Lower Bounds for Kernelizations

Yijia Chen [*]
Shanghai Jiaotong University

Jörg Flum [†]
Albert-Ludwigs-Universität Freiburg

Moritz Müller [‡]
Albert-Ludwigs-Universität Freiburg

Abstract

Among others, we show that every parameterized problem with a "linear OR" and with NP-hard underlying classical problem does not have a polynomial reduction to itself that assigns to every instance x with parameter k an instance y with $|y| = k^{O(1)} \cdot |x|^{1-\varepsilon}$ unless the polynomial hierarchy collapses to its third level (here ε is any given real number greater than zero).

1. Introduction

Often, if a computationally hard problem must be solved in practice, one tries, in a preprocessing step, to reduce the size of the input data. This approach has been widely studied and applied in parameterized complexity and it is known as *kernelization* there. We recall the basic concepts.

Parameterized complexity is a refinement of classical complexity theory, in which one measures the complexity of an algorithm not only in terms of the total input length n, but also takes into account other aspects of the input codified as the parameter k. Central to parameterized complexity theory is the notion of fixed-parameter tractability. It relaxes the classical notion of tractability by allowing algorithms whose running time can be exponential but only in terms of the parameter. This is based on the idea to choose the parameter in such a way that it can be assumed to be small for the instances one is interested in. To be precise, a problem is said to be *fixed-parameter tractable* if it can be decided by an *fpt-algorithm*, that is, an algorithm whose running time is $f(k) \cdot p(n)$, where f is an arbitrary computable function and p a polynomial.

A *kernelization* $\mathbb{K}$ of a parameterized problem is a polynomial time algorithm that computes for every instance x of the problem an equivalent instance $\mathbb{K}(x)$ of a size bounded in terms of k (the parameter of the instance x). This suggests a new method for designing fpt-algorithms: To decide a given instance x, we compute the kernel $\mathbb{K}(x)$ and then decide if $\mathbb{K}(x)$ is a yes-instance by brute-force. The converse holds, too: Every fixed-parameter tractable problem has a kernelization. The proof of this fact is easy; however it gives only a "trivial" kernel with no algorithmic impact.

Besides efficient computability, an important quality of a good kernelization is *small kernel size*. The notion of *polynomial kernelization* is an abstract model for small kernel size. A kernelization $\mathbb{K}$ is polynomial if there is a polynomial p such that for all instances x (with parameter k), the size of $\mathbb{K}(x)$ is bounded by $p(k)$.

Polynomial kernelizations are known for many parameterized problems (compare [12]). However, till recently, besides artificial problems, only few natural problems were known to have *no* polynomial kernelization (one being the model-checking for monadic second-order logic on trees parameterized by the length of the second-order formula). This has changed, since a machinery has been developed showing that no problem "having an OR" has a polynomial kernelization unless the polynomial hierarchy collapses (cf. [4, 6]). Various applications of this machinery were given in [4, 6], in particular, in [6] it was shown that the problem SAT parameterized by the number of propositional variables of the input formula has no polynomial kernelization.

In this paper we refine a central ingredient of this machinery to obtain better lower bounds. Applied to the SAT problem we show:

[*]Email: yijia.chen@cs.sjtu.edu.cn
[†]Email: joerg.flum@math.uni-freiburg.de
[‡]Email: moritz.mueller@math.uni-freiburg.de

Assume that the polynomial hierarchy does not collapse. Then for every $\varepsilon > 0$ there is no polynomial time algorithm that for every instance α of SAT with k variables computes an equivalent instance α' with

$$|\alpha'| \leq k^{O(1)} \cdot |\alpha|^{1-\varepsilon}. \tag{1}$$

This result is a particular instance of a general theorem that yields lower bounds of the type in (1) for every problem "having a linear OR" (compare Theorem 34 for the precise statement). For problems satisfying an apparently weaker condition, namely only "having an OR for instances with constant parameter" we still get quite good lower bounds; in case of SAT it would be:

$$|\alpha'| \leq k^{O(1)} \cdot |\alpha|^{o(1)}. \tag{2}$$

As already mentioned, concrete kernelizations yield algorithms for solving parameterized problems efficiently for small parameter values. Conceptually similar are compression algorithms, even though the intention is slightly different: the question is whether one can efficiently compress every "long" instance x of a problem Q with "a short witness" to a shorter equivalent instance x' of a problem Q' (here equivalent means that $x \in Q$ if and only if $x' \in Q'$). "Such compression enables to succinctly store instances until a future setting will allow solving them, either via a technological or algorithmic breakthrough or simply until enough time has elapsed" (see [10]). By suitably generalizing the notion of a kernelization of a parameterized problem to the notion of a kernelization from some parameterized problem to another one, Fortnow and Santhanam [6] introduce a framework which allows to deal with kernelizations and compressions at the same time (in [6] a different terminology is used). Nevertheless we stick to the traditional notion of kernelization as we mainly address problems of parameterized complexity.

More precisely, the content of the different sections is the following. After recalling some definitions and fixing our notation in Section 2, we consider and analyze some basic questions concerning kernelizations in Section 3. In particular, we shall see that "most" parameterized problems have a polynomial kernelization if and only if they are self-compressible.

A kernelization is *strong* if the parameter of $\mathbb{K}(x)$ is less than or equal to the parameter of x. It is known that every parameterized problem that has a kernelization already has a strong kernelization. In Section 4 we derive a general result (Corollary 13) that shows that parameterized problems satisfying certain conditions have no strong *polynomial* kernelizations. As an application we get that the problem SAT has no strong polynomial kernelization if P $\neq$ NP and no strong subexponential kernelization if the exponential time hypothesis (ETH) holds.

In Section 5 we recall the results of Bodlaender et al. [4] and of Fortnow and Santhanam [6] relevant in our context and give some new applications. Section 6 and Section 7 are devoted to the generalizations of these results of type (1) and of type (2), respectively, already mentioned above.

2. Preliminaries

The set of natural numbers (that is, nonnegative integers) is denoted by $\mathbb{N}$. For a natural number n let $[n] := \{1, \ldots, n\}$. By $\log n$ we mean $\lceil \log n \rceil$ if an integer is expected. For $n = 0$ the term $\log n$ is undefined. We trust the reader's common sense to interpret such terms reasonably.

We identify problems (or languages) with subsets Q of $\{0,1\}^*$. Clearly, as done mostly, we present concrete problems in a verbal, hence uncodified form or as a set of strings over an arbitrary finite alphabet. We use both P and PTIME to denote the class of problems Q such that $x \in Q$ is solvable in polynomial time.

A *reduction from a problem Q to a problem Q'* is a mapping $R : \{0,1\}^* \to \{0,1\}^*$ such that for all $x \in \{0,1\}^*$ we have $(x \in Q \iff R(x) \in Q')$. We write $R : Q \leq^p Q'$ if R is a reduction from Q to Q' computable in polynomial time, and $Q \leq^p Q'$ if there is a polynomial time reduction from Q to Q'.

2.1. Parameterized Complexity. A *parameterized problem* is a pair (Q, κ) consisting of a classical problem $Q \subseteq \{0,1\}^*$ and a *parameterization* $\kappa : \{0,1\}^* \to \mathbb{N}$, which is required to be polynomial time computable even if the result is encoded in unary.

We introduce some parameterized problems, which will be used later, thereby exemplifying our way to represent parameterized problems. We denote by p-SAT the parameterized problem

p-SAT	
Instance:	A propositional formula α in conjunctive normal form.
Parameter:	Number of variables of α.
Question:	Is α satisfiable?

By p-PATH and p-CLIQUE we denote the problems:

p-PATH	
Instance:	A graph G and $k \in \mathbb{N}$.
Parameter:	k.
Question:	Does G have a path of length k?

p-CLIQUE	
Instance:	A graph G and $k \in \mathbb{N}$.
Parameter:	k.
Question:	Does G have a clique of size k?

