

# Nisan's Pseudorandom Generator for RL

$$\text{BPL} \subseteq \text{SC} = \text{DTISP}(\text{poly}(n), \log^2(n))$$

---

Soham Chatterjee

December 2, 2025

Pseudorandomness Course (CSS.413.1) Presentation, STCS

# Pseudorandom Generator

## Definition (Pseudorandom Generator)

A map  $\mathcal{G} : \{0, 1\}^l \rightarrow \{0, 1\}^n$ , where  $n \geq l$  is called a PRG for a class  $\mathcal{C}$  with a parameter  $\epsilon > 0$  if for any  $f \in \mathcal{C}$ ,

$$\left| \mathbb{P}_{y \in \{0,1\}^n} [f(y) = 1] - \mathbb{P}_{x \in \{0,1\}^l} [f(\mathcal{G}(x)) = 1] \right| \leq \epsilon$$

- Here  $l$  is called the seed-length of the PRG.
- $n - l$  is called the stretch of the PRG.
- We call  $\mathcal{G}$ ,  $\epsilon$ -fools  $\mathcal{C}$ .
- Typically, we want  $n \gg l$  and  $\mathcal{G}$  to be efficiently computable.

Let  $T$  be a BPL machine which uses  $n^c$  random bits on inputs of length  $n$  and runs in polynomial time and uses  $S = O(\log n)$  space.

# Finite State Automata

Let  $T$  be a BPL machine which uses  $n^c$  random bits on inputs of length  $n$  and runs in polynomial time and uses  $S = O(\log n)$  space.

- There are at most  $2^{O(S)} = \text{poly}(n)$  configurations of  $T$ .

# Finite State Automata

Let  $T$  be a BPL machine which uses  $n^c$  random bits on inputs of length  $n$  and runs in polynomial time and uses  $S = O(\log n)$  space.

- There are at most  $2^{O(S)} = \text{poly}(n)$  configurations of  $T$ .
- Each random bit is used to make a transition between two configurations.

# Finite State Automata

Let  $T$  be a BPL machine which uses  $n^c$  random bits on inputs of length  $n$  and runs in polynomial time and uses  $S = O(\log n)$  space.

- There are at most  $2^{O(S)} = \text{poly}(n)$  configurations of  $T$ .
- Each random bit is used to make a transition between two configurations.
- The starting configuration is fixed for any input.

# Finite State Automata

Let  $T$  be a BPL machine which uses  $n^c$  random bits on inputs of length  $n$  and runs in polynomial time and uses  $S = O(\log n)$  space.

- There are at most  $2^{O(S)} = \text{poly}(n)$  configurations of  $T$ .
- Each random bit is used to make a transition between two configurations.
- The starting configuration is fixed for any input.
- Input  $x$  is accepted if  $T$  reaches a state representing acceptance.

# Finite State Automata

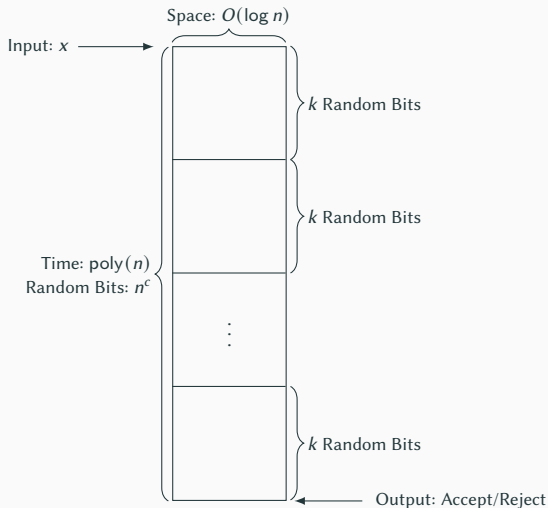
Let  $T$  be a BPL machine which uses  $n^c$  random bits on inputs of length  $n$  and runs in polynomial time and uses  $S = O(\log n)$  space.

- There are at most  $2^{O(S)} = \text{poly}(n)$  configurations of  $T$ .
- Each random bit is used to make a transition between two configurations.
- The starting configuration is fixed for any input.
- Input  $x$  is accepted if  $T$  reaches a state representing acceptance.

