Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 148 (2015)

Nondeterministic extensions of the Strong Exponential Time Hypothesis and consequences for non-reducibility

Marco L. Carmosino*† Jiawei Gao*† Russell Impagliazzo*†
Ivan Mihajlin* Ramamohan Paturi* Stefan Schneider*

{mcarmosi, jiawei, russell, imikhail, paturi, stschnei}@cs.ucsd.edu
Department of Computer Science and Engineering
UC, San Diego
La Jolla, CA 92093

September 9, 2015

Abstract

We introduce the Nondeterministic Strong Exponential Time Hypothesis (NSETH) as a natural extension of the Strong Exponential Time Hypothesis (SETH). We show that both refuting and proving NSETH would have interesting consequences.

In particular we show that disproving NSETH would give new nontrivial circuit lower bounds. On the other hand, NSETH implies non-reducibility results, i.e. the absence of (deterministic) fine-grained reductions from SAT to a number of problems. As a consequence we conclude that unless this hypothesis fails, problems such as 3-SUM, APSP and model checking of a large class of first-order graph properties cannot be shown to be SETH-hard using deterministic or zero-error probabilistic reductions.

1 Introduction

Traditionally, complexity theory has been used to distinguish very hard problems, such as NP-complete problems, from relatively easy problems, such as those in P. However, over the past few decades, there has been progress in understanding the exact complexities of problems, both for very hard problems and those within P, under plausible assumptions. For example, under hypotheses such as the k-SUM conjecture [GO95] from computational geometry or the Strong Exponential Time Hypothesis for the complexity of SAT [IPZ01, IP01], it follows that the known algorithms for many basic problems within P, including Fréchet distance [Bri14], edit distance [BI15], string matching [ABW15], k-dominating set [PW10], orthogonal vectors [Wil04], stable marriage for low dimensional ordering functions [MPS15], and many others [BCH14], are essentially optimal.

Unfortunately, as our understanding of the relationship between the exact complexities of problems grows, so does the complexity of the web of known reductions and the number of distinct

*This research is supported by NSF grant CCF-1213151 from the Division of Computing and Communication Foundations. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

†This work was done [in part] while the author was visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant #CNS-1523467.

ISSN 1433-8092

conjectures these results are based on. Ideally, we would like to show that many of these conjectures are in fact equivalent, or that all follow from some basic unifying hypothesis, thereby improving our understanding and simplifying the state of knowledge. For example, it would be nice to show that the 3-SUM conjecture follows from SETH (Strong Exponential Time Hypothesis). It would also be nice to show that SETH implies that HITTINGSET and MAXFLOW require nonlinear time. Can we prove that APSP takes n^3 time under SETH?

In this paper, we introduce a new technique which provides evidence that such a simplification (i.e. hardness results under one unifying hypothesis such as SETH) is *unlikely*, at least when restricted to deterministic reductions. Just as one can show that a problem is unlikely to be NP-complete by showing that it belongs to a presumably smaller complexity class (such as $\mathsf{NP} \cap \mathsf{coNP}$), we can get *non-reducibility* results by comparing the complexity of problems in other models of computation.

To obtain our non-reducibility results, we consider the nondeterministic and co-nondeterministic complexities of the problem under question. If a problem has a smaller nondeterministic and co-nondeterministic complexities, we show that if there were to be a deterministic *fine-grained* reduction from SAT to such a problem, it follows that SAT can be solved faster in co-nondeterministic time, which may be unlikely. More precisely, we introduce the following variant of SETH for nondeterministic models.

Nondeterministic Strong Exponential Time Hypothesis (NSETH): For every $\epsilon > 0$, there exists a k so that k-TAUT is not in $\mathsf{NTIME}[2^{n(1-\epsilon)}]$, where k-TAUT is the language of all k-DNF which are tautologies.

We feel that NSETH is plausible for many of the same reasons as SETH. Just as many algorithmic techniques have been developed for k-SAT, all of which approach exhaustive search for large k, many proof systems have been considered for k-TAUT, and none have been shown to have significantly less than purely exponential complexity for large k. In fact, the tree-like ([PI00]) and regular resolution ([BI13]) proof systems have been proved to require such sizes. Moreover, we observe that results of [JMV13] that obtain circuit lower bounds assuming SETH is false yield the same bounds assuming that NSETH is false. So disproving NSETH would be both a breakthrough in proof complexity and in circuit complexity.

We consider problems together with their *presumed* or *conjectured* complexities. Let the pair (L, T) denote the language L with (presumed) deterministic time complexity T. We use the notion of *fine-grained reducibility* (a generalization of subcubic reducibility from [WW10]) introduced by Vassilevska Williams [Wil] to reduce problems with their complexities to one another. We say that (L_1, T_1) is fine-grained reducible (denoted as $\leq_{\text{FGR}}$) to (L_2, T_2) if there is a Turing reduction from L_1 to L_2 such that improvement of the sort $T_2^{1-\epsilon}$ for $\epsilon > 0$ in the complexity of L_2 leads to an improvement of $T_1^{1-\delta}$ in the complexity of L_1 for some $\delta > 0$. We say that a language L with time complexity T is SETH-hard if there is a fine-grained reduction from CNFSAT with time 2^n to (L, T).

Using fine-grained reductions, an intricate web of relationships between improving basic algorithms within polynomial time have been established. By considering the nondeterministic and co-nondeterministic complexities of such problems, we show, under NSETH, that deterministic fine-grained reductions between many of these problems *do not exist*. In particular,

- HITTINGSET for sets of total size m and time $T(m) = m^{1+\gamma}$ is not SETH-hard for any $\gamma > 0$, and no problem that is SETH-hard reduces to HITTINGSET for any such time complexity.
- 3-SUM for $T(n) = n^{1.5+\gamma}$ is not SETH-hard for any $\gamma > 0$.
- MAXFLOW on a graph with m edges and $T(m) = m^{1+\gamma}$ is not SETH-hard.

- All-pairs shortest path on a graph with n vertices and $T(n) = n^{2+\frac{6+\omega}{9}+\gamma}$ is not SETH-hard.

While there are many known SETH-hard problems, few are graph problems, and those few have the same logical structure. In addition to specific problems, our method can be used to explain why the structure of SETH-hard graph problems are all similar. In particular, we consider first-order definable graph properties on sparse graphs (where we view the input size as the number of edges m). We show that, under SETH, the maximum time complexity for such a property expressible with k quantifiers will be close to $O(m^{k-1})$. On the other hand, if NSETH, all SETH-hard properties have the same logical structure: $k-1$ quantifiers of one type, followed by a single quantifier of the other type.

These results are only valid for deterministic or zero-error probabilistic fine-grained reductions. We introduce a non-uniform variant NUNSETH under which they also hold for randomized reductions with bounded error. However, some care should be used to evaluate whether this hypothesis is true, since it has not been the subject to previous study and Williams has recently shown related hypotheses about Merlin-Arthur complexity of k-TAUT are false ([Wil15]).

2 Outline of the paper

In section 3, we provide definitions of fine-grained reducibilities and establish basic closure properties of these reductions. In section 4, we outline reasons why disproving NSETH is nontrivial. In section 5, we examine the nondeterministic and co-nondeterministic complexities of several problems within polynomial time whose exact complexities have been extensively studied, and show that, under NSETH, none of these problems are SETH-hard. In section 6, we explain why all the known maximally hard SETH-hard first-order graph properties have the same logical structure.

In section 7, we show that NSETH also implies that certain new problems are hard, especially those involving verifying solutions to known SETH-hard problems. Finally, section 8 presents our conclusions and open problems.

3 Definitions and basic properties

Fine-grained reductions are defined with the motivation to control the exact complexity of the reducibility. For this purpose, we consider languages together with their *presumed* or *conjectured* complexities. We use the pair (L, T) to denote a language together with its time complexity T. Intuitively, if (L_1, T_1) fine-grained reduces to (L_2, T_2), then any constant savings in the exponent of the time complexity of L_2 implies some constant savings in the exponent of the time complexity of L_1.

Definition 3.1 (Fine-Grained Reductions ($\leq_{FGR}$))**.** Let L_1 and L_2 be languages, and let T_1 and T_2 be time bounds. We say that (L_1, T_1) *fine-grained reduces* to (L_2, T_2) (denoted $(L_1, T_1) \leq_{FGR} (L_2, T_2)$) if

(a) $\forall \epsilon > 0 \quad \exists \delta > 0$, $\exists \mathcal{M}^{L_2}$, a deterministic Turing reduction from L_1 to L_2, such that

$$\mathsf{TIME}[\mathcal{M}] \leq T_1^{1-\delta}$$

(b) Let $\tilde{Q}(\mathcal{M}, x)$ denote the set of queries made by $\mathcal{M}$ to the oracle on an input x of length n. The query lengths obey the following time bound.

$$\sum_{q \in \tilde{Q}(\mathcal{M},x)} (T_2(|q|))^{1-\epsilon} \leq (T_1(n))^{1-\delta}$$

If a fine-grained reduction exists from (L_1, T_1) to (L_2, T_2), algorithmic savings for L_2 can be transferred to L_1. The definition gives us exactly what is needed to establish savings for L_1 by simulating the machine $\mathcal{M}^{L_2}$ using the faster algorithm for L_2. The role of each parameter in the definition of fine-grained reducibility makes this clear.

