Strong ETH Breaks With Merlin and Arthur: Short Non-Interactive Proofs of Batch Evaluation

August 19, 2020

Reminder

- Conjunctive normal form of Boolean formula is a conjunction of clauses, where a clause is a disjunction of literals.
- Example: $(x_1 \vee \neg x_2) \wedge \neg x_1$
- **CNFSAT** problem: Given a Boolean formula in conjunctive normal form. Is there any satisfying assignment of it?
- For example, formula above is satisfied if $x_1 = 0$ and $x_2 = 0$.
- **k-SAT** problem: Given a Boolean formula in conjunctive normal form where each clause has at most k literals. Is there any satisfying assignment of it?

Hardness of k-SAT

- How fast can we solve **k-SAT**?
- Naive algorithm requires $O(2^n)$ time.
- There is a $O(1.334^n)$ algorithm for **3-SAT**.
- **k-SAT** is known to be NP-complete for $k > 2$, so we don't have a polynomial time algorithm for it unless $\mathrm{P} = \mathrm{NP}$, which is unlikely.
- Can we solve **3-SAT** in $O(2^{\sqrt{n}})$ time?
- Can we solve **k-SAT** in $O(1.999^n)$ time for any k?
- Are there any (conditional) lower bounds for **k-SAT**?

Hardness of k-SAT

- P $\neq$ NP hypothesis: **3-SAT** cannot be solved in polynomial time.
- **ETH** (exponential time hypothesis): **3-SAT** requires $O(2^{\delta n})$ time for some $\delta > 0$.
- **SETH** (strong exponential time hypothesis): **k-SAT** requires $O(2^{n-o(n)})$ time for unbounded k.

Co-nondeterministic algorithms

- **k-SAT** is NP problem, so for any YES-instance there is a proof that can be verified in polynomial time.
- How fast can we verify that instance of **k-SAT** is a NO-instance?
- If there is a polynomial-time proof that instance is a NO-instance, NP $=$ co-NP which leads to PH collapse, which is unlikely.
- Trivial proof of k-CNF unsatisfability is a Boolean formula itself and cannot be verified in $O(2^{n-o(n)})$ time for unbounded k under **SETH**.
- Is there more efficient proof of k-CNF unsatisfability?

NSETH

- **NSETH** (nondeterministic strong exponential time hypothesis): co-nondeterministic algorithm for **k-SAT** requires $O(2^{n-o(n)})$ time for unbounded k.
- NB: Verifying algorithm should be deterministic.
- What if we allow randomization in verifier?

Arthur and Merlin games

- Interactive proof system, introduced by Babai in 1985.
- Arthur is a verifier that has an access to random numbers.
- Merlin is a prover with infinite computational power.

Arthur

Merlin

MA protocol

- MA (Merlin-Arthur) protocol is a 1-message protocol in which Merlin just sends Arthur the proof.
- Protocol is correct if for any YES instance there is a proof, that Arthur will accept with probability at least $\frac{2}{3}$ and for any NO instance Arthur will accept any proof with probability at most $\frac{1}{3}$.
- Hardness of protocol is a time Arthur should spend to verify the proof in worst case.
- MA is a class of languages that has polynomial MA protocol.
- Quite similar to NP but allows randomization.

AM[2] protocol

- AM[2] (Arthur-Merlin) protocol is a 2-message protocol.
- At first, Arthur tosses some random coins and sends their outcomes to Merlin.
- After that Merlin sends Arthur the proof.
- Protocol is correct if for any YES-instance there is a proof, that Arthur will accept with probability at least $\frac{2}{3}$ and for any NO-instance Arthur will accept any proof with probability at most $\frac{1}{3}$.
- Hardness of protocol is a time Arthur should spend to verify the proof in worst case.
- AM is a class of languages that has polynomial AM[2] protocol.

