Electronic Colloquium on Computational Complexity, Report No. 9 (2019)

The Fine-Grained Complexity of Strengthenings of First-Order Logic

Jiawei Gao
jiawei@cs.ucsd.edu
University of California, San Diego

Russell Impagliazzo[*]
russell@cs.ucsd.edu
University of California, San Diego

Abstract

The class of model checking for first-order formulas on sparse graphs has a complete problem with respect to fine-grained reductions, Orthogonal Vectors (OV) [GIKW17]. This paper studies extensions of this class or more lenient parameterizations. We consider classes obtained by allowing function symbols; first-order on ordered structures; adding various notions of transitive closure operations; and stratifications of first-order properties by quantifier depth and variable complexity, rather than number of quantifiers. For some of these classes, OV is still a complete problem, in that significant improvement for the entire class is equivalent to significant improvement for OV algorithms. For these classes, we can also use the improved OV algorithm of [AWY15, CW16] to get moderate improvements on algorithms for the entire class. For other classes, we show that model checking becomes harder than for first-order, under well-studied conjectures such as SETH. For other classes, we show hardness follows from weaker assumptions than SETH.

Surprisingly, whether an extension increases the complexity of model checking seems independent of whether it increases the expressive power of the logic. For example, adding function symbols does not change which problems are expressible by first-order, but does increase the time for model checking under SETH. On the other hand, adding an ordering does not change the fine-grained complexity of model checking, although it increases the logic's expressive power.

1 Introduction

Fine-grained complexity is a relatively new sub-area within theoretical computer science that aims to not only qualitatively classify problems as "easy" or "hard", but (to the extent possible) pinpoint their exact complexities. There are now a wide variety of standard algorithmic problems where no significant improvements in algorithmic running time can be made without refuting one of a few conjectures about well-studied problems, such as the k-SUM problem [GO95], All Pairs Shortest Paths [WW10, AGW14, LWW18], SAT, or Orthogonal Vectors [AWW14, Bri14, ABW15, BI15, BK15, MPS16, KPS17, AR16, ABDN18, BRS$^+$18].

In traditional complexity, classes of problems are related to each other, and individual problems understood by identifying classes for which they are complete. In contrast, most of the results in fine-grained complexity were obtained on a problem-by-problem basis. One reason for this is that results in fine-grained complexity cut across traditional classes, with NP-complete problems reducing to problems within P or even smaller classes. This raises the questions: is it possible to give a fine-grained complexity of classes of problems? Is the notion of completeness useful in fine-grained complexity?

[*]Work supported by a Simons Investigator Award from the Simons Foundation.

ISSN 1433-8092

Some preliminary answers were given in [Wil14b, GIKW17]. Both of these papers consider the class of *first-order definable properties*, the first for the dense case (where each relation is given as a matrix, aka adjacency matrix format), and the second for the sparse case (where the input is given as a list of tuples in the relations, e.g., for graphs, adjacency list format). This class is natural both in terms of computational complexity, where it is the uniform version of AC_0, and in database theory, because these are the queries expressible in basic SQL [AHV95]. First-order logic can also express many polynomial time computable problems: Orthogonal Vectors, k-Orthogonal Vectors, k-Clique, k-Independent set, k-Dominating set, etc. Not only were the likely complexities of the hardest problems (as a function of number of quantifiers) given, but in the second paper, a natural complete problem was identified, the orthogonal vectors problem (discussed below in more detail). The conclusion was that there were substantial improvements possible in the worst-case complexity of model checking for first-order properties if and only if the known Orthogonal Vectors algorithms can be substantially improved. Using a recent sub-polynomial improvement in OV algorithms by [AWY15], they obtained a similar improvement in model-checking every first order property.

There are also some work on classes of problems that are related in spirit, but do not form a well-studied complexity class. [WW10] studies problems related to shortest paths in graphs, and show that many are equivalent as far as having sub-cubic algorithms. [KPS17] studies dynamic programming algorithms with a similar structure, and gives a unified treatment of their fine-grained complexities.

Here, we extend this class-based approach to fine-grained complexity. We consider the fine-grained complexities of various well-studied extensions of first-order logic. Surprisingly, we find that some extensions of first-order that *increase* the expressive power greatly *do not change* the fine-grained complexity of the corresponding model checking, while some extensions that *do not change* expressibility *substantially increase* the complexity of model checking. (This may not be as paradoxical as it would appear at first glance. Parity is not expressible in first-order, but a pre-processing stage to compute the parity would not greatly change the running time of most queries. In the other direction, although at a qualitative level, expressive power might be the same in an extension, the simulation might change quantitative aspects, such as the number of quantifiers. Our bounds give limits on the quantitative price that must be paid in doing the conversion from one logic to another.)

The basic logic that we consider is first-order relational logic. There are a finite list $R_1, ..R_t$ of relation symbols, each with a non-negative integer arity a_i. A formula is built from these symbols applied to variables, with Boolean connectives and the quantifers $\forall$ and $\exists$. A finite model is specified as a universe U and for each R_i a set of tuples $(u_1, ..u_{a_i}) \in U^{a_i}$. For algorithmic purposes, we can either represent a relation as a matrix or tensor of Booleans, which for every tuple, specifies whether or not it is in the relation, or as a list of possible relations. In this paper, we generally use the list representation, but sometimes need to refer to algorithms using the matrix representation. We use n to mean the size of U, and m to be $|U|$ plus the total number of tuples in all relations.

For k-quantifier first-order definable dense graph properties (i.e., structures with unary and binary relations represented in adjacency matrix format), [Wil14b] shows that on a graph of n vertices:

1. If the OV conjecture is true, then for every $\epsilon > 0$, worst-case model checking requires time $\Omega(n^{k-1-\epsilon})$.
2. For $k \geq 9$, model checking can be performed in time $O(n^{k-1})$.

Thus, the likely complexity is resolved under the assumption (which follows from the Strong Exponential Time Hypothesis), for $k \geq 9$ and graph problems. For $3 \leq k \leq 8$ or for non-graph problems, the exact complexity is still open, but upper and lower bounds differ by at most a linear factor.

For sparse first order properties, with arbitrary arities, the characterization is even stronger.

[GIKW17] shows that on a hypergraph of m edges:

1. If the OV conjecture is true, then for every $\epsilon > 0$, worst-case model checking requires time $\Omega(m^{k-1-\epsilon})$. (This follows from the earlier result).
2. OV is complete for first-order: If the OV conjecture is false, then there is an $\epsilon > 0$ so that for every $k \geq 3$, model checking can be done in time $O(m^{k-1-\epsilon})$.
3. Model checking can be improved using the fastest OV algorithm: Unconditionally, model checking can be done in time $m^{k-1}/2^{\Omega(\sqrt{\log m})}$.

So improvements in the entire class of model checking problems are captured by the extent to which OV algorithms can be improved.

We show, somewhat suprisingly, that different natural extensions of first-order sometimes maintain the complexity of model checking exactly, but sometimes change the complexity in an interesting way. Whether the complexity of the model checking problem is changed seems completely independent of whether the extension increases the expressive power of the logic.

In particular we consider:

First-order logic with unary function symbols: In general, function symbols can be replaced with relations representing their graphs, so adding functions does not change the expressive power of first-order logic. However, in doing so, we might increase the number of quantifiers needed to express a property. We show that, assuming the low-dimension OV conjecture (which follows from SETH), the model checking problem increases by almost a linear factor when functions are added. More precisely, we show that assuming the low-dimension OV conjecture holds, then $\forall k \geq 3, \forall \epsilon > 0$, there exists a k-quantifier first-order formula φ with unary functions, so that no algorithm can decide φ in time $O(m^{k-\epsilon})$.

This approaches a factor of m over the upper bound for model checking such formulas without function symbols.

First-order logic on ordered structures: The next three extensions all increase the expressive power of the logics in different ways, but do not change the complexity of the corresponding model-checking problems.

In many applications, the elements of the universe are ordered, e.g., by time or location in memory. First order properties on ordered structures have strictly more expressive power than those on un-ordered structures, even for properties not involving the ordering itself (by Gurevich, Theorem 5.3 in [Lib13]). However, we show that adding orderings does not change the complexity of model checking:

1. If the OV conjecture is true, then for every $\epsilon > 0$, worst-case model checking of ordered structures requires time $\Omega(m^{k-1-\epsilon})$. (This follows from the earlier result for unordered structures).
2. OV is complete for first-order on ordered structures: If the OV conjecture is false, then there is an $\epsilon > 0$ so that for every $k \geq 3$, model checking for structures with ordering can be done in time $O(m^{k-1-\epsilon})$.
3. Model checking can be improved using the fastest OV algorithm: Unconditionally, model checking of formulas over structures with ordering can be done in time $m^{k-1}/2^{\Omega(\sqrt{\log m})}$.

Transitive closures of symmetric relations: Adding transitive closure operations to first-order logic enhances its power significantly. However, here we find a special class involving transitive closure that is still equivalent to first-order: when transitive closure operations are only taken on symmetric input relations. This allows us to query whether two elements are in the same connected component of an undirected graph. We show that the same conclusions above for first-order logic on ordered structures also hold for this class. Let φ be a first-order formula with a fixed number of transitive closure operations that are only taken on symmetric input relations, φ has $k \geq 3$ quantifiers, then

1. If the OV conjecture is true, then for every $\epsilon > 0$, worst-case model checking for φ requires time $\Omega(m^{k-1-\epsilon})$. (This follows from the earlier result for FO without transitive closure).
2. OV is complete for first-order: If the OV conjecture is false, then there is an $\epsilon > 0$ so that for every $k \geq 3$, model checking for φ can be done in time $O(m^{k-1-\epsilon})$.
3. Model checking can be improved using the fastest OV algorithm: Unconditionally, model checking for φ can be done in time $m^{k-1}/2^{\Omega(\sqrt{\log m})}$.

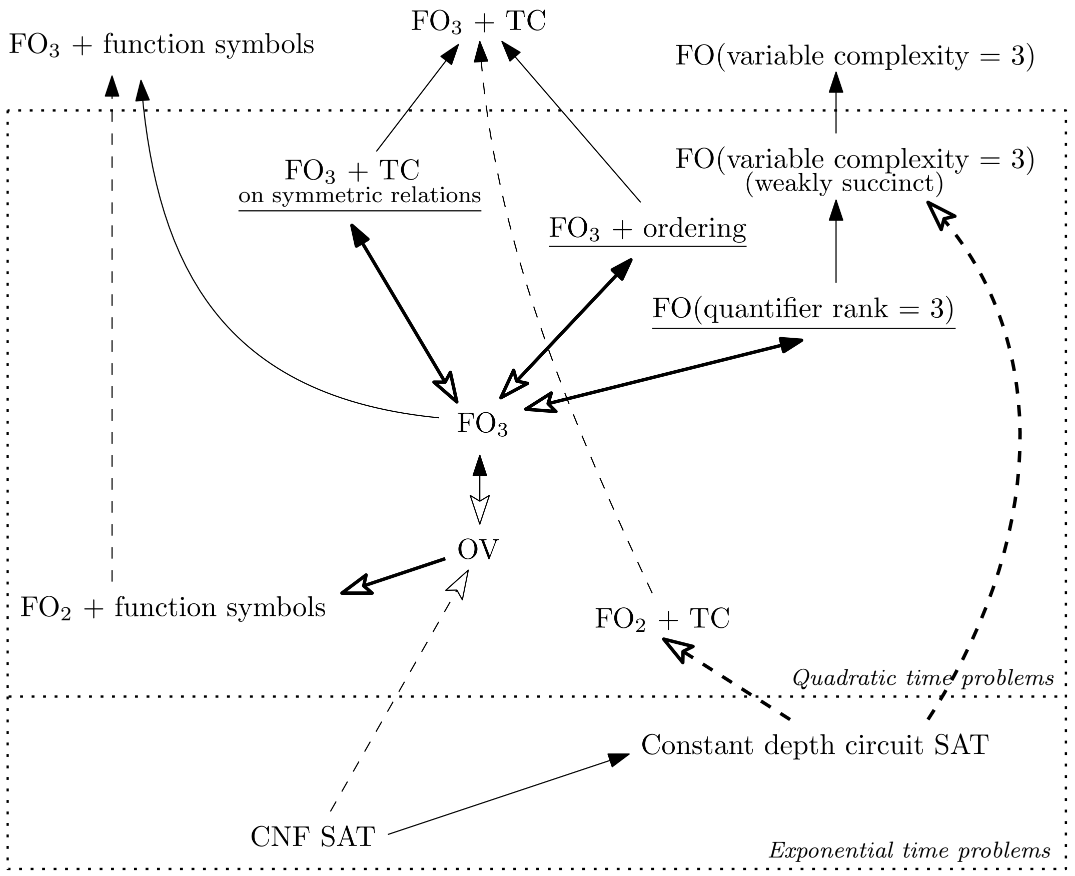
First-order formulas of quantifier rank k: We get a similar equivalence if we measure the complexity of first-order formulas by quantifier rank rather than number of quantifiers. Quantifier rank counts only the depth of nesting of quantifiers, rather than the total number of them. For example $\forall x[(\exists y R(x,y)) \wedge (\forall z S(x,z))]$ has three quantifiers, but since the two inside quantifiers are parallel rather than nested, has only quantifier rank two. However, we show that problems of any fixed quantifier rank, have the same complexity of model checking as for that number of total quantifiers. More precisely, let φ be a first-order formula of quantifier rank k, where $k \geq 3$, then