T_1**:** The presumed time to decide L_1, usually given by a trivial algorithm.

T_2**:** The presumed time to decide L_2.

ϵ**:** Any savings (assumed or real) on computing L_2.

δ**:** The savings (as a funciton of ϵ) that can be obtained over T_1 when deciding L_1 by reducing to L_2.

Definition 3.2 (Randomized Fine-Grained Reductions ($\leq^s_{rFGR}$))**.** Exactly as in the deterministic case, except the Turing reduction from (L_1, T_1) to (L_2, T_2) is a probabilistic machine with some two-sided error bound

$$\Pr[\mathcal{M}^{L_2}(x) = L_1(x)] \geq s$$

We denote a randomized fine grained reduction from L_1 to L_2 with error bound s by $(L_1, T_1) \leq^s_{rFGR} (L_2, T_2)$. Generally, we will use $s = 2/3$, so we denote $\leq^{2/3}_{rFGR}$ by $\leq_{rFGR}$.

We will have occasion to consider FGRs between function problems. This poses the problem that, in certain situations, just writing down the solution to a problem could exceed the time bound and wipe out fine-grained savings. In the deterministic case, we cope with this by adding another restriction to the definition of a fine-grained reduction:

Definition 3.3 (Fine-Grained Reductions for Functions ($\leq_{fFGR}$))**.** Exactly as in the decision deterministic case, except that the Turing reduction $\mathcal{M}^{f_2}$ is to a function problem f_2 and is expected to produce a functional output. In addition to the existing resource bounds, we bound the size of answers given by the f_2 oracle.

$$\sum_{q \in \tilde{Q}(\mathcal{M},x)} (|f_2(q)|) \leq (T_1(n))^{1-\delta}$$

The bound on query answer size ensures that each proof about decision FGRs goes through in the function FGR case, with an additional step corrosponding to the bound on query answers that is identical to checking the bound on query sizes of the definition of a decision FGR.

We will also consider FGRs between nondeterministic computation of function problems. Defining exactly what it means for a nondeterministic machine to compute a function is fairly involved, so we sidestep this issue by using the *graph* of the function as a decision problem. That is, by convention we use the language $gr(f) = \{\langle x, f(x)\rangle | x \in \{0,1\}^*\}$ to assess the nondeterministic complexity of every function f we are interested in. Since here we only only study $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}$ complexity, this convention does not unduly simplify our model. It is equivalent to being able to print the ith bit of $f(x)$ on input x in $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}$, which we would have anyway. Thus, using the graph of a function, all properties of FGRs between the nondeterministic complexity of decision problems hold between function problems as well.

3.1 Deterministic Fine-grained Reductions

The properties of deterministic fine-grained reductions are exactly what one would expect and follow by standard methods. See Appendix C for proofs.

Lemma 3.4 (Fine-grained reductions translate savings for DTIME)**.** *Let* $(L_1, T_1) \leq_{FGR} (L_2, T_2)$*, and* $L_2 \in \mathsf{DTIME}[T_2(n)^{1-\epsilon}]$ *for* $\epsilon > 0$*. There exists* $\delta > 0$ *such that*

$$L_1 \in \mathsf{DTIME}[T_1(n)^{1-\delta}]$$

Lemma 3.5 (Fine-grained reductions transfer savings for $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}$)**.** *Let* $(L_1, T_1) \leq_{FGR} (L_2, T_2)$*, and* $L_2 \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T_2(n)^{1-\epsilon}]$ *for some* $\epsilon > 0$*. Then there exists a* $\delta > 0$ *such that*

$$L_1 \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T_1(n)^{1-\delta}]$$

To prove both of these "savings" lemmas, we simply run the reduction TM and simulate oracle calls to L_2 using the efficient algorithm for L_2 to get savings for L_1.

Corollary 3.6 (Fine-grained reductions translate savings from reductions)**.** *When the true complexity of a problem is meaningfully smaller than the time bound used in a fine-grained reduction, savings are translated.*

1. *Let* $(L_1, T_1) \leq_{FGR} (L_2, T_2^{1+\gamma})$*, and* $L_2 \in \mathsf{DTIME}[T_2]$*. Then there exists* $\delta > 0$ *such that*
$$L_1 \in \mathsf{DTIME}[T_1^{1-\delta}]$$
2. *Let* $(L_1, T_1) \leq_{FGR} (L_2, T_2^{1+\gamma})$*, and* $L_2 \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T_2]$*. Then there exists a* $\delta > 0$ *such that*
$$L_2 \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T_1^{1-\delta}]$$

The above follows from the saving transfer lemmas by a simple substitution.

Lemma 3.7 (Fine-grained reductions are closed under composition)**.** *Let* $(A, T_A) \leq_{FGR} (B, T_B)$ *and* $(B, T_B) \leq_{FGR} (C, T_C)$*. It then follows* $(A, T_A) \leq_{FGR} (C, T_C)$*.*

Finally, composition is proved by carefully verifying time and query bounds on the obvious "nested" simulation of A using the algorithm for C.

3.2 Randomized FGRs

As we will show, many of the problems such as k-SUM and HITTINGSET which have served as starting points for fine-grained reductions have substantially smaller nondeterministic complexities than their conjectured deterministic complexities. From the above closure properties, it will follow that if NSETH is true, none of these problems is SETH-hard under deterministic (or zero-error probabilistic) fine-grained reductions. This leaves a major loophole: these problems might still be SETH-hard under randomized reductions. In this section, we will outline a reason why even randomized SETH-hardness would be somewhat surprising. We introduce a non-uniform version of NSETH, NUNSETH, and show that this hypothesis would imply the non-existence of even randomized SETH-hardness results.

Definition 3.8. Let k-TAUT be the tautology problem restricted to k-DNF's. The *Non-uniform Nondeterministic Strong Exponential Time Hypothesis* (NUNSETH) is the statement : $\forall \epsilon > 0\, \exists k \geq 0$, so that there are no nondeterministic circuit families of size $O(2^{n(1-\epsilon)})$ recognizing the language k-TAUT.

While we do not have any general conservation of non-uniform nondeterministic time by randomized reductions, we do have a limit for the special case of problems that are SETH-hard under randomized reductions.

Lemma 3.9. *Assume L is* -SETH*-hard with $T(N)$ via a randomized reduction. If* NUNSETH*, then there is no $\delta > 0$ so that $L \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T^{1-\delta}(n)]$.*

Proof. Let ϵ be the constant corresponding to δ in the reduction, and let $\mathcal{M}^L$ be the corresponding randomized oracle machine. Let $m < n^k$ be the length in bits of a description of a k-SAT formula on n inputs. By repeating $\mathcal{M}^L$ $O(m)$ times and taking the majority answer, we can make the error probability less than 2^{-m}. Therefore, there is one random tape that has no errors, using the standard argument that $\mathsf{BPP} \in \mathsf{P/poly}$. Since $\mathcal{M}$ runs in total time $2^{(1-\epsilon)n)}$, this tape will have length at most $m2^{(1-\epsilon)n)}$, and so will be an exponential improvement over 2^n. Once we have fixed the tape, we can simulate the oracle queries nondeterministically as in the case of deterministic reductions, with total complexity $O(m)$ times what it is for one run. Thus, we get a nondeterministic circuit with total size $O(m2^{(1-\epsilon)n})$. □

Note that the above argument, in addition to needing advice, multiplies the complexity by an amount polynomial in the input size. While this is not an issue for SAT, it would render the consequences of randomized reductions for problems within P moot, since we are trying to preserve exact polynomial complexities.

While NUNSETH seems plausible, we should exercise some caution before adopting it as an axiom. First, there are no known consequences if NUNSETH fails to be true. Secondly, we originally were going to add equally plausible (to us) hypotheses concerning the total time for bounded round interactive protocols for k-TAUT. However, Williams recently showed that even the general formula counting problem has a Merlin-Arthur protocol of total complexity $\tilde{O}(2^{n/2})$. Because there is a polynomial overhead in making such a protocol a nondeterministic algorithm with advice, this does not contradict NUNSETH. However, it does remind us that counter-intuitive things can happen when randomness and nondeterminism are combined, so we should be cautious in assuming non-uniformity might not speed up computation in this circumstance.

4 What if ¬NSETH?

SETH is an interesting hypothesis because both ¬SETH and SETH have interesting consequences that seem difficult to prove unconditionally. In this section, we show that the same proofs that show "¬SETH implies circuit lower bounds" can be applied to ¬NSETH as well. This is evidence that NSETH will be hard to refute.

