Locally sampleable uniform symmetric distributions

Talk by Kewen Wu

Notes by Sanjana Das

February 20, 2025

This is joint work with Daniel Kane and Anthony Ostuni.

§1 Introduction

There are lots of adjectives in the title, which we'll explain soon.

§1.1 The question

We're broadly interested in the following question:

Question 1.1. What distributions can a shallow circuit produce?

We'll focus on circuits with Boolean inputs and outputs (with multiple output gates). We have negation, and AND and OR gates with bounded fan-in (we'll work with fan-in 2). By 'shallow' we mean that the circuit has shallow depth — that any computation from the input has bounded (i.e., constant) length.

We can also view such a circuit as a mapping from inputs to outputs. The output bits will be low-degree polynomials (because the circuit is shallow, so each output bit only touches a limited number of input bits), and they're produced by small decision trees. Or you can view these outputs as weakly correlated — intuitively, a shallow thing can't generate much correlation. So we stated the problem in terms of shallow circuits, but you can recast it in many ways.

What is a distribution? Suppose we have a fixed circuit mapping input bits to output bits. We consider feeding the circuit *random* input bits $r_i \sim \mathsf{Unif}\{0,1\}$. This induces some distribution $b_1 \dots b_n$ for the output; we'll use $\mathcal{D}$ to denote this output distribution.

Question 1.1 is very broad — how can we possibly describe a distribution? So for our purposes, we'll focus on *uniform symmetric* distributions; we'll see some examples for why we care about such distributions.

Definition 1.2. We say $\mathcal{D}$ is uniform if it's the uniform distribution on its support — i.e., we have the set $\operatorname{supp}(\mathcal{D})$, and we pick a uniform random thing from it.

Definition 1.3. We say a distribution $\mathcal{D}$ is symmetric if permuting the coordinates of a string $x \in \{0,1\}^n$ doesn't affect its probability.

Example 1.4

- The uniform distribution on $\{0,1\}^n$ is uniform and symmetric.
- The $1/3$-biased distribution (where we independently set each coordinate to be 1 with probability $1/3$) is symmetric but not uniform.
- The distribution of a uniform random string of Hamming weight $n/2$ is uniform and symmetric.
- The point distribution on 0^n is uniform (*any* point distribution is uniform) and symmetric (all permutations of 0^n are still 0^n).

(In fact, you can describe any uniform symmetric distribution by specifying which Hamming weights are allowed, and then taking a uniform thing from the set of strings with those Hamming weights.)

§1.2 Motivation

Why do we care about this? For one thing, this seems a natural question on its own — we care about the computation power of our computational models, and this is one type of question we typically study. For example, we can ask the general question of what shallow circuits can do. In terms of *computation*, this is not very interesting — for example, they can't compute $x_1 \oplus \cdots \oplus x_{n-1}$. This is because the shallow circuit has constant depth, so every output bit just depends on a constant number of input bits; this means you can't generate a correlation spanning all $n-1$.

But you could also talk about sampling, or generating a distribution:

Question 1.5. Can we generate the distribution $(x_1, \ldots, x_{n-1}, x_1 \oplus \cdots \oplus x_{n-1})$?

If you could compute $x_1 \oplus \cdots \oplus x_{n-1}$, then you could certainly generate this distribution — sample $n-1$ random bits, and do this computation to get the last one. Here we *can't* do the computation, but can we still sample from this distribution? It turns out the answer is yes! If we forget about this specific computational expression for our string, what is it? It's just a uniform random string of even parity (i.e., of even Hamming weight). And we can produce such a string in a different way — sample random bits $y_1, \ldots, y_n$, and take

$$(y_1 \oplus y_2, y_2 \oplus y_3, \ldots, y_n \oplus y_1).$$

This distribution is the same as the one we started with — so even though we can't do the computation, we can sample this distribution.

Question 1.6. Are there other examples like this — where there's a symmetric function that you can't compute, but there's some distribution naturally induced by it that's easy to sample from?

And this work basically shows the answer is no — this is the only nontrivial example!

