An alternative presentation of the analysis
of Nisan’s pseudorandom generator of space-bounded machines

The following description of the analysis of Nisan’s construction [3] is inspired by [1], and differs from the presentation in [2, Sec. 8.4.2.1]. Specifically, the construction is the same, but rather than being analyzed by looking at contracted versions of the distinguisher (see [2, p. 321]), we consider a sequence of distributions that this distinguisher may examine.

Our description is meant to replace the text in [2, pp. 320-321], which means that it relies on the definitions and notations of [2, Sec. 8.4].

Sketch of the proof of [2, Thm. 8.21]. The main technical tool used in this proof is the “mixing property” of pairwise independent hash functions (see [2, Apdx. D.2]). A family of functions $H_n$, which map $\{0,1\}^n$ to itself, is called mixing if for every pair of subsets $A,B \subseteq \{0,1\}^n$ for all but very few (i.e., $\exp(-\Omega(n))$) fraction) of the functions $h \in H_n$, it holds that

$$\Pr[U_n \in A \land h(U_n) \in B] \approx \frac{|A|}{2^n} \cdot \frac{|B|}{2^n}$$

where the approximation is up to an additive term of $\exp(-\Omega(n))$. (See the generalization of [2, Lem.D.4], which implies that $\exp(-\Omega(n))$ can be set to $2^{-n/3}$.)

We may assume, without loss of generality, that $s(k) = \Omega(\sqrt{k})$, and thus $\ell \defeq \ell(k) \leq 2^{s(k)}$ holds. For any $s(k)$-space distinguisher $D_k$ as in [2, Def. 8.20], we consider its computation when fed with $\ell$-long sequences that are taken from various distributions. The first distribution is the uniform distribution over $\{0,1\}^n$; that is, $U_\ell \equiv U_n^{(1)}U_n^{(2)} \cdots U_n^{(\ell')}$, where $\ell' = \ell/n$ and the $U_n^{(j)}$’s are independent random variables each uniformly distributed over $\{0,1\}^n$. The last distribution will be the one produced by the pseudorandom generator, and a generic (hybrid) distribution will have the form

$$H_i \equiv \defeq G_i(U^{(1)}_n)G_i(U^{(2)}_n) \cdots G_i(U^{(\ell'/2^i-1)}_n)G_i(U^{(\ell'/2^i)}_n)$$

where $G_i$ is an arbitrary mapping of $n$-bit strings to $2^i \cdot n$-bit strings (and $i \in \{0,1,\ldots, \log_2 \ell'\}$).

That is, the $i$th hybrid is obtained by applying $G_i : \{0,1\}^n \rightarrow \{0,1\}^{2^i \cdot n}$ to a sequence of $\ell'/2^i$ independently and uniformly distributed $n$-bit long strings. Note that $H_0 \equiv U_\ell$ (with $G_0$ being the identity function), whereas $H_{\log_2 \ell'} = G_{\log_2 \ell'}(U_n)$ is a distribution that is obtained by stretching random $n$-bit long strings into $\ell$-bit long strings.

The key observation is that, for every $i$, the automata $D_k$ cannot distinguish between $H_i$ and a distribution obtained by selecting a typical $h \in H_n$ and outputting

$$G_i(U^{(1)}_n)G_i(h(U^{(1)}_n)) \cdots G_i(U^{(\ell'/2^i+1)}_n)G_i(h(U^{(\ell'/2^i+1)}_n)).$$

Note that the foregoing distribution is similar to $H_i$, except that the $2j$th block is set to $G_i(h(U^{(j)}_n))$ rather than to $G_i(U^{(j)}_n)$ as in $H_i$.

