On the Range Avoidance Problem for Circuits

Hanlin Ren
Oxford

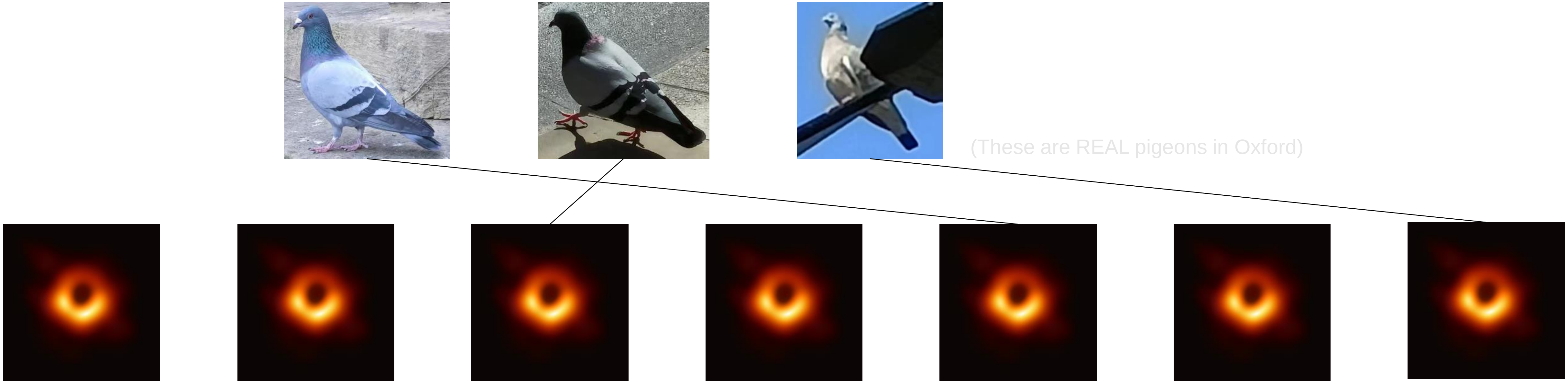
Rahul Santhanam
Oxford

Zhikun Wang
Xi'an Jiaotong

July 5, 2022

The Empty Pigeonhole Principle

- If you throw N pigeons into M holes, and $N < M$, then there is an empty pigeonhole.
- Weak version: if you throw N pigeons into M holes, and $2N < M$, then there is an empty pigeonhole.



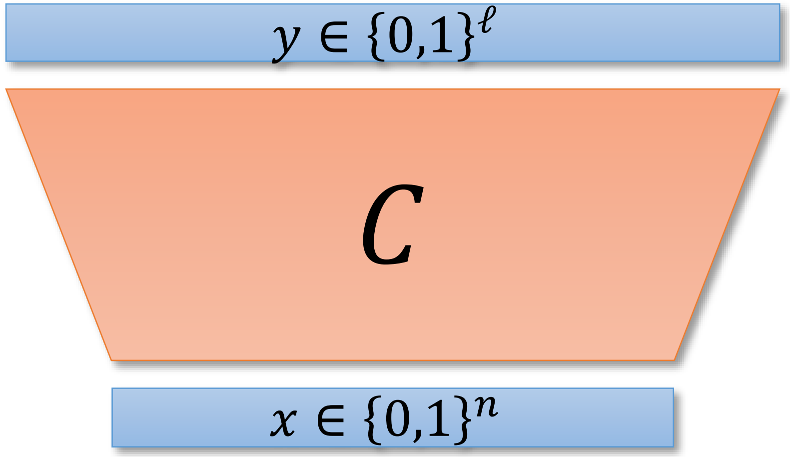
empty empty empty empty

Range Avoidance Problem

- Input: a (multi-output) circuit $C: \{0,1\}^n \to \{0,1\}^\ell$ ($\ell > n$)
- Output: any string $y \notin \mathrm{Range}(C)$
  - I.e. for every $x \in \{0,1\}^n$, $C(x) \neq y$

A "non-output" of C.