The second motivation for this problem comes from data structure lower bounds. Here, we want to see how much we can compress the input in our data structure while still supporting efficient queries.

Problem 1.7 (Dictionary problem)

You're given a n-bit string of Hamming weight $0 \bmod 127$, and you want to store it as a s-bit string h, in such a way that you can reconstruct each x_i easily from h.

So you have an input x and compress it to h, and you want to be able to reconstruct x efficiently.

Remark 1.8. The number 127 is not special; it's just because Kewen finished the slides on January 27.

There's a direct way to do this to maximize efficiency — just store $h = x$. Then if you want to find x_i, you can just read h_i. So here, the storage is n bits (i.e., $s = n$), but it's extremely efficient — you just need to read one bit.

On the other hand, we can minimize storage. Not all n-bit strings are possible — there's roughly $2^n/127$ possibilities for x. So information-theoretically, we can store x with $s = \lceil \log_2(2^n/127) \rceil = n - 6$ bits — take h to be the index pointing to x in the list of all possible strings. So we save 6 bits in the storage. But we pay for this in efficiency — even if we don't care about the entire string x but only one bit, we need to know the *whole* h to figure out that one bit of x. This is very inefficient.

Question 1.9. Can we achieve both? Specifically, can we save *something* in storage, while still doing reasonably good in decoding?

And this work shows that the answer is no! In particular:

Theorem 1.10 (Kane–Ostuni–Wu 2025+)
Either you need to read $\omega(1)$ bits of h to compute one bit of x, or h has length n.

(And if h has length n, then you have the trivial construction where you can read just one bit.)

We won't go into the details of the connection to the original problem, which is nontrivial. But the basic idea is that if you have a small data structure, you can imagine taking h to be a uniform random string (of the appropriate length) and decoding it to get output x. Then the distribution of x will be reasonably close to the uniform distribution on strings of Hamming weight 0 mod 127. So to show that this is impossible, we have to show that you can't sample this uniform distribution on strings of weight 0 mod 127 with a shallow circuit — more precisely, that you'll have large distance from this distribution.

A third motivation comes from some quantum vs. classical questions that we won't talk about too much.

Question 1.11. Can we use quantum shallow circuits to produce some distributions that are hard for classical ones?

For example, you have the classes NC^0 (classical constant-depth circuits) and QNC^0 (quantum constant-depth circuits); are they different? This work shows that the answer is yes — there are uniform symmetric distributions we can produce with quantum circuits but not classical ones.

§1.3 Setup and results

This is the background story and motivation; now let's set up the problem.

Let $f : \{0,1\}^m \to \{0,1\}^n$ be a *local* function, meaning that every output bit depends on a constant number of input bits. (This is the same as being a shallow circuit.) We don't assume any relation between m and n; you can have as many input bits as you want. We define $\mathcal{U}$ to be the uniform distribution on $\{0,1\}^m$, and we use $f(\mathcal{U})$ to denote the output distribution of f under $\mathcal{U}$.

Question 1.12. What uniform symmetric distributions can be produced as $f(\mathcal{U})$ for some f?

You can try some examples. We'll start with a trivial one, the point distribution 0^n (for a point distribution to be symmetric, it has to be one of 0^n or 1^n). For this, you don't even need input bits — you can just output 0^n.

And because both 0^n and 1^n are doable, their mixture $f(\mathcal{U}) \sim \mathsf{Unif}\{0^n, 1^n\}$ is also doable (and local) — every output bit looks at one input bit, and decides based on this whether to output 0 or 1.

What about other constructions? The parity construction from earlier gives a slightly nontrivial example: We can output a random string of even parity, by sampling random bits and XORing neighboring ones — i.e., we define

$$f(r_1, \ldots, r_n) = (r_1 \oplus r_2, r_2 \oplus r_3, \ldots, r_n \oplus r_1).$$

And because we can do strings of even parity, we can also do strings of odd parity — just put $r_n \oplus r_1 \oplus 1$ on the last coordinate, instead of $r_n \oplus r_1$.