Therefore the configuration graph of  $T$  on input  $x$  represents a finite state automata with  $N = \text{poly}(n)$  states.



# Computational Tableau of BPL machine



**Figure 1:** Computational Tableau of BPL machine  $T$ .

## Dividing BPL Computation into Blocks

- Let the BPL machine  $T$  run in time  $n^c$  using  $n^{c'}$  random bits and  $O(\log n)$  space on input  $x$  of length  $n$ .
- Let  $k \ll n^c$  be a parameter to be fixed later.
- Divide the computation of  $T$  into  $t = n^c/k$  blocks, where each block uses  $k$  random bits.
- We can treat each block as a separate BPL machine  $T_i$  (in some sense) which takes input as the final configuration of  $T_{i-1}$  and  $k$  random bits.

# FSA for Computation Blocks

- We can think of  $T$  as a finite state automata with  $N$  states.
- Each state makes  $2^k$  transitions where each transition corresponds to a choice of  $k$  random bits.

# FSA for Computation Blocks

- We can think of  $T$  as a finite state automata with  $N$  states.
- Each state makes  $2^k$  transitions where each transition corresponds to a choice of  $k$  random bits.
- Let  $Q$  be this automata. For transition we write  $Q(i; r) = j$  for any  $r \in \{0, 1\}^k$  if  $Q$  goes from state  $i$  to state  $j$  when fed  $r$ .

# FSA for Computation Blocks

- We can think of  $T$  as a finite state automata with  $N$  states.
- Each state makes  $2^k$  transitions where each transition corresponds to a choice of  $k$  random bits.
- Let  $Q$  be this automata. For transition we write  $Q(i; r) = j$  for any  $r \in \{0, 1\}^k$  if  $Q$  goes from state  $i$  to state  $j$  when fed  $r$ .
- Let  $M$  be the matrix where  $M[i, j] = \mathbb{P}_{r \in \{0, 1\}^k} [Q(i; r) = j]$

# FSA for Computation Blocks

- We can think of  $T$  as a finite state automata with  $N$  states.
- Each state makes  $2^k$  transitions where each transition corresponds to a choice of  $k$  random bits.
- Let  $Q$  be this automata. For transition we write  $Q(i; r) = j$  for any  $r \in \{0, 1\}^k$  if  $Q$  goes from state  $i$  to state  $j$  when fed  $r$ .
- Let  $M$  be the matrix where  $M[i, j] = \mathbb{P}_{r \in \{0, 1\}^k} [Q(i; r) = j]$

$$\mathbb{P}_{r \in \{0, 1\}^{k \times t}} [T(x, r) = \text{Acc}] = \sum_{j: \text{Accepting state}} M^t[1, j]$$

**Goal:** Approximate  $M^t$  using PRG.

# Approximate Automata Matrix

Suppose we have a pseudorandom generator  $\mathcal{G} : \{0, 1\}^k \rightarrow \{0, 1\}^{k \cdot t}$ . Let  $Q$  be a finite state automata with  $N$  states and its matrix be  $M$  as defined above.

- From definition of  $M$ ,  $M^t[i, j] = \mathbb{P}_{r_1, \dots, r_t \in \{0, 1\}^k} [Q(i; r_1 \dots r_t) = j]$
- Using  $\mathcal{G}$ , let  $M_{\mathcal{G}}[i, j] = \mathbb{P}_{r \in \{0, 1\}^k} [Q(i; \mathcal{G}(r)) = j]$
- Want to construct  $\mathcal{G}$  such that

$$\|M^t - M_{\mathcal{G}}\| < \epsilon$$

for small  $\epsilon$  with high probability.