Algorithms for CKT-SAT or CKT-TAUT imply circuit lower bounds (see [Wil13] and [Wil14b]). For some restricted circuit classes $\mathcal{C}$, we can reduce satisfiability or tautology of $\mathcal{C}$-circuits to k-SAT or k-TAUT by decomposing $\mathcal{C}$ circuits into a "big OR" of CNF formulas. Thus, both ¬SETH and ¬NSETH imply faster $\mathcal{C}$-circuit analysis algorithms (tautology or satisfiability) for these classes, which imply lower bounds.

The proofs of [JMV13] optimize the reduction of arbitrary nondeterministic time languages to 3-SAT to obtain new "failure of a hardness hypothesis about k-SAT implies circuit lower bounds" results for a variety of circuit classes. The following (see Appendix B for details) is implicit in their work:

Theorem 4.1. *We have the following implications from failure of a k-TAUT hardness hypothesis to circuit lower bounds for restricted classes:*

1. *If the nondeterministic exponential time hypothesis (*NETH*) is false; i.e., for every $\epsilon > 0$,* 3-TAUT *is in time $2^{\epsilon n}$, then $\exists f \in \mathsf{E}^{\mathsf{NP}}$ such that f does not have linear-size circuits.*

2. *If the nondeterministic strong exponential time hypothesis (*NSETH*) is false; i.e., there is a $\delta < 1$ such that for every k, k-TAUT is in time $2^{\delta n}$, then $\exists f \in \mathsf{E}^{\mathsf{NP}}$ such that f does not have linear-size series-parallel circuits.*

3. *If there is $\alpha > 0$ such that n^{α}-TAUT is in time $2^{n-\omega(n/\log\log n)}$, then $\exists f \in \mathsf{E}^{\mathsf{NP}}$ such that f does not have linear-size log-depth circuits.*

Since (by item 2 above) refuting NSETH would give nontrivial circuit lower bounds, it is unlikely to be easy to refute.

5 The nondeterministic time complexity of problems in P

How could we show that one language is not reducible to another language? There is an ever-growing web of problems, hypotheses, and reductions that reflect the fine-grained complexity approach to explaining hardness. Could this structure collapse into a radically simpler graph, with just a few equivalence classes? If we assume NSETH, the answer to this question is *probably not as much as one might hope.*

We can broadly categorize computational problems into two sets. In the first category, the deterministic time complexity is higher than both the nondeterministic and co-nondeterministic time complexity. In the second category, at least one of nondeterminism or co-nondeterminism does not help in solving the problem more efficiently. Corollary 3.6 shows that savings in $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}$ are preserved under deterministic fine-grained reductions. As a result, we can rule out tight reductions from a problem that is hard using nondeterminism or co-nondeterminism to a problem that is easy in $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}$.

If NSETH holds, then k-TAUTk is in the category of problems that do not benefit fron nondeterminism.. benefit from co-nondeterminism. So, any problem that is SETH-hard under deterministic reductions also falls in this category.

In this section we explore problems that do benefit from $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}$, i.e. we give nondeterministic algorithms that are faster than their presumed deterministic time complexities. This rules out deterministic fine-grained reductions from CNFSAT to these problems with their presumed time complexities. As a consequence, it is not possible to show that these problems are SETH-hard using a deterministic reduction.

We begin by formalizing the notion of non-reducibility.

Theorem 5.1 (NSETH implies no reduction from SAT)**.** *If* NSETH *and $C \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T_C]$ for some problem C, then $(\mathsf{SAT}, 2^n) \not\leq_{FGR} (C, T_C^{1+\gamma})$ for any $\gamma > 0$.*

Proof. Assume NSETH, $(\mathsf{SAT}, 2^n) \leq_{FGR} (C, T_C^{1+\gamma})$, and $C \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T_C]$. By Corollary 3.6, preservation of $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}$ savings under fine-grained reductions, there exists $\delta > 0$ such that $\mathsf{SAT} \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{n(1-\delta)}]$. This contradicts NSETH, therefore it cannot be the case (under NSETH) that $(\mathsf{SAT}, 2^n) \leq_{FGR} (C, T_C)$. □

Corollary 5.2 (NSETH implies no reductions from SETH-hard problems)**.** *If* NSETH *and $C \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T_C]$, then for any B that is* SETH*-hard under deterministic reductions with time T_B, and $\gamma > 0$, we have*

$$(B, T_B) \not\leq_{FGR} (C, T_C^{1+\gamma})$$

Proof. Assume NSETH, and that (B, T_B) is SETH-hard. Therefore, we know $(\mathsf{SAT}, 2^n) \leq_{FGR} (B, T_B)$. Now assume $(B, T_B) \leq_{FGR} (C, T_C^{1+\gamma})$. Then by Lemma 3.7, composition of fine-grained reductions, we have that $(\mathsf{SAT}, 2^n) \leq_{FGR} (C, T_C)$. But by Theorem 5.1 above, this is impossible under NSETH. □

We now give the main result of this section.

Theorem 5.3. *Under* NSETH*, there is no deterministic or zero-error fine-grained reduction from* SAT *or any* SETH*-hard problem to the following problems with the following time complexities for any* $\gamma > 0$*.*

- MAXFLOW *with* $T(m) = m^{1+\gamma}$
- HITTINGSET *with* $T(m) = m^{1+\gamma}$
- 3-SUM *with* $T(n) = n^{1.5+\gamma}$
- *All-pairs shortest path with* $T(n) = n^{2+\frac{6+\omega}{9}+\gamma}$

Note that for graph problems, n refers to the number of vertices, m refers to the number of edges, and ω is the matrix multiplication exponent.

To prove Theorem 5.3 we give both nondeterministic and co-nondeterministic algorithms for these problems.

5.1 Maximum Flow

The maximum flow problem has been an extensively studied problem for decades and has a large number of theoretical and practical applications. While approximate maximum flow on undirected graphs has a $\tilde{O}(m)$ algorithm [KLOS14], where m is the number of edges, no linear time algorithm is known for the exact version of the problem.

A natural question from the point of conditional hardness is if we can prove a superlinear lower bound by proving that the problem is SETH-hard.

In this section we use the max-flow/min-cut theorem to give a $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}$ algorithm for the decision version of max-flow with time linear in the number of edges. Assuming NSETH, we can then conclude that there is no deterministic fine-grained reduction from any SETH-hard problem to maximum flow with a superlinear time bound.

Definition 5.4 (Maximum Flow Problem)**.** Let $G = (V, E)$ be a connected directed graph, $s, t \in V$ be vertices and $k \in \mathbb{R}$.

The maximum flow problem (MAXFLOW) is to decide if there exists a flow from s to t of value at least k.

The nondeterministic algorithm for maximum flow is straight-forward and the co-nondeterministic algorithm follows directly for the max-flow/min-cut theorem.

Lemma 5.5. MAXFLOW $\in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[O(m)]$

Proof. For the nondeterministic algorithm, nondeterministically guess the flow on each edge. We can verify in linear time that the value of the flow is at least k, that no edge flow exceeds the edge capacity, and that for all nodes the inflow is equal to the outflow.

For the co-nondeterministic algorithm, nondeterministically guess a cut (S, T) such that $s \in S$ and $t \in T$ with value l where $l < k$. By the max-flow/min-cut theorem there is no flow with value strictly greater than l. The value of a cut can be computed in $O(m)$ time. □

This completes the part of Theorem 5.3 concerning maximum flow. In contrast, the single-source maximum flow problem requires quadratic time under SETH [AVWY15]. In the single-source maximum flow problem we are given a source s and need to output the maximum flow from s to all other nodes. As a consequence, there is no deterministic fine-grained reduction from single-source maximum flow to maximum flow under NSETH.

5.2 Hitting Set

Given two families of non-empty sets $\mathcal{S}$ and $\mathcal{T}$ defined on universe U, a set $S \in \mathcal{S}$ is a *hitting set* if it has nonempty intersections with all members in $\mathcal{T}$. The HITTINGSET problem accepts input $(\mathcal{S}, \mathcal{T}, U)$ iff

$$\exists S \in \mathcal{S}\ \forall T \in \mathcal{T}\ \exists u \in U\ \ ((u \in S) \wedge (u \in T))$$

Let the size of input be $m = \sum_{S \in \mathcal{S}} |S| + \sum_{T \in \mathcal{T}} |T|$. Assume for any $u \in U$, we can in constant time decide if $u \in S$ or $u \in T$. We show that HITTINGSET and its negation are both solvable in nondeterministic linear time.

Lemma 5.6. HITTINGSET $\in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[O(m)]$

HITTINGSET can be solved nondeterministically in linear time, by guessing an S, enumerating all $T \in \mathcal{T}$, and guessing a $u \in T$.

The negation of the HITTINGSET problem ¬HITTINGSET, which is defined as

$$\forall S \in \mathcal{S}\ \exists T \in \mathcal{T}\ \forall u \in U\ \ ((u \notin S) \vee (u \notin T))$$

can be solved by the following algorithm.

```
for each S ∈ 𝒮 do
    Nondeterministically select T from 𝒯;
    for each u ∈ S do
        if u ∈ T then
            Reject.
        end
    end
end
Accept.
```

Algorithm 1: Algorithm for ¬HITTINGSET

The algorithm runs in time $O(\sum_{S \in \mathcal{S}} |S|) = O(m)$.