And because we can do both evens and odds, we can also do their mixture, which is the uniform distribution on all strings. Of course, you could also get the uniform distribution just by sampling directly, where every input bit leads to one output bit (i.e., $f(r_1, \ldots, r_n) = (r_1, \ldots, r_n)$); either construction is local.

To summarize:

Example 1.13

We can sample from the following six distributions:

- The point distribution 0^n (obtained by $f = 0^n$).
- The point distribution 1^n (obtained by $f = 1^n$).
- The distribution $\mathsf{Unif}\{0^n, 1^n\}$, obtained by
$$f(r_1) = (r_1, r_1, \ldots, r_1).$$
- The uniform distribution on even-weight strings, obtained by
$$f(r_1, \ldots, r_n) = (r_1 \oplus r_2, r_2 \oplus r_3, \ldots, r_n \oplus r_1).$$
- The uniform distribution on odd-weight strings, obtained by
$$f(r_1, \ldots, r_n) = (r_1 \oplus r_2, r_2 \oplus r_3, \ldots, r_n \oplus r_1 \oplus 1).$$
- The uniform distribution on all strings (obtained by the identity function).

But certainly these are not the only uniform symmetric distributions. For example:

Question 1.14. Can we sample a uniform random string of weight at most 10, or a uniform string of fixed weight $n/2$, or a uniform string whose weight is 0 mod 3?

These are all valid uniform symmetric distributions, and you can try constructing a shallow circuit to output them, but it seems hard — you don't quite get them. And indeed, it was a conjecture that you can't:

Conjecture 1.15 (Filmus–Leigh–Riazanov–Sokolov 2023) **—** There are no other examples of uniform symmetric distributions you can sample using local functions (other than the ones in Example 1.13).

Theorem 1.16 (Kane–Ostuni–Wu 2025+)

Conjecture 1.15 is true.

So far, this is about *perfect* sampling — to perfectly sample a uniform symmetric distribution locally (when

n is large), these are the only examples we can have. This is an exact statement, but we'll now talk about a robust and quantitative version.

Definition 1.17. We say f is d-local if each output bit only depends on d input bits.

We still define $\mathcal{U}$ and $f(\mathcal{U})$ in the same way. But instead of asking $f(\mathcal{U})$ to be *perfectly* uniform and symmetric, we just ask it to be ε-close to a uniform symmetric distribution (in total variation distance).

Theorem 1.18 (Kane–Ostuni–Wu 2025+)

If $f(\mathcal{U})$ is ε-close to some uniform symmetric distribution, then it must be $O_d(\varepsilon)$-close to one of the six distributions in Example 1.13.

We'll talk a bit about the dependence on d later; but the dependence is linear in ε (for fixed d).

The takeaway is that there's six special distributions that are easy — the point distributions on 0^n and 1^n, $\mathsf{Unif}\{0^n, 1^n\}$, a uniform even string, a uniform odd string, or a uniform string. And as seen in Example 1.13, it's not hard to construct them — their localities are 0, 0, 1, 2, 2, and 1 (respectively). So one way to view this result is that if you want to generate uniform symmetric distributions, having the locality d be a larger constant is the same as having it be 2.

§2 Proof overview

Next we'll talk a bit about the proof ideas. Instead of directly telling how we pick out these six distributions and rule out all others, we'll start with some examples of how to rule out specific concrete uniform symmetric distributions (whose ideas will be helpful in general). The first example will be the uniform distribution over strings of Hamming weight $n/3$; and the second example will be the majority distribution (the uniform distribution on strings of weight at least $n/2$). Then we'll talk a little about how you combine these ideas for the general case.

§2.1 Example 1: Granularity

Let's begin with the weight $n/3$ distribution. Assume $3 \mid n$; we want to sample from n-bit strings with Hamming weight exactly $n/3$. Why can't we do this?

The idea is a granularity argument. If you look at the marginal distribution of each bit in this target distribution, every individual bit will have bias $1/3$ (i.e., it's 1 with probability $1/3$). But $1/3$ is not a dyadic rational that you can generate using uniform bits — for example, if the output bit b_1 only reads the three input bits r_1, r_2, and r_5, then the probability density function of b_1 will have to be an integer multiple of $1/8$, which cannot approximate $1/3$ very well.

