$\mathsf{BPL} \subseteq \mathsf{SC}$ **and the Saks-Zhou theorem (lecture notes)**

Course: Derandomizing Space-Bounded Computation, Winter 2025, University of Chicago
Instructor: William Hoza (`williamhoza@uchicago.edu`)

---

In these lecture notes, we sketch the proofs of the following two classic theorems.

**Theorem 0.1** ([Nis94])**.** $\mathsf{BPL} \subseteq \mathsf{SC}$.[1] *To be more specific, every language in* $\mathsf{BPL}$ *can be decided by a deterministic algorithm that uses polynomial time and* $O(\log^2 n)$ *bits of space.*

**Theorem 0.2** ([SZ99])**.** $\mathsf{BPL} \subseteq \mathsf{DSPACE}((\log n)^{3/2})$.

If we insist on a polynomial-time simulation, then the $O(\log^2 n)$ space bound in Theorem 0.1 is still the best bound known today. If we don't worry about time complexity, the space bound in Theorem 0.2 has only been slightly improved, namely to $O((\log n)^{3/2}/\sqrt{\log\log n})$ [Hoz21].

The proofs of Theorems 0.1 and 0.2 both use Nisan's PRG, which we studied previously. In these lecture notes, we continue using the same notation to reason about Nisan's PRG.

# 1 $\mathsf{BPL} \subseteq \mathsf{SC}$: Searching for good hash functions

When we analyzed Nisan's PRG, we showed that for any automaton $M$, if we pick a hash function $h$ from a pairwise uniform family, then with high probability, $h$ is "good" for $M$, meaning that $G_h$ fools $M$. To prove $\mathsf{BPL} \subseteq \mathsf{SC}$, we will show that we can actually *find* a good hash function:

**Lemma 1.1** (Finding a good hash function)**.** *Suppose we are given the truth table of a finite automaton* $M\colon [w] \times \{0,1\}^k \to [w]$. *Using* $O(k + \log w)$ *bits of space, it is possible to deterministically find the* $O(k)$*-bit description of an explicit hash function* $h\colon \{0,1\}^k \to \{0,1\}^k$ *such that* $G_h$ *fools* $M$ *with* $\ell_1$ *error at most* $w^{2.5} \cdot 2^{-k/2}$.

*Proof sketch.* Let $\mathcal{H}$ be an explicit pairwise uniform family of hash functions. For each $h$ in $\mathcal{H}$, define

$$e_h = \max_{u \in [w]} \sum_{v \in [w]} \Big| \, |\{(x,y) \in \{0,1\}^{2k} : M^2[u, xy] = v\}| - |\Sigma| \cdot |\{x \in \{0,1\}^k : M_h[u,x] = v| \, \Big|.$$

Observe that $e_h$ is an integer between 0 and $w \cdot 2^{2k}$. For any fixed $h$, the value $e_h$ can be computed and stored in $O(k + \log w)$ bits of space by straightforward counting. The algorithm outputs the $h$ that minimizes $e_h$, which can be found by exhaustive search over all $h$. Our previous analysis of a random $h \sim \mathcal{H}$ ("Lemma 2.2" in the lecture notes on Nisan's PRG) implies that the best hash function $h$ satisfies $\|M^2 - M_h\|_1 \leq w^{2.5} \cdot 2^{-k/2}$. □

**Corollary 1.2** (Finding a sequence of good hash functions)**.** *Suppose we are given the truth table of a finite automaton* $M\colon [w] \times \{0,1\}^k \to [w]$ *and a power of two* $n \in \mathbb{N}$. *Using* $O(k \cdot \log n + \log w)$ *bits of space and* $\mathrm{poly}(n, w, 2^k)$ *time, it is possible to deterministically find the* $O(k)$*-bit descriptions of explicit hash functions* $h_1, \dots, h_{\log n}$ *such that* $G_{h_1,\dots,h_{\log n}}$ *fools* $M$ *with* $\ell_1$ *error at most* $w^{2.5} \cdot 2^{-k/2} \cdot n$.