Section A.3 generalizes his algorithm for model checking of arbitrary k-quantifier sentences with at least one existential quantifier and ending with a universal quantifier.

5.3 3-sum

The conjecture that the 3-SUM problem admits no $O(n^{2-\epsilon})$ algorithm for any $\epsilon > 0$ has proven immensely useful to show the conditional hardness of a large number of problems (e.g. [GO95, DBdGO97, AW14]), most of which are not known to be hard under SETH. A fine-grained reduction from SAT to 3-SUM would therefore have a large impact, proving the 3-SUM conjecture under SETH.

We give a subquadratic algorithm for 3-SUM in $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}$, which rules out a deterministic fine-grained reduction from SAT to 3-SUM under NSETH.

Definition 5.7. Given n integers $a_1 \dots a_n$ in the range $[-n^c, n^c]$ for some constant c, the 3-*Sum problem* (3-SUM) is the problem of determining if there is a triple $1 \le i, j, k \le n$ such that $a_i + a_j + a_k = 0$.

Lemma 5.8. 3-SUM $\in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[\tilde{O}(n^{1.5})]$

Proof. There is a trivial constant time nondeterministic algorithm of guessing the triplet of indices. The more interesting part is to show that there is an efficient nondeterministic algorithm to show that there is no such triplet.

We nondeterministically guess a proof of the form (p, t, S), such that

- p is a prime number, such that $p \le \mathbf{prime}_{n^{1.5}}$, where $\mathbf{prime}_i$ is i-th prime number.
- t is a nonnegative integer with $t \le 3cn^{1.5} \log n$ such that $t = |\{(i, j, k) \mid a_i + a_j + a_k = 0 \mod p\}|$ is the number of three-sums modulo p.
- $S = \{(i_1, j_1, k_1), \dots, (i_t, j_t, k_t)\}$ is a set of t triples of indices, such that for all $r : \ 0 < r \le t$ we have $a_{i_r} + a_{j_r} + a_{k_r} = 0 \mod p$ and $a_{i_r} + a_{j_r} + a_{k_r} \neq 0$

We first show that such a proof exists. Let us assume that there is no triple of elements that sum up to zero. Let R be set of all pairs $((i, j, k), p)$, such that p is a prime $\le \mathbf{prime}_{n^{1.5}}$ and $a_i + a_j + a_k = 0 \mod p$. Then $|R| \le n^3 \log(3n^c) < 3cn^3 \log n$, as any integer z can have at most $\log(z)$ prime divisors. Then, by a simple counting argument, there indeed exists a prime $p_0 \le \mathbf{prime}_{n^{1.5}}$, such that the number of pairs of the form $((i, j, k), p_0)$ in R is at most $\frac{3cn^3 \log n}{n^{1.5}} = 3cn^{1.5} \log n)$.

To verify a proof of that form we first need to check that for all $r \le t$:

$$a_{i_r} + a_{j_r} + a_{k_r} = 0 \mod p$$

$$a_{i_r} + a_{j_r} + a_{k_r} \neq 0$$

Then we compute the number of 3-sums modulo p and compare it with t. In order to do this we expand the following expression using Fast Fourier Transform in time $\tilde{O}(t)$:

$$\left(\sum_i x^{(a_i \mod p)} \right)^3$$

Let b_j be a coefficient before x^j. We need to check that

$$b_0 + b_p + b_{2p} = t$$

If it is true, then the proof is accepted, otherwise it is rejected.

The time complexity of verification is $\tilde{O}(n^{1.5})$ for reading and checking the properties of all the triples and $\tilde{O}(t) = \tilde{O}(n^{1.5})$ for counting the number of triples that sum to 0 modulo p. Therefore the total time complexity is $\tilde{O}(n^{1.5})$.

□

5.4 All-pairs shortest paths and related problems

The All-pairs shortest path problem (APSP) is to find the shortest path in a graph between any pair of nodes. Like the 3-SUM conjecture and SETH, the conjecture that APSP does not admit an $O(n^{3-\epsilon})$ time algorithm for any $\epsilon > 0$ has been used successfully to show the conditional hardness of a number of problems, e.g. [WW10, VW09].

We use a similar technique as in the algorithm for 3-SUM to show that the Zero Weight Triangle problem (ZWT), which is hard under APSP, admits an efficient algorithm in $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}$.

Definition 5.9. Given a tripartite graph $G(V_1, V_2, V_3, E)$ with $|V_1| = |V_2| = |V_3| = n$ and edge weights in $[-n^a, n^a]$, the *Zero Weight Triangle problem* is the problem of determining if there is a triangle such that the sum of the edge weights is 0.

We first show that if the range is small enough, then we can count the number of zero weight triangles efficiently.

Lemma 5.10. *There is a deterministic algorithm for counting the number of zero weight triangles in time* $O(n^{\omega+4a})$

Proof. Consider all $O(n^{4a})$ triples $x, y, z \in [-n^a, n^a]$ such that $x + y + z = 0$. For each triple restrict the graph to edges between V_1 and V_2 of edge weight x, edges between V_2 and V_3 of weight y, and edges between V_1 and V_3 of weight z. We can then count the number of triangles in the resulting unweighted graph using matrix multiplication in time $O(n^\omega)$. The total time is $O(n^{\omega+4a})$ as claimed. □

In particular, we will be using Lemma 5.10 when working modulo a prime p, where the time complexity of counting the number of zero weight triangles modulo p is then $O(n^\omega p^2)$.

Lemma 5.11. *The Zero Weight Triangle Problem is in* $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[O(n^{2+\omega/3})]$.

Proof. As for 3-SUM, the nondeterministic algorithm is trivial and we concentrate on the co-nondeterministic algorithm.

Let $\mu = 1 - \omega/3$. Further let c be a large constant such that there are at least n^μ primes in the range $R = [n^\mu, cn^\mu \log n]$. We assume that there is no zero weight triangle and consider any fixed triangle. The total weight of the triangle is in the range $[-3n^a, 3n^a]$ and the number of primes $p \in R$ such that the triangle has weight $0 \mod p$ is at most $\log(3n^a)/\log(n^\mu) < \frac{2}{\mu}a$. Since R contains at least n^μ primes, there is a prime $p \in R$ such that the number of triangles with weight $0 \mod p$ is at most $\frac{2}{\mu}an^{2+\omega/3}$.

The nondeterministic algorithm now proceeds as follows: Nondeterministically pick p as above. By Lemma 5.10 we can deterministically count the number s of triangles with weight $0 \mod p$ in time $O(n^\omega p^2) = O(n^{2+\omega/3})$. Nondeterministically pick s distinct triangles and check that each of them has weight $w \neq 0$ with $w = 0 \mod p$.

The total time is bounded by $O(n^{2+\omega/3})$ as claimed. □

Corollary 5.12. APSP $\in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[\tilde{O}(n^{2+\frac{6+\omega}{9}})]$.

Proof. A deterministic fine-grained reduction from the problem of finding a negative weight triangle to ZWT can be found in [VW09], such that the negative weight triangle problem is also in $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[\tilde{O}(n^{2+\omega/3})]$. Finally, [WW10] give a deterministic fine-grained reduction from APSP to the negative weight triangle problem with time $\tilde{O}(n^2 T(n^{1/3}))$, where $T(n)$ is the time complexity of the negative weight triangle problem. □

Note that [WW10] in fact give a sizable list of problems that are equivalent to APSP under subcubic deterministic fine-grained reductions (including negative weight triangle, but not zero weight triangle). Our non-reducibility result therefore applies to all of these problems.

6 Characterizing the quantifier structure of SETH-hard graph problems

There are many problems within P that are known to be SETH-hard, but few of them are graph problems. And of the ones that are, they tend to have similar logical forms. For instance, k-Dominating Set [PW10] is definable by a $\forall^k\exists$ quantified formula; Graph Diameter-2 and Bipartite Graph Dominated Vertex [BCH14] are definable by $\forall\forall\exists$ quantified formulas. Here we study the relations between SETH-hardness and the logical structures of model checking problems. The paper by Ryan Williams [Wil14a] explored the first-order graph properties on dense graphs, while in this paper, we look into sparse graphs whose input is a list of edges.

We define "graph property" quite broadly. The input to a graph property is a many-sorted universe that we view as sets of vertices, together with a number of unary relations (node colors), and binary relations, viewed as different categories or colors of edges. The binary relations can in general be directed. We specify the problem to be solved by a first order sentence. Let φ be a first order sentence in prenex normal form, which has k quantifiers.

$$\varphi = Q_1 x_1 \in X_1, Q_2 x_2 \in X_2, \ldots Q_k x_k \in X_k \psi$$

or shortened as

$$\varphi = Q_1 x_1 Q_2 x_2 \ldots Q_k x_k \psi$$

where φ is a quantifier-free formula whose atoms are unary or binary predicates on $x_1, \ldots, x_k$.