So by granularity, each output bit already produces some noticeable distance from the target distribution. And it turns out that by looking at *many* output bits, we can boost this to a distance that's close to 1. Specifically, if you can find K output bits that are independent, then by concentration your distance will accumulate to $1 - e^{-\Omega(K)}$. (We're still looking at the total variation distance.)

How do we get independent output bits? Just having a shallow circuit (i.e., a local function) doesn't guarantee this — they could all share a common input bit, for example. However, we *can* show the following:

Lemma 2.1 (Structural lemma)

By conditioning on $o(n)$ input bits, we can find $\Omega(n)$ independent output bits.

(This is a graph-theoretic result.)

Now let's do this conditioning. Then after the conditioning, we can find $\Omega(n)$ independent output bits, so we'll have *extremely* large distance, specifically $1-e^{-\Omega(n)}$. And we can union bound over all $2^{o(n)}$ possibilities for the conditioning, so the final distance will be

$$1 - 2^{o(n)} \cdot e^{-\Omega(n)} = 1 - e^{-\Omega(n)}.$$

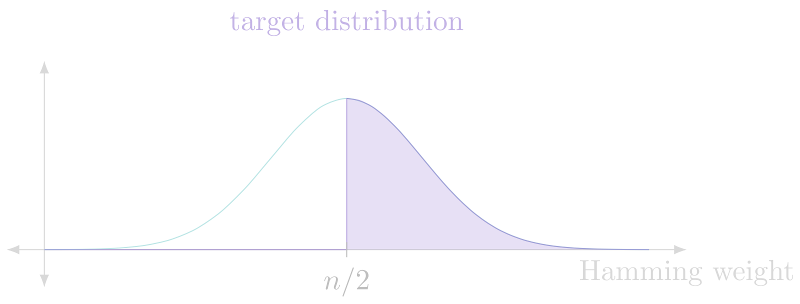
(It's crucial that the tradeoff in Lemma 2.1 has the $\Omega(n)$ term dominating the $o(n)$ term, so that the union bound doesn't become useless.)

So this is the idea for ruling out the distribution on strings of weight $n/3$, by exploiting granularity. But there are distributions that can't be ruled out by granularity, and we'll see one next.

§2.2 Example 2: Sharp cutoffs

Now we'll consider the uniform distribution over strings of Hamming weight at least $n/2$. Here, granularity doesn't really work — the marginal bias of each bit is extremely close to $1/2$, and producing a marginal bias of $1/2$ using unbiased bits is totally doable. So we'll need a different approach to rule this out.

The idea is to explore the sharp cutoff phenomenon in this distribution. Here there's a sharp cutoff at the point $n/2$ — on the right you have all the strings, and on the left you have none.



This doesn't really seem doable — $f(\mathcal{U})$ is the output of a shallow circuit, which means it shouldn't be able to generate a strong enough correlation to identify this cutoff at $n/2$.

How do we formalize this? We'll go back to the structural lemma (Lemma 2.1), which says that conditioning on a sublinear number of input bits, we have linearly many output bits. And if we sum up these independent output bits, then by the central limit theorem we'll basically get a Gaussian distribution. And a Gaussian distribution can't locate this cutoff at $n/2$, because a Gaussian is so smooth and has no sharp cutoff.

So this is the idea, but there's a caveat — this argument doesn't always work. The issue is that our total Hamming weight isn't just the sum of the independent output bits given by Lemma 2.1 — it *also* sums over the correlated output bits. So there being many independent output bits doesn't necessarily imply that the Hamming weight distribution of the *entire* string is Gaussian.

Example 2.2

Suppose our output is $(r_1, \neg r_1, r_2, \neg r_2, \ldots, r_{n/2}, \neg r_{n/2})$. Certainly there are a lot of independent bits, but the total Hamming weight is always exactly $n/2$.

So even if you don't do any conditioning, even if you're able to find lots of independent output bits, the Hamming weight could still be very concentrated.