Similarly we define p-DOMINATING-SET. If C is a class of graphs, then p-PATH(C) denotes the problem

p-PATH(C)	
Instance:	A graph G in C and $k \in \mathbb{N}$.
Parameter:	k.
Question:	Does G have a path of length k?

We use similar notations for other problems.

We recall the definitions of the classes FPT, EXPT, EPT and SUBEPT. A parameterized problem (Q, κ) is *fixed-parameter tractable* (or, in FPT) if $x \in Q$ is solvable in time $f(\kappa(x)) \cdot |x|^{O(1)}$ for some computable $f : \mathbb{N} \to \mathbb{N}$. If f can be chosen such that $f(k) = 2^{\kappa(x)^{O(1)}}$, then (Q, κ) is in EXPT. If f can be chosen such that $f(k) = 2^{O(k)}$, then (Q, κ) is in EPT. If f can be chosen such that $f(k) = 2^{o^{\text{eff}}(k)}$, then (Q, κ) is in SUBEPT.

Here o^{eff} denotes the effective version of little oh: For computable functions $f, g : \mathbb{N} \to \mathbb{N}$ we say that f *is effectively little oh of* g and write $f = o^{\text{eff}}(g)$ if there is a *computable,* nondecreasing and unbounded function $\iota : \mathbb{N} \to \mathbb{N}$ such that for sufficiently large $k \in \mathbb{N}$

$$f(k) \leq \frac{g(k)}{\iota(k)}.$$

As usual we often write $f(k) = o^{\text{eff}}(g(k))$ instead of $f = o^{\text{eff}}(g)$.

At some places in this paper, it will be convenient to consider *preparameterized problems*; these are pairs (Q, κ), where again Q is a classical problem and κ is a *preparametrization,* that is, an arbitrary function from $\{0,1\}^*$ to the set $\mathbb{R}_{\geq 0}$ of nonnegative real numbers.

3. Kernelizations

In this section we start by recalling the notion of kernelization and by introducing some refinements. We study some basic properties of kernelizations and its relationship to the notion of compression.

Definition 1. Let (Q, κ) be a parameterized problem and $f : \mathbb{N} \to \mathbb{N}$ be a function. An *f-kernelization* for (Q, κ) is a polynomial time algorithm $\mathbb{K}$ that on input $x \in \{0,1\}^*$ outputs $\mathbb{K}(x) \in \{0,1\}^*$ such that

$$(x \in Q \iff \mathbb{K}(x) \in Q) \quad \text{and} \quad |\mathbb{K}(x)| \leq f(\kappa(x)).$$

In particular, $\mathbb{K}$ is a polynomial time reduction from Q to itself. If in addition for all $x \in \{0,1\}^*$

$$\kappa(\mathbb{K}(x)) \leq \kappa(x),$$

then $\mathbb{K}$ is a *strong* f-kernelization.

We say that (Q, κ) has a *linear, polynomial, subexponential, simply exponential, and exponential kernelization* if there is an f-kernelization for (Q, κ) with $f(k) = O(k)$, $f(k) = k^{O(1)}$, $f(k) = 2^{o^{\text{eff}}(k)}$, $f(k) = 2^{O(k)}$, and $f(k) = 2^{k^{O(1)}}$, respectively.

The following result is well-known:

Proposition 2. *Let (Q, κ) be a parameterized problem with decidable Q. The following statements are equivalent.*

(1) (Q, κ) is fixed-parameter tractable.

(2) (Q, κ) has an f-kernelization for some computable f.

(3) (Q, κ) has a strong f-kernelization for some computable f.

Furthermore, if f is computable and $x \in Q$ is solvable in time $f(\kappa(x)) \cdot |x|^{O(1)}$, then (Q, κ) has a strong f-kernelization.

The recent survey [9] contains examples of natural problems whose currently best known kernelizations are polynomial, simply exponential and exponential.