Hardness of k-SAT, again

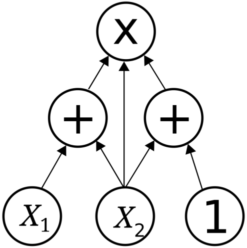
- P $\neq$ NP hypothesis: **3-SAT** cannot be solved in polynomial time.
- **ETH** (exponential time hypothesis): **3-SAT** requires $O(2^{\delta n})$ time for some $\delta > 0$.
- **SETH** (strong exponential time hypothesis): **k-SAT** requires $O(2^{n-o(n)})$ time for unbounded k.
- **NSETH** (nondeterministic strong exponential time hypothesis): co-nondeterministic algorithm for **k-SAT** requires $O(2^{n-o(n)})$ time for unbounded k.
- **MASETH** (Merlin-Arthur strong exponential time hypothesis): There is no $O(2^{n(1-\varepsilon)})$ MA protocol for **k-UNSAT** for any $\varepsilon > 0$ and unbounded k.
- **AMSETH** (Arthur-Merlin strong exponential time hypothesis): There is no $O(2^{n(1-\varepsilon)})$ AM protocol for **k-UNSAT** for any $\varepsilon > 0$ and unbounded k.

Today's results

- **AMSETH** and **MASETH** are false! Today we will mostly talk about that.
- In particular we will show a $O^*(2^{n/2})$ **MA** protocol for **UNSAT** which requires only $\frac{n}{2} + o(n)$ coin tosses for Arthur.
- If we have more time we will talk about some important consequences of **NSETH**.

Arithmetic Circuits Complexity

- Arithmetic circuit is a way to represent polynomials over some field.



Arithmetic circuit, representing polynomial $x2(x1 + x2)(x2 + 1)$

- Each gate is either a variable, element from field or operator $+$ or $\times$.
- **Size** of a arithmetic circuit is a number of gates in it.
- **Degree** of a arithmetic circuit is a degree of a polynomial that it represents.

Multipoint Arithmetic Circuit Evaluation

- Suppose we have a circuit $C(x_1, x_2, \ldots, x_n)$ of size s, degree d and bounded fan-in over a field $\mathbb{F}$ and k points $a_1, a_2, \ldots, a_k \in \mathbb{F}^n$.
- We want to compute k values $c_i = C(a_i)$.
- Naive algorithm requires $\Theta(ks)$ time.
- We want to do it faster.
- However we don't know how to do it :(
- Maybe our friend Merlin can help us?

Verifying Multipoint Arithmetic Circuit Evaluation

- Suppose we have a circuit $C(x_1, x_2, \ldots, x_n)$ of size s, degree d and bounded fan-in over a field $\mathbb{F}$ and k points $a_1, a_2, \ldots, a_k \in \mathbb{F}^n$.
- We want to design an efficient **MA** proof system for arithmetic circuit multipoint evaluation.

Theorem 3.1

For every prime power q and $\varepsilon > 0$, multipoint arithmetic circuit evaluation for k points in $(\mathbb{F}_q)^n$ on an arithmetic circuit C of n inputs, s gates, and degree d has an MA-proof system where:

- Merlin sends a proof of $O(k \cdot d \cdot \log(kqd/\varepsilon))$ bits, and
- Arthur tosses at most $\log(kqd/\varepsilon)$ coins, outputs $(C(a_1), \ldots, C(a_k))$ incorrectly with probability at most ε, and runs in time $(k \cdot \max\{d, n\} + s \cdot poly(\log s)) \cdot poly(\log(kqd/\varepsilon))$.

Theorem 3.1 (without screamers)

- Theorem 3.1 seems scary. However we need it to refute **MASETH**, and will prove it.
- Let's see some intuition behind this creepy formulas.
- Size of the problem instance is $O((kn + s\log s) \cdot logq)$, so protocol complexity is almost linear up to the d factor.
- Hence protocol is almost optimal for small degrees.
- Surprisingly, many important problems have small circuit degrees.
- We will prove theorem 3.1 later and now will refute **MASETH** using it.

All-input Arithmetic Circuit Sum

We are given a prime p, an $\varepsilon > 0$ and an arithmetic circuit $C(x1, x2, \dots, x_n)$ with degree d and size s and want to compute the sum $\sum_{(x1,x2,\dots,x_n)\in\{0,1\}^n} C(x1, x2, \dots, x_n) \mod p$.
It can be done by a Merlin-Arthur protocol running in $O(2^{\frac{n}{2}} \cdot poly(n, s, d, log(\frac{p}{\varepsilon})))$ time and tossing $\frac{n}{2} + O(log(\frac{pd}{\varepsilon}))$ coins.