1. If the OV conjecture is true, then for every $\epsilon > 0$, worst-case model checking for φ requires time $\Omega(m^{k-1-\epsilon})$. (This follows from the earlier result for quantifier number.)
2. OV is complete for first-order: If the OV conjecture is false, then there is an $\epsilon > 0$ so that for every $k \geq 3$, model checking for φ can be done in time $O(m^{k-1-\epsilon})$.
3. Model checking can be improved using the fastest OV algorithm: Unconditionally, model checking for φ can be done in time $m^{k-1}/2^{\Omega(\sqrt{\log m})}$.

Transitive closure of arbitrary binary relations: For the next two extensions, we give some evidence that the model checking problems are not reducible to unextended first-order logic. While we do not give stronger conditional lower bounds in terms of running time, we show that these conditional bounds hold under a substantially weaker assumption than OVC, the analog of SETH for constant depth circuits.

Allowing general transitive closures of arbitrary relations increases the power of first-order logic dramatically. If TC can be negated then the formula can express a even larger class of problems. For the model checking problem on graph problems, and for binary relations, however, the increase is less dramatic. We can compute each new transitive closure of an existing relation in $O(n^3)$ time. Then we can solve the model checking problem on the corresponding dense graph problem. This gives an $O(n^{k-1})$ algorithm for model checking k-quantifier problems for $k \geq 9$, and $O(n^k)$ for any $k \geq 3$. While we were unable to close this gap, or to show that transitive closures increase the complexity, we were able to prove the corresponding lower bound under a weaker assumption than SETH, SETH for polynomial-sized bounded depth formulas. Under this hypothesis, the class of 2-quantifier first-order formulas allowing transitive closure on arbitrary relations cannot be solved in time $O(m^{2-\epsilon})$ for any $\epsilon > 0$. More specifically,

1. The SETH of depth 2 circuit SAT implies the quadratic time hardness of 2-quantifier FO with positive TC only on original relations.
2. The SETH of depth 3 circuit SAT implies the quadratic time hardness of 2-quantifier FO with positive TC on subformulas containing TC only on original relations.



FO$_3$ + function symbols
FO$_3$ + TC
FO(variable complexity = 3)
FO$_3$ + TC on symmetric relations
FO(variable complexity = 3) (weakly succinct)
FO$_3$ + ordering
FO(quantifier rank = 3)
FO$_3$
OV
FO$_2$ + function symbols
FO$_2$ + TC
Quadratic time problems
Constant depth circuit SAT
CNF SAT
Exponential time problems

Figure 1: The expressive power and complexity of problems and classes of problems. FO$_k$ stands for FO formulas with k quantifiers. The solid arrows are reductions directly implied from the expressive power, and the hollow arrows are non-trivial reductions. The solid lines are reductions preserving subquadratic time (except for the one between exponential time problems), while the dashed lines are reductions between problems with different conjectured running time. The thick lines are new results in this paper, and the underlined problems get improved algorithms from these results.

3. In general, the SETH of depth d circuit SAT implies the quadratic time hardness of 2-quantifier FO with $d-1$ nested layers of TC operations.

First-order formulas of variable complexity k: We get a similar result for another measure of the logical depth of forumlas. When the variable names can be reused in different scopes, but there is no restrictions on the quantifier rank, then the situation is similar as when we allow transitive closures on arbitrary binary relations: on formulas of variable complexity k, it is not known whether it can be computed faster than $O(n^{k-1+\omega})$ for $3 \leq k < 9$, or faster than $O(n^{k-1+o(1)})$ when $k \geq 9$, even if a large number of problems in this class can be solved in quadratic time (if they are "weakly succinct", see Appendix B). Here we prove that under the SETH for polynomial-sized constant depth formulas, the class of first-order formulas of variable complexity 3 cannot be solved in time $O(m^{2-\epsilon})$ for any $\epsilon > 0$.

Figure 1 shows the reductions among problems.

2 Organization of the paper

Section 3 introduces the problem definitions, notations, conjectures and basic reduction techniques. Section 4 shows adding function symbols to first-order logic will make the model checking problem strictly harder by a reduction from OV. In Section 5 we show that introducing ordering to FO will not make the model checking harder. Section 6 shows when we allow transitive closures taken on symmetric input relations, the complexity stays the same. Section 7 proves that quantifier rank k formulas are no harder than formulas with k quantifiers in prenex normal form. In Section 8 we study two classes of problems that are hard under the SETH of constant depth circuits: formulas of variable complexity 3, and 2-quantifier formulas with transitive closure operations. In Section 9 we will talk about the open problems. Appendix A gives the baseline algorithm for first-order with ordering and first-order formulas of fixed quantifier rank. In Appendix B we will discuss the running time for formulas of fixed variable complexity.

3 Preliminaries

The Orthogonal Vectors (OV) problem is a well-studied problem in the field of fine-grained complexity. Its hardness is implied by the hardness of CNF-SAT, and implies the hardness of many problems (A list of hardness results under OV conjecture is compiled in [Wil18]). It is defined as follows: Given n boolean vectors of dimension d, decide whether there exists two vectors so that their inner product is zero. k-Orthogonal Vectors (k-OV) is a generalization of OV, where the goal is to find k vectors so that their inner product is zero. Usually $d = \omega(\log n)$. A naive algorithm runs in time $O(n^k d)$ and the current best algorithm is $n^{k-\Omega(1/\log(d/\log n))}$. [AWY15, CW16].

Next we will give the definitions and notations regarding the model checking problems.

Let φ be a fixed formula and let G be an input structure, the model checking problem for φ is to decide whether G satisfies φ. When φ is a first-order formula, we also call it a first-order property problem.

φ is a fixed formula without free variables (i.e. all variables are quantified by either $\exists$ or $\forall$). We use MC(φ) to denote the model checking problem for formula φ. In this paper we usually use letter φ for formulas containing quantifiers, and use letter ψ for quantifier-free subformulas. Letter Q is used to represent a quantifier, either $\exists$ or $\forall$.

The input structure has multiple variable domains and multiple relations. It can be considered as a hypergraph (represented by adjacency list): the elements of the structure correspond to vertices, and the relation tuples of the structure correspond to edges.

The *domain of a variable* is a fixed set of vertices so that a variable in φ can be assigned to any one of the vertices. The total number of vertices is n.

A *relation* is a fixed set of edges (or hyperedges) so that a binary (or t-ary) predicate in φ can correspond to one of the edges. The total number of edges is m. We only consider the case that $m \geq n$.

Because the number of edges is important in describing problem size, in the rest of this paper we define the *weight* of a vertex v to stand for the total number of edges the vertex is in.

Types of first-order property problems and their extensions

- ***k*-quantifier problems**
  Here φ has form $Q_1 x_1 \dots Q_k x_k \psi(x_1, \dots, x_k)$. Without loss of generality, we assume φ is in prenex normal form. For example, the sparse OV problem can be represented by
  $$\varphi_{OV} = \exists x \exists y \forall z (\neg One(x, z) \vee \neg One(y, z)),$$

where x, y are vectors and z is a coordinate of the vectors. $One(x, z)$ is true iff a vector x has a one on its z-th coordinate. The sparse k-OV problem is equivalent to

$$\varphi_{k-OV} = \exists x_1 \dots \exists x_k \forall z \bigvee_{i=1}^{k} (\neg One(x_k, z)).$$

We use the notation FOP_k for the class of $\mathsf{MC}(\varphi)$ where φ is a first-order formula with k quantifiers.

- **Quantifier rank k problems**
  When we can reuse the same variable name in different scopes, for example,
  $$\varphi = \exists x(\exists y(\exists z\psi_1 \wedge \forall z\psi_2) \wedge \forall y(\exists z\psi_3 \vee \forall z\psi_4))$$
  has only three variable names x, y, z, but they represent different variables in different scopes of the formula. The *quantifier rank* of a formula is the maximum depth of nesting of its quantifiers. The above formula is of quantifier rank 3.
  We use the notation $\mathsf{FOP}_{\text{qr}=k}$ for the class of $\mathsf{MC}(\varphi)$ where φ is a FO formula with quantifier rank k. $\mathsf{FOP}_{\text{qr}=k}$ contains FOP_k. It can be solved in time $O(mn^{k-2})$ for any $k \geq 2$ (Lemma A.1).
- **Variable complexity k problems**
  If we do not bound the quantifier rank of φ, it will have even more expressive power, for example, a formula of form
  $$\varphi = \exists x \exists y(R(x, y) \wedge \exists x(R(y, x) \wedge \exists y \exists(R(x, y) \wedge \dots)))$$
  can represent a path of any constant length using only 2 variables.
  A formula with k different variable names is referred to as of *variable complexity k*. A formula is of variable complexity k iff it can be computed by a straightline program, each line has at most k distinct variables (even if written in prenex form). That is, it's equivalent to the result of a sequence of first-order queries of form $R_i = \{(x_1, \dots, x_a) \mid \varphi_i(x_1, \dots, x_a)\}$ where each R_i is an intermediate relation of arity a ($0 \leq a \leq k$), and each φ_i is a first-order formula with at most k variables, including $x_1, \dots, x_a$, which appear as free variables in φ_i.
  If in each line of the corresponding straightline program φ, there is at most one occurrence of an intermediate binary relation computed in a previous line, then φ can be solved in $O(mn)$ time for sparse graphs, which will be shown in Appendix B. In this case we call this variable complexity 3 problem *weakly succinct*.
  We use the notation $\mathsf{FOP}_{\text{vc}=k}$ for the class of $\mathsf{MC}(\varphi)$ where φ is a FO formula with variable complexity k. The class of weakly succinct problems in $\mathsf{FOP}_{\text{vc}=k}$ contains $\mathsf{FOP}_{\text{qr}=k}$ (Each line of the straightline program creates an new intermediate relation on x, y by an intermediate relation on x, y and some original relations on some z. We will not elaborate the details here).
- **First-order on ordered structures**
  If all the elements in the domain of some variable have a total order so that for any two elements a, b it may be $a > b$, $a < b$ or $a = b$, then the comparison relation on two elements is a dense relation. But unlike general dense relations, it can be represented succinctly in the input by $O(n)$ space.
  We use the notation $\mathsf{FOP}_k(\leq)$ for the class of $\mathsf{MC}(\varphi)$ where φ is a k-quantifier FO formula with comparison predicates. $\mathsf{FOP}_k(\leq)$ contains FOP_k. It can be solved in time $O(mn^{k-2})$ for any $k \geq 2$ (Lemma A.1).
- **First-order with transitive closure**
  The transitive closure of a sparse relation may be a dense relation. The comparison relation is a special case, which is the transitive closure of the successor relation. We use TC_R to denote the transitive closure of relation R.
  We use the notation $\mathsf{FOP}_k(\text{TC})$ for the class of $\mathsf{MC}(\varphi)$ where φ is a k-quantifier FO formula allowing transitive closure operations. $\mathsf{FOP}_k(\text{TC})$ contains $\mathsf{FOP}_k(\leq)$.