# Matrix Norm

For any vector  $v \in \mathbb{R}^N$ , define  $\|v\| = \sum_{i \in [N]} |v(i)|$ . Then for any matrix  $M \in \mathbb{R}^{N \times N}$  define

$$\|M\| = \sup_{0 \neq v \in \mathbb{R}^N} \frac{\|Mv\|}{\|v\|}$$



# Matrix Norm

For any vector  $v \in \mathbb{R}^N$ , define  $\|v\| = \sum_{i \in [N]} |v(i)|$ . Then for any matrix  $M \in \mathbb{R}^{N \times N}$  define

$$\|M\| = \sup_{0 \neq v \in \mathbb{R}^N} \frac{\|Mv\|}{\|v\|}$$

## Properties:

- $\|M\| = \max_{i \in [N]} \sum_{j \in [N]} |M[i, j]|$
- $\|M + N\| \leq \|M\| + \|N\|$
- $\|MN\| \leq \|M\| \cdot \|N\|$

# Universal Hash Family

## Definition (Universal Hash Family (Carter-Wegman))

$\mathcal{H} = \{h : \{0, 1\}^k \rightarrow \{0, 1\}^m\}$  is a *universal hash family* if for any  $x_1 \neq x_2 \in \{0, 1\}^k$  and  $y_1, y_2 \in \{0, 1\}^m$ ,

- $\mathbb{P}_{h \in \mathcal{H}} [h(x_1) = y_1] = \frac{1}{2^m}$
- $\mathbb{P}_{h \in \mathcal{H}} [h(x_1) = y_1 \wedge h(x_2) = y_2] = \frac{1}{2^{2m}}$

# Universal Hash Family

## Definition (Universal Hash Family (Carter-Wegman))

$\mathcal{H} = \{h : \{0, 1\}^k \rightarrow \{0, 1\}^m\}$  is a *universal hash family* if for any  $x_1 \neq x_2 \in \{0, 1\}^k$  and  $y_1, y_2 \in \{0, 1\}^m$ ,

- $\mathbb{P}_{h \in \mathcal{H}} [h(x_1) = y_1] = \frac{1}{2^m}$
- $\mathbb{P}_{h \in \mathcal{H}} [h(x_1) = y_1 \wedge h(x_2) = y_2] = \frac{1}{2^{2m}}$

- For our purpose, we have  $k = m$ .

# Universal Hash Family

## Definition (Universal Hash Family (Carter-Wegman))

$\mathcal{H} = \{h : \{0, 1\}^k \rightarrow \{0, 1\}^m\}$  is a *universal hash family* if for any  $x_1 \neq x_2 \in \{0, 1\}^k$  and  $y_1, y_2 \in \{0, 1\}^m$ ,

- $\mathbb{P}_{h \in \mathcal{H}} [h(x_1) = y_1] = \frac{1}{2^m}$
- $\mathbb{P}_{h \in \mathcal{H}} [h(x_1) = y_1 \wedge h(x_2) = y_2] = \frac{1}{2^{2m}}$

- For our purpose, we have  $k = m$ .
- We can construct such a family with  $|\mathcal{H}| = 2^{O(k)}$  where

$$\mathcal{H} = \{a \cdot x + b \mid a, b \in \{0, 1\}^k\}$$

over  $GF(2^k)$ .

# Property of Universal Hash Family

## Definition $((\epsilon, A, B)$ -good hash function)

Let  $A \subseteq \{0, 1\}^k$ ,  $B \subseteq \{0, 1\}^m$ ,  $\epsilon > 0$ ,  $h : \{0, 1\}^k \rightarrow \{0, 1\}^m$  is said to be  $(\epsilon, A, B)$ -good if

$$\left| \mathbb{P}_{x \in \{0, 1\}^k} [x \in A \wedge h(x) \in B] - \alpha \cdot \beta \right| \leq \epsilon$$

where  $\alpha = \frac{|A|}{2^k}$  and  $\beta = \frac{|B|}{2^m}$ .

# Property of Universal Hash Family

## Definition (( $\epsilon$ , $A$ , $B$ )-good hash function)

Let  $A \subseteq \{0, 1\}^k$ ,  $B \subseteq \{0, 1\}^m$ ,  $\epsilon > 0$ ,  $h : \{0, 1\}^k \rightarrow \{0, 1\}^m$  is said to be  $(\epsilon, A, B)$ -good if

$$\left| \mathbb{P}_{x \in \{0, 1\}^k} [x \in A \wedge h(x) \in B] - \alpha \cdot \beta \right| \leq \epsilon$$

where  $\alpha = \frac{|A|}{2^k}$  and  $\beta = \frac{|B|}{2^m}$ .