All-input Arithmetic Circuit Sum (proof)

- We will prove it using Theorem 3.1.
- Suppose that n is even.
- Consider an expression $\sum_{(x_1, x_2, \dots x_{\frac{n}{2}}) \in \{0,1\}^{\frac{n}{2}}} C(x_1, x_2, \dots, x_n)$.
  This is a some arithmetic circuit $C'(\frac{n}{2}+1, \frac{n}{2}+1, \dots, n)$.
- It's clear that some of values of a C' over all possible arguments is a sum of values of a C over all possible values. So instead of getting 2^n values of C and summing them, we can get $2^{\frac{n}{2}}$ values of C' and sum them.
- How can we get these $2^{\frac{n}{2}}$ values? We can ask Merlin and theorem 3.1 for help!

All-input Arithmetic Circuit Sum (proof)

- Suppose $deg(C) = d$ and $size(C) = s$.
- Then $deg(C') \leq d$ and $size(C') \leq 2^{\frac{n}{2}} \cdot s$.
- By simple substitution of these values into theorem 3.1 we get **MA** protocol for All-input Arithmetic Circuit Sum with $O(2^{\frac{n}{2}} \cdot poly(n, s, d, log(\frac{p}{\varepsilon})))$ time and $\frac{n}{2} + O(log(\frac{pd}{\varepsilon}))$ coins.
- QED.

MA protocol for **UNSAT**

- We can transform *CNF* formula F into arithmetic circuit C such that $\forall(x_1, x_2, \ldots, x_n) : F(x_1, x_2, \ldots, x_n) = C(x_1, x_2, \ldots, x_n)$.
- Transformation rules are quite simple.
- $C(x) = x$.
- $C(\neg F) = 1 - C(F)$.
- $C(F_1 \wedge F_2) = C(F_1) \cdot C(F_2)$.
- $C(F_1 \vee F_2) = C(F_1) + C(F_2) - C(F_1) \cdot C(F_2)$.
- Suppose F has n variables and m clauses.
- Then $deg(C(F)) = O((n+m)^{O(1)})$ and $size(C(F)) = O(n+m)$.

MA protocol for **UNSAT**

- Now we just want to check if $\sum_{(x1,x2,\ldots,x_n)\in\{0,1\}^n}(C(F))(x1,x2,\ldots,x_n) > 0$.
- Pick a prime p between 2^n+1 and $2^{n+1}+2$ (such a prime exists because of Bertrand postulate) and compute $\sum_{(x1,x2,\ldots,x_n)\in\{0,1\}^n}(C(F))(x1,x2,\ldots,x_n) \mod p$. This prime can be sent by Merlin and verified by Arthur.
- This can be done using All-input Arithmetic Circuit Sum **MA** protocol described above.
- And thus **MASETH** (and **AMSETH**) is false!

Proof of the theorem 3.1

- Now we are going to prove theorem 3.1.
- But before we have to do one important thing.


CHILL TIME!
ADDTEXT.COM

Proof of the theorem 3.1

- Proof of the theorem consists of some nice ideas and pretty technical work.
- At first high-level structure of the proof will be described to get the key idea.
- Then technical part will be described briefly.

Proof of the theorem 3.1

- Reminder: we have an arithmetic circuit $C(x_1, x_2, \ldots, x_n)$ over $\mathbb{F}_q$ with degree d, s gates and bounded fan-in and k points $a_1, a_2, \ldots a_k \in \mathbb{F}_q^n$. We want to know $C(a_i)$ for all i with error probability at most ε.
- We build an extension field $F = \mathbb{F}_{q^l}$ for *some* l.
- Then we take arbitrary subset S of F with cardinality k and build arbitrary bijection between elements of S and a_i. Let $\Phi : S \to a$ be such a mapping.
- Now let's build a family of polynomials $\Psi_j : F \to F$ such that $\Psi_j(\alpha) = \Phi(\alpha)[j] \; \forall \alpha \in S$.
- Define a univariate polynomial $R(x) = C(\Psi_1(x), \Psi_2(x), \ldots, \Psi_n(x))$.
- It's clear that $R(\alpha) = C(\Psi_1(\alpha), \Psi_2(\alpha), \ldots, \Psi_n(\alpha)) = C(\Phi(\alpha)[1], \Phi(\alpha)[2], \ldots, \Phi(\alpha)[n]) = C(\Phi(\alpha))$, so $C(a_i) = R(\Phi^{-1}(a_i))$.

