsilon \leq \left(\frac{1}{3^{\lg(L)}}\right)^2$

Now, by induction we conclude that (so $\log(\frac{1}{\varepsilon}) \approx \log(L) \approx \log(n)$)

$$\left| \mathbb{P}\left\{ f^{L}_{START}(U_{rk}) = \text{accept} \right\} - \mathbb{P}\left\{ f^{L'}_{START}(G_k(U_{kr})) = \text{accept} \right\} \right| \leq 3^k \varepsilon \leq \frac{1}{3^{\lg(L)}}$$

aka, if g is the fn computed by M, then

$$\left| \mathbb{P}\{ g(U_{rk}) = 1 \} - \mathbb{P}\{ g(G_k(U_{rk})) = 1 \} \right| \leq 1/\text{poly}(n)$$

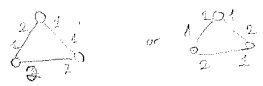
So that proves the thm.

Applications of Nisan's PRG.

1) Universal Traversal sequences: $w \in [d]^r$ is a UTS if

for <u>all</u> d-regular, n-node connected, undirected graphs G, and any labeling of the edges, and any starting vertex, following w will hit all the vertices.

eg, $d=2$, $n=3$,



- Probabilistic method shows these exist w/ poly(n) length. (Just choose one uniformly at random, which takes poly(n) random bits and $\log(n)$ space) 122 should do it.
- Via Nisan's PRG, we can use only $O(\log^2(n))$ random bits instead. Exhausting over all possible $2^{O(\log^2(n))}$ sequences and gluing them together gives an explicit sequence of len $\text{poly}(n) \cdot n^{O(\log(n))} = n^{O(\log(n))}$

This is the best known.

2) Choosing random primes.

- You can choose a random n-bit prime w/ space $O(n)$, so you can also do it w/ $O(n\log(n))$ random bits.

and randomness something like n^2; or n^3 if I use randomized primality testing. (Alg is, pick rand. n-bit #s, probably 1 is prime).

3) Random expander walks.

- If I want to walk on an expander for a long time, say poly(n) steps, then this takes $\log(n)$ space and poly(n) random bits.

w/ Nisan I can do it in $\log^2(n)$ random bits.

4) Amplification of BPP.

Say I have a BPP alg w/ r bits of randomness. I reduce the error by a length t walk on an expander w/ 2^r vertices. This takes space r and randomness $r+t$, to get failure prob. 2^{-t}.

w/ Nisan, I can instead take $r\log(r+t) = O(r\log(t))$ to get failure prob 2^{-t}.

If $t \gg r$, say

Question: Nisan's PRG says that if an alg uses space $S(n) \geq \log(n)$, we can stretch

$O(S(n)\log(n))$ random bits $\longrightarrow$ $2^{S(n)}$ random bits.

But what if we don't want $2^{S(n)}$ random bits?
Eg, what if we want to take a walk on an expander of len $\text{polylog}(n)$?
This org still gives

$O(S(n)\log(n))$ random bits $\longrightarrow$ $\text{polylog}(n)$ random bits.

We'd really like $S(n)$ here. That way, if $S(n)=\log(n)$, we can completely derandomize in poly time.

Can we do better?

Thm (Nisan–Zuckerman) For any $S(n) \geq \log(n)$,

$\exists\ G: \{0,1\}^{O(S)} \to \{0,1\}^{\text{poly}(S)}$ s.t. for all fns g computable in space S (and time $T \geq n$)

$$\left| \mathbb{P}\{g(G(U))=1\} - \mathbb{P}\{g(U)=1\} \right| \leq 1/\text{poly}(S)$$

(and G runs in time $T+\text{poly}(S)$ and space S.)

Cor Any alg running in time T and space S using only $\text{poly}(S)$ random bits can be derandomized to get an alg that runs in time $2^S(T+\text{poly}(S))$ w/ space S.

aka, (BPL w/ very few random bits) $\subseteq$ L

For the applications we saw earlier, we can

2) Choose a random prime w/ $O(n)$ random bits instead of $O(n\log(n))$

3) Take a walk of len $\log^2(n)$ on an expander using only $\log(n)$ random bits.

Nisan – Zuckerman PRG: main idea

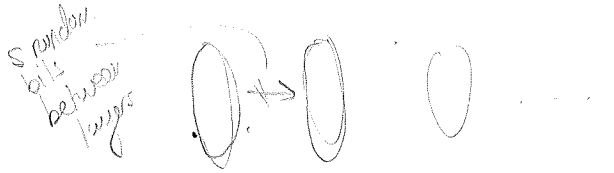
To get the idea, let's just stretch $O(S)$ bits to $O(S^2/\log(S))$ bits.
You can then build the final PRG out of this.

Construction: Let $EXT: \{0,1\}^{4S} \times \{0,1\}^{d} \to \{0,1\}^{S}$ be a $(2S,\epsilon)$ extractor, $d = O(\log(S))$

Define $G: \{0,1\}^{4S} \times (\{0,1\}^{d})^{L} \to (\{0,1\}^{S})^{L}$ by

$G(x, y_1, \dots, y_L) = EXT(x, y_1) \circ EXT(x, y_2) \circ \dots \circ EXT(x, y_L)$, where $L = S/\log(S)$

Proof that this works: As before, consider a bunch of layers.



$L = \frac{S}{\log(S)}$ layers

Let $W_i = EXT(X, Y_1) \circ \dots \circ EXT(X, Y_i)$ (random variable, where X, Y_j's are uniform)

Inductive hypothesis:

$$\Delta\left(f_{START}^{(i-1)}(W_{i-1}),\ f_{START}^{(i-1)}(U_{(i-1)S}) \right) \le (i-1)\epsilon$$

($f_{START}^{(i-1)}$: now this means, stuck in i^{th} layer, not $(i-1)^{th}$ layer; $\epsilon = 1/\text{poly}(S)$)

- Base case is clear ✓
- Induction proceeds as before. Informally, we consider the two RVs

START $\xrightarrow{W_{i-1}}$ ($(i-1)$'st layer) $\xrightarrow{EXT(X, Y_i)}$ $f_{START}^{i}(W_i)$ (uniform on $(f_{START}^{(i-1)})^{-1}(\cdot)$)

vs

START $\xrightarrow{U_{(i-1)S}}$ V $\xrightarrow{U_{iS}}$ $f_{START}^{i}(U_{iS})$

W_{i-1} ↑ these guys are close by induction

U_{iS} ↑ these are close by recycling randomness lemma

and then use Δ-inequality.

At the end of the day, get error $1/\text{poly}(S)$, seed length $4S + dL = 4S + O(\log(S) \cdot \frac{S}{\log(S)}) = O(S)$