An instance of the model checking problem of φ gives k ($k \geq 3$) specifies sets $X_1, \ldots X_k$, where variable x_i is an element of set X_i, and unary or binary relations on these sets. (X_i needn't be disjoint, so allowing them to be viewed as distinct only increases the expressive power. We assume equality is one of the relations, so we can tell when $x_i = x_j$.)) The sets $X_1, \ldots, X_k$ can be considered as the sets of nodes in a k-partite graph, and the values of a binary predicate can be considered as edges in the graph, i.e. for predicate P, $P(x_i, x_j) = \mathit{true}$ means there is an edge between nodes x_i and x_j. We refer to the k-partite graph with edges defined by predicate P as G_P, and the union of graphs defined on all predicates as G. The data structures used to code the relations are as follows: For each unary relation, an array of Booleans indexed by the vertices saying whether the relation holds, and for each binary predicate, the list representation of the corresponding directed graph. We want to see if φ is true for the input model.

Examples of this problem include k-Clique, which is defined by

$$\varphi = \exists x_1 \ldots \exists x_k \bigwedge_{i,j \in \{1,\ldots,k\}, i \neq j} E(x_i, x_j)$$

and k-Dominating Set, defined by

$$\varphi = \exists x_1 \ldots \exists x_k \forall x_{k+1} \left(E(x_1, x_{k+1}) \vee \cdots \vee E(x_k, x_{k+1})\right)$$

and Graph Radius-2, defined by

$$\varphi = \exists x_1 \forall x_2 \exists x_3 \left(E(x_1, x_3) \wedge E(x_3, x_2)\right)$$

We let $n = \max_i |X_i|$ be the maximum size of the node parts, and m be the number of edges in the union of the graphs. The size is $n + m$, but for convenience, we will assume $m > n$ and use m as the size.

The maximum deterministic complexity of a k-quantifier formula for $k \geq 2$ is $O(m^{k-1})$. For $k = 2$, this is just linear in the input size, so matching lower bounds follow. So the interesting case is $k \geq 3$. If SETH is true, some formulas require approximately this time. But if NSETH holds, all such formulas that are SETH hard are of the same logical form. This is made precise as follows:

Theorem 6.1. *Let $k \geq 3$. If* NSETH *is true, then there is a k-quantifier formula whose model checking problem is $O(m^{k-1})$* SETH*-hard, but all such formulas have the form $\forall^{k-1}\exists$ or $\exists^{k-1}\forall$.*

Theorem 6.1 comes directly from the following lemmas:

Lemma 6.2. *If* SETH *or* NSETH *is true, then there are $\forall^{k-1}\exists$ problems that are* SETH*-hard for time $O(m^{k-1})$.*

Thus by negating φ, the $\exists^{k-1}\forall$ problems are also hard under SETH.

On the other hand if a problem is of any form other than $\forall^{k-1}\exists$, we will show it has smaller non-deterministic complexity. Such a problem has either exactly one existential quantifier not in the innermost position, no existential quantifiers, or at least two existential quantifiers.

Lemma 6.3. *If φ has exactly one existential quantifier, but it is not on the innermost position, then it can be solved in $O(m^{k-2})$ nondeterminisitic time.*

Lemma 6.4. *If φ has more than one existential quantifiers, then it can be solved in time $O(m^{k-2})$ nondeterministically.*

These problems can be solved by guessing the existentially quantified variables, and exhaustive search on universally quantified variables. Because there are at most $k - 2$ universial quantifiers, the algorithm runs in time $O(m^{k-2})$.

Lemma 6.5. *If all quantifiers are universal, then it can be solved in deterministic time $O(m^{k-1.5})$*

Thus, only $\forall^{k-1}\exists$ formulas require $O(m^{k-1})$ non-deterministic time, and by looking at the complements, only $\exists^{k-1}\forall$ formulas require $O(m^{k-1})$ co-non-deterministic time. Thus, assuming NSETH, only these two types of first-order properties might be SETH-hard for the maximum difficulty of a k-quantifier formula.

Lemmas 6.2, 6.3 and 6.5 will be proved in Appendix A.

7 Consequences for verification of solutions

Besides implying that some problems are not SETH-hard, NSETH also implies some new lower bounds on problems in P. Namely, if NSETH is true, then problems such as Fréchet distance, edit distance, and longest common substring also require quadratic co-nondeterministic time (i.e., to show that the optimal solution has cost that exceeds a given value). This immediately implies that, even given a solution, testing optimality requires quadratic time. We can formalize this as follows:

Theorem 7.1. *Let* Opt(x) *be the optimization problem, given x, find $\max_{y,|y|=l(|x|)} F(x, y)$, for some F that is computable in time $T_F(n + l(n)) \geq n + l(n)$. The verification problem* Ver *is: given x and y, is y an optimal solution for* Opt*, i.e., is there no y' with $F(x, y') > F(x, y)$. Assume that* Opt *is* SETH*-hard for some $T(n)$ which is greater than $T_F^{1+\gamma}(n + l(n))$ for some $\gamma > 0$. Then if* NSETH*,* Ver *cannot be solved in any time T' so that $T_{\text{Ver}}(n + l(n)) < T^{1-\epsilon}(n)$ for any $\epsilon > 0$.*

Proof. Assume not, that Ver can be solved in some time T' with $T_{\text{Ver}}(n + l(n)) < T^{1-\epsilon}(n)$. Then we can compute the function Opt in $\mathsf{NTIME}((T_{\text{Ver}}(n + l(n)) + T_F(n + l(n))))$ as follows:

Non-deterministically guess an optimal solution y and run the algorithm for Ver on the pair (x, y). If it is optimal (i.e., in Ver), return $F(x, y)$. The total time is $l(n)$ to guess y, plus $T_F(n+l(n))$ to compute F, plus $T_{\text{Ver}}(n + l(n))$.

From the assumption that Opt is SETH-hard for time $T(n)$, and since the time complexity of the above procedure is $O(T(n)^{1-\epsilon})$ for some $\epsilon > 0$, it follows that TAUT is in time $2^{n(1-\delta)}$ for some $\delta > 0$. This contradicts NSETH. □

So NSETH gives us a way to argue that not only finding but verifying optimal solutions is computationally intensive.

8 Conclusions and open problems

A theme running through computational complexity is that looking at general relationships between models of computing and complexity classes can frequently shed light on the difficulty of specific problems. In this paper, we introduce this general technique to the study of fine-grained complexity by comparing nondeterministic complexities of problems. This raises the more general question of what other notions and models of complexity might be useful in distinguishing the fine-grained complexity of problems. For example, we show that neither 3-SUM or all-pairs shortest path can be SETH-hard if NSETH holds. This still leaves open the possibility that the two conjectures are equivalent to each other (if not to SETH). One might be able to prove such an equivalence, or give evidence against it by showing a different notion of complexity that distinguishes the two and is preserved by FGR.

Acknowledgments: We would like to thank Ryan Williams for the many helpful comments on an earlier draft, and Virginia Vassilevska Williams.

References

[ABW15] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Quadratic-time hardness of LCS and other sequence similarity measures. *CoRR*, abs/1501.07053, 2015.

[AVWY15] Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC '15, pages 41–50, New York, NY, USA, 2015. ACM.

[AW14] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 434–443, 2014.

[BCH14] Michele Borassi, Pierluigi Crescenzi, and Michel Habib. Into the square - on the complexity of quadratic-time solvable problems. *CoRR*, abs/1407.4972, 2014.

[BI13] Christopher Beck and Russell Impagliazzo. Strong ETH holds for regular resolution. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 487–494, 2013.

[BI15] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 51–58, 2015.

[Bri14] Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. *CoRR*, abs/1404.1448, 2014.

[DBdGO97] Mark De Berg, Marko M de Groot, and Mark H Overmars. Perfect binary space partitions. *Computational Geometry*, 7(1):81–91, 1997.

[GO95] Anka Gajentaan and Mark H Overmars. On a class of o (n 2) problems in computational geometry. *Computational geometry*, 5(3):165–185, 1995.

[IP01] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367 – 375, 2001.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512 – 530, 2001.

[JMV13] Hamidreza Jahanjou, Eric Miles, and Emanuele Viola. Local reductions. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:99, 2013.

[KLOS14] Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pages 217–226. SIAM, 2014.

[MPS15] Daniel Moeller, Ramamohan Paturi, and Stefan Schneider. Subquadratic algorithms for succinct stable matching. 2015.

[PI00] Pavel Pudlak and Russell Impagliazzo. A lower bound for dll algorithms for k-sat. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2000, San Francisco, CA, USA, January 9-11, 2000*, pages 128–136, 2000.

[PW10] Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1065–1075, 2010.

[VW09] Virginia Vassilevska and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 455–464. ACM, 2009.

[Wil] Virginia Vassilevska Williams. Stoc tutorial: Hardness and equivalences in p. `http://theory.stanford.edu/~virgi/stoctutorial.html`. Accessed: 2010-09-30.