We are mainly interested in polynomial kernelizations. First we show that the notions of polynomial kernelization and of strong polynomial kernelization are distinct:

Proposition 3. *There is a parameterized problem (Q, κ) that has a polynomial kernelization but no strong polynomial kernelization.*

Proof: Let Q be a classical problem that is not solvable in time $2^{O(|x|)}$. We define a parameterized problem (P, κ) with $P \subseteq \{0,1\}^* \times \{1\}^*$ and with $\kappa((x, 1^k)) = k$. By 1^k we denote the string consisting of k many 1s. For each $k \in \mathbb{N}$ we define the *k-projection* $P[k] := \{x \mid (x, 1^k) \in P\}$ *of* P by:

– If $k = 2\ell + 1$, then
$$P[k] := Q_{=\ell} \ (:= \{x \in Q \mid |x| = \ell\}).$$
Hence, all elements in $P[k]$ have length ℓ.

– If $k = 2\ell$, then
$$P[k] := \left\{x1^{2^\ell} \mid x \in Q_{=\ell}\right\},$$
where $x1^{2^\ell}$ is the concatenation of x with the string 1^{2^ℓ}. Hence, all elements in $P[k]$ have length $\ell + 2^\ell$.

Intuitively, an element in the 2ℓ-projection is an element in the $(2\ell + 1)$-projection padded with 2^ℓ many 1s. It is not hard to see that P has a linear kernelization (which on the even projections increases the parameter).

We claim that P has no strong polynomial kernelization. Assume $\mathbb{K}$ is such a kernelization and $c \in \mathbb{N}$ such that
$$|\mathbb{K}((z, 1^m))| \leq m^c.$$
We use $\mathbb{K}$ to solve $x \in Q$ in time $2^{O(|x|)}$:

Let x be an instance of Q and let $\ell := |x|$. We may assume that
$$(2\ell)^c < 2^\ell$$
(note that there are only finitely many x not satisfying this inequality). We compute (in time $2^{O(\ell)}$)
$$(u, k) := \mathbb{K}\big((x1^{2^\ell}, 2\ell)\big).$$
We know that $k \leq 2\ell$ and $|u| \leq (2\ell)^c < 2^\ell$. If u does not have the length of the strings in $P[k]$, then $(u, k) \notin P$ and therefore $x \notin Q$. In particular, this is the case if $k = 2\ell$ (as $|u| < 2^\ell$). If u has the length of the strings in $P[k]$ and hence $k < 2\ell$, then it is easy to read off from u an instance y with $|y| < |x|$ and $(y \in Q \iff x \in Q)$. We then apply the same procedure to y. □

Remark 4. Let $c \in \mathbb{N}$. It is not hard to generalize the previous example and to show that there is a parameterized problem with a polynomial kernelization but with no polynomial kernelization $\mathbb{K}$ satisfying for all $x \in \{0,1\}^*$
$$\kappa(\mathbb{K}(x)) \leq \kappa(x)^c.$$

The next result shows that a parameterized problem (Q, κ) in FPT$\setminus$EXPT with $Q \in$ NP cannot have polynomial kernelizations. We show a little bit more. Recall that EXP is the class of classical problems Q such that $x \in Q$ is solvable in deterministic time $2^{|x|^{O(1)}}$.

Proposition 5. *Assume that the problem (Q, κ) has a polynomial kernelization and that $Q \in$ EXP. Then $(Q, \kappa) \in$ EXPT.*

Proof: Let $\mathbb{K}$ be a polynomial kernelization of (Q, κ). As $Q \in$ EXP there is an algorithm $\mathbb{A}$ solving $x \in Q$ in time $2^{|x|^{O(1)}}$. The algorithm that on $x \in \{0,1\}^*$ first computes $\mathbb{K}(x)$ and then applies $\mathbb{A}$ to $\mathbb{K}(x)$ solves $x \in Q$ in time $|x|^{O(1)} + 2^{|\mathbb{K}(x)|^{O(1)}} = 2^{|\kappa(x)|^{O(1)}} \cdot |x|^{O(1)}$. □

The model-checking of monadic second-order logic on the class of trees is in EXP. By a result of [8] the corresponding parameterized problem with the length of the formula as parameter is in FPT$\setminus$EXPT unless P = NP. Hence, by the preceding proposition, it has no polynomial kernelization (unless P = NP).