*Proof sketch.* Suppose we have already found and stored $h_1, \dots, h_{i-1}$. To find $h_i$, we apply Lemma 1.1 to the automaton $M_{h_1,\dots,h_{i-1}}$. Each time the Lemma 1.1 algorithm asks about some transition $M_{h_1,\dots,h_{i+1}}[u,x]$, we compute it by feeding $G_{h_1,\dots,h_{i+1}}(x)$ into $M$. Our previous analysis of the efficiency of Nisan's PRG ("Lemma 3.1" in the lecture notes on Nisan's PRG) implies that this process takes only $O(k + \log n)$ bits of space, hence $\mathrm{poly}(2^k, n)$ time. The Lemma 1.1 algorithm takes $O(k + \log w)$ bits of space, hence $\mathrm{poly}(2^k, w)$ time. Finally, our analysis of the accumulation of error in Nisan's PRG ("Lemma 3.3" in the lecture notes on Nisan's PRG) implies that $G_{h_1,\dots,h_{\log n}}$ fools $M$ with $\ell_1$ error at most $w^{2.5} \cdot 2^{-k/2} \cdot (n-1)$. □

---

[1] By definition, a language is in $\mathsf{SC}$ if it can be decided by a deterministic algorithm that simultaneously uses polynomial time and polylogarithmic space.

Corollary 1.2 readily implies $\mathsf{BPL} \subseteq \mathsf{SC}$. Some details follow.

*Proof of Theorem 0.1.* Given an instance $x$ of some language in $\mathsf{BPL}$, in deterministic log-space, we can compute the truth table of a finite automaton $M\colon [w] \times \{0,1\} \to [w]$ that describes how the $\mathsf{BPL}$ algorithm's configuration updates when it reads a single random bit. Here $w = \mathrm{poly}(|x|)$. We can ensure that the start state is 1, the unique accepting state is $w$, and the unique rejecting state is $w-1$. We can also ensure that $M[w,0] = M[w,1] = w$ and $M[w-1,0] = M[w-1,1] = w-1$.

Let $n$ be a bound on the number of random bits that the $\mathsf{BPL}$ algorithm uses on $x$. For example, we can always take $n = w$. To decide whether $x$ is in the language, it suffices to estimate the $(1,w)$ entry of $M^n$ to within additive error $\varepsilon = 0.1$.

We begin by artificially enlarging the alphabet of $M$, producing an automaton $M'\colon [w] \times \{0,1\}^k \to [w]$ for a suitable $k = O(\log(wn/\varepsilon))$. Next, we use Corollary 1.2 to deterministically find hash functions $h_1, \dots, h_{\log n}$ such that $G_{h_1,\dots,h_{\log n}}$ fools $M'$ with $\ell_1$ error at most $w^{2.5} \cdot 2^{-k/2} \cdot n$, which is at most $\varepsilon$ provided we choose a suitable value $k = O(\log(wn/\varepsilon))$.

Finally, we compute $(M')^n[1, G_{h_1,\dots,h_{\log n}}(y)]$ for all $y \in \{0,1\}^k$. Altogether, this process uses $O(k \cdot \log n + \log w) = O(\log(wn/\varepsilon) \cdot \log n)$ bits of space, which is $O(\log^2 |x|)$ if $n = w$ and $\varepsilon = 0.1$, and it uses $\mathrm{poly}(n, w, 2^k) = \mathrm{poly}(nw/\varepsilon)$ time, which is $\mathrm{poly}(|x|)$ if $n = w$ and $\varepsilon = 0.1$. ☐

## 2 $\mathsf{BPL} \subseteq \mathsf{DSPACE}((\log n)^{3/2})$: Reusing hash functions

What goes wrong if we sample just *one* hash function $h$ in Nisan's PRG and reuse it in every round of the recursion? It is tempting to think that the PRG should still work by a simple union bound. For any *fixed* $M$, it is indeed true that $G_h$ fools $M$ with high probability. However, if we reuse $h$ in the first and second rounds of the recursion, then in the second round of the recursion, we want $G_h$ to fool $M_h$. Obviously, $M_h$ is *correlated with* $h$, and hence there is no reason to think that $G_h$ is likely to fool $M_h$.

Despite this issue, Saks and Zhou managed to figure out a way to reuse hash functions in Nisan's PRG. Their idea is that maybe $M_h$ is not so correlated with $h$ after all, because $M_h \approx M^2$ with high probability, and $M^2$ is independent of $h$. To make this idea make sense, Saks and Zhou modify the algorithm, not just the analysis. The key new ingredient is a randomized procedure for modifying $M_h$ so that it still approximates $M^2$, but now it is essentially independent of $h$.

