No non-uniform lower bounds implies CSAT is extremely hard.

Amnon Ta-Shma and Dean Doron

In the previous lecture we have seen that if there are no non-uniform lower bounds (namely, if $\mathsf{NEXP} \subseteq \mathsf{P/poly}$ and PERM $\in \mathsf{AP/poly}$) then PIT is hard (and in particular does not derandomize completely). In this lecture we will see a result of Ryam Williams [1] showing that if there are no non-uniform bounds (namely, $\mathsf{NEXP} \subseteq \mathsf{P/poly}$) then CSAT is extremely hard.

1 Succinct representations

Definition 1. *We say a string $w \in \{0,1\}^n$ is represented by a circuit C with $\log n$ inputs, if $C(i) = w_i$.*

For example, if $f : \{0,1\}^n \to \{0,1\}$ is a Boolean function then its truth table is a string $T_f \in \{0,1\}^{2^n}$, where the entry indexed by $x \in \{0,1\}^n$ is $T_f(x) = f(x)$. Now, suppose C is a Boolean circuit with n inputs computing f, then C represents the truth-table of f, because for every x, $T_f(x) = f(x) = C(x)$.

Similarly, we can represent formulae.

Definition 2. *Suppose $\varphi(x_1, \ldots, x_n)$ is a* 3SAT *formula on n variables,*

$$\varphi = \wedge_{i=1}^{m} C_i(x_1, \ldots, x_n).$$

We say a circuit C represents φ if on input $i \in [m]$ C_i returns the i-th clause.

Such a representation is called *succinct*. Succinct representations are *local* in the sense that it provides means for efficiently computing the local segments of φ.

The reason for the name *succinct* is that succinct representations might be much smaller than "global" representations that output the whole formula φ. To see that consider the case where C is a polynomial size circuit (polynomial in its input length $\log m$) succinctly representing φ. Then, any circuit "globally" representing φ must have m size, as it has to at least output the m output bits representing φ, whereas the circuit C has size $\mathrm{poly}(\log m)$. A simple counting argument shows almost all formulas over n variables with m clauses are *not* succinct. Thus, succinct formulas form a very special and restricted subclass of formulas.

We now define Succinct3SAT:

Definition 3. *The input to the language* Succinct3SAT *is a circuit C succinctly representing a* 3SAT *formula φ. The input is in the language iff $\varphi \in$* SAT.

Succinct3SAT is NEXP complete and with reductions that almost preserve size:

Theorem 4. Succinct3SAT *is NEXP-complete (under polynomial-time reductions). Furthermore, for every language $L \in \mathsf{NTIME}(2^n)$ there is a reduction $L \leq_{\varphi}$* Succinct3SAT *such that:*

- *For every $x \in \{0,1\}^n$, $\varphi(x)$ is a circuit on $\ell = \log m$ bits of size $O(\ell^4)$, describing a* 3SAT *formula with $m = cn^4 2^n$ clauses, where c and and d are constants depending on the language L alone, and,*
- *$\varphi(x)$ runs in poly(ℓ) time.*

The moral is that uniform computation is succinct by nature, and when we reduce to SAT on many variables, we better remember to record this property.

We remark that a scaled down version of this also exits for $\mathsf{NTIME}(n)$. Also, the PCP theorem gives a version where we end up with a *gap*, i.e., either C describes a satisfiable formula, or a formula where every assignment does not satisfy a constant fraction of its clauses. A *succinct* version of $\mathsf{NTIME}[n]$ with a *constant gap* and *almost no expansion* exists and follows from the existence of short PCPs. We do not need this for this lecture.

2 The IKW result revisited

A few lectures ago we saw the easy witness method and the IKW result. The result was that if $\mathsf{NEXP} \subseteq \mathsf{P/poly}$ then $\mathsf{NEXP} = \mathsf{EXP}$ (hence $\mathsf{NEXP} = \mathsf{MA}$). The reasoning was as follows: suppose:

- $\mathsf{NEXP} \subseteq \mathsf{P/poly}$ (and therefore also $\mathsf{EXP} \subseteq \mathsf{P/poly}$ and $\mathsf{EXP} = \mathsf{MA}$), but,
- $\mathsf{NEXP} \neq \mathsf{EXP}$

then there must be some language $L \in \mathsf{NEXP}$ that is solved by some non-deterministic machine $M(x,y)$, such that (infinitely often) there exists an input $x_n \in L$ that has no *easy* witness, i.e., a witness that represents the truth table of an easy function.