Assumptions

The complexity is measured in the word RAM model with $O(\log n)$ bit words. The notation $\tilde{O}$ notation is generally used for time complexity hiding sub-polynomial time factors. But in this paper we usually consider savings factors in running time that grow faster than polylogarithmic, so we still use the big-O notation but let it hide polylogarithmic factors.

In this paper, without loss of generality we make the following assumptions.

- Assume $m = n^{1+o(1)}$, because otherwise the $O(mn^{k-2})$ time baseline algorithm (Lemma A.1) is better than the conjectured time $m^{k-o(1)}$.
- Assume that different variables are in different domains. (For instance, the universe of x is X, the universe of y is Y, etc.) In other words, a structure for a k-quantifier formula can be considered as a k-partite graph. However, in the case where transtive closure operations can be taken on a relation, we will assume both variables of the relation are in the same universe.
- We assume that for any tuple of elements, the value of a predicate on this tuple can be queried in constant time. Also, assume that the neighbors of any element v can be enumerated in time linear to the degree of v.

Conjectures

Below is a list of the conjectures about the hardness of the problems mentioned in this paper.

- Strong Exponential Time Hypothesis(SETH) for CNF-SAT: For all $\epsilon > 0$, there exists a k so that k-CNF-SAT cannot be solved in time $O(2^{n(1-\epsilon)})$. [IPZ98]
- Strong Exponential Time Hypothesis(SETH) for circuit class C: For all $\epsilon > 0$, the satisfiability of C cannot be solved in time $O(2^{n(1-\epsilon)})$. [AHWW16]
- Low-dimension OV conjecture(LDOVC), or Strong OV conjecture: For all $\epsilon > 0$, there is no $O(n^{2-\epsilon})$ time algorithm for OV with dimension $d = \omega(\log n)$.
- Moderate-dimension OV conjecture(MDOVC): For all $\epsilon > 0$, there is no $O(n^{2-\epsilon}\mathsf{poly}(d))$ time algorithm that solves OV with dimension d.
- Sparse OV conjecture(SOVC): For all $\epsilon > 0$, there is no $O(m^{2-\epsilon})$ time randomized algorithm for OV where m is the total Hamming weight of all the input vectors. [GIKW17]
- First-order property conjecture(FOPC): There exists integer $k \geq 2$, so that FOP_{k+1} cannot be solved in time $O(m^{k-\epsilon})$ for any $\epsilon > 0$. [GIKW17]

The SETH of CNF-SAT implies LDOVC [Wil05], and LDOVC implies MDOVC because low-dimension OV is a special case of moderate-dimension OV. MDOVC, SOVC and FOPC are equivalent [GIKW17]. The SETH of depth d circuits, where d is a constant greater than 2, is weaker than the SETH of CNF-SAT.

Reductions and reduction techniques

The *fine-grained reduction* was introduced in [WW10]. Let $(\Pi_1, T_1(m)) \leq_{\text{FGR}} (\Pi_2, T_2(m))$ denote that if there is some $\epsilon_2 > 0$ such that problem Π_2 is in $\mathsf{TIME}((T_2(m))^{1-\epsilon_2})$, then problem Π_1 is in $\mathsf{TIME}((T_1(m))^{1-\epsilon_1})$ for some ϵ_1. This means if Π_2 can be solved substantially faster than T_2 then Π_1 can be solved substantially faster than T_1. If both T_1 and T_2 are $O(m^2)$, the reduction is called a subquadratic reduction.

If a reduction from Π_1 to Π_2 is more strict that can preserve any savings factor that can even be sub-polynomial, it is called an *exact-complexity reduction.* Let $(\Pi_1, T_1(m)) \leq_{\text{EC}} (\Pi_2, T_2(m))$ denote that if problem Π_2 is in $\mathsf{TIME}(T_2(m))$, then problem Π_1 is in $\mathsf{TIME}(T_1(m))$. We use these notations not only on single problems but also on classes of problems. Let C_1 and C_2 be classes problems. $(C_1, T_1(m)) \leq_{\text{EC}} (C_2, T_2(m))$ means if all problems in class C_2 are in time $T_2(m)$, then we can solve any problem in C_1 in time $T_1(m)$.

A useful reduction is that $(\mathsf{FOP}_{k+1}, n \cdot T(m, n)) \leq_{\text{EC}} (\mathsf{FOP}_k, T(m, n))$, where $T(m, n)$ is the running time on m edges and n vertices. This is because we can exhaustively search the outermost quantified variable, and use the value as a constant in the rest of the formula, thus reducing it to n

instances of the model checking for k-quantifier formulas. The same technique can also be applied in the reductions for formulas with comparisons, and formulas of quantifier rank k.

The *grouping-reduction technique* is another useful reduction technique, which was introduced in [AWY15], that reduces the Batch OV problem to OV, and in [AWW16], that reduces Hitting Set to OV. In [GIKW17] a weighted version is used on the model checking on sparse structures.

The template of this reduction looks like: Assume the model checking for $\varphi = \exists x_1 \dots \exists x_k P(x_1, \dots, x_k)$ can be decided in time $O(m^k/s(m))$ for some savings factor s, where P is a property on $x_1, \dots, x_k$ that can be decided in time linear to the total weight of $x_1, \dots, x_k$. Then we can list all x such that $\exists x_2 \dots \exists x_k P(x_1, \dots, x_k)$ can be decided in time $O(m^k/s(\mathsf{poly}(m)))$.

The idea is as follows: Pick a threshold size g, which is usually a polynomial of m. For elements of weight greater than g, we enumerate all of them, and for each of them we run a $O(m^{k-1})$ algorithm to decide if P holds on x by doing exhaustive search on all other variables. So the total time is $O(m/g) \cdot O(m^{k-1}) = O(m^k/g)$.

Next, partition the each of the domain of $x_1, \dots, x_k$ into groups so that a group has total weight at most g, and there are $O(m/g)$ groups for each of $x_1, \dots, x_k$. Every time we take a k-tuple of groups, and query φ on this smaller instance. As long as the query returns true, we can find a satisfying x_1 in $O(\log g)$ queries. On finding a satisfying x_1, we mark this x_1 and remove this x_1 from its group. Continue this process until either all x_1 are marked or no combination of groups satisfies φ. There are at most $O(m \log g + (m/g)^k)$ calls to the oracle of φ. The running time is $O(m \log g \cdot m \cdot (m/g)^k \cdot g^k/s(g)) = O(\log g \cdot m^k/s(g))$.

Thus the final running time is $O(m^k/g + \log g \cdot m^k/s(g))$. The logarithmic factor can be omitted if function s grows much faster than polylog functions.

4 FO with unary function symbols

Consider adding function symbols to first order logic. Unary functions can be represented as arrays of linear size, but binary functions increase the input size to quardratic, and so on for higher arities. So to measure complexity in terms of the input size, we only consider unary functions. To simulate higher arities, we could increase the universe to a Cartesian power and then have unary functions on this product space.

While we can simulate any function by a relation coding the graph of the function, $R_f(x, y) \iff f(x) = y$, to express , for example that $f(x_1) = f(x_2)$ we would need to write $\exists y, R_f(x_1, y) \wedge R_f(x_2, y)$. So the number of quantifiers in the translated forumlas would increase, possibly up to the number of function symbols apprearing in the original. However, there is still a trivial $O(n^k)$ algorithm for model-checking a k-quantifier formula with functions. So, assuming OV conjecture, the complexity grows by at most a linear amount over that for first-order without functions.

In this section, we show that this increase is necessary. Compared to the linear time baseline algorithm for the model checking of 2 quantifier formulas, when we introduce function symbols, a 2 quantifier formula may require quadratic time to solve.

Theorem 1. *The low-dimension OV conjecture implies that for any $\epsilon > 0$, there exists 2-quantifier formula φ with function symbols, whose model checking cannot be decided in time $O(m^{2-\epsilon})$.*

The low-dimension k-OV conjecture implies that for any $\epsilon > 0$, there exists k-quantifier formula φ with function symbols, whose model checking cannot be decided in time $O(m^{k-\epsilon})$.

Consider an OV instance where the dimension of vectors $d = c \log n$ for some constant c. We construct a hypergraph for model checking as follows.

Let each vector correspond to an element. Besides these elements, we create $O(\sqrt{n})$ extra elements, each corresponding to a distinct boolean vector of dimension $\frac{1}{2}\log n$. For all pairs of these short vectors that are orthogonal to each other, we create an edge of relation $R_\perp$ between them. There are at most $(\sqrt{n})^2 = O(n)$ pairs of short vectors, so the relation $R_\perp$ is sparse.

Each vector of length $c\log n$ can be partitioned into $2c$ blocks of length $\frac{1}{2}\log n$. We define functions $f_1, \dots f_{2c}$ so that $f_i : \{0,1\}^{c\log n} \to \{0,1\}^{\frac{1}{2}\log n}$ be the mapping from a vector to its i-th block. Therefore, there exists a pair of orthogonal vectors x, y iff in FO with function symbols,

$$\varphi = \exists x \exists y \bigwedge_{i=1}^{2c} R_\perp(f_i(x), f_i(y))$$

is satisfied.

If we can decide such φ in time $m^{2-\epsilon}$, then for any constant $c > 0$, OV of dimension $d = c\log n$ can be solved in time $O(n^{2-\epsilon})$, contradicting the low-dimension OV conjecture. Furthermore, the SETH of CNF-SAT will also be refuted.

The reduction from k-OV is similar, where we create a set of $O(\sqrt[k]{n})$ extra elements representing all boolean vectors of length $(1/k)\log n$, and create kc functions mapping vectors to its blocks of length $(1/k)\log n$. Here, the relation $R_\perp$ is k-ary, defined on the k-tuples of short vectors whose inner product is zero. The total number of these k-ary relations is still $(\sqrt[k]{n})^k = O(n)$.

5 FO with comparison on ordered structures

In this section we will consider the case where elements are given a total pre-ordering, and there are three predicates expressing that an element is greater than, less than, or equivalent to another element in the ordering. The comparison relation is an implicit dense relation but can be represented in $O(n)$ space in the input, by giving a table indexed by element, giving the element's rank within the ordering, with equivalent elements given the same rank. (If we were not given this table, we could use any sorting algorithm to construct it in $O(n\log n)$ time.) Using this table, we can list the elements by this ordering in time $O(n)$, and given any two elements, we can compare them in time $O(1)$. The following theorem shows that adding comparison to first-order logic does not increase the fine-grained complexity because it is equivalent to first-order properties without comparison.

Theorem 2. *For non-decreasing* $2^{\Omega(\sqrt{\log m})} \le s(m) \le m$,

$$(\mathsf{FOP}_k(\le), m^{k-1}/s(\mathsf{poly}(m))) \le_{EC} (\mathsf{FOP}_3, m^2/s(m)).$$

We will show that $(\mathsf{FOP}_3(\le), m^2/s(\mathsf{poly}(m))) \le_{\mathrm{EC}} (\mathsf{FOP}_3, m^2/s(m))$. The reduction from $\mathsf{FOP}_k(\le)$ follows from the quantifier-eliminating downward self-reduction.

Assume in time $T_{FO}(m)$ we can list all x satisfying any formula of form $\exists y Q_3 z \psi_{FO}(x,y,z)$, where ψ_{FO} is a quantifier-free first-order formula without ordering. By the grouping-reduction technique, it is reducible to FOP_3.