[Wil04] Ryan Williams. A new algorithm for optimal constraint satisfaction and its implications. *Electronic Colloquium on Computational Complexity (ECCC)*, (032), 2004.

[Wil13] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013.

[Wil14a] Ryan Williams. Faster decision of first-order graph properties. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 80:1–80:6, 2014.

[Wil14b] Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014.

[Wil15] Ryan Williams. Personal communication. 2015.

[WW10] Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 645–654. IEEE, 2010.

Appendix A Proofs of lemmas in Section 6

A.1 $\forall^{k-1}\exists$ problems are hard under NSETH

In this section we prove Lemma 6.2 by showing that if the problem defined by formula

$$\forall x_1 \dots \forall x_{k-1} \exists x_k \left(P(x_1, x_k) \wedge \dots \wedge P(x_{k-1}, x_k)\right)$$

can be solved in $O(m^{k-1-\epsilon})$ nondeterministic time for some $\epsilon > 0$, then NSETH is false. We construct a fine-grained reduction from DNF-TAUT to the above problem.

Given a formula D in DNF, with variable set V, let $V_1 \dots V_{k-1}$ be a partition of V such that $|V_1| = \dots = |V_{k-1}| = n/(k-1)$. For each variable set V_i, we create a node set X_i. For each assignment α for the k variables in V_i, we create a node x_α in X_i. For each term t in D, we create node x_t in X_k. Whenever a term t is satisfied by a partial assignment α, we let there be an edge $P(x_\alpha, x_t)$.

D is tautology if and only if for all $k-1$ tuples of partial assignments for $V_1 \dots V_{k-1}$, there exists a term t such that t is satisfied by all the partial assignments. Thus D is tautology iff the model checking instance we have created is satisfiable.

The number of nodes and edges in the constructed graph is $O^*(2^{n/k})$. Thus we have set up an fine-grained reduction from $O(2^n)$ time to $O(m^k)$ time.

A.2 All-existential and all-universal problems

For simplicity, in sections A.2 and A.3, we will restrict all predicates to be binary. The algorithms can easily be modified to work for unary predicates.

In this section we prove Lemma 6.5 by showing an algorithm for all-existential formulas that run in time $O(m^{k-1.5})$.

Assume there is at most one predicate on each pair of variables. Otherwise for a pair of variable we can take the disjunction of all the predicates to be the value of the only predicate. Let the only predicate be E.

First, we write ψ in DNF with t terms, i.e. $\psi = \bigvee_{i=1}^{t} \psi_i$, and then split its terms, so that $\varphi = \bigvee_{i=1}^{t} (\exists x_1 \dots \exists x_k \psi_i)$. Thus the problem becomes a constant number of model checking subproblems with form $\exists x_1 \dots \exists x_k \psi_i$ where ψ_i is a conjunction of either positive or negative predicates.

Let the number of predicates be p. We give the predicates some canonical order, so that their truth value correspond to strings in $\{0,1\}^p$. We call the strings *colors*. Specifically, the color 0^p (where all predicates are false) is called the *background color*. The *color interpretation function*

χ maps a $(x_1, \ldots, x_k)$ tuple to the color c corresponding to the predicate values decided by this assignment.

For each ψ_i, we find all colors satisfying it, and then for each color c, we use the following algorithm to count the number of $(x_1, \ldots, x_k)$ tuples such that $\chi(x_1, \ldots, x_k) = c$.

Case 1: The color c contains at least two positive predicates.

Case 1-1: There exists four distinct variables a, b, c, d such that $E(a, b)$ and $E(c, d)$ are true in c. Then we could exhaustively search all $k - 4$ variables other than a, b, c, d in time $O(n^{x^{k-4}})$, and enumerate all $E(c, d)$ and $E(a, b)$ edges, which takes time $O(m^2)$. The overall running time is $O(m^{k-2})$. For each k-tuple enumerated, if $\chi(x_1, \ldots, x_k) = c$, we add 1 to the counter for color c.

Case 1-2: There are three distinct variables a, b, c such that $E(a, b)$ and $E(b, c)$ are true in c. We take time $O(m^{k-3})$ to exhaustively search all $k - 3$ other variables. Then we consider the a nodes of large and small degree separately.

- For a's whose degree is at most $\sqrt{m}$, we enumerate all $E(a, b)$ edges. Then for each b, we enumerate all (b, c) edges. The running time is $O(\sum_a deg(a)) \leq O(m\sqrt{m}) = O(m^{1.5})$.
- For a's whose degree is greater than $\sqrt{m}$, we enumerate all such a's and all (b, c) edges. Because there are at most $O(m/\sqrt{m}) = \sqrt{m}$ such a's, the running time is $O(\sqrt{m}\, m) = O(m^{1.5})$.

For each k-tuple enumerated, check if $\chi(x_1, \ldots, x_k) = c$. The overall running time is $O(m^{k-3}\, m^{1.5}) = O(m^{k-1.5})$.

Case 2: The color c contains only one positive predicate. Let the positive predicate be $E(a, b)$.

As described in Case 1, we can compute the number of $(x_1, \ldots, x_k)$ with colors where where both $E(a, b)$ and some non-(a, b) predicates are true, Also, we can easily compute the number of k tuples satisfying $E(a, b)$ regardless of other predicates, by multiplying the number of $E(a, b)$ edges and every other vertex set's size. Then, by inclusion-exclusion we can exclude the cases where some other predicates are true. Because there are only a constant number of colors, the running time is $O(m^{k-1.5})$.

Case 3: c is the background color. We compute the number of $(x_1, \ldots, x_k)$ of all other colors, and use inclusion-exclusion to exclude the cases where some predicates are true. The running time is $O(m^{k-1.5})$.

A.3 Single-$\exists$ formulas ending with $\forall$

In this section we prove Lemma 6.3, in which case φ has only one $\exists$ and the innermost quantifier is $\forall$. Let the existentially quantified variable be x_j.

For a pair of nodes (x_u, x_v), we define its color $\chi(x_u, x_v)$ to be binary strings corresponding to the truth values of all predicates on them. By preprocessing we can set up a table that allows us to check whether $\chi(x_u, x_v) = c$ for given x_u, x_v and color c in constant time. We also define $\chi(x_k | x_1 \ldots x_{k-1})$ to be the concatenation of $\chi(x_1, x_k), \ldots, \chi(x_{k-1}, x_k)$. For colors composed of only 0 (where all the related predicates are false), we call them "background".

Our algorithm can count the number of x_k's with color $\chi(x_k | x_1 \ldots x_{k-1}) = c$ for all $(x_1, \ldots, x_{k-1})$ and c in time $O(m^{k-2})$. The main idea of the algorithm is to nondeterministically guess x_j and count valid values of x_k, so that it saves the exhaustive search on x_j and x_k.

1. For each combination of $(x_1, \ldots, x_{j-1})$ nodes, we nondeterministically bundle a fixed x_j value to it. This takes time $O(m^{j-1})$, which is at most $O(m^{k-2})$ because $j < k$. In the rest of this algorithm, given any $(x_1, \ldots, x_{j-1})$ values we can find their corresponding x_j value in constant time.

2. We do a $(k-2)$-layer nested loop. On each layer we loop through all (x_i, x_k) edges where x_i is a variable other than x_j or x_k. Then inside all the loops, for each x_k in the $(k-2)$ current (x_i, x_k) edges, we record the color $\chi(x_k|x_1 \ldots x_{k-1})$, where the values of x_j come from the current $(x_1, \ldots, x_{j-1})$.

   Then after the loops we can count the number of x_k's of for each $(x_1, \ldots, x_{k-1})$ and each color.

   This step can be done in time $O(m^{k-2})$.

3. The previous step did not count the x_k's that only appear in (x_j, x_k) edges, i.e. whose $\chi(x_k|x_1 \ldots x_{k-1})$ is all-zero on all non-(x_j, x_k) predicate positions, but not all-zero on some (x_k, x_j) predicate positions. We will count these x_k's in this step.

   For each x_j, we can enumerate all the (x_j, x_k) edges to count the number of x_k's where $\chi(x_j, x_k) = c_{jk}$ for any not all-zero c_{jk}. Also, from the previous step, we can count the number of x_k's s where $\chi(x_k|x_1 \ldots x_{k-1})$ is not all-zero on some non-(x_j, x_k) predicate positions, and also equals c_{ij} on its (x_j, x_k) predicate positions. By subtraction we can get the number of x_k's, where $\chi(x_k|x_1 \ldots x_{k-1})$ is all-zero on non-(x_j, x_k) predicate positions, and equals c_{ij} on its (x_j, x_k) predicate positions. Similarly as the previous step, this process runs in time $O(m^{k-2})$.

4. Now we have counted the x_k's with all non-background colors for all $(x_1, \ldots, x_{k-1})$. The number of x_k's where $\chi(x_k|x_1 \ldots x_{k-1})$ is background can be computed by $|X_k|$ subtracting the numbers of all non-background x_k's.