- A total problem in $\mathbf{TF\Sigma_2}$, complete for the class $\mathbf{APEPP}$ [ITCS'21]
  - Abundant Polynomial Empty Pigeonhole Principle



A problem that captures the complexity of weak empty pigeonhole principle!

$C(\text{index of pigeon}) = \text{index of hole}$, find an empty hole!

Explicit Constructions

- Big goal in TCS: construct pseudorandom objects deterministically
  - Ramsey graphs, rigid matrices, expander graphs, hard truth tables…
- Explicit construction problems:
  - RIGID: On input 1^n, output an $n \times n$ matrix that is rigid
  - RAMSEY: On input 1^n, output an n-vertex Ramsey graph
  - HARD: On input 1^{2^n}, output a length-2^n truth table without small circuits
- [Korten'21]: Range avoidance captures explicit constructions!
  - $\mathrm{RIGID}, \mathrm{RAMSEY}, \mathrm{HARD}, \ldots \in \mathbf{APEPP}$
- Sparse APEPP ($\boldsymbol{S}\mathbf{APEPP}$): unary problems reducible to Avoid

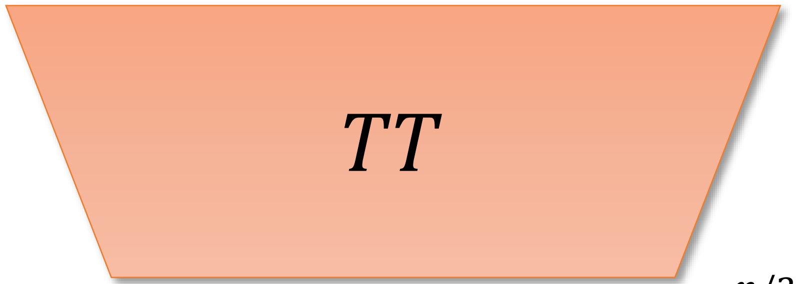
Example: Circuit Lower Bounds

- Weak empty PHP: most Boolean functions require circuits of > $2^{n/2}$ size!
- Embarrassing open Q: $\mathbf{E}^{\mathbf{NP}} \subseteq \mathbf{SIZE}[10n]$?

Truth table of C

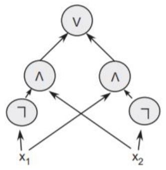
$01100010011 \ldots \ldots \ldots \ldots \ldots \ldots 001010$

2^n bits



$2^{n/2}\mathrm{poly}(n)$ bits

Description of a size-$2^{n/2}$ circuit C



If Avoid $\in$ $\mathbf{FP}$, then $\mathbf{E} \not\subseteq \mathbf{SIZE}\left[2^{n/2}\right]$.

If Avoid $\in$ $\mathbf{FP}^{\mathbf{NP}}$, then $\mathbf{E}^{\mathbf{NP}} \not\subseteq \mathbf{SIZE}\left[2^{n/2}\right]$.

(Trivial complexity upper bounds:
Avoid $\in$ $\mathbf{FBPP}$ and Avoid $\in$ $\mathbf{FZPP}^{\mathbf{NP}}$)

TT stands for “truth table”.

The Complexity of Avoid

- Korten (FOCS'21): $\text{Avoid} \in \mathbf{FP}^{\mathbf{NP}}$ if and only if $\mathbf{E}^{\mathbf{NP}} \not\subseteq \mathbf{SIZE}[2^{0.1n}]$

Under "plausible"(?) assumptions, $\text{Avoid} \in \mathbf{FP}^{\mathbf{NP}}$
"Plausible" = widely studied and believed

Actually, the following are equivalent:

- $\text{Avoid} \in \mathbf{FP}^{\mathbf{NP}}$
- There is some $\epsilon > 0$ such that $\mathbf{E}^{\mathbf{NP}} \not\subseteq \mathbf{SIZE}[2^{\epsilon n}]$
- $\mathbf{E}^{\mathbf{NP}} \not\subseteq \mathbf{SIZE}[2^n/2n]$

Question 1: is $\text{Avoid} \in \mathbf{FP}$?