In later sections, under some complexity-theoretic assumptions, we will present various examples of natural problems that are in EPT and have no polynomial kernelization. Here we give a simple, artificial example without polynomial kernelizations which holds unconditionally. Bodlaender et al. [4] claim the existence of a problem in EPT without subexponential kernelizations.

Example 6. Let Q be a classical problem not in PTIME but solvable in time $O(|x|^{\log |x|})$. Let κ be the parameterization mapping x to $(\log |x|)^2$. Then $(Q, \kappa) \in$ EPT, because $2^{\kappa(x)} = |x|^{\log |x|}$.

For the sake of contradiction assume that (Q, κ) has a polynomial kernelization $\mathbb{K}$. Then to decide if $x \in Q$ it suffices to decide if $\mathbb{K}(x) \in Q$. Since $|\mathbb{K}(x)| = (\log |x|)^{O(1)}$ this can be done in time

$$|\mathbb{K}(x)|^{\log |\mathbb{K}(x)|} \leq (\log |x|)^{O(\log \log |x|)} \leq 2^{(\log \log |x|)^{O(1)}} \leq |x|^{O(1)}.$$

Thus $Q \in$ PTIME, a contradiction.

However, if we would allow kernelizations to have slightly superpolynomial running time, then *every* EPT problem would have subexponential kernelizations:

Proposition 7. *Let $(Q, \kappa) \in$ EPT and $\iota : \mathbb{N} \to \mathbb{N}$ be a nondecreasing unbounded and computable function.*[1] *Then there is algorithm $\mathbb{K}$ that for every instance x of Q outputs an instance $\mathbb{K}(x)$ in time*

$$|x|^{O(\iota(\kappa(x)))}$$

such that

$$(x \in Q \iff \mathbb{K}(x) \in Q) \qquad \text{and} \qquad |\mathbb{K}(x)| = 2^{o(\kappa(x))}.$$

To obtain this proposition we just refine the "standard" proof of the implication $(1) \Rightarrow (2)$ of Proposition 2 and show that every problem in EPT has arbitrarily small exponential kernelizations, that is, for every $\varepsilon \in \mathbb{R}$ there is a polynomial kernelization with kernels of size $\leq (1+\varepsilon)^{\kappa(x)}$, even more:

Lemma 8. *Let (Q, κ) be a parameterized problem in* EPT. *There is an algorithm $\mathbb{I}$ that takes as inputs an instance x of Q and $\ell \in \mathbb{N}$ and outputs an instance $\mathbb{I}(x, \ell)$ of Q in time $|x|^{O(\ell)}$ such that*

$$(x \in Q \iff \mathbb{I}(x, \ell) \in Q) \qquad \text{and} \qquad |\mathbb{I}(x, \ell)| = 2^{O(\kappa(x))/\ell}.$$

Proof: We choose $c \in \mathbb{N}$ and an algorithm $\mathbb{A}$ solving $x \in Q$ is in time $2^{c \cdot \kappa(x)} \cdot |x|^{O(1)}$. Furthermore we fix $x_+ \in Q$ and $x_- \notin Q$. Then the following is the desired algorithm.

$\mathbb{I}(x, \ell)$ // x an instance of Q and $\ell \in \mathbb{N}$.

1. **if** $|x| \leq 2^{\kappa(x)/\ell}$ **then** output x.
2. **else** simulate $\mathbb{A}$ on x
   // the running time is bounded by $2^{c \cdot \kappa(x)} \cdot |x|^{O(1)} \leq |x|^{c \cdot \ell + O(1)}$.
3. **if** $\mathbb{A}$ accepts x **then** output x_+ **else** output x_-.

[1] To get a "slightly superpolynomial running time" we choose as ι an "extremely slowly" growing function.

□

Proof of Proposition 7: We choose a polynomial time computable $\nu : \mathbb{N} \to \mathbb{N}$ with $\nu \leq \iota$ and set $\mathbb{K}(x) := \mathbb{I}(x, \nu(\kappa(x)))$, where $\mathbb{I}$ is the algorithm of the preceding lemma. □

Next we show that the different degrees of kernelizability introduced in Definition 1 are indeed different.