How do we handle this? Instead of looking at individual bits, we'll group them together. In this example, instead of just looking at r_1, we'd group r_1 with $\neg r_1$, and r_2 with $\neg r_2$, and so on. We'll refer to these groups as *neighborhoods*. We won't formally define neighborhoods in general; but essentially, for each output bit you look at all the other output bits that are correlated with it, and that defines its neighborhood.

If $f(\mathcal{U})$ is uniform (on strings with weight at least $n/2$), then not only is every *individual* output bit close to unbiased, but every *pair* of outbut bits should also be roughly unbiased. And that's not the case here — each of these neighborhoods is far from unbiased (it'll always be either 01 or 10).

So this is a larger scale granularity argument — in the past we used individual output bits, but now we use granularity for neighborhoods. For this, we'll use an upgraded version of the structural lemma (Lemma 2.1), which guarantees that not only are our output *bits* independent, but their neighborhoods are too.

Lemma 2.3 (Upgraded structural lemma)

By conditioning on $o(n)$ input bits, we can find $\Omega(n)$ independent output neighborhoods.

Then there are two cases. One is that many of these neighborhoods are not close to unbiased (while they're supposed to be close to unbiased in the target distribution). Then each of these neighborhoods is at $\Omega(1)$ distance from its desired distribution; and because they're independent, the entire output distribution is at $1 - e^{-\Omega(n)}$ distance from the target by concentration.

Meanwhile, if most of these neighborhoods are close to unbiased, then our intuition works out — the weight distribution will look like a Gaussian, because we put all the correlations inside these neighborhoods and the neighborhoods are independent of each other. And then because a Gaussian is smooth, it can't generate this large cutoff.

So either way, we witness some large distance.

Remark 2.4. The intuition is that a sum of independent output bits is Gaussian, but this isn't enough because those independent bits could be correlated with stuff on the outside (which could cancel out their variance, as in Example 2.2); so we fix this by going one step further and taking neighborhoods. Then we're saying that the individual output *bits* whose neighborhoods are independent are random enough to generate something Gaussian-looking. We're *not* saying that *every* bit in these neighborhoods has sufficiently random contribution; this isn't true (some bits in these neighborhoods can have correlations with stuff on the outside; to prevent this, you'd need to expand one step, then two steps, and so on). But what we're saying is that because of neighborhood independence, these *single* bits are sufficiently random and independent. (This certainly needs proof.)

So that's the proof for strings of weight at least $n/2$. It introduces the new idea of using neighborhoods instead of individual output bits, and the idea of exploring the cutoff phenomenon in the target distribution.

§2.3 The general case

Now we'll talk about how we rule out any 'weird' uniform symmetric distribution $\mathcal{D}$. For example, maybe $\mathcal{D}$ has the strings of Hamming weight $n/2$ to $n/2 + \sqrt{n}$, but doesn't contain strings of Hamming weight $n/2 - \sqrt{n}$ to $n/2$. Or maybe you have Hamming weights $n/2$ and $n/2 + 10$, but you skip over $n/2 + 2$, $n/2 + 4$, and so on. These are all weird uniform symmetric distributions that we want to rule out.

First, we design a potential function h to witness the cutoffs. We should have

$$\mathbb{E}_{x \sim \mathcal{D}}[h(|x|)] \approx 1$$

(where $|x|$ is the Hamming weight of x) — so h is supposed to witness the cutoffs, but also to have significant mass under $\mathcal{D}$. On the other hand, for the Gaussian distribution $\mathcal{G}$, we should have

$$\mathbb{E}_{z \sim \mathcal{G}}[h(z)] \ll 1.$$

Then equipped with this potential function, we can review the previous analyses. We again use the upgraded structural lemma (Lemma 2.3) to find many independent neighborhoods. Then there's two cases — either the neighborhoods are close to the correct marginals, or they're far. If they're far from the correct marginals, then because they're independent, you can use concentration to show that you're far from the target distribution. On the other hand, if most are close to the correct marginals, then because they're independent, their weight distribution will be like a Gaussian by the central limit theorem. And this is impossible because of the potential function h (you shouldn't have $\mathbb{E}_{x\sim\mathcal{D}}[h(|x|)] \approx 1$ and $\mathbb{E}_{z\sim\mathcal{G}}[h(z)] \ll 1$ if $|x|$ is like a Gaussian).