Is "$\text{Avoid} \in \mathbf{FP}$" or its negation implied by any plausible assumption?

Question 2: is $\text{Avoid} \in \mathbf{FNP}$?

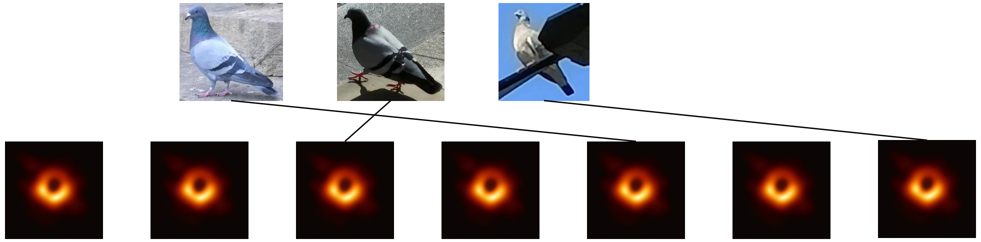
Is "$\text{Avoid} \in \mathbf{FNP}$" or its negation implied by any plausible assumption?

It's entirely conceivable that you can take a circuit as input, "scramble" it using some fancy crypto stuff, and somehow produce a non-output in poly-time…

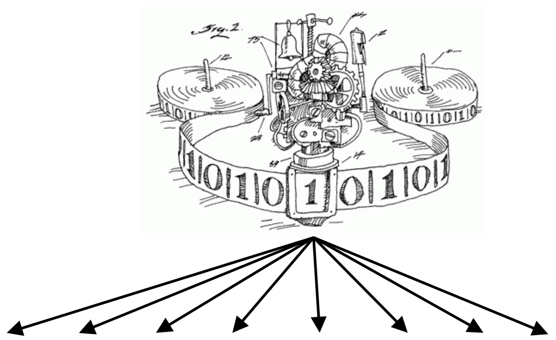
Recap: $\mathbf{FNP}$ Algorithms

- Nondeterministic poly-time algorithms which
  - Accepts at least one nondeterministic branch
  - On each accepted branch, outputs a valid answer

Input:



Valid answers: $\{1,2,4,6\}$



Recap: Proof Complexity

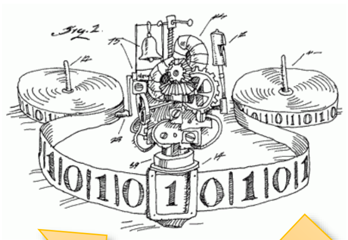
- Propositional Proof Systems (PPS): nondet. algorithms for $\overline{SAT}$
  - A deterministic poly-time algorithm $P(\varphi, y)$ that gets a formula φ and a "proof" y (of the claim that φ is **un**satisfiable)
  - φ is **un**satisfiable iff $\exists y$, $P(\varphi, y) = 1$.
  - $\mathrm{PfLen}_P(\varphi) = \min\{|y|: P(\varphi, y) = 1\}$

If φ is satisfiable then $\mathrm{PfLen}_P(\varphi) = +\infty$.

- Proof complexity generators: $G: \{0,1\}^n \to \{0,1\}^\ell$
  - Want: it is hard to prove that "$y \notin \mathrm{Range}(G)$" for every y, despite this formula being true for most y
  - A sequence of generators $\left\{G_n: \{0,1\}^n \to \{0,1\}^{\ell(n)}\right\}_{n\in\mathbb{N}}$ is **hard** for P if for every $y \in \{0,1\}^{\ell(n)}$, $\mathrm{PfLen}_P(\text{"}y \notin \mathrm{Range}(G_n)\text{"}) \geq \ell(n)^{\omega(1)}$.
  - Uniform generator: there is an algorithm that given x, computes $G_{|x|}(x)$