On the other hand, the foregoing distribution has the form of $H_{i+1}$ (i.e., let $G_{i+1}(s) = G_i(s)G_i(h(s))$). To prove that this replacement has little effect on the movement of $D_k$, we consider an arbitrary pair of vertices, $u$ and $v$ in layers $(2j - 2) \cdot 2^i \cdot n$

---

1. Indeed, while at this point $G_i$ is to be thought of as arbitrary, later we shall use specific choices of $G_i$.
2. Setting the $(2j-1)^{st}$ block to $G_i(h(U^{(j)}_n))$ rather than to $G_i(U^{(2j-1)}_n)$ as in $H_i$ is immaterial.
and $(2j - 1) \cdot 2^i \cdot n$, respectively, and denote by $L_{u,v} \subseteq \{0,1\}^n$ the set of the $n$-bit long strings $s$ such that the automaton moves from vertex $u$ to vertex $v$ upon reading $G_i(s)$ (from locations $(2j - 2) \cdot 2^i \cdot n + 1, ..., (2j - 1) \cdot 2^i \cdot n$ in its input). Similarly, for a vertex $w$ at layer $2j \cdot 2^i \cdot n$, we let $L'_{v,w}$ denote the set of the strings $s$ such that $D_k$ moves from $v$ to $w$ upon reading $G_i(s)$. By Eq. (1), for all but very few of the functions $h \in H_n$, it holds that

$$\Pr[U_n \in L_{u,v} \land h(U_n) \in L'_{v,w}] \approx \Pr[U_n \in L_{u,v}] \cdot \Pr[U_n \in L'_{v,w}],$$

where “very few” and $\approx$ are as in Eq. (1). Thus, for all but $\exp(-\Omega(n))$ fraction of the choices of $h \in H_n$, replacing the coins in the second transition (i.e., the transition from layer $(2j - 1) \cdot 2^i \cdot n$ to layer $2j \cdot 2^i \cdot n$) with the value of $h$ applied to the outcomes of the coins used in the first transition (i.e., the transition from layer $(2j - 2) \cdot 2^i \cdot n$ to $(2j - 1) \cdot 2^i \cdot n$), approximately maintains the probability that $D_k$ moves from $u$ to $w$ via $v$. Using a union bound (on all triples $(u,v,w)$ as in the foregoing), we note that, for all but $2^{3s(k)} \cdot \ell' \cdot \exp(-\Omega(n))$ fraction of the choices of $h \in H_n$, the foregoing replacement approximately maintains the probability that $D_k$ moves through any specific triple of vertices that are $2^i \cdot n$ apart. (We stress that the same $h$ can be used in all these approximations.)

Thus, at the cost of extra $|h|$ random bits, we can reduce the number of true random coins used in transitions on $D_k$ by a factor of two, without significantly affecting the final decision of $D_k$ (where again we use the fact that $\ell' \cdot \exp(-\Omega(n)) < \exp(-\Omega(n))$, which implies that the approximation errors do not accumulate to too much). That is, fixing a good $h$ (i.e., one that provides a good approximation to the transition probability over all $2^{3s(k)} \cdot \ell'$ triples), we can replace the amount of randomness in the hybrid (from $\ell'/2^i \cdot n$ in $H_i$ to $\ell'/2^{i+1} \cdot n$ in $H_{i+1}$, which is defined based on this $h$), while approximately preserving the acceptance probability of $D_k$ (i.e., $\Pr[D_k(H_i) = 1] \approx \Pr[D_k(H_{i+1}) = 1]$).

Applying the foregoing process can for $i = 0, ..., \log_2 \ell - 1$, we repeatedly reduce the randomness of the hybrid by a factor of two, by randomly selecting (and fixing) a new hash function. Thus, repeating the process for a logarithmic (in $\ell'$) number of times, we obtain a distribution that depends on $n$ random bits, at which point we stop. In total, we have used $t \overset{\text{def}}{=} \log_2 \ell' < \log_2 \ell(k)$ random hash functions, denoted $h^{(1)}, ..., h^{(t)}$. This means that we can generate a (pseudorandom) sequence that fools the original $D_k$ by using a seed of length $n + t \cdot \log_2 |H_n|$ (see [2, Fig. 8.3] and [2, Exer. 8.28]). Using $n = \Theta(s(k))$ and an adequate family $H_n$ (e.g., [2, Const. D.3]), we obtain the desired $(s, 2^{-s})$-pseudorandom generator, which indeed uses a seed of length $O(s(k) \cdot \log_2 \ell(k)) = k$.

\[\square\]

References