Let φ be in prenex normal form, and assume $\varphi = Q_1 x \exists y Q_3 z \psi(x,y,z)$, for otherwise if y is quantified by $\forall$, we will decide the negation of the formula. We show that we can list all x such that $\exists y Q_3 z \psi(x,y,z)$ holds.

We let the *weight* of an element be 1 plus the number of tuples that element occurs in. Let g be a threshold value. First , we decide whether to include x on our list for all x of weight greater than g. There are at most m/g of them. For each large weight x, treating it as a constant we get a 2-quantifier problem. By the baseline algorithm (Lemma A.1), it can be decided in time $O(m)$. So the total time spent is $O(m^2/g)$.

Next, for all y of weight greater than g, we list the set of x that cause the one-quantifier subformula to be true. For each large weight y, treating it as a constant we compute the problem: list all x such that $Q_3 z \psi(x, y, z)$. Using the baseline algorithm we can list all x satisfying $Q_3 z \psi(x, y, z)$ in time $O(m)$. Finally we merge all lists of x computed on each y. For any x in the lists, there must exist a y where $Q_3 z \psi(x, y, z)$ holds.

By triplicating elements, we can assume that the x, y, and z variables are quantified over disjoint domains X, Y, and Z. The ordering relation is defined on the union of all three sets. We remove from X and Y the elements of weight higher than g. We partition the whole universe $X \cup Y \cup Z$ into intervals where for each interval the total weight of elements in $X \cup Y \cup Z$ is between g and $2g$, the interval is a single element of Z of weight higher than g. Thus, there are $O(m/g)$ intervals.

Note that we allow elements to be equivalent in the pre-ordering. When we group the elements into intervals, we will keep all equivalent elements in the same interval, unless there are so many such elements that their total weight is more than g. If that is the case, we break the set of equivalent elements arbitraritly into groups of total weight between g and $2g$, and do not include any non-equivalent elements in these groups.

Let the elements of X, Y, Z in the i-th interval form sets X_i, Y_i, Z_i, respectively. For each pair of groups X_i, Y_j, we need to create a list $L_{i,j}$ of those x in X_i for which there exists a y in Y_j such that the rest of the formula holds true. Our final output will be the union of all these lists.

We call a pair i, j *special* if either $i = j$ or there is some tuple in one of the non-ordering input relations involving elements from both intervals. There are at most $O(m)$ special pairs, since each tuple involves at most a constant number of intervals. We handle special and non-special pairs differently.

Note that each element z not in either of the two intervals Z_i or Z_j has the same ordering relationship with all $x_i \in X_i$ and the same ordering relationship with all $y_j \in Y_j$. We say two elements z_1, z_2 in Z are *indistinguishable* if they have the same ordering relationships to elements in X_i and Y_j, the same evaluation on unary relations, and have no non-ordering relationships involving any elements of X_i or Y_j. For any two indistinguishable z_1, z_2, for any $x \in X_i$, $y \in Y_j$ if the inner-most formula is true for x, y, z_1, then it is also true for x, y, z_2. Thus, the formula, including the innermost quantifier, is true for x, y if and only if it is true relative to a maximal set of distinguishable elements. There are only $O(g)$ elements z with some relation to the two intervals, $O(g)$ in the two intervals, and only constantly many equivalence classes of others under indistinguishability. As a preprocessing step, in linear time, we can for each boolean combination of unary predicates for z, create a sorted list of intervals with such elements. Then we can use binary search to see whether such an element exists in some interval before i, between i and j, or after j. We can find z's with a relation to either X_i or Y_j by searching all tuples involving such elements. Thus, we can construct in time $O(g + \log n)$ a maximal set of $O(g)$ distinguishable elements for a given pair of intervals.

For special intervals, we use the quadratic time baseline algorithm on them, taking time g^2. So the total time is $m \cdot g^2$.

For non-special intervals, we know there are no relations involving elements in the two intervals. Assume without loss of generality that $i \leq j$. We create unary predicates on z that say whether z comes from an interval before i, interval i, an interval between i and j, j, or after j. Call these five predicates $A_1(z), \ldots, A_5(z)$.

Recall that the formula is of the form: $\exists y Q_3 z \psi(x, y, z)$. We can further divide the sets X_i and Y_j into constantly many subsets $X_{i,\alpha}$ and $Y_{j,\beta}$ according to the set of unary relations α and β that are true for x and for y respectively. If for each pair of α, β, we compute the list of $x \in X_{i,\alpha}$ so that $\exists y \in Y_{j,\beta} Q3z\psi(x, y, z)$, the final output is the union of these lists. For each, we can replace all unary predicates of x and of y in ψ to get equivalent formulas $\psi_1(x, y, z), \ldots, \psi_5(x, y, z)$ when

$A_i(z)$ is true on z.

We can divide the z's up into the five cases given by the new unary predicates $A_1, \dots, A_5$. In each case, we will show that ψ can be simplified.

For the different cases, ψ_i will be simpler in different ways. ψ_1, ψ_3, and ψ_5 will have no occurences of the ordering relation. ψ_2 will be a function only of x and z, and ψ_4 only of y and z, but may have ordering predicates.

If Q_3 is $\exists$, our output list is the union of the corresponding lists for the five cases above. (The list for the fourth case is either all x or none.) We will use the baseline algorithm to compute those for cases 2 and 4 in $O(g)$ time. The other three cases are size $O(g)$ instances of 3-quantifier unordered first order statements, and so we can use the best algorithm for model checking on unordered structures for these.

If Q_3 is $\forall$, we only want to consider x, y so that all five conditions are true simultaneously. We can use the baseline algorithm to compute the subset X' of x so that the second case holds, and the subset Y' of y so that the fourth case holds. Then we restrict the universe to those subsets, and solve the other three cases: find the set of $x \in X'$ so that $\exists y \in Y' \forall z \wedge_{l=1,3,5} (A_l(z) \rightarrow \psi_l(x, y, z))$

To finish the proof, we need to construct the simplified formulas φ_l for the five cases. First note that in all cases, we have fixed all unary relations on x and y, and for each two distinct intervals, any x and y in the intervals have the same order, so we can also fix ordering relations between x and y.

For any z not in interval i or j, the ordering between z and any x in X_i is fixed , and the same for any $y \in Y_j$. (For some z that occur before i, we might have equivalence to elements in X_i, for some, they are strictly smaller, so what this fixed relationship is does depend on z.) We can introduce six new unary relations on z coding this ordering with respect to x and with respect to y. So for cases 1, 3 and 5, we can replace any ordering relation between x and z or y and z with the corresponding unary predicate of z. This removes all ordering relations to obtain ψ_1, ψ_3 and ψ_5.

To create ψ_2, we are restricting to z in interval i. This fixes the ordering information between z and any $y \in Y_j$, but not between z and x. However, since the pair is not special, there are no relations that involve both y and z. Thus, we can replace those relations by *false* in ψ. Similarly, there are no relations that involve both x and y, so we can replace those relations by $False$ as well. In addition, we have already fixed the unary predicates of y, and comparisons between y and x or z. Thus, the restricted formula now has no occurences of y at all, so is a predicate $\psi_2(x, z)$.

The case for ψ_4 is symmetric.

In summary, fix a combination of α, β, then for a pair of non-special interval X_i, Y_j, we first decide ψ_2: list x so that a formula on x, z holds (in this case, y is indistinguishable to x and $z \in Z_i$ so the variable y is omitted). Next, decide ψ_4: list y so that a formula on y, z holds (here x is indistinguishable to y and $z \in Z_i$ so we omit the variable x). Here we get a list of x and a list of y, then on these two lists, we decide ψ_1, ψ_3, ψ_5.

Thus, the total time we spend on this sub-problem is $O(q)$ to solve the second and fourth cases, and then the best algorithm to solve a three quantifier unordered query on $O(q)$ sized inputs for the other three cases . So for each non-special pair of i, j, the time for φ_2 is $O(g) + T_{FO}(O(g))$. For special pairs, the time is $T_{FO}(g^2)$. There are at most $O(m)$ special pairs, and at most $O((m/g)^2)$ non-special pairs of i, j. We spent time $O(m^2/g)$ to solve the "high degree" case. So the total time is $O(m + m^2/g + (m/g)^2 \cdot T_{FO}(g) + m \cdot g^2)$.

If $s(m)$ is a polynomial improvement factor, i.e., $T_{FO}(m) = m^{2-\epsilon}$, then by letting $g = m^{\frac{1}{2+\epsilon}}$ the running time is $O(m^{2-\frac{\epsilon}{2+\epsilon}})$.

By the algorithm for OV in [AWY15, GIKW17], we get an improved algorithm for $\mathsf{FOP}_k(\leq)$ in time $m^k/2^{\Theta(\sqrt{\log m})}$.

6 FO with transitive closure on symmetric input relations

Consider the model checking of a first-order formula with transitive closures, where the transitive closure operation can only be taken on symmetric input relations. In this case $TC_R(x, y)$ is true iff x and y are in the same connected component by edges of undirected edge set R. Thus the formula can have binary predicates about whether two variables are in the same connected component or not. Note that there can be more than one symmetric relations that the transitive closure operation can be taken on.

Theorem 3. *For non-decreasing $2^{\Omega(\sqrt{\log m})} \leq s(m) \leq m$, if FOP_3 is in time $O(m^2/s(m))$, then model checking for a fixed k-quantifier FO formula with transitive closures only taken on symmetric input binary relations is solvable in time $O(m^{k-1}/s(\mathsf{poly}(m)))$.*

Proof. Like the previous section, here we consider the case $k = 3$, and demonstrate a subquadratic time reduction to FOP_3.

Let there be t different TC relations, where t is a constant integer. Let $T_t(m)$ be the running time on instances with t TC relations. We will reduce an instance with t TC relations to instances with $t-1$ TC relations. Here we pick one of the TC relations, and will only deal with this TC relation. The goal is to reduce to instances without this TC relation.

Here we assume that in the input, each vertex has a "category", indicating which connected component it is in. The formula φ contains predicates of form $TC_R(x, y)$ to represent that x and y are in the same connected component by edges of relation R.

Let $\varphi = Q_1 x Q_2 y Q_3 z \psi(x, y, z)$ Let the quantifier Q_2 be $\exists$, otherwise we will decide the negation of the formula.

ψ can be transformed into a disjunction of itself with $TC_R(x, y)$,$TC_R(x, z)$ and $TC_R(y, z)$ replaced by all combinations of *true* and *false* respectively. Let C be the set of all the combinations (there are at most $2^3 = 8$ combinations).

So we consider the model checking of

$$\varphi = Q_1 x \exists y Q_3 z \left[\bigvee_{c \in C}(\psi_c \wedge (x, y, z \text{ satisfy } c))\right]$$

ψ_c is ψ with all TC predicates replaced by *true* or *false* according to c.

Let g be a threshold value. First of all, find out all x of weight greater than g. On each of the x, we solve a 2 quantifier problem in time $O(m)$ by the baseline algorithm. Next, find all y of weight greater than g. On each of the y, treating y as a constant we using the baseline algorithm we can list all satisfying x in time $O(m)$, and thus we can check for all x whether there exists a satisfiable y. There are at most $O(m/g)$ such large weight elements, so the total time to deal with these elements is $O(m^2/g)$.

Next, list all elements in $X \cup Y$ so that elements in the same category are listed consecutively. Furthermore, we list all categories of total weight greater than g before the other small categories. Partition the big list into $O(m/g)$ groups so that each has total weight at most g. We make sure that each category of total weight at least g is broken into groups that contain no elements from other categories. The small-weight categories are merged together to make groups of total weight as near to g as possible. In the i-th group, let the sets of elements in X, Y be X_i, Y_i respectively.

For each pair of sets (X_i, Y_j) and each truth value combination c, do the following case analysis.