## Lemma

If  $\mathcal{H}$  is a universal hash family, then for any  $A \subseteq \{0, 1\}^k$ ,  $B \subseteq \{0, 1\}^m$ ,  $\epsilon > 0$ ,

$$\mathbb{P}_{h \in \mathcal{H}} [h \text{ is not } (\epsilon, A, B)\text{-good}] \leq \frac{\alpha \cdot \beta}{2^k \epsilon^2}$$

# Property of Universal Hash Family

## Definition $((\epsilon, A, B)$ -good hash function)

Let  $A \subseteq \{0, 1\}^k$ ,  $B \subseteq \{0, 1\}^m$ ,  $\epsilon > 0$ ,  $h : \{0, 1\}^k \rightarrow \{0, 1\}^m$  is said to be  $(\epsilon, A, B)$ -good if

$$\left| \mathbb{P}_{x \in \{0, 1\}^k} [x \in A \wedge h(x) \in B] - \alpha \cdot \beta \right| \leq \epsilon$$

where  $\alpha = \frac{|A|}{2^k}$  and  $\beta = \frac{|B|}{2^m}$ .

## Lemma

If  $\mathcal{H}$  is a universal hash family, then for any  $A \subseteq \{0, 1\}^k$ ,  $B \subseteq \{0, 1\}^m$ ,  $\epsilon > 0$ ,

$$\mathbb{P}_{h \in \mathcal{H}} [h \text{ is not } (\epsilon, A, B)\text{-good}] \leq \frac{\alpha \cdot \beta}{2^k \epsilon^2}$$

For  $k = m$ , by union bound an uniformly random  $h$  is  $\left(\frac{\epsilon}{k^2}, A, B\right)$ -good for all  $A, B$ .

## Nisan's Generator

Let  $\mathcal{H}$  be an universal hash family from  $\{0, 1\}^k$  to  $\{0, 1\}^k$ . For any integer  $m \geq 0$  define the function  $\mathcal{G}_m: \{0, 1\}^k \times \mathcal{H}^m \rightarrow \{0, 1\}^{k \cdot 2^m}$  recursively as follows:



# Nisan's Generator

Let  $\mathcal{H}$  be an universal hash family from  $\{0, 1\}^k$  to  $\{0, 1\}^k$ . For any integer  $m \geq 0$  define the function  $\mathcal{G}_m: \{0, 1\}^k \times \mathcal{H}^m \rightarrow \{0, 1\}^{k \cdot 2^m}$  recursively as follows:

- $\mathcal{G}_0(x) = x$
- $\mathcal{G}_m(x, h_1, \dots, h_m) = (\mathcal{G}_{m-1}(x, h_1, \dots, h_{m-1}), \mathcal{G}_{m-1}(h_m(x), h_1, \dots, h_{m-1}))$

# Nisan's Generator

Let  $\mathcal{H}$  be an universal hash family from  $\{0, 1\}^k$  to  $\{0, 1\}^k$ . For any integer  $m \geq 0$  define the function  $\mathcal{G}_m: \{0, 1\}^k \times \mathcal{H}^m \rightarrow \{0, 1\}^{k \cdot 2^m}$  recursively as follows:

- $\mathcal{G}_0(x) = x$
- $\mathcal{G}_m(x, h_1, \dots, h_m) = (\mathcal{G}_{m-1}(x, h_1, \dots, h_{m-1}), \mathcal{G}_{m-1}(h_m(x), h_1, \dots, h_{m-1}))$

For example:

$$\mathcal{G}_1(x, h) = (x, h(x)), \quad \mathcal{G}_2(x, h_1, h_2) = (x, h_1(x), h_2(x), h_1 \cdot h_2(x))$$

# Nisan's Generator

Let  $\mathcal{H}$  be an universal hash family from  $\{0, 1\}^k$  to  $\{0, 1\}^k$ . For any integer  $m \geq 0$  define the function  $\mathcal{G}_m: \{0, 1\}^k \times \mathcal{H}^m \rightarrow \{0, 1\}^{k \cdot 2^m}$  recursively as follows:

- $\mathcal{G}_0(x) = x$
- $\mathcal{G}_m(x, h_1, \dots, h_m) = (\mathcal{G}_{m-1}(x, h_1, \dots, h_{m-1}), \mathcal{G}_{m-1}(h_m(x), h_1, \dots, h_{m-1}))$

For example:

$$\mathcal{G}_1(x, h) = (x, h(x)), \quad \mathcal{G}_2(x, h_1, h_2) = (x, h_1(x), h_2(x), h_1 \cdot h_2(x))$$

$$\mathcal{G}_3(x, h_1, h_2, h_3) = (x, h_1(x), h_2(x), h_1 \cdot h_2(x), h_3(x), h_1 \cdot h_3(x), h_2 \cdot h_3(x), h_1 \cdot h_2 \cdot h_3(x))$$

# Nisan's Generator

Let  $\mathcal{H}$  be an universal hash family from  $\{0, 1\}^k$  to  $\{0, 1\}^k$ . For any integer  $m \geq 0$  define the function  $\mathcal{G}_m: \{0, 1\}^k \times \mathcal{H}^m \rightarrow \{0, 1\}^{k \cdot 2^m}$  recursively as follows:

- $\mathcal{G}_0(x) = x$
- $\mathcal{G}_m(x, h_1, \dots, h_m) = (\mathcal{G}_{m-1}(x, h_1, \dots, h_{m-1}), \mathcal{G}_{m-1}(h_m(x), h_1, \dots, h_{m-1}))$

For example:

$$\mathcal{G}_1(x, h) = (x, h(x)), \quad \mathcal{G}_2(x, h_1, h_2) = (x, h_1(x), h_2(x), h_1 \cdot h_2(x))$$

$$\mathcal{G}_3(x, h_1, h_2, h_3) = (x, h_1(x), h_2(x), h_1 \cdot h_2(x), h_3(x), h_1 \cdot h_3(x), h_2 \cdot h_3(x), h_1 \cdot h_2 \cdot h_3(x))$$

- We want  $k \cdot 2^m = n^c \implies m = \log t$ .
- This gives a stretch of  $k \cdot (t - 1)$  bits.

# Proof Flow

Let  $h_1, \dots, h_s$  be some fixed hash functions from  $\mathcal{H}$ . Define the matrix

$$M_{h_1, \dots, h_s}[i, j] = \mathbb{P}_{x \in \{0,1\}^k} [Q(i; \mathcal{G}_s(x, h_1, \dots, h_s)) = j]$$

- Using  $h_1, \dots, h_s$  we had  $2^s$  many transitions in  $Q$ . So we should compare  $M_{h_1, \dots, h_s}$  with  $M^{2^s}$ .

**Goal:** For 'good' choice of  $h_1, \dots, h_m$ ,  $\|M^{2^m} - M_{h_1, \dots, h_m}\| < \epsilon$

**Approach:**

**Step 1:** Suppose we have  $h_1, \dots, h_{s-1}$ . We will find  $h_s \in \mathcal{H}$  such that for all  $i, j \in [N]$ ,

$$\left\| M_{h_1, \dots, h_{s-1}}^2 - M_{h_1, \dots, h_s} \right\| \leq \delta$$

**Step 2:** Using above property show for all  $s \in [m]$ ,

$$\left\| M_{h_1, \dots, h_s} - M^{2^s} \right\| \leq (2^s - 1)\delta$$

## Find good $h_s$ from $h_1, \dots, h_{s-1}$

Suppose we have  $h_1, \dots, h_{s-1} \in \mathcal{H}$  such that,

$$\left\| M_{h_1, \dots, h_{s-1}} - M^{2^{s-1}} \right\| \leq (2^{s-1} - 1)\delta$$

If we can find  $h_s$  such that  $\|M_{h_1, \dots, h_{s-1}}^2 - M_{h_1, \dots, h_s}\| \leq \delta$  then we are done.