Proof of the theorem 3.1

- It turns out that R has small degree, so Arthur can evaluate it in all $\alpha \in S$.
- However Arthur can't get R's coefficients explicitly: he only knows that $R(x) = C(\Psi_1(x), \Psi_2(x), \ldots, \Psi_n(x))$, but getting coefficients of R from circuit is too slow.
- And here comes Merlin which sends coefficients of $R(x)$ to Arthur.
- Now Arthur have to check whether polynomial represented by circuit $R(x)$ and polynomial represented by Merlin's coefficients are the same.
- Arthur just takes random point from F and evaluates both circuit and Merlin's polynomial in this point.
- If values are the same, proof is accepted and rejected otherwise.

Proof of the theorem 3.1

- Now let's discuss technical aspects of the protocol described above.
- We are working with polynomials over the field $\mathbb{F}_{q^l}$.
- Two univariate polynomials can be multiplied in $mult(n) = O(n \log^2 n)$ time.
- Univariate polynomial of a degree n can be interpolated by values in $n+1$ points in $O(mult(n) \cdot logn) = O(n \log^3 n)$ time.
- Univariate polynomial of a degree n represented by vector of coefficients can be evaluated in n points in $O(mult(n) \cdot logn) = O(n \log^3 n)$ time.

Proof of the theorem 3.1

- We are working in an extension field $F = \mathbb{F}_{q^l}$ for such l that $l > \frac{dk}{\varepsilon}$.
- Field can be described by a irreducible polynomial generated by Merlin and Arthur can verify irreducibility by Kedlaya-Umans' test efficiently.
- Polynomials Ψ_i can be evaluated by Arthur using simple polynomial interpolation.
- Values of $R(x)$ for all $\alpha \in S$ can be evaluated by Arthur using univariate multipoint polynomial evaluation.
- Time analysis won't be presented here since it's quite long and boring.
- (almost) QED.

Summary

- We have an efficient **MA** protocol for multipoint arithmetic circuit evaluation and all-input arithmetic circuit sum.
- Using it we refuted **MASETH**, however presented framework is quite generic and can be used for creation of many different **MA** protocols.
- For example, we can create efficient **MA** protocol for permanent (and thus any problem in **VP** class), number of Hamiltonian cycles and much more...
- It's exciting that our protocol use really few coin tosses.
- Maybe it can be derandomized and **NSETH** can be refuted?

Derandomization

- Derandomization seems hard.
- Obvious way is the replacement of k coin tosses with checking 2^k variants leads to $O^*(2^n)$ algorithm with is no better than trivial.
- One can see that bottleneck of a derandomization is a polynomial identity testing problem. This problem was studied for a long time, however we still don't have efficient deterministic algorithms for it.
- Interesting consequence is that **NSETH** implies that polynomial identity testing problem cannot be solved in $O(n^{2-\varepsilon})$ time for any $\varepsilon > 0$.

The End

That's it!

Or not? (if we have time)

Why NSETH is important?

- People really want to refute **SETH**, however **NSETH** is more strong than **SETH** and refuting **NSETH** before **SETH** seems a good idea.
- However, there is another exciting consequence of a **NSETH**. One can rule out deterministic fine-grained reductions under **NSETH**!

3-SUM conjecture

- **3-SUM** problem: given n integers $a_i \in [-n^3; n^3]$ are there three distinct indices i, j, k such that $a_i + a_j + a_k = 0$?
- $O(n^3)$ algorithm is straightforward, $O(n^2)$ algorithm is easy too.
- Can we do better?
- **3-SUM** conjecture: **3-SUM** cannot be solved in $O(n^{2-\varepsilon})$ time for any $\varepsilon > 0$.
- Can we reduce **CNFSAT** to **3-SUM** and thus show that there is no $O(n^{2-\varepsilon})$ algorithm for **3-SUM** under **SETH**?