- **Case 1:** If all elements of X_i and Y_j are from the same category, and c implies $x \in X_i, y \in Y_i$ are in the same category, then all pairs of x, y satisfy c. Next, find the set edges between

X_i, Z and between Y_i, Z satisfying c, and make a query to the oracle solving $t-1$ TC relation problems. The running time is $T_{t-1}(g)$.

- **Case 2:** If X_i and Y_j have elements from the same categories and also elements from different categories, then i and j must be equal or adjacent integers. In this case we just query the baseline algorithm, so the time is $O(g^2)$.
- **Case 3:** If elements in X_i and elements in Y_j are from completely different categories, and c implies x, y should be in different categories, then all pairs of x, y satisfy c. This case is similar as the first case.

By some preprocessing, for any pair of i, j, it is easy to tell which of the above three cases the pair (X_i, Y_j) is in.

There are $O((m/g)^2)$ instances of time $T_{t-1}(g)$, and $O(m/g)$ instances of time $O(g^2)$. The total time is $O((m/g)^2 \cdot T_{t-1}(g) + (m/g) \cdot g^2 + m^2/g)$. If $T_{t-1}(m) = m^2/s_{t-1}(m)$ then by taking $g = m^{1/2}$ there is $T_t(m) = O(m^2/s_{t-1}(m^{1/2}))$. Letting $T_t(m) = m^2/s_t(m)$, we get $s_t(m) = O(s_{t-1}(m^{1/2})) = O(s_{t-2}((m^{1/2})^{1/2})) = \cdots = O(s(m^{1/2^t}))$. Thus $T_t(m) = O(m^2/s(m^{1/2^t}))$. □

7 FO formulas of quantifier rank k

A formula with quantifier rank k may have more than k variables when converted to prenex normal form, but the following theorem shows that even if it seems more powerful, it is reducible to quantifier number k problems.

Theorem 4. *For non-decreasing $2^{\Omega(\sqrt{\log m})} \le s(m) \le m$,*

$$(\mathsf{FOP}_{qr=k}, m^{k-1}/\mathsf{poly}(s(m))) \le_{EC} (\mathsf{FOP}_3, m^2/s(m)).$$

We will show that $(\mathsf{FOP}_{\mathrm{qr}=3}, m^3) \le_{\mathrm{FGR}} (\mathsf{FOP}_3, m^2)$. The reduction from $\mathsf{FOP}_{\mathrm{qr}=k}$ follows from the quantifier-eliminating downward self-reduction.

We will use a "Normal Problem" as an intermediate problem in the reduction. It is defined as follows:

List all x such that $\varphi_N(x)$ holds, where $\varphi_N(x) = \exists y[(\bigwedge_{1\le i\le L} \exists z_i \psi_i(x, y, z_i)) \wedge (\forall z' \psi'(x, y, z'))]$. Each z_i is a distinct variable from scope Z_i. Each ψ_i is a conjunction of predicates[1]. The predicates can appear either positively or negated.

In Lemma 7.1 we show a reduction from the Normal Problem to FOP_3, and in Lemma 7.2 we show a reduction from an $\mathsf{FOP}_{\mathrm{qr}=3}$ problem to the Normal Problem.

Lemma 7.1. *(Normal Problem, $m^2/\mathsf{poly}(s(m))) \le_{EC} (\mathsf{FOP}_3, m^2/s(m))$.*

Proof. Let d be a threshold value of the weight of elements.
Step 1. Decide for x of weight at least d on some Z_i.

For those x of degree greater than d on some Z_i, there can be at most m/d of them. We enumerate all such x and then solve corresponding 2-quantifier problems in linear time by the baseline algorithm. The total time is $O(m^2/d)$.
Step 2. Decide for y of weight at least d on some Z_i.

For those y of weight greater than d on some z_i, there can be at most m/d of them. We enumerate all such y's and then list all the x in the corresponding 2-quantifier problems in linear time by the baseline algorithm. Finally we can merge all the lists of x. The total time is $O(m^2/d)$.
Step 3. Decide for x and y of weight less than d.

[1]Here we assume all predicates are either unary or binary. A ternary predicate on x, y, z_i can be treated in a similar way as a binary predicate on either x, z_i or y, z_i.

For each ψ_i, we consider the two cases based on whether it contains any positive occurrences of binary predicates on z_i.

Step 3-1. Consider the ψ_i's where all binary predicates involving z_i are negative.

In this case, we will show that if $|Z_i|$ is large enough, then the conjunction of all negative predicates on z_i is always true, otherwise the problem is easy to be transformed to the case in Step 3-2.

If $|Z_i|$ is greater than $2d$, then for any pair of x and y, at least one $z_i \in Z_i$ is not adjacent to x or y by any relation, because x and y together have no more than $2d$ neighbors. Thus all the negative predicates on z_i are true on the triple (x, y, z_i). So we can replace the conjunction of all these negative predicates by *true* in ψ_i, leaving only binary predicates on x, y and unary predicates.

If $|Z_i|$ is less than $2d$, then in time $O(m \cdot 2d)$ we can create a new relation $R_c(x, z_i)$. $R_c(x, z_i)$ is true iff all binary relations on (x, z_i) evaluate to false. We remove all binary relations on (x, z_i) from ψ_i, and append "$\wedge R_c(x, z_i)$" to the end of it. Because $R_c(x, z_i)$ appears positively, we will decide it in the next step. After creating the new relation, the size of the structure has become $m' = m \cdot 2d$.

Step 3-2. Now in all of the remaining ψ_i's, either some binary predicate on (x, z_i) is positive or some on (y, z_i) is positive.

Without loss of generality, assume that in the formulas ψ_i, for $1 \le i \le u$ the pairs (x, z_i) are in some positive predicates, and for $i > u$, the pairs (x, z_i) do not appear in any positive binary predicate. Similarly, assume for $j > t$ the pairs (y, z_j) are in some positive predicates, and for $j \le t$, the pairs (y, z_j) do not appear in any positive binary predicate. It must be $t \le u$, because for any z_i there is at least one positive predicate on z_i.

The idea of this reduction is to let the big variable $\widetilde{x}$ contain variables z_i for $i \le u$, and let the big variable $\widetilde{y}$ contain variables z_j for $j > u$. For each pair of $\widetilde{x}$ and $\widetilde{y}$, there is a unique tuple of $(z_1, \ldots, z_L)$. Therefore, each $\exists z_i$ can be replaced by $\forall z_i$.

On each x, for all u tuples of neighbors $z_1 \in Z_1, \ldots, z_u \in Z_u$, we create a new element $\widetilde{x} = (x, z_1, \ldots, z_u)$ in the domain $\widetilde{X}$. Because x has at most d neighbors, the number of distinct $\widetilde{x}$ is bounded by d^u. Next, we create the following new relations on $\widetilde{x}$. Define an auxiliary relation $R_\ni$ where $R_\ni(\widetilde{x}, x)$ is true iff the tuple represented by $\widetilde{x}$ contains x, and $R_\ni(\widetilde{x}, z_i)$ is true iff $\widetilde{x}$ contains z_i.

Next we replace all relations on x by relations on $\widetilde{x}$.

- For each unary relation $R_k(x)$, we replace it by new unary relation R_k^* where $R_k^*(\widetilde{x})$ is true iff $R_k(x) \wedge R_\ni(\widetilde{x}, x)$.
- For each unary relation $R_k(z_i)$ where $i \le u$, we replace it by new unary relation R_k^* where $R_k^*(\widetilde{x})$ is true iff $R_k(z_i) \wedge R_\ni(\widetilde{x}, z_i)$.
- For each binary relation $R_k(x, z_i)$ where $i \le u$, we replace it by new unary relation R_k^* where $R_k^*(\widetilde{x})$ is true iff $R_k(x, z_i) \wedge R_\ni(\widetilde{x}, x) \wedge R_\ni(\widetilde{x}, z_i)$.
- For each binary relation $R_k(x, z_j)$ where $j > u$, we replace it by new binary relation R_k^* where $R_k^*(\widetilde{x}, z_j)$ is true iff $R_k(x, z_j) \wedge R_\ni(\widetilde{x}, x)$.
- For each binary relation $R_k(x, z')$, we replace it by new binary relation R_k^* where $R_k^*(\widetilde{x}, z')$ is true iff $R_k(x, z') \wedge R_\ni(\widetilde{x}, x)$.

There are at most $m \cdot d^u$ distinct elements of $\widetilde{x}$, so the unary relations on it is also bounded by this value. From each edge on x we have created at most d^u new edges. Since there are m' edges, the total size of new binary relations is $O(m' \cdot d^L)$.

Similarly, on each y, for all $L - u$ tuples of neighbors $z_1 \in Z_{u+1}, \ldots, z_L \in Z_L$, we create a new element $\widetilde{y} = (y, z_{u+1}, \ldots, z_L)$ in the domain $\widetilde{Y}$. Again we replace old relations by new relations on $\widetilde{y}$ in a similar way as those on $\widetilde{x}$.

- For each unary relation $R_k(y)$, we replace it by new unary relation R_k^* where $R_k^*(\widetilde{y})$ is true iff $R_k(y) \wedge R_\ni(\widetilde{y}, y)$.
- For each unary relation $R_k(z_j)$ where $j > u$, we replace it by new unary relation R_k^* where $R_k^*(\widetilde{y})$ is true iff $R_k(z_j) \wedge R_\ni(\widetilde{y}, z_j)$.
- For each binary relation $R_k(y, z_j)$ where $j > u$, we replace it by new unary relation R_k^* where $R_k^*(\widetilde{y})$ is true iff $R_k(y, z_j) \wedge R_\ni(\widetilde{y}, y) \wedge R_\ni(\widetilde{y}, z_j)$.
- For each binary relation $R_k(y, z_i)$ where $i \leq u$, we replace it by new binary relation R_k^* where $R_k^*(\widetilde{y}, z_i)$ is true iff $R_k(y, z_i) \wedge R_\ni(\widetilde{y}, y)$.
- For each binary relation $R_k(y, z')$, we replace it by new binary relation R_k^* where $R_k^*(\widetilde{y}, z')$ is true iff $R_k(y, z') \wedge R_\ni(\widetilde{y}, y)$.

Similarly, the total size of the new relations is also $O(m' \cdot d^L)$.

Next, we deal with relations between $\widetilde{x}$ and $\widetilde{y}$.

- For each binary relation $R_k(x, y)$, we replace it by new binary relation R_k^* where $R_k^*(\widetilde{x}, \widetilde{y})$ is true iff $R_k(x, y) \wedge R_\ni(\widetilde{x}, x) \wedge R_\ni(\widetilde{y}, y)$.

Because the number of $\widetilde{x}$ corresponding to an x is d^u and the number of $\widetilde{y}$ corresponding to an y is d^{L-u}, and because there are m' relations on x, y, the total size of the new relations is $O(m' \cdot d^L)$.

After modifying the input structure, next we will modify the formula $\varphi_N(x)$. We change the predicates in each formula ψ_i to get ψ_i^*, and change the predicates in ψ' to get ψ'^*.

- For each occurrence of predicate $R_k(x)$, we replace it by $R_k^*(\widetilde{x})$. For each occurrence of predicate $R_k(x)$, we replace it by $R_k^*(\widetilde{y})$.
- For each occurrence of predicate $R_k(z_i)$ where $i \leq u$, we replace it by $R_k^*(\widetilde{x})$. For each occurrence of predicate $R_k(z_j)$ where $j > t$, we replace it by $R_k^*(\widetilde{y})$.
- For each occurrence of predicate $R_k(x, z_i)$ where $i \leq u$, we replace it by $R_k^*(\widetilde{x})$.
- For each occurrence of predicate $R_k(x, z_j)$ where $j > u$, we replace it by $R_k^*(\widetilde{x}, z_j)$.
- For each occurrence of predicate $R_k(y, z_j)$ where $j > u$, we replace it by $R_k^*(\widetilde{y})$.
- For each occurrence of predicate $R_k(y, z_i)$ where $i \leq u$, we replace it by $R_k^*(\widetilde{y}, z_i)$.
- For each occurrence of predicate $R_k(x, y)$, we replace it by $R_k^*(\widetilde{x}, \widetilde{y})$.
- For each occurrence of predicate $R_k(x, z')$, we replace it by $R_k^*(\widetilde{x}, z')$. For each occurrence of predicate $R_k(y, z')$, we replace it by $R_k^*(\widetilde{y}, z')$.