We first define a procedure for modifying a single entry $\widehat{p}$ in the transition probability matrix of $M_h$ in order to destroy the correlation with $h$. For $\widehat{p} \in [0,1]$, $d \in \mathbb{N}$, and $r \in [2^d]$, we define

$$\widehat{p} \ominus_d r = 2^{-d} \cdot \lfloor 2^d \cdot \max\{0, \widehat{p} - r \cdot 2^{-2d}\} \rfloor.$$

We think of $r$ as randomness. The $\ominus$ operation begins by randomly *perturbing* $\widehat{p}$; we subtract a random small amount (at most $2^{-d}$) without letting it become negative. Then it *truncates* the perturbed value, retaining $d$ bits of precision. The following two propositions show that if $\widehat{p} \approx p$, then $\widehat{p} \ominus_d r \approx p$ as well, and furthermore, applying the $\ominus$ operation effectively *destroys* any information that is stored in $\widehat{p}$ rather than in $p$.

**Proposition 2.1** ($\ominus$ doesn't introduce much error). *For every $\widehat{p} \in [0,1]$, $d \in \mathbb{N}$, and $r \in [2^d]$, we have*

$$\widehat{p} \ominus_d r \in [\widehat{p} - 2^{-d+1}, \widehat{p}] \cap [0,1].$$

*Proof.* This is immediate from the definition. ☐

**Proposition 2.2** ($\ominus$ destroys information). *For every $p \in [0,1]$ and $d \in \mathbb{N}$, there exists $r_{\mathrm{bad}} \in [2^d]$ such that for every $\widehat{p} \in [0,1]$ and every $r \in [2^d] \setminus \{r_{\mathrm{bad}}\}$, if $|p - \widehat{p}| < 2^{-2d-1}$, then $\widehat{p} \ominus_d r = p \ominus_d r$.*

*Proof.* Let $r_{\mathrm{bad}}$ be the unique value such that $p - r \cdot 2^{-2d}$ is at distance less than $2^{-2d-1}$ from an integer multiple of $2^{-d}$, or let $r_{\mathrm{bad}} = 1$ if no such value exists. If $r \neq r_{\mathrm{bad}}$ and $|p - \widehat{p}| < 2^{-2d-1}$, then there is no integer multiple of $2^{-d}$ between $p - r \cdot 2^{-2d}$ and $\widehat{p} - r \cdot 2^{-2d}$, hence $\widehat{p} \ominus_d r = p \ominus_d r$. ☐

Now we extend the $\ominus$ operation to operate on automata. Suppose $\widehat{M}\colon [w] \times \Sigma \to [w]$ is an automaton. For $d \in \mathbb{N}$ and $r \in [2^d]$, we define an automaton $(\widehat{M} \ominus_d r)\colon [w] \times [2^d] \to [w]$ as follows.

1. Let $A \in [0,1]^{w \times w}$ be the transition probability matrix of $\widehat{M}$.
2. Define $B \in [0,1]^{w \times w}$ by $B_{u,v} = A_{u,v} \ominus_d r$. Note that $B$ is substochastic, i.e., the entries of each row add up to *at most* one, and furthermore each entry of $B$ is an integer multiple of $2^{-d}$.
3. Define $C \in [0,1]^{w \times w}$ by increasing the last entry of each row of $B$ so that $C$ is stochastic, i.e., the entries of each row add up to exactly one. Note that each entry of $C$ is still an integer multiple of $2^{-d}$.
4. Let $(\widehat{M} \ominus_d r)[u, x] = v$ if

$$C_{u,1} + \cdots + C_{u,v-1} < \frac{x}{2^d} \leq C_{u,1} + \cdots + C_{u,v}.$$

   Note that the transition probability matrix of $\widehat{M} \ominus_d r$ is precisely the matrix $C$.

The following two propositions show that if $\widehat{M}$ approximates $M$, then applying the $\ominus$ operation to $\widehat{M}$ produces another automaton that approximates $M$, and in the process, it destroys any information that is stored in $\widehat{M}$ rather than in $M$.