**Algorithm:** Go over all  $h \in \mathcal{H}$  and all  $i, j \in [N]$ :

**Step 1:** Compute

- $p_1 = M_{h_1, \dots, h_{s-1}, h}[i, j]$
- $p_2 = \sum_{l \in [N]} M_{h_1, \dots, h_{s-1}}[i, l] \cdot M_{h_1, \dots, h_{s-1}}[l, j]$

**Step 2:** Check if  $|p_1 - p_2| > \frac{\delta}{N}$  go to next  $h$  else return  $h$ .

## Algorithm returns *good* $h_s$

### Claim

*If  $h_s$  is returned by the above algorithm, then*

$$\|M_{h_1, \dots, h_s} - M^{2^s}\| \leq (2^s - 1)\delta$$

## Algorithm returns *good* $h_s$

### Claim

*If  $h_s$  is returned by the above algorithm, then*

$$\|M_{h_1, \dots, h_s} - M^{2^s}\| \leq (2^s - 1)\delta$$

We have  $\|M_{h_1, \dots, h_{s-1}}^2 - M_{h_1, \dots, h_s}\| \leq \delta$ .



## Algorithm returns *good* $h_s$

### Claim

*If  $h_s$  is returned by the above algorithm, then*

$$\|M_{h_1, \dots, h_s} - M^{2^s}\| \leq (2^s - 1)\delta$$

We have  $\|M_{h_1, \dots, h_{s-1}}^2 - M_{h_1, \dots, h_s}\| \leq \delta$ .

$$\|M_{h_1, \dots, h_s} - M^{2^s}\| \leq \|M_{h_1, \dots, h_s} - M_{h_1, \dots, h_{s-1}}^2\| + \|M_{h_1, \dots, h_{s-1}}^2 - M^{2^s}\|$$

## Algorithm returns *good* $h_s$

### Claim

*If  $h_s$  is returned by the above algorithm, then*

$$\|M_{h_1, \dots, h_s} - M^{2^s}\| \leq (2^s - 1)\delta$$

We have  $\|M_{h_1, \dots, h_{s-1}}^2 - M_{h_1, \dots, h_s}\| \leq \delta$ .

$$\|M_{h_1, \dots, h_s} - M^{2^s}\| \leq \|M_{h_1, \dots, h_s} - M_{h_1, \dots, h_{s-1}}^2\| + \|M_{h_1, \dots, h_{s-1}}^2 - M^{2^s}\|$$

$$\begin{aligned} \|M_{h_1, \dots, h_{s-1}}^2 - M^{2^s}\| &\leq \|M_{h_1, \dots, h_{s-1}}\| \cdot \|M_{h_1, \dots, h_{s-1}} - M^{2^{s-1}}\| \\ &\quad + \|M_{h_1, \dots, h_{s-1}} - M^{2^{s-1}}\| \cdot \|M^{2^{s-1}}\| \end{aligned}$$

## Algorithm returns *good* $h_s$

### Claim

*If  $h_s$  is returned by the above algorithm, then*

$$\|M_{h_1, \dots, h_s} - M^{2^s}\| \leq (2^s - 1)\delta$$

We have  $\|M_{h_1, \dots, h_{s-1}}^2 - M_{h_1, \dots, h_s}\| \leq \delta$ .

$$\|M_{h_1, \dots, h_s} - M^{2^s}\| \leq \|M_{h_1, \dots, h_s} - M_{h_1, \dots, h_{s-1}}^2\| + \|M_{h_1, \dots, h_{s-1}}^2 - M^{2^s}\|$$

$$\begin{aligned} \|M_{h_1, \dots, h_{s-1}}^2 - M^{2^s}\| &\leq \|M_{h_1, \dots, h_{s-1}}\| \cdot \|M_{h_1, \dots, h_{s-1}} - M^{2^{s-1}}\| \\ &\quad + \|M_{h_1, \dots, h_{s-1}} - M^{2^{s-1}}\| \cdot \|M^{2^{s-1}}\| \\ &\leq 1 \cdot (2^{s-1} - 1)\delta + (2^{s-1} - 1)\delta \cdot 1 = (2^s - 2)\delta \end{aligned}$$

**Thank You**