FNP Algorithms for Avoid vs. Proof Complexity Generators

- **Theorem:** The following are equivalent:
  - Avoid $\in$ **FNP**
  - There exists a PPS breaking every (non-uniform) proof complexity generator

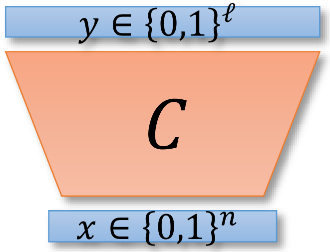


A PPS breaking every generator

FNP algorithm for Avoid: Guess y along with a proof that $y \notin \mathrm{Range}(C)$

Input circuit = Proof complexity generator

FNP algorithm = PPS



a proof that $y \notin \mathrm{Range}(C)$ for some particular y

FNP Algorithms for $\mathbf{Avoid}$ vs. Proof Complexity Generators

- **Theorem:** The following are equivalent:
  - $\mathbf{SAPEPP} \subseteq \mathbf{FNP}$
  - There exists a PPS breaking every uniform proof complexity generator

Recap: $\mathbf{SAPEPP}$ = unary explicit construction problems
"Given 1^n, find a Ramsey graph over n nodes"

The K^t Generator

- Fix a polynomial t. For a string x, $\mathrm{K}^t(x)$ is the length of the shortest program that generates x in $t(|x|)$ steps.

“time-bounded Kolmogorov complexity”

- The K^t generator with stretch $\alpha(n)$:
  - Input is a program prog of length $n - \alpha(n)$
  - Simulate prog for $t(n)$ steps
  - If the output of prog has length exactly n, output whatever it outputs
  - Otherwise, output 0^n
- Corresponding problem in $\mathbf{SAPEPP}$: K^t-HARD
  - On input 1^n, generate a string $y \in \{0,1\}^n$ such that $\mathrm{K}^t(y) > n - \alpha(n)$

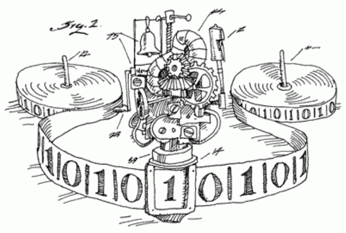
FNP Algorithms for Avoid vs. Proof Complexity Generators

- **Theorem:** The following are equivalent:*
  - $\mathbf{SAPEPP} \subseteq \mathbf{FNP}$
  - There exists a PPS breaking every uniform proof complexity generator
  - There exists a PPS breaking the K^t generator for $t(n) = n^2$
  - $\mathrm{K}^t\text{-HARD} \in \mathbf{FNP}$

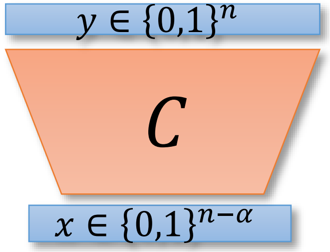
*: up to $\omega(1)$ factors in the stretch

ou can replace n^2 by any
easonable" polynomial ☺

There is a hardest proof complexity generator (K^t)!



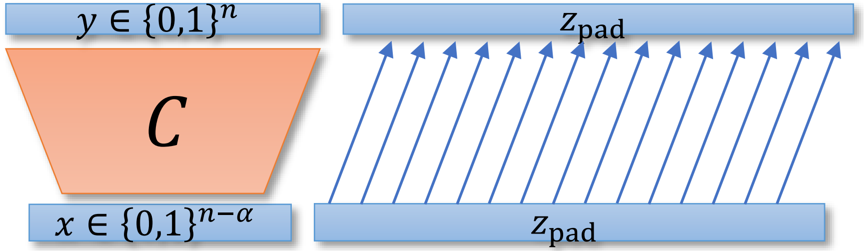
A PPS breaking the K^t generator



A uniform generator

Easy case: the time complexity of C is at most $t(n)$
… Then for every $y \in \mathrm{Range}(C)$, we have $\mathrm{K}^t(y) \le n - \alpha + O(1)$!!!

Harder case: the time complexity of C is $\gg t(n)$
… Use a padding argument!