**Proposition 2.3** ($\ominus$ doesn't introduce much error when applied to automata)**.** *For every automaton $\widehat{M}\colon [w] \times \Sigma \to [w]$, every $d \in \mathbb{N}$, and every $r \in [2^d]$, we have*

$$\|\widehat{M} - (\widehat{M} \ominus_k r)\|_1 \leq 2^{-d+2} \cdot w.$$

*Proof.* This is immediate from Proposition 2.1. □

**Proposition 2.4** ($\ominus$ destroys information when applied to automata)**.** *For every automaton $M\colon [w] \times \Sigma \to [w]$ and $d \in \mathbb{N}$, there exists $R_{\mathrm{bad}} \subseteq [2^d]$ of size at most $w^2$ such that for every $\widehat{M}\colon [w] \times \Sigma' \to [w]$ and every $r \in [2^d] \setminus R_{\mathrm{bad}}$, if $\|M - \widehat{M}\|_{\max} < 2^{-2d-1}$, then $\widehat{M} \ominus_d r = M \ominus_d r$.*

*Proof.* This is immediate from Proposition 2.2 (let $R_{\mathrm{bad}}$ contain each $r_{\mathrm{bad}}$ associated with each entry of the transition probability matrix of $M$). □

Furthermore, $\widehat{M} \ominus_d r$ is efficiently computable:

**Proposition 2.5** ($\widehat{M} \ominus_d r$ is efficiently computable)**.** *Given the truth table of $\widehat{M}\colon [w] \times \{0,1\}^k \to [w]$, given $d \in \mathbb{N}$, and given $r \in [2^d]$, it is possible to compute the truth table of $\widehat{M} \ominus_d r$ using $O(k + d + \log w)$ bits of space.*

Proposition 2.5 is more or less immediate from the definitions; we omit the tedious proof. Now we present the Saks-Zhou algorithm. Suppose we are given $M\colon [w] \times \{0,1\} \to [w]$, $n \in \mathbb{N}$, and $\varepsilon \in (0,1)$. Our goal is to approximate the transition probability matrix of $M^n$, to within $\ell_1$ error $\varepsilon$. Let $\mathcal{H}$ be a pairwise uniform family of hash functions $h\colon \{0,1\}^k \to \{0,1\}^k$ for a suitable value $k = O(\log(wn/\varepsilon))$. Let $s, t \in \mathbb{N}$ such that $st = \log n$. Sample $h_1, \ldots, h_s \sim \mathcal{H}$, and let $\vec{h} = (h_1, \ldots, h_s)$. Sample $r_1, \ldots, r_t \in [2^d]$ independently and uniformly at random for a suitable value $d \leq k$, and let $\vec{r} = (r_1, \ldots, r_t)$. Define the following sequence of automata:

$$\widehat{M}^{(0)} = M$$
$$\widehat{M}^{(i)} = \widehat{M}^{(i-1)}_{\vec{h}} \ominus_d r_i.$$

(Note: The alphabet of $\widehat{M}^{(i-1)}$ is $\{0,1\}$ or $\{0,1\}^d$, but we can treat it as an automaton over the alphabet $\{0,1\}^k$ that simply ignores some of the bits of each symbol it reads. This allows us to apply the hash functions $\vec{h}$.) Crucially, we reuse the same vector of hash functions $\vec{h}$ in each round. Finally, the output of the Saks-Zhou algorithm consists of the transition probability matrix of $\widehat{M}^{(t)}$.

**Proposition 2.6** (Efficiency of the Saks-Zhou algorithm)**.** *The Saks-Zhou algorithm uses* $O(\log(wn/\varepsilon)\cdot(s+t))$ *bits of space and* $O(\log(wn/\varepsilon)\cdot(s+t))$ *bits of randomness.*

We omit the tedious proof.

**Proposition 2.7** (Correctness of the Saks-Zhou algorithm)**.** *Except with probability* $w^2\cdot t\cdot 2^{-d}+w^5\cdot\log n\cdot 2^{O(d+s)}\cdot 2^{-k}$ *over the choices of* $\vec{h}$ *and* $\vec{r}$*, we have*

$$\|\widehat{M}^{(t)}-M^n\|_1\leq 4nw\cdot 2^{-d}.$$

*Proof.* Purely for the sake of analysis, we define the following sequence of automata:

$$\begin{aligned}M^{(0)}&=M\\ M^{(i)}&=(M^{(i-1)})^{2^s}\ominus_d r_i.\end{aligned}$$

By Proposition 2.4, the probability that $r_i$ falls into the $R_{\text{bad}}$ set the automaton $(M^{(i-1)})^{2^s}$ is at most $w^2\cdot t\cdot 2^{-d}$. Fix any $r_1,\ldots,r_t$ such that this does not occur.