We merge all z_i elements and z' elements into the same universe Z, and create unary predicates $IsZ_1(z), \ldots, IsZ_L(z)$ and $IsZ'(z)$ to represent whether a z is originally from some certain Z_i or in Z'.

Now the goal of the new problem is to list all $\widetilde{x}$ such that $\exists \widetilde{y} \forall z$, all of the following holds.

- For i in range 1 to t, $(IsZ_i(z) \wedge R_\ni(\widetilde{x}, z)) \to \psi_i^*(\widetilde{x}, \widetilde{y}, z)$
- For i in range $u + 1$ to L, $(IsZ_i(z) \wedge R_\ni(\widetilde{y}, z)) \to \psi_i^*(\widetilde{x}, \widetilde{y}, z)$
- $IsZ'(z) \to \psi'^*(\widetilde{x}, \widetilde{y}, z)$

So it is a "List-∃-∀" type problem of size $O(m \cdot 2d \cdot d^L) = O(md^{L+1})$. By the grouping-reduction technique, it is reducible to FOP_3 [GIKW17]. Assume "List-∃-∀" problems are in time $m^2/s(m)$. Then the Normal Problem is in time $O(m^2/d + (md^{L+1})^2/s(md^{L+1})) \leq O(m^2/d + (md^{L+1})^2/s(m))$. By choosing $d = s(m)^{\frac{1}{2L+3}}$, we get running time $O(m^2/s(m)^{\frac{1}{2L+3}})$.

□

Lemma 7.2. *There is a linear time Turing reduction from any* $\mathsf{FOP}_{qr=3}$ *problem to the Normal Problem.*

Proof. Let the variables defined in the outermost layer be named x, let the variables defined in the middle layer be named y and let the variables defined in the innermost layer be named z.

The structure outside the outermost quantifiers

For a quantifier rank 3 formula φ that is composed of form $(Q_i x \varphi_i(x))$ connected by ANDs and ORs, we decide each $Q_i x \varphi_i(x)$ separately. For each $Q_i x \varphi_i(x)$, we will show that we can compute a list of x such that $\varphi_i(x)$ is true, so that we can do union and intersection operations on all the lists and decide the value of φ.

The outermost layer of quantifiers

To decide the truth value of $\varphi_i(x)$ for every x, we write $\varphi_i(x)$ in DNF, treating any quantified subformulas as atoms, so that $\varphi_i(x)$ has form $\bigvee_{j\in[J]} \bigwedge_{k\in[K]} (Q_{ijk} y \varphi'_{ijk}(x,y))$ for some constants J and K. For each x, we will decide for all j,k the truth value of $Q_{ijk} y \varphi'_{ijk}(x,y)$. Thus finally we can decide whether for each x there exists some j such that for all k, $Q_{ijk} y \varphi'_{ijk}(x,y)$ holds. If Q_{ijk} is $\forall$, we decide its negation (that is $\exists y \neg\varphi'_{ijk}(x,y)$), and after we finally get a list of x, we complement the list. Thus we can only consider the case $Q_{ijk} = \exists$.

The second layer of quantifiers

We fix the x, i, j, k in the previous step, and consider subformula of form $\exists y \varphi'(x,y)$. We write it in DNF so that it has form $\exists y \bigvee_{g\in[G]} \bigwedge_{h\in[H]} (Q_{gh} z \psi'_{gh}(x,y,z)) = \bigvee_{g\in[G]} \exists y \bigwedge_{h\in[H]} (Q_{gh} z \psi_{gh}(x,y,z))$. Then $\bigwedge_{h\in[H]} (Q_{gh} z \psi'_{gh}(x,y,z))$ can be written in form $(\bigwedge_\ell \exists z \psi_\ell(x,y,z)) \wedge \forall z \psi'(x,y,z)$ by merging all the $\forall z$ subformulas into one big $\forall z$ connected by $\wedge$. Next, write each ψ_ℓ in DNF, and move the $\vee$'s outside the the"$\exists z$"s, and distribute with the "$\bigwedge_\ell$", so that $(\bigwedge_\ell \exists z \psi_\ell(x,y,z))$ is equivalent to a disjunction of $(\bigwedge_{\ell'} \exists z \psi_{\ell'}(z))$ where each $\psi_{\ell'}$ is a conjunction of predicates and negated predicates. Because "$\vee$" commutes with "$\exists$", the big disjunction before "$\exists z$" can be moved outside "$\exists y$". So in the end, we only need to solve a constant number of instances of form "List all x such that $\exists y[(\bigwedge_{\ell'} \exists z \psi_{\ell'}(z)) \wedge (\forall z \psi'(x,y,z))]$. □

8 Conditional hardness under the SETH of constant depth circuits

The satisfiability of higher depth circuits may be harder than the satisfiability of CNF. Thus, the Strong Exponential Time Hypothesis of a circuit of depth greater than 2 is weaker than the SETH of CNF-SAT. It may be possible that even if the SETH of CNF-SAT is refuted, the SETH of higher depth circuits still holds true. This section shows that in this case, variable complexity 3 formulas and 2-quantifier formulas with transitive closures would require quadratic time.

8.1 Hardness of variable complexity 3 formulas

Thw following theorem shows that the SETH of constant depth circuit implies the quadratic-time hardness of $\mathsf{FOP}_{\mathrm{vc}=3}$.

Theorem 5. *If $\mathsf{FOP}_{vc=3}$ is solvable in time $O(m^{2-\epsilon})$ for some $\epsilon > 0$, then the satisfiabilty of constant depth circuits of size M with N variables is solvable in time $2^{(1-\epsilon/2)N} \cdot \mathsf{poly}(M)$.*

In Circuit SAT, without loss of generality we assume that the circuit is in De Morgan form: it has d levels, where the gates of level $(i+1)$ only have input wires from gates of level i. The NOT gates only appear in the bottom level, which we call level 0. Let the level of AND and OR gates nearest to the input wires be level 1, and the output gate be level d.

Now we reduce the satisfiability of this circuit to a property defined by an FO formula of variable complexity 3. For the N input variables, we split them into two sets of size $N/2$ each. Let α represent partial assignments of the first $N/2$ variables, and let β represent those of the rest $N/2$ variables. So there are $2^{N/2}$ distinct α and β. For each gate g and each partial asssignment α, create a variable g_α representing the tuple (g,α). Define predicate $Same(g_\alpha, g)$ to be true iff the gate in g_a is the same gate as g. Define unary predicate $\mathit{IsAND}(g_\alpha)$ to true iff g is an AND gate, and similarly define $\mathit{IsOR}(g_\alpha)$ for whether g is an OR gate. For any tuple (g'_α, g_α) sharing the same α where gate g' is input to g, we let relation $\mathit{Input}(g'_\alpha, g_\alpha)$ be true on this tuple.

For each level 1 gate g and each α, consider the two cases. If g is an AND gate, we evaluate whether the partial assignment α does not falsify g. If so, then let the relation $\mathit{Sat}_1(g_\alpha)$ be true, otherwise false. If g is an OR gate, we evaluate whether the partial assignment already makes g true. If so, let the relation $\mathit{Sat}_1(g_\alpha)$ be true, otherwise false.

Similarly, for each level 1 gate g and each β, if g is an AND gate and β does not falsify g, then let the relation $\mathit{Sat}_2(g,\beta)$ be true, otherwise false. If g is an OR gate and β makes g true, then we let the relation $\mathit{Sat}_2(g,\beta)$ be true, otherwise false.

Next we will compute d intermediate relations from $\mathit{TrueGates}_1$ to $\mathit{TrueGates}_d$. $\mathit{TrueGates}_i(g_\alpha,\beta)$ holds iff g is a level-i gate, and the assignment by α and β satisfies g.

$$\begin{aligned}
\mathit{TrueANDGates}_1 &= \{(g_\alpha,\beta) \mid (g_\alpha \in \mathit{Level}_1) \wedge \mathit{IsAND}(g_\alpha) \wedge \mathit{Sat}_1(g_\alpha) \wedge \exists g(\mathit{Same}(g_\alpha, g) \wedge \mathit{Sat}_2(g,\beta))\}\\
\mathit{TrueORGates}_1 &= \{(g_\alpha,\beta) \mid (g_\alpha \in \mathit{Level}_1) \wedge \mathit{IsOR}(g_\alpha) \wedge (\mathit{Sat}_1(g_\alpha) \vee \exists g(\mathit{Same}(g_\alpha, g) \wedge \mathit{Sat}_2(g,\beta)))\}\\
\mathit{TrueGates}_1 &= \{(g_\alpha,\beta) \mid \mathit{TrueANDGates}_1(g_\alpha,\beta) \vee \mathit{TrueORGates}_1(g_\alpha,\beta)\}
\end{aligned}$$

For $i = 2$ to $i = d$, we define the following intermediate relations:

$$\begin{aligned}
\mathit{TrueANDGates}_i = \{(g_\alpha,\beta) \mid &(g_\alpha \in \mathit{Level}_i)\\
&\wedge \mathit{IsAND}(g_\alpha) \wedge \forall g'_\alpha \in \mathit{Level}_{i-1}(\mathit{Input}(g'_\alpha, g_\alpha) \to \mathit{TrueGates}_{i-1}(g'_\alpha,\beta))\}\\
\mathit{TrueORGates}_i = \{(g_\alpha,\beta) \mid &(g_\alpha \in \mathit{Level}_i)\\
&\wedge \mathit{IsOR}(g_\alpha) \wedge \exists g'_\alpha \in \mathit{Level}_{i-1}(\mathit{Input}(g'_\alpha, g_\alpha) \wedge \mathit{TrueGates}_{i-1}(g'_\alpha,\beta))\}\\
\mathit{TrueGates}_i = \{(g_\alpha,\beta) \mid &\mathit{TrueANDGates}_i(g_\alpha,\beta) \vee \mathit{TrueORGates}_i(g_\alpha,\beta)\}
\end{aligned}$$

Finally, the circuit is satisfiable iff $\exists g_\alpha \exists \beta\, \mathit{TrueGates}_d(g_\alpha,\beta)$.

Here, each intermediate relation is defined by 3 variables, therefore, the total variable complexity is 3. Also, in the definition formula of each intermediate relation, there is at most one occurence of any previously computed intermediate binary relations. By Appendix B, the formula is weakly succinct, therefore it can be solved in quadratic time.

The total number of elements is $2^{N/2}\mathsf{poly}(M)$. The total number of original relations is also $2^{N/2}\mathsf{poly}(M)$. So if $\mathsf{FOP}_3(\mathrm{TC})$ is time $O(m^{2-\epsilon})$, then the Circuit SAT instance is in $2^{(1-\epsilon/2)N} \cdot \mathsf{poly}(M)$ time, contradicting the SETH of constant depth circuits.

8.2 Hardness of 2 variable formulas with transitive closure

This section proves the conditional hardness for the case where the formula has only two quantifiers, where transitive closure operations can appear arbitrarily.