So that's the argument.

Remark 2.5. You might recall that there are some interesting distributions that also seem to be ruled out by this argument but actually aren't, the even and odd distributions (i.e., strings of even weight or odd weight, as in Example 1.13). The reason why mod 2 isn't ruled out but mod 3 and mod 4 are is subtle, and goes into the technical details of the analysis. But at a very high level, what happens is if you take the binomial distribution mod 2, the resulting distribution over 0 and 1 is uniform. But if we take it mod other numbers (e.g., 3 or 4), it'll be *close* to uniform, but it's never actually uniform. So this is the key distinction that comes up in the calculation that distinguishes 2 from 3 or 4.

§3 Concluding remarks

To summarize, this result characterizes all locally sampleable uniform symmetric distributions — the ones in Example 1.13 are the only six examples.

Question 3.1. What if we drop the uniformity assumption, and only ask the distribution to be symmetric — what kinds of symmetric distributions can you sample with local functions?

This is work in progress. The authors think that the answer should be mixtures of evens, odd, and p-biased distributions where p is a dyadic rational, where the mixing weights are also dyadic rationals. It's not hard to see that you can sample these with local functions — you first use a few input bits to fix which distribution to sample from according to the mixing weights. Then given this, you sample from the even or odd or p-biased distribution, each of which is individually doable. The hard part is to say that these are the *only* examples; they're 80% confident this is true, and it's in progress.

Question 3.2. Can we improve the quantitative bounds?

So far, we've been ignoring all the constants in the asymptotic notation. But the formal statement is:

Theorem 3.3 (Kane–Ostuni–Wu 2025+)

If f is d-local and $n \geq \mathsf{tower}(d)$, and $f(\mathcal{U})$ is ε-close to a uniform symmetric distribution, then $f(\mathcal{U})$ is $(\varepsilon \cdot \mathsf{tower}(d))$-close to one of the six distributions in Example 1.13.

(The tower function is $\mathsf{tower}(d) = 2^{2^{2^{\cdot^{\cdot^{\cdot}}}}}$, with d 2's.)

This statement has two towers. The authors suspect that the second tower should actually be constant, and they have some idea for how to do that. But the first tower, where we say $n \geq \mathsf{tower}(d)$, turns out to be necessary for our analysis, because of the structural lemma. The structural lemma really *requires* tower-type dependence (in the structural lemma, you're fixing some input bits and want to make sure there's a lot of independent output neighborhoods; and the dependence between these really is tower-type). So a tower-type dependence is necessary for this analysis, and the authors don't know how to improve it; but they suspect that it should actually be *exponential* in d.

This is an interesting question because you can rephrase it in the following way: We know only the distributions in Example 1.13 are locally sampleable. But what about the other distributions — for example, what if I want you to output something close to the uniform distribution on Hamming weights 0 mod 4? I know you're not going to do it in *constant* locality, but how much locality do you need? This result rules out locality like $\log^* n$. But you might think the locality has to be much more — maybe it has to be linear (i.e., the output bits need to read linearly many input bits)? Improving this tower dependence would improve this type of result (what's the locality needed for other uniform symmetric distributions?), which is interesting.

Remark 3.4. The place where the tower-type dependence comes from (in the structural lemma) is that at each step, either you can already find a lot of independent output neighborhoods — which is good, and you can stop — or you can find some small set of inputs such that conditioning on them (i.e., removing them) will drop the average degree of the dependency graph. We iteratively do this; in the end, if you have the empty graph then everything is independent, so you must stop at some point in the process.

The lower bound construction for the structural lemma is constructed based on this iterative algorithm. We have this upper bound proof where every time, you prune something out and drop the average degree. And the construction has layers of stuff going on inspired by the upper bound construction, which leads to tower-type tradeoffs. (This construction is in the paper.)

It's possible that if we explored weak correlation — where instead of independence we have some type of weak dependence — maybe quantitative bounds from graph theory could help; but we don't know how to do that.