$\mathbf{FNP}$ Algorithms for $\mathbf{Avoid}$ vs. Proof Complexity Generators

*: up to $\omega(1)$ factors in the stretch

- **Theorem:** The following are equivalent:*
  - $\mathrm{Avoid} \subseteq \mathbf{FNP}$
  - There exists a PPS breaking every (non-uniform) proof complexity generator
  - cK^t-$\mathrm{HARD} \in \mathbf{FNP}$

- Conditional K^t complexity:
  - $\mathrm{K}^t(x \mid y) =$ length of the shortest program that given y outputs x in time $t(|x| + |y|)$
- cK^t-HARD: On input $(1^n, y)$, output any string $x \in \{0,1\}^n$ such that $\mathrm{K}^t(x \mid y) > n - \alpha(n)$

$\mathbf{FP}$ Algorithms for $\mathbf{Avoid}$ vs. Time Hierarchy against Advice

*: up to $\omega(1)$ factors in the stretch

- **Theorem:** The following are equivalent:*
  - $\mathbf{SAPEPP} \subseteq \mathbf{FP}$
  - $\mathrm{K}^t\text{-HARD} \in \mathbf{FP}$
  - There is a language $L \in \mathbf{E} \setminus \mathrm{i.o.}\mathbf{DTIME}[2^{n+1}]_{/(2^n-\omega(1))}$

Time hierarchy against (near-maximum) advice...

Other Results (Advertisement)

- An "Algorithmic Method" to solve Avoid unconditionally in $\mathbf{FP}^{\mathbf{NP}}$
  - Generalising the "Algorithmic Method" for proving lower bounds for $\mathbf{E}^{\mathbf{NP}}$
- Characterisation of lower bounds for $\mathbf{E}^{\mathbf{NP}}$

Williams'11:
$\mathbf{NEXP} \not\subseteq \mathbf{ACC}^0$

- Reductions between Avoid for low-complexity circuits
  - Avoiding $\mathbf{NC}_4^0$ (4-local) circuits is as hard as avoiding $\mathbf{NC}^1$ circuits!
  - Uses the randomised encodings of Applebaum-Ishai-Kushilevitz (SICOMP'06)
- Welcome to talk to me about these results!

Finally, A Hypothesis…

- Hypothesis: $\mathrm{K}^{\mathrm{poly}}$-$\mathrm{HARD}$ is solvable in $\mathbf{FP}$.
  - Given $(1^t, 1^n)$, one can find, in det. polynomial time, a string $x \in \{0,1\}^n$ such that $\mathrm{K}^t(x) \geq s(n)$.
  - Essentially equivalent to $\mathbf{SAPEPP} \subseteq \mathbf{FP}$ for $C : \{0,1\}^{s(n)} \to \{0,1\}^n$.
- Q: How plausible is this hypothesis?
  - For $s(n) = \log^2 n$? $n^{0.1}$? $0.1n$? $0.999n$? $n - n^{0.9}$? $n - \log^2 n$? $n - 1$???
  - If you believe in circuit lower bounds (hard truth tables are easy to generate), should you also believe in this hypothesis ($\mathrm{K}^{\mathrm{poly}}$-random strings are easy to generate)?
- Q: How is this hypothesis connected to other parts of complexity theory?
- Q: Is cK^t-HARD in $\mathbf{FP}$? Less secure, but how much less?

New hypothesis for derandomization?

Summary

- Range avoidance problem
  - Captures explicit constructions!
  - “Plausibly” in $\mathbf{FP}^{\mathbf{NP}}$, but unknown if it’s in $\mathbf{FNP}$ or even $\mathbf{FP}$
- $\mathrm{Avoid} \in \mathbf{FNP}$ if and only if there is a PPS breaking every proof complexity generator
  - $(\mathrm{c})\mathrm{K}^t$ generator is the hardest one!
- Hypothesis: we can generate strings of large K^t complexity, deterministically

Thank you!

Questions are welcome! ☺