Theorem 6. *Let SAT_d be the satisfiability problem of depth d circuits with N variables of size M.*

1. *If the model checking for 2-quantifier FO formula with positive TC only on original relations is in time $O(m^{2-\epsilon})$ for some $\epsilon > 0$, then SAT_2 can be solved in time $2^{(1-\epsilon/2)N} \cdot \mathsf{poly}(M)$.*
2. *If the model checking for 2-quantifier FO formula with positive TC on subformulas containing TC on original relations can be solved in time $O(m^{2-\epsilon})$ for some $\epsilon > 0$, then SAT_3 can be solved in time $2^{(1-\epsilon/2)N} \cdot \mathsf{poly}(M)$.*
3. *In general, if the model checking for 2-quantifier FO formula with d nested layers of TC operations can be solved in time $O(m^{2-\epsilon})$ for some $\epsilon > 0$, then SAT_{d-2} can be solved in time $2^{(1-\epsilon/2)N} \cdot \mathsf{poly}(M)$.*

The reduction is similar to the one for $\mathsf{FOP}_{\text{vc}=3}$. We again use partial assignments α, β, and let variable g_α to represent tuple (g, α). Relations *Same*, *Input* are also defined in the same way. This time we assume that in the circuit, on each level either all gates are AND or all gates are OR.

The bottom 2 levels of gates

• If the bottom level are AND gates, and the next level are OR gates:

First, define a relation *Sat* as follows:

$$\begin{aligned} \mathit{Sat} =& \{(g,\beta) \mid g \in \mathit{Level}_1 \wedge (\beta \text{ does not make } g \text{ false})\} \\ & \cup \{(g_\alpha, g) \mid g_\alpha \in \mathit{Level}_1 \wedge \mathit{Same}(g_\alpha, g) \wedge (\alpha \text{ does not make } g \text{ false})\} \\ & \cup \{(g'_\alpha, g_\alpha) \mid g'_\alpha \in \mathit{Level}_2 \wedge g_\alpha \in \mathit{Level}_1 \wedge \mathit{Input}(g_\alpha, g'_\alpha)\} \end{aligned}$$

The above relation can be created in time $2^{n/2}\mathsf{poly}(m)$, where m is the size of the circuit. This relation is like a union of $\mathit{Sat}_1, \mathit{Sat}_2$, $\mathit{TrueGates}_1$ and *Input* relations in the previous section.

Next, we define an intermediate relation for level 2 gates:

$$\begin{aligned} \mathit{TrueGates}_2 =& \{(g_\alpha,\beta) \mid g_\alpha \in \mathit{Level}_2 \wedge TC_{\mathit{Sat}}(g_\alpha, \beta)\} \\ & \cup \{(g'_\alpha, g_\alpha) \mid g'_\alpha \in \mathit{Level}_3 \wedge g_\alpha \in \mathit{Level}_2 \wedge \mathit{Input}(g_\alpha, g'_\alpha)\} \end{aligned}$$

We claim that level 2 gate g' is satisfied by α and β iff $\mathit{TrueGates}_2$ is true on tuple g'_α, β. This is because $\mathit{TrueGates}_2$ is true on g'_α, β where g' is a level 2 gate iff there is a path $g'_\alpha \to g_\alpha \to g \to \beta$ by *Sat*, where g is a level 1 gate. This means neither α and β make the AND gate g false, so g is satisfied by α and β. Also g is input to the OR gate g', so g' is also satisfied. In the other direction, if g' is satisfied by α and β then there must be such a path.

• If the bottom level are OR gates, and the next level are AND gates:

This case is analogous to the previous case because AND is the negation of OR. First, define a relation *Falsify* as follows:

$$\begin{aligned} \mathit{Falsify} =& \{(g,\beta) \mid g \in \mathit{Level}_1 \wedge (\beta \text{ does not make } g \text{ true})\} \\ & \cup \{(g_\alpha, g) \mid g_\alpha \in \mathit{Level}_1 \wedge \mathit{Same}(g_\alpha, g) \wedge (\alpha \text{ does not make } g \text{ true})\} \\ & \cup \{(g'_\alpha, g_\alpha) \mid g'_\alpha \in \mathit{Level}_2 \wedge g_\alpha \in \mathit{Level}_1 \wedge \mathit{Input}(g_\alpha, g'_\alpha)\} \end{aligned}$$

Then we define an intermediate relation for level 2 gates:

$$\begin{aligned} \mathit{TrueGates}_2 =& \{(g_\alpha,\beta) \mid g_\alpha \in \mathit{Level}_2 \wedge \neg TC_{\mathit{Falsify}}(g_\alpha, \beta)\} \\ & \cup \{(g'_\alpha, g_\alpha) \mid g'_\alpha \in \mathit{Level}_3 \wedge g_\alpha \in \mathit{Level}_2 \wedge \mathit{Input}(g_\alpha, g'_\alpha)\} \end{aligned}$$

We claim that level 2 gate g' is satisfied by α and β iff $\mathit{TrueGates}_2$ is true on tuple g'_α, β. This is because $\mathit{TrueGates}_2$ is false on g'_α, β where g' is a level 2 gate iff there exists a path $g'_\alpha \to g_\alpha \to g \to \beta$ by *Falsify*, where g is a level 1 gate. This means neither α and β make the OR gate g true, so g is falsified by α and β. Also g is input to the AND gate g', so g' is also falsified.

Higher levels of gates

From level 3 up, if the current level i are OR gates and the level $i-1$ are AND gates, then define intermediate relations

$$\begin{aligned} \mathit{TrueGates}_i =& \{(g_\alpha,\beta) \mid g_\alpha \in \mathit{Level}_i \wedge TC_{\mathit{TrueGates}_{i-1}}(g_\alpha, \beta)\} \\ & \cup \{(g'_\alpha, g_\alpha) \mid g'_\alpha \in \mathit{Level}_{i+1} \wedge g_\alpha \in \mathit{Level}_i \wedge \mathit{Input}(g_\alpha, g'_\alpha)\} \end{aligned}$$

Otherwise, if the current level i are AND gates and the level $i-1$ are OR gates, then define

$$\begin{aligned}
\mathit{TrueGates}_i =& \{(g_\alpha, \beta) \mid g_\alpha \in \mathit{Level}_i \wedge \neg TC_{\mathit{FalseGates}_i}(g_\alpha, \beta)\} \\
\mathit{FalseGates}_i =& \{(g_\alpha, \beta) \mid g_\alpha \in \mathit{Level}_i \wedge \neg\, \mathit{TrueGates}_i(g_\alpha, \beta)\} \\
& \cup \{(g'_\alpha, g_\alpha) \mid g'_\alpha \in \mathit{Level}_{i+1} \wedge g_\alpha \in \mathit{Level}_i \wedge \mathit{Input}(g_\alpha, g'_\alpha)\}
\end{aligned}$$

For OR gates g'_α where g' is on level i, $\mathit{TrueGates}_i(g'_\alpha, \beta)$ is true iff there exists some g_α where g is on level $i-1$ satisfying $\mathit{TrueGates}_{i-1}(g_\alpha, \beta)$ and $\mathit{TrueGates}_{i-1}(g'_\alpha, g_\alpha)$. This means g is satisfied by α, β and g is input to g'.

For AND gates g'_α where g' is on level i, $\mathit{TrueGates}_i(g'_\alpha, \beta)$ is false iff there exists some g_α where g is on level $i-1$ satisfying $\mathit{FalseGates}_{i-1}(g_\alpha, \beta)$ and $\mathit{FalseGates}_{i-1}(g'_\alpha, g_\alpha)$. This means g is falsified by α, β and g is input to g'.

Finally, the circuit is satisfiable iff $\exists g_\alpha \in \mathit{Level}_d \exists \beta\, \mathit{TrueGates}_d(g_\alpha, \beta)$.

Like the previous section, the total number of elements is $2^{N/2}\mathsf{poly}(M)$, and the total number of original relations is also $2^{N/2}\mathsf{poly}(M)$.

9 Open problems

1. A very general type of open problem is to see whether other complexity classes have complete problems under fine-grained reductions, or give evidence that there are no such complete problems. To make sense, we need a stratification of the class where each layer in the stratification has a reasonable conjecture for its worst-case complexity. For example, we could look at $\mathsf{SPACE}(k \log n)$, with conjectured complexity n^k, if we restrict the tape alphabet to binary.

2. $\mathsf{MC}(\varphi)$ where φ is a variable complexity k formula with only unary and binary predicates can be written in a straightline program whose intermediate relations have arity at most 2. Thus it can be solved in $\tilde{O}(n^{k-3+\omega})$ time, and when $k \geq 9$, it can be solved in time $n^{k-1+o(1)}$, by [Wil14b]. When the input is sparse, will there be better algorithms? From the space complexity point of view, it possible to succinctly represent the intermediate relations without explicitly listing $O(n^2)$ tuples?

3. What is the best algorithm for FO formulas with t function symbols, where t is a fixed small constant, such as 2 or 3? In this case, can a 2-quantifier problem be solved in time m^2 or faster? One idea is to use grouping-reduction to eliminate one function each time, but this time we cannot guarantee that on any pair of groups, the relations on their function values are still sparse compared to the group size.

4. On a directed graph with edge set E, is there a reduction from a problem of form $(\exists x \in V)(\exists y \in V)[\mathrm{TC}_E(x, y) \wedge P(x, y)]$ where P is a property decidable in time linear to the sum of weights of x and y, to a problem of form $(\exists x \in V)(\exists y \in V)P(x, y)$? This would be interesting because the former problem is highly sequential and the latter is highly parallel.

5. The first-order logic can be extended to more expressive classes in many ways, including least fixed point, and temporal logic. It would be interesting if they can be studied in the fine-grained complexity context.

Acknowledgements

The authors sincerely thank Marco Carmosino for comments on improving this paper.

References

[ABDN18] Amir Abboud, Karl Bringmann, Holger Dell, and Jesper Nederlof. More consequences of falsifying SETH and the orthogonal vectors conjecture. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 253–266. ACM, 2018.

[ABW15] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 59–78. IEEE, 2015.

[AGW14] Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. Subcubic equivalences between graph centrality problems, apsp and diameter. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 1681–1697. SIAM, 2014.

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of databases: the logical level*. Addison-Wesley Longman Publishing Co., Inc., 1995.

[AHWW16] Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends: Or: A polylog shaved is a lower bound made. In *Proc. STOC*, pages 375–388. ACM, 2016.

[AR16] Udit Agarwal and Vijaya Ramachandran. Fine-grained complexity and conditional hardness for sparse graphs. *arXiv preprint arXiv:1611.07008*, 2016.

[AWW14] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *International Colloquium on Automata, Languages, and Programming*, pages 39–51. Springer, 2014.

[AWW16] Amir Abboud, Virginia Vassilevska Williams, and Joshua Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete Algorithms*, pages 377–391. SIAM, 2016.

[AWY15] Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 218–230. SIAM, 2015.

[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 symposium on Theory of computing*, pages 51–58. ACM, 2015.

[BK15] Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 79–97. IEEE, 2015.

[Bri14] Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless seth fails. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 661–670. IEEE, 2014.

[BRS$^+$18] Arturs Backurs, Liam Roditty, Gilad Segal, Virginia Vassilevska Williams, and Nicole Wein. Towards tight approximation bounds for graph diameter and eccentricities. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 267–280. ACM, 2018.

[CW16] Timothy M Chan and Ryan Williams. Deterministic APSP, Orthogonal Vectors, and More: Quickly derandomizing Razborov-Smolensky. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1246–1255. SIAM, 2016.

[GIKW17] Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and Ryan Williams. Completeness for first-order properties on sparse structures with algorithmic applications. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 2162–2181, 2017.

[GO95] Anka Gajentaan and Mark H Overmars. On a class of o (n2) problems in computational geometry. *Computational geometry*, 5(3):165–185, 1995.

[IP99] Russell Impagliazzo and Ramamohan Paturi. Complexity of k-SAT. In *Computational Complexity, 1999. Proceedings. Fourteenth Annual IEEE Conference on*, pages 237–240. IEEE, 1999.

[IPZ98] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 653–662. IEEE, 1998.

[KPS17] Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. On the Fine-Grained Complexity of One-Dimensional Dynamic Programming. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:15, 2017.

[Lib13] Leonid Libkin. *Elements of finite model theory.* Springer Science & Business Media, 2013.