Having that IKW prove that $\mathsf{EXP} = \mathsf{MA} \subseteq io{-}\mathsf{NTIME}(2^{n^a})|n$ for some constant number a. The idea is that the n bits of advice give the input x_n (for infinitely often input lengths), the machine guesses an accepting witness, and any accepting witness must represent a polynomially-hard function that can reduce the complexity of 2^{n^b} of L to 2^{n^a}. However, we proved by diagonalization that this is false. Hence if $\mathsf{NEXP} \subseteq \mathsf{MA}$ we must have $\mathsf{NEXP} = \mathsf{EXP}$.

We now state the same argument a bit differently:

Definition 5. *A* verifier *for a language $L \in \mathsf{NTIME}(t(n))$ is a TM $M(x,y)$ s.t.*

- *$x \in L$ iff $\exists y\ M(x,y) = 1$, and,*
- *$M(x,y)$ terminates in time $O(t^2(|x|))$, and,*
- *$|y| \leq t(|x|)$.*

Definition 6. *A witness $y \in \{0,1\}^m$ is d-easy if it represents the truth table of a function $f_y : \{0,1\}^{\log m} \to \{0,1\}$ that can be solved by a circuit (over $\log m$ bits) of size at most $(\log m)^d$.*

Definition 7. *A language $L \in \mathsf{NTIME}(t(n))$* always has d-easy witnesses*, if for every verifier $M(x,y)$, for every $x \in L$ (except perhaps for finitely many) there exits a d-easy witness y such that $M(x,y) = 1$.*

In this notation we see that the IKW argument says that:

Theorem 8. *Suppose* $\mathsf{NEXP} \subseteq \mathsf{P/poly}$*. Then for every language* $L \in \mathsf{EXP}$ *there exists a constant* d *such that* L *always has* d*-easy witnesses.*

Proof. Suppose not. Fix a language $L \in \mathsf{NEXP}$ such that for every constant d, L does not always have d-easy witnesses, i.e., there exists a verifier V for L with (infinitely often) no d-easy witnesses. Then, for any $L_2 \in \mathsf{MA}$ we choose d large enough, and use L and V to partially derandomize L_2, giving $\mathsf{EXP} = \mathsf{MA} \subseteq io - \mathsf{NTIME}(2^{n^a})|n$ and a contradiction. ☐

3 If $\mathsf{NEXP} \subseteq \mathsf{P/poly}$ then CSAT is extremely hard

We are used to measuring complexity as a function of the input size. We would like now to measure the complexity of CSAT using a slightly refined measure. We say CSAT has complexity $T(\ell, s)$ if there exits an algorithm that given a circuit C on ℓ variables and size s (where size is the number of edges in the circuit, or alternatively, the description size of C) determines whether C has a satisfying assignment in time $T(\ell, s)$.

The trivial algorithm for CSAT tries all 2^ℓ assignments and has complexity $T(\ell, s) = 2^\ell \cdot poly(s)$. We now show that if $\mathsf{NEXP} \subseteq \mathsf{P/poly}$ then no real speedup is possible.

Theorem 9. *Suppose* $\mathsf{NEXP} \subseteq \mathsf{P/poly}$*. Let* f *be any function such that* $f = \omega(1)$ *(i.e.,* f *is super-constant). Then there is no algorithm solving* CSAT *in time* $T(\ell, s) \leq \frac{2^\ell \cdot s}{\ell^{f(\ell)}}$*.*

Proof. Suppose $\mathsf{NEXP} \subseteq \mathsf{P/poly}$.

Fix any $L \in \mathsf{NTIME}(2^n)$. By Theorem 4 we can reduce L to Succinct3SAT with slight expansion. I.e., given $x \in \{0,1\}^n$ we can compute a circuit $C = \varphi_L(x)$ succinctly describing a formula with $m = O(n^4 2^n)$ variables, where the hidden constant depends only on L. The circuit C is over $\ell = \log m$ variables and has size $O(\ell^4)$ and $\varphi(x)$ runs in $poly(\ell)$ time. The reduction is correct, i.e., $x \in L$ iff the 3SAT formula defined by C is satisfiable. C describes the formula.