By our analysis from last time (see "Lemma 2.2" and "Lemma 3.3" in the lecture notes on Nisan's PRG), except with probability $w^5\cdot\log n\cdot 2^{O(d+s)}\cdot 2^{-k}$ over the choice of $\vec{h}$, we have $\|M_{\vec{h}}^{(i-1)}-(M^{(i-1)})^{2^s}\|_1<2^{-2d-1}$ for every $i\in[t]$. Assume that this occurs.

Under these assumptions, let us show by induction on $i$ that $\widehat{M}^{(i)}=M^{(i)}$ for every $i\in[t]$. In the base case $i=0$, this is true by definition. Now assume by induction that $\widehat{M}^{(i-1)}=M^{(i-1)}$. Our assumption about $\vec{h}$ tells us that $\|(M^{(i-1)})^{2^s}-\widehat{M}_{\vec{h}}^{(i-1)}\|_1<2^{-2d-1}$. Consequently, our assumption about $r_i$, together with Proposition 2.4, tells us that $\widehat{M}^{(i)}=M^{(i)}$.

To complete the proof, we show by induction on $i$ that $\|M^{(i)}-M^{2^{si}}\|_1\leq\frac{2^{si}-1}{2^s-1}\cdot 2^{-d+2}\cdot w$. In the base case $i=0$, this is trivial. For $i>0$, we have

$$\|M^{(i)}-M^{2^{si}}\|_1\leq\|M^{(i)}-(M^{(i-1)})^{2^s}\|_1+\|(M^{(i-1)})^{2^s}-M^{2^{si}}\|_1.$$

The first term is at most $2^{-d+2}\cdot w$ by Proposition 2.3. The second term is at most $2^s\cdot\|M^{(i-1)}-M^{2^{s\cdot(i-1)}}\|_1$, because

$$\begin{aligned}\|A^m-B^m\|_1\leq\sum_{i=1}^m\|A^{m-i+1}B^{i-1}-A^{m-i}B^i\|_1&=\sum_{i=1}^m\|A^{m-i}\cdot(A-B)\cdot B^{i-1}\|_1\\ &\leq\sum_{i=1}^m\|A^{m-i}\|_1\cdot\|A-B\|_1\cdot\|B^{i-1}\|_1\\ &\leq m\cdot\|A-B\|_1\end{aligned}$$

for any stochastic matrices $A,B$. Therefore, by induction, we get

$$\|M^{(i)}-M^{2^{si}}\|_1\leq\left(2^s\cdot\frac{2^{s\cdot(i-1)}-1}{2^s-1}+1\right)\cdot 2^{-d+2}\cdot w=\frac{2^{si}-1}{2^s-1}\cdot 2^{-d+2}\cdot w.$$

□

To complete the proof of Theorem 0.2, the idea is to set $s=t=\sqrt{\log n}$, $w=n$, and $\varepsilon=0.1$, and try all possible settings of the $O((\log n)^{3/2})$ many random bits.

## References

[Hoz21] William M. Hoza. "Better Pseudodistributions and Derandomization for Space-Bounded Computation". In: *Proceedings of the 25th International Conference on Randomization and Computation (RANDOM)*. Vol. 207. 2021, 28:1–28:23. ISBN: 978-3-95977-207-5. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2021.28.

[Nis94] Noam Nisan. “RL $\subseteq$ SC”. In: *Comput. Complexity* 4.1 (1994), pp. 1–11. ISSN: 1016-3328. DOI: 10.1007/BF01205052.

[SZ99] Michael Saks and Shiyu Zhou. “$\mathrm{BP_H SPACE}(S) \subseteq \mathrm{DSPACE}(S^{3/2})$”. In: *J. Comput. System Sci.* 58.2 (1999), pp. 376–403. ISSN: 0022-0000. DOI: 10.1006/jcss.1998.1616.