[LWW18] Andrea Lincoln, Virginia Vassilevska Williams, and Ryan Williams. Tight hardness for shortest cycles and paths in sparse graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1236–1252. Society for Industrial and Applied Mathematics, 2018.

[MPS16] Daniel Moeller, Ramamohan Paturi, and Stefan Schneider. Subquadratic algorithms for succinct stable matching. In *International Computer Science Symposium in Russia*, pages 294–308. Springer, 2016.

[PW10] Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. In *SODA*, volume 10, pages 1065–1075. SIAM, 2010.

[Wil05] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2):357–365, 2005.

[Wil14a] Richard Ryan Williams. The polynomial method in circuit complexity applied to algorithm design (invited talk). In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 29. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.

[Wil14b] Ryan Williams. Faster decision of first-order graph properties. In *Proceedings of the 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)*, page 80. ACM, 2014.

[Wil18] Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the ICM*, 2018.

[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.

A Baseline Algorithms

Lemma A.1. *For any integer $k \geq 2$, FOP_k, $\mathsf{FOP}_{qr=k}$ and $\mathsf{FOP}_k(\leq)$ are in time $O(mn^{k-2})$. The model checking for k-quantifier formulas with transitive closure operations only on symmetric input binary relations is also in time $O(mn^{k-2})$.*

The baseline algorithm for FOP_k is proved in [GIKW17]. For $\mathsf{FOP}_{\mathrm{qr}=k}$ and $\mathsf{FOP}_k(\leq)$, we only need to show the case where $k = 2$. The cases for $k > 2$ follows from the quantifier-eliminating downward self-reduction.

The linear time baseline algorithm for $\mathsf{FOP}_{\mathrm{qr}=2}$ is straightforward from the baseline algorithm for FOP_2. Let the variables in the outer scopes be x and the one in the inner scopes be y. For each variable named x and each variable named y, we can compute $\#(x) = |\{y \in y \mid \psi(x, y)\}|$ for any quantifier-free formula $\psi(x, y)$. Thus, for a variable x, we can list all elements in the domain of x satisfying any $\varphi(x)$ where φ has quantifier rank 1. Thus we can decide $\exists x\varphi(x)$ and $\forall x\varphi(x)$ for any variable named x.

The linear time baseline algorithm for $\mathsf{FOP}_2(\leq)$ is also adapted from the baseline algorithm for FOP_2. We prove that $\mathsf{FOP}_2(\leq) \subseteq \mathsf{TIME}(m)$. Let $\varphi = Q_1 x\ Q_2 y\ \psi(x, y)$, where

$$\psi(x, y) = ((x < y) \wedge \psi_<(x, y)) \vee ((x = y) \wedge \psi_=(x, y)) \vee ((x > y) \wedge \psi_>(x, y))$$

for each x, we let $Y_<, Y_=, Y_>$ be the subsets of Y less than x, equal to x or greater than x. These sets are disjoint. The proof for FOP_2 showed that each time taking one $x \in X$, we can compute $\#_<(x) = |\{y \in Y_< \mid \psi(x, y)\}|$ (and similarly $\#_=(x), \#_>(x)$ for $Y_=, Y_>$ respectively) in time linear to the degree of x. Thus for each x, if y is quantified by $\exists$ then we check whether $\#_<(x) + \#_=(x) + \#_>(x) > 0$. If y is quantified by $\forall$ then we check whether $\#_<(x) + \#_=(x) + \#_>(x) = |Y|$.

For k-quantifier formulas with transitive closure operations only on symmetric input relations, it is easy to see that using the same method, for each x we can count the number of y satisfying a relation and also satisfying a constant number of predicates about whether x and y are in the same connected component by a certain undirected edge relation.

B Baseline algorithm for variable complexity k

B.1 Variable complexity 2

Lemma B.1. $\mathsf{FOP}_{vc=2} \subseteq \mathsf{TIME}(m)$

For the complement of a sparse relation, we call it a *co-sparse relation.* If a relation R is co-sparse, we can represent it by its complement $\bar{R}$, which takes only $O(m)$ space.

First, convert the variable complexity 2 formula φ into a constant number of first-order queries of at most 2 variables. Each query is one of the following forms.

1. $R(x,y) = R_i(x,y) \wedge R_j(x,y)$
2. $R(x,y) = R_i(x,y) \vee R_j(x,y)$
3. $R(x,y) = \neg R_i(x,y)$
4. $R(x) = \exists y R_i(x,y)$
5. $R(x) = \forall y R_i(x,y)$
6. $R = \exists x R_i(x)$
7. $R = \forall x R_i(x)$
8. $R = \neg R_i$

We will show that if we have already computed all of the previous queries in $O(m)$ time, then we can compute the current query (or its negation, if it is co-sparse) in $O(m)$ time.

1. Intersection
   - The intersection of sparse relation $R_i(x,y)$ and sparse relation $R_j(x,y)$ can be computed in time $O(\min(|R_i|, |R_j|))$, by going through the shorter list of R_i and R_j, and check if the tuple satisfies the other relation.
   - The intersection of sparse $R_i(x,y)$ and co-sparse $R_j(x,y)$ can be computed in time $O(|R_i|)$, by going through all tuples of R_i, and check if the tuple satisfies the other relation.
   - The intersection of co-sparse $R_i(x,y)$ and co-sparse $R_j(x,y)$ can be computed in time $O(|\bar{R}_i + \bar{R}_j|)$, by letting $\bar{R}_t$ be the union of $\bar{R}_i$ and $\bar{R}_j$.
2. Union

   It is reducible to the intersection of two relations, by De Morgan's Law.
3. Existential quantifier
   - $R(x) = \exists y R_i(x,y)$ for sparse $R_i(x,y)$ can be computed in time $O(|R_i|)$ by going through all tuples of R_i and list all x that appear in some tuple.
   - $R(x) = \exists y R_i(x,y)$ for co-sparse $R_i(x,y)$ can be computed by taking the complement of $\forall y \bar{R}_i(x,y)$.
4. Universal quantifier
   - $R(x) = \forall y R_i(x,y)$ for sparse $R_i(x,y)$ can be computed in time $O(|R_i|)$ by going through all tuples of R_i, for each x count how many tuples it is in, and finally list all x that appear in $|Y|$ tuples.
   - $R(x) = \forall y R_i(x,y)$ for co-sparse $R_i(x,y)$ can be computed by taking the complement of $\exists y \bar{R}_i(x,y)$.

B.2 3 and more variables

Any formula of 3 quantifiers is can be solved in time $O(n^{\omega})$ [Wil14b]. By applying this algorithm on every line, we can decide a formula of variable complexity 3 in the amount of time. This can be generalized to any $k \geq 3$.

Theorem 7. *(Strengthening of Williams' algorithm) Any first-order property defined by a formula of variable complexity k is decidable in time $\tilde{O}(n^{k-3+\omega})$ for $c \geq 2$. For $k \geq 9$, it can be decided in $n^{k-1+o(1)}$ time.*

SETH implies k-OV for $k \geq 2$ requires time $n^{k-o(1)}$. Therefore, assuming SETH, for $2 \leq k < 8$, it is impossible to express k-OV in FO using only k variables, without blowing up the input size by a polynomial factor.

B.3 Case analysis on FO with three variables

Now we consider formulas of variable complexity 3.

If a ternary intermediate relation is created in the straightline program in a line of form $R(x,y,z) = \psi_i(x,y,z)$, then ψ must be quantifier-free. Whenever this ternary intermediate relation is used in other places, we can just replace the relation by ψ_i. Thus we can without loss of gererality assume that all intermediate relations are unary or binary.

We define three types of variable complexity 3 formulas.

1. **Strongly Succinct:** The formula is equivalent to a straightline program where all intermediate relations are unary. One example is to decide if there exists a length ℓ chain of orthogonal vectors: $v_1 \perp v_2, v_2 \perp v_3, \dots, v_{\ell-1} \perp v_\ell$ for some constant ℓ.

   Because in computing the straighline program, no dense relation is created, we can use the algorithm for "List-$\exists$-$\forall$" problems for each line. Thus, it is subquadratic time reducible to OV by the reduction in [GIKW17].

2. **Weakly Succinct:** The formula is equivalent to a straightline program where in each line there is at most one occurrence of an intermediate binary relation. Most natural problems of variable complexity 3 are in this case. The formula we have constructed for the constant-depth circuit satisfiability is weakly succinct. $\mathsf{FOP}_{\text{qr}=3}$ problems are also weakly succinct.

   In this case, it can be solved in $O(mn)$ time for sparse graphs. The proof will be presented in the end of this section.

3. **Non-succinct:** The formula is not weakly succinct. In this case, it is solvable in matrix multiplication time, and in $O(n^3)$ using combinatorial algorithms.

   An example is that, in a sparse graph, decide whether there is a pair $(x \in X, y \in Y)$ such that for all $z \in Z$, z either has a neighbor not adjacent to x, or has a neighbor not adjacent to y. In FO, it is

   $$\exists x \exists y \forall z (\exists y' \neg E(x,y') \wedge E(z,y')) \vee (\exists x' \neg E(y,x') \wedge E(z,x'))$$

   which is equivalent to the straightline program

   $$\begin{aligned} R_1(y,z) &= \exists y'(\neg E(x,y') \wedge E(z,y')) \\ R_2(x,z) &= \exists x'(\neg E(y,x') \wedge E(z,x')) \\ \varphi &= \exists x \exists y \forall z (R_1(y,z) \vee R_2(x,z)) \end{aligned}$$

   An equivalent problem is: Given three families of sets, A, B, C, decide if for all $\forall S_1 \in A, \forall S_2 \in B$, there is a set $S_3 \in C$ contained in the intersection of A and B.

   $$\forall S_1 \in A \forall S_2 \in B \exists S_3 \in C (S_3 \subseteq S_1 \cap S_2)$$

   In FO,

   $$\forall x \in A \forall y \in B \exists z \in C (\forall y'(y' \in x \vee \neg y' \in z) \wedge \forall x'(x' \in y \vee \neg x' \in z))$$

   It is open whether non-succinct formulas can be decided in $O(m^2)$ time.

Proof for the $O(mn)$ upper bound for weakly succinct formulas.

Consider each line of the straightline program $R(x, y) = Q_z\psi(x, y, z)$. Without loss of generality assume Q_z is $\exists$, for otherwise we can compute the complement of R. Furthermore, we also assume $\psi(x, y, z)$ is a conjunction. For otherwise, we will write $\psi(x, y, z)$ in DNF, and consider each conjunction separately.

Case 1: The intermediate relation is on x, y.

Enumerate all x, for each x, we can list y such that $Q_z\psi(x, y, z)$ holds for x, y, using the $O(m)$ time baseline algorithm. The time is $O(mn)$.

Case 2: The intermediate relation is on y, z.

Case 2-1: Some binary predicate on x, z appears positively in the conjunction.

We enumerate the edges of the positive predicate, to get the pairs of (x, y), and for each of them, enumerate all z, and check if $\psi(x, y, z)$ holds. This takes time $O(mn)$.

Case 2-2: All binary predicates on x, z appear negatively in the conjunction.

Let ψ' be the subformula of the conjunction that contains all predicates on y, z, including the intermediate relation.

For each y, count how many z satisfy ψ' with y. Let the sum be $f(y)$. The total time is $O(n^2)$.

Enumerate all edges on x, z and enumerate all y. In this way we can count for each pair of (x, y) the number of z so that ψ' is satisfied on y, z and also there is an edge between x, z. Let the sum be $g(x, y)$. The total time is $O(mn)$.

For each pair of x, y, the value $f(y) - g(x, y)$ is the number of z such that ψ' is true and also there are no edges between x, z.

Case 2-3: There are no binary predicates on x, z.

Then there must be binary predicates on x, y, or otherwise x is isolated from other variables in the formula, making it easier to decide. We do a similar counting argument as Case 2-2.

Case 3: The intermediate relation is on x, z.

This case is equivalent to Case 2, because we can switch variables x and y in the formula.