5. Finally, for all $(x_1, \ldots, x_{k-1})$, we sum the number of x_k's of all colors that satisfy φ. If it always equals $|X_k|$, then the algorithm accepts.

Appendix B Proof Sketches for Section 4

In this appendix, we sketch the details of Theorem 4.1, which is implicit in [JMV13]. Essentially, they reduce CKT-SAT for certain circuit classes $\mathcal{C}$ to k-CNF, and then use the CKT-SAT algorithm to circuit lower bound framework plus some assumed failure of k-SAT hardness (which gives a k-SAT algorithm) to imply circuit lower bounds. We observe that a reduction to k-TAUT is implicit in their proofs, and therefore failure of k-TAUT hardness implies identical circuit lower bounds.

The following theorem (and others like it) is proved in [Wil13] and applied in [Wil14b] to prove breakthrough circuit lower bounds against ACC:

Theorem B.1. *Let $s(n)$ be super-polynomial, and let $\mathcal{C} \subseteq \mathsf{P/poly}$ a class of circuit closed under composition that contains AC^0. If satisfiability or tautology of $\mathcal{C}$-circuits on n variables and n^c gates can be solved in co-nondeterministic time $(2^n \cdot \mathsf{poly}(n^c))/s(n)$, then NE does not have polynomial size $\mathcal{C}$-circuits.*

The final step in proving the above simulates arbitrary $L \in \mathsf{NTIME}[2^n]$ using a reduction to CKT-SAT or CKT-TAUT; the only difference between these two arguments is a single negation.

Let ϕ_x be the 3-CNF such that $x \in L \iff \phi_x \in$ 3-SAT, and let W_x encode a canonical assignment to $\text{var}(\phi_x)$ witnessing that $\phi_x \in$ 3-SAT, if such an assignment exists. The reduction to CKT-SAT constructs a circuit $D \in \mathcal{C}$ such that:

$$D(j) \iff \text{clause } j \text{ of } \phi_x \text{ is satisfied by assignment } W_x$$

Let $\mathcal{I}(\phi_x)$ be some index set for the clauses of ϕ_x. So, we have $x \in L \iff \forall j \in \mathcal{I}(\phi_x)\ D(j)$. Therefore:

$$x \in L \iff D \in \text{CKT-TAUT} \iff \neg D \notin \text{CKT-SAT}$$

The standard reduction runs a fast CKT-SAT algorithm on $\neg D$ and *accepts* iff this algorithm *rejects*. Of course, if we instead make the assumption that there is a fast algorithm for CKT-TAUT, we can run fast CKT-TAUT on D and output the result, which will also be a correct simulation for L.

These techniques were pushed further to the following classes $\mathcal{C}$, which are *not* closed under composition, by optimizing the reduction of arbitrary nondeterministic time languages to 3-SAT [JMV13].

Theorem B.2. *For each of the following classes $\mathcal{C}$, if satisfiability or tautology of circuits in $\mathcal{C}$ can be solved in time $2^n/n^{\omega(1)}$ then there is a problem $f \in \mathsf{E}^{\mathsf{NP}}$ that is not solvable by circuits in $\mathcal{C}$:*

1. *linear-size circuits*
2. *linear-size series-parallel circuits,*
3. *linear-size log-depth circuits*
4. *quasi-polynomial-size SYM-AND circuits.*

The proof of part (1) of the above was already present in [Wil13], but items 2 - 4 required more efficient reductions to 3-SAT. The authors of [JMV13] also showed that if k-SAT is easier than expected, this implies circuit lower bounds for classes 1 - 3 listed above:

Corollary B.3. *In Items (1), (2), and (3) of Theorem B.2 we can, in place of the assumption about $\mathcal{C}$-SAT, substitute (respectively) the following assumptions and obtain a problem in E^{NP} that is not solvable in $\mathcal{C}$*

1. *The exponential time hypothesis (*ETH*) [IP01] is false; i.e., for every $\epsilon > 0$,* 3-SAT *is in time $2^{\epsilon n}$*
2. *The strong exponential time hypothesis (*SETH*) is false [IPZ01]; i.e., there is a $\delta < 1$ such that for every k, k-SAT is in time $2^{\delta n}$*
3. *There is $\alpha > 0$ such that n^α-SAT is in time $2^{n-\omega(n/\log\log n)}$*

The failure of NETH, NSETH, and the k-TAUT version of (3) above imply identical lower bounds, because each argument of Corollary B.3 concludes with a simulation of arbitrary $L \in \mathsf{NTIME}[2^n]$ that reduces to k-SAT. As with Theorem B.1, a single negation transforms this into a reduction to k-TAUT. We sketch the details below.

Build $D \in \mathcal{C}$, the same clause-evaluation circuit used by the proof of Theorem B.1, and then use a structural decomposition of circuits in the class $\mathcal{C}$ to re-express $\neg D$ as the OR of a (relatively)

small number of CNFs. Then, run the assumed faster k-SAT algorithm on each CNF. The overall $\neg D = \bigvee$CNF formula is satisfiable iff one of the CNF formulas is satisfiable, so the simulation of arbitrary $L \in \mathsf{E}^{\mathsf{NP}}$ will *reject* if any CNF is found to be satisfiable, correctly deciding L in too little time.

To reduce to k-TAUT instead of k-SAT, we simply negate the structural decomposition of $\neg D$ to see that $D = \bigwedge$DNF. Recall that $x \in L \iff D \in$ CKT-TAUT. Therefore if we assume a fast (non)deterministic k-TAUT algorithm, we can check if D is a tautology by running a fast k-TAUT algorithm on each "leaf" DNF, thus obtaining a fast simulation of L.

This outlines the proof of:

Corollary B.4. *In Items (1), (2), and (3) of Theorem B.2 we can, in place of the assumption about $\mathcal{C}$-SAT, substitute (respectively) the following assumptions and obtain a problem in E^{NP} that is not solvable in $\mathcal{C}$*

1. *The nondeterministic exponential time hypothesis (*NETH*) is false; i.e., for every $\epsilon > 0$,* 3-TAUT *is in time $2^{\epsilon n}$*
2. *The nondeterministic strong exponential time hypothesis (*NSETH*) is false ; i.e., there is a $\delta < 1$ such that for every k, k-*TAUT *is in time $2^{\delta n}$*
3. *There is $\alpha > 0$ such that n^{α}-*TAUT *is in time $2^{n-\omega(n/\log\log n)}$*

Appendix C Proofs of lemmas in Section 3

We prove the lemmas from section 3 about deterministic FGRs.

Definition C.1 (Fine-Grained Reductions ($\leq_{FGR}$))**.** Let L_1 and L_2 be languages, and let T_1 and T_2 be time bounds. We say that (L_1, T_1) *fine-grained reduces* to (L_2, T_2) (denoted $(L_1, T_1) \leq_{FGR} (L_2, T_2)$) if

(a) $\forall \epsilon > 0 \quad \exists \delta > 0$, $\exists \mathcal{M}^{L_2}$, a deterministic Turing reduction from L_1 to L_2, such that

$$\mathsf{TIME}[\mathcal{M}] \leq T_1^{1-\delta}$$

(b) Let $\tilde{Q}(\mathcal{M}, x)$ denote the set of queries made by $\mathcal{M}$ to the oracle on an input x of length n. The query lengths obey the following time bound.

$$\sum_{q \in \tilde{Q}(\mathcal{M},x)} (T_2(|q|))^{1-\epsilon} \leq (T_1(n))^{1-\delta}$$

Definition C.2 (Randomized Fine-Grained Reductions ($\leq^s_{rFGR}$))**.** Exactly as in the deterministic case, except the Turing reduction from (L_1, T_1) to (L_2, T_2) is a probabilistic machine with some two-sided error bound

$$\Pr[\mathcal{M}^{L_2}(x) = L_1(x)] \geq s$$

We denote a randomized fine grained reduction from L_1 to L_2 with error bound s by $(L_1, T_1) \leq^s_{rFGR} (L_2, T_2)$. Generally, we will use $s = 2/3$, so we denote $\leq^{2/3}_{rFGR}$ by $\leq_{rFGR}$.

Definition C.3 (Fine-Grained Reductions for Functions ($\leq_{fFGR}$))**.** Exactly as in the decision deterministic case, except that the Turing reduction $\mathcal{M}^{f_2}$ is to a function problem f_2 and is expected to produce a functional output. In addition to the existing resource bounds, we bound the size of answers given by the f_2 oracle.

$$\sum_{q \in \tilde{Q}(\mathcal{M},x)} (|f_2(q)|) \leq (T_1(n))^{1-\delta}$$

C.1 Deterministic Fine-grained Reductions

We prove several properties of deterministic fine-grained reductions. They are exactly what one would expect and follow by standard methods.

Lemma C.4 (Fine-grained reductions translate savings for DTIME)**.** *Let $(L_1, T_1) \leq_{FGR} (L_2, T_2)$, and $L_2 \in \mathsf{DTIME}[T_2(n)^{1-\epsilon}]$ for $\epsilon > 0$. There exists $\delta > 0$ such that*

$$L_1 \in \mathsf{DTIME}[T_1(n)^{1-\delta}]$$

Proof. We use the reduction and simulate the oracle calls to L_2 with the efficient algorithm for L_2 to get savings for L_1.