3-SUM conjecture and NSETH

- At least one of the following is true:
  1. **3-SUM** conjecture is false.
  2. **NSETH** is false.
  3. There are no **deterministic** fine-grained $(2^{n-\varepsilon}, n^{2-\delta})$ reductions from **CNFSAT** to **3-SAT** and thus **3-SUM** is hard not because of **CNFSAT**.
- So **NSETH** ruled out reduction.
- Let's prove it!

Anatomy of a Reduction

- Suppose we have a $(2^{n(1-\varepsilon)}, n^{2-\delta})$ reduction from **CNFSAT** to **3-SUM**.
- It converts instance P of **CNFSAT** of size n into instances Q_i of **3SUM** such that $\sum |Q_i|^{2-\delta} \leq 2^{n(1-\varepsilon)}$.
- Answers for Q_i are sufficient to determine the answer for P.
- Both times to create Q_i's and restore the answer are $O(2^{n(1-\varepsilon)})$.

Anatomy of a Reduction

- Suppose we want to design a co-nondeterministic algorithm for **CNFSAT** using reduction to **3-SUM**.
- This algorithm consists of three phases:
  1. Create instances Q_i of **3-SUM** from instance P of **CNFSAT** as in usual reductions.
  2. Check proof for instances Q_i of **3-SUM** that they are YES or NO instances.
  3. Prove that **CNFSAT** instance is a NO instance using Q_i's outcomes as in usual reductions.
- If second phase can be done in $O(n^{2-\delta})$ time for both YES and NO instances, we get a $O(2^{n(1-\varepsilon)})$ co-nondeterministic algorithm for **CNFSAT** which does not exists under **NSETH**.
- Nondeterministic algorithm for **3-SUM** is trivial: proof is just a required triple, so the only interesting thing here is co-nondeterministic algorithm for **3-SUM**.

Anatomy of a Reduction

- So under **NSETH** at least one of the following is true.
  1. There are no deterministic $(2^{n(1-\varepsilon)}, n^{2-\delta})$ reductions from **CNFSAT** to **3-SAT**.
  2. There is no $O(n^{2-\varepsilon})$ co-nondeterministic algorithm for **3-SUM**.
- Now we will show that (2) is not the case by showing $O(n^{\frac{3}{2}} \cdot poly(logn))$ co-nondeterministic algorithm for **3-SUM**.

Co-nondeterministic algorithm for **3-SUM**

- Consider a modulo p relaxation of a **3-SUM** problem.
- We call a triple of different indices (i, j, k) *good* iff $a_i + a_j + a_k \equiv 0 \pmod{p}$.
- Algorithm guesses a prime p such that there are at most $\alpha \cdot n^{\frac{3}{2}} \cdot \log n$ good triples in modulo p relaxation and guesses these triples.
- Such a proof can be easily validated using FFT in $O(p \log p)$ time.
- After that we just check whether $a_i + a_j + a_k \neq 0$ for all good triples.
- Now we have to prove that such a prime exists for all NO instances and find the α constant.

Prime existence proof

- Since $a_i \in [-n^3; n^3]$ for all triples (i, j, k) $a_i + a_j + a_k \in [-3n^3; 3n^3]$.
- For all triples (i, j, k) such that $a_i + a_j + a_k \neq 0$ there are at most $\log(3n^3)$ primes p such that $a_i + a_j + a_k \equiv 0 \pmod p$.
- By prime number distribution theorem there are at least $\frac{n^{\frac{3}{2}}}{\beta}$ primes below $n^{\frac{3}{2}} \log n$.
- So, if there are no triples that sums up to zero, there exists a prime p with at most $\frac{n^3 \log(3n^3)}{n^{\frac{3}{2}}/\beta} = n^{\frac{3}{2}} \log(3n^3)\beta$ good triples.
- That finishes our proof and shows the way to get α constant.

The End