Next we define a verifier V for L. Given $x \in \{0,1\}^n$ it computes $C = \varphi(x)$. Next it guesses $y \in \{0,1\}^m$ and takes it to be an assignment for C (which has at most $n \leq 2^\ell = m$ variables). It then goes over all the clauses $C_i = C(i)$ the circuit C defines, and check they are all satisfied. V runs in time $O(2^{2\ell})$. Hence V is a verifier for L. By Theorem 8 there exits a constant D such that V always has d-easy witnesses. I.e., for every large enough n and every $x \in \{0,1\}^n$ there exits a circuit A_x on $\ell = \log(m)$ bits, of size at most $O(\ell^d)$, such that the assignment $y = A_x(\cdot)$ satisfies $V(x, \cdot)$. A describes the assignment and notice that A is very small, $\mathsf{Size}(A) = O(\ell^d) = O(n^d)$.

We now define a new (and faster) algorithm M' for L. Given $x \in \{0,1\}^n$ we first compute $C = \varphi(x)$. We then guess a circuit A of size $O(n^d)$. We define a new circuit UNSAT_A as follows. UNSAT_A has $\log m$ variables. For every $i \in [m]$ it first compute $C(i)$ which is the i'th literal C_i in the 3SAT formula C describes. It finds the three variables i_1, i_2, i_3 in that literal, and computes their value $A(i_1), A(i_2), A(i_3)$ in the assignment described by A. It finally checks that C_i is *false* under that assignment. Therefore,

Claim 10. *The assignment defined by* A *satisfies the sentence defined by* C *iff the circuit* UNSAT_A *does not have a satisfying assignment.*

Proof. If A satisfies C, then it satisfies every clause C_i of C, hence for all i $\mathrm{UNSAT}_A(i) = false$, and UNSAT_A is unsatisfied. Similarly, if A does not satisfy C, then for some i the clause clause C_i is unsatisfied by A, hence $\mathrm{UNSAT}_A(i) = true$, and UNSAT_A is satisfied. □

Therefore,

Corollary 11. $x \in L$ *iff* $\exists_A \mathrm{UNSAT}_A \notin \mathrm{CSAT}$.

Proof. $x \in L$ iff there exists a small A s.t. The assignment defined by A satisfies the sentence defined by C, iff $\exists_A \mathrm{UNSAT}_A \notin \mathrm{CSAT}$. □

Therefore all we have to do to solve L is to guess A, compute the circuit UNSAT_A (which we can do in time $poly(\ell) = poly(n)$ which is negligible) and then solve CSAT on UNSAT_A. The circuit UNSAT_A has $\ell = \log(m) = \log(O(2^n n^4))$ variables and size $s = O(\mathsf{Size}(C) + \mathsf{Size}(A)) = O(n^4 + n^d)$. Let us assume w.l.o.g. that $d \geq 4$ and $s = \mathsf{Size}(\mathrm{UNSAT}_A) = O(n^d)$. Altogether, M' runs in $\mathsf{NTIME}(T(\ell, s))$.[1]

Now suppose CSAT can be solved in time $T(\ell, s) = \frac{2^\ell s}{\mathrm{SP}(\ell,s)}$. Then $T(\ell, s) = \frac{2^\ell s}{\mathrm{SP}(\ell,s)} = O(\frac{ms}{\mathrm{SP}(\ell,s)}) = O(\frac{2^n n^4 n^d}{\mathrm{SP}(\ell,s)})$. Therefore, for every $f(n) = \omega(1)$,

$$\mathsf{NTIME}(2^n) \subseteq \bigcup_d \mathsf{NTIME}(O(\frac{2^n n^4 n^d}{\mathrm{SP}(\ell, s)})) \subseteq \mathsf{NTIME}(O(\frac{2^n n^4 n^{f(n)}}{\mathrm{SP}(\ell, s)})).$$

Since there is a tight non-deterministic hierarchy, we conclude that $O(\frac{2^n n^4 n^{f(n)}}{\mathrm{SP}(\ell,s)}) \geq 2^n$ and $\mathrm{SP}(\ell, s) \leq O(n^{f(n)})$. Finally, $\ell = \log m = n + 4\log n + O(1)$ so ℓ is about n and we can exchange n with ℓ. □

Notice that the proof also gives an explicit way of putting NEXP in Σ_2. This is not surprising since by IKW if $\mathsf{NEXP} \subseteq \mathsf{P/poly}$ then $\mathsf{NEXP} = \mathsf{MA}$ and $\mathsf{NEXP} = \Sigma_2$.

Also notice that we also get the assertion that if $\mathsf{NEXP} \subseteq \mathsf{P/poly}$ then CSAT is extremely hard (with the same parameters) for co-non-deterministic computation.

References

[1] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM Journal on Computing*, 42(3):1218–1244, 2013.

[1] We remark that even though the running time of M' is exponential, and has to be so, M' makes only $poly(n)$ non-deterministic guesses.