Let $\mathcal{A}_{L_2}$ be the algorithm for L_2 that runs in deterministic time $T_2(n)^{1-\epsilon}$. Let $\mathcal{M}^{L_2}$ be the oracle Turing machine that achieves the fine-grained reducibility from L_1 to L_2. Since $\mathcal{M}^{L_2}$ achieves a fine-grained reduction from L_1 to L_2, we know that $\mathcal{M}^{L_2}$ runs in time $T_1^{1-\delta}$ for some $\delta > 0$. We simulate $\mathcal{M}^{L_2}$ by running $\mathcal{A}_{L_2}$ whenever $\mathcal{M}^{L_2}$ queries L_2. The total time spent running $\mathcal{A}_{L_2}$ when oracle calls are made is less than $T_2^{1-\delta}$. Overall, the simulation of $\mathcal{M}^{L_2}$ will be completed in time $O(T_1^{1-\delta})$, which shows $L_1 \in \mathsf{DTIME}[T_1(n)^{1-\delta}]$. □

Lemma C.5 (Fine-grained reductions transfer savings for $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}$)**.** *Let $(L_1, T_1) \leq_{FGR} (L_2, T_2)$, and $L_2 \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T_2(n)^{1-\epsilon}]$ for some $\epsilon > 0$. Then there exists a $\delta > 0$ such that*

$$L_1 \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T_1(n)^{1-\delta}]$$

Proof. We proceed as in the proof of Lemma C.4, but we need to work with the nondeterministic and co-nondeterministic algorithms for L_2. Let $\mathcal{M}^{L_2}$ be the deterministic oracle Turing machine that achieves a fine-grained reduction from L_2 to L_1. We design a nondeterministic machine $\mathcal{M}'$ for deciding L_1 to show that $L_1 \in \mathsf{NTIME}[T_1(n)^{1-\delta}]$. $\mathcal{M}'$ guesses a table of queries to L_2 (by the machine $\mathcal{M}^{L_2}$) and their answers. $\mathcal{M}'$ will use the nondeterministic algorithm for L_2 to verify the answers to the queries which belong to the language L_2 and the nondeterministic algorithm for the complement $\bar{L}_2$ of L_2 to verify the answers to the queries which are not in L_2. It then simulates $\mathcal{M}^{L_2}$, looking up queries in the verified guess table. If $\mathcal{M}^{L_2}$ ever asks a query which is not in the table, $\mathcal{M}'$ *rejects*. The simulation itself takes time at most $T_2^{1-\delta}$. The additional cost is writing down the query table and checking the answers. Guessing and verification of the query table takes at most $O(T_1^{1-\delta})$ time. Therefore, $L_1 \in \mathsf{NTIME}[T_1(n)^{1-\delta}]$.

Similarly, we can design a nondeterministic machine for deciding the complement $\bar{L}_2$ of L_2 to conclude $L_1 \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T_1(n)^{1-\delta}]$. □

Corollary C.6 (Fine-grained reductions translate savings from reductions)**.** *When the true complexity of a problem is meaningfully smaller than the time bound used in a fine-grained reduction, savings are translated.*

1. *Let* $(L_1, T_1) \leq_{FGR} (L_2, T_2^{1+\gamma})$*, and* $L_2 \in \mathsf{DTIME}[T_2]$*. Then there exists* $\delta > 0$ *such that*

$$L_1 \in \mathsf{DTIME}[T_1^{1-\delta}]$$

2. *Let* $(L_1, T_1) \leq_{FGR} (L_2, T_2^{1+\gamma})$*, and* $L_2 \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T_2]$*. Then there exists a* $\delta > 0$ *such that*

$$L_2 \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T_1^{1-\delta}]$$

Proof. For both cases, perform the following substitution. Let $\hat{T}_2 = T_2^{1+\gamma}$,and set $\epsilon = \frac{\gamma}{1+\gamma} < 1$. Observe that $T_2 = \hat{T}_2^{1/(1+\gamma)} = \hat{T}_2^{1-\epsilon}$. Therefore, if $(L_1, T_1) \leq_{FGR} (L_2, T_2^{1+\gamma})$, we have $(L_1, T_1) \leq_{FGR} (L_2, \hat{T}_2)$, and if $L_2 \in \mathsf{TIME}[T_2]$ then $L_2 \in \mathsf{TIME}[\hat{T}_2^{1-\epsilon}]$. Therefore, either Lemma C.4 or Lemma C.5 applies, and we have the required savings for L_1. □

Lemma C.7 (Fine-grained reductions are closed under composition)**.** *Let* $(A, T_A) \leq_{FGR} (B, T_B)$ *and* $(B, T_B) \leq_{FGR} (C, T_C)$*. It then follows* $(A, T_A) \leq_{FGR} (C, T_C)$*.*

Proof. Let $\mathcal{M}_{AB}^B$ be the machine that achieves a fine-grained reduction from (A, T_A) to (B, T_B). Also let $\mathcal{M}_{BC}^C$ be the machine that achieves a fine-grained reduction from (B, T_B) to (C, T_C). We construct a machine $\mathcal{M}_{AC}^C$ that achieves a fine-grained reduction from (A, T_A) to (C, T_C).

$\mathcal{M}_{AC}^C$ simulates the machine $\mathcal{M}_{AB}^B$ and when it makes an oracle call to B, $\mathcal{M}_{AC}^C$ simulates the machine $\mathcal{M}_{BC}^C$. We argue that $\mathcal{M}_{AC}^C$ satisfies the requied time and query length bounds so it does indeed achieve a fine-grained reduction from (A, T_A) to (C, T_C).

Let x be an input. Let $\tilde{Q}_{AB}(x)$ be the set of queries that the fine-grained reduction machine $\mathcal{M}_{AB}^B$ makes on x. Fix arbitrary $\epsilon_C > 0$. $\mathcal{M}_{AC}^C$ on input x behaves as follows.

1. Simulate $\mathcal{M}_{AB}^B(x)$
2. For any query $q \in B$ to the oracle, run $\mathcal{M}_{BC}^C(q)$

We will show that $\mathcal{M}_{AC}^C$ satisfies the time and query length bounds as desired.

Time bound We need to show that $\mathcal{M}_{AC}^C$ runs in time $T_A^{1-\delta}(|x|)$ for some $\delta > 0$. Let $\mathsf{TIME}[\mathcal{M}(x)]$ denote the number of steps of $\mathcal{M}$ on input x. Let δ_A and δ_B be the constants guaranteed by the corresponding fine-grained reductions

$$\begin{aligned}
\mathsf{TIME}[\mathcal{M}_{AC}^C(x)] &= \mathsf{TIME}[\mathcal{M}_{AB}^B(|x|)] + \sum_{q \in \tilde{Q}_{AB}(x)} \mathsf{TIME}[\mathcal{M}_{BC}^C(q)] \\
&\leq \mathsf{TIME}[\mathcal{M}_{AB}^B(|x|)] + \sum_{q \in \tilde{Q}_{AB}(x)} T_B^{1-\delta_B}(|q|) && \text{since } (B, T_B) \leq_{FGR} (C, T_C) \\
&\leq \mathsf{TIME}[\mathcal{M}_{AB}^B(|x|)] + T_A(|x|)^{1-\delta_A} \\
&\leq O(T_A(|x|)^{1-\delta_A}) && \text{since } A \leq_{FGR} B
\end{aligned}$$

Bounded Queries We need to bound the time complexity of deciding all queries to C made by $\mathcal{M}^C_{AC}$, i.e., we need to show

$$\sum_{q \in \tilde{Q}_{AC}(x)} T_C(|q|)^{1-\epsilon_C} \leq T_A(|x|)^{1-\delta_A}$$

First, observe that since each query to C made by $\mathcal{M}^C_{AC}$ is simulated by a run of $\mathcal{M}^C_{BC}$, we have $\tilde{Q}_{AC}(x) = \bigcup_{z \in \tilde{Q}_{AB}(x)} \tilde{Q}_{BC}(z)$.

It follows

$$\begin{aligned}
\sum_{q \in \tilde{Q}_{AC}(x)} T_C(|q|)^{1-\epsilon_C} &= \sum_{z \in \tilde{Q}_{AB}(x)} \sum_{q \in \tilde{Q}_{BC}(z)} T_C(|q|)^{1-\epsilon_C} \\
&\leq \sum_{z \in \tilde{Q}_{AC}(x)} T_B(|z|)^{1-\delta_B} && \text{since } B \leq_{FGR} C \\
&\leq T_A(|x|)^{1-\delta_A} && \text{since } A \leq_{FGR} B
\end{aligned}$$

Since ϵ_C is arbitrary, we conclude $A \leq_{FGR} C$.

□

ECCC ISSN 1433-8092

http://eccc.hpi-web.de