Proposition 9. *The classes of parameterized problems with a linear, a polynomial, a subexponential, a simply exponential, and an exponential kernelization are pairwise different.*

The claim immediately follows from the following lemma.

Lemma 10. *Let $g : \mathbb{N} \to \mathbb{N}$ be nondecreasing and unbounded and $f : \mathbb{N} \to \mathbb{N}$ such that $f(k) \leq g(k-1)$ for all sufficiently large k. Then there is a $Q \subseteq \{0,1\}^*$ and a preparameterization κ such that (Q, κ) has a g-kernelization but no f-kernelization.*

If in addition g is increasing and time-constructible, then we can choose κ to be a parameterization.

Proof: Let g and f be as in the statement. We choose k_0 such that $f(k) \leq g(k-1)$ for all $k \geq k_0$. We consider the "inverse function" ι_g of g given by

$$\iota_g(m) := \min\{s \in \mathbb{N} \mid g(s) \geq m\}.$$

Then for all $n \in \mathbb{N}$

$$n \leq g(\iota_g(n)) \qquad \text{and} \qquad \text{if } \iota_g(n) \geq 1, \text{then } \; g(\iota_g(n)-1) < n. \tag{3}$$

Let Q be a problem not in PTIME and define the parameterization κ by $\kappa(x) := \iota_g(|x|)$. By the first inequality in (3) the identity is a g-kernelization of (Q, κ).

Assume that there is an f-kernelization $\mathbb{K}$ of (Q, κ). As ι_g is unbounded, we have $\iota_g(|x|) \geq k_0$ for sufficiently long $x \in \{0,1\}^*$. Then

$$|\mathbb{K}(x)| \leq f(\kappa(x)) = f(\iota_g(|x|)) \leq g(\iota_g(|x|)-1) < |x|.$$

Thus applying $\mathbb{K}$ at most $|x|$ times we get an equivalent instance of length at most $f(k_0)$. Therefore, $Q \in$ PTIME, a contradiction.

If g is increasing and time-constructible, then ι_g is polynomial time computable and hence κ is a parameterization. □

Polynomial Kernelization and Compression. Most natural problems $Q \in$ NP have a *canonical* representation of the form

$$x \in Q \quad \Longleftrightarrow \quad \text{there is } y \in \{0,1\}^{g(x)} \text{ such that } (x, y) \in Q_0 \tag{4}$$

for some polynomial time computable function $g : \{0,1\}^* \to \mathbb{N}$ and some $Q_0 \in$ PTIME. In [3] the problem (Q, g) has been called the *canonical parameterization* of Q (more precisely, one should speak of the canonical parameterization induced by the representation (4) of Q). Clearly (Q, g) is fixed-parameter tractable, it is even in EPT. If (Q, κ) was a parameterized problem, then (Q, g) is called the *canonical reparameterization* of (Q, κ).

The canonical reparameterization of p-SAT is p-SAT itself; the canonical reparameterizations of the problems p-PATH, p-CLIQUE and p-DOMINATING-SET are the problems *uni*-PATH, *uni*-CLIQUE and *uni*-DOMINATING-SET, respectively, where in the three cases, we have $g((G,k)) = k \cdot \log |V|$; hence in particular,

uni-PATH	
Instance:	A graph $G = (V, E)$ and $k \in \mathbb{N}$.
Parameter:	$k \cdot \log \lvert V \rvert$.
Question:	Does G have a path of length k?

Many fixed-parameter tractable problems, namely all in EXPT and hence, in particular, p-PATH, have a polynomial kernelization if and only if their canonical reparameterizations have. This is shown by the following proposition.

Proposition 11. *Let $(Q,\kappa) \in$ EXPT and let (Q,g) be the canonical reparameterization of (Q,κ). Assume that g has the form*

$$g(x) = \kappa(x) \cdot \log h(x) \text{ with } h(x) = |x|^{O(1)}$$

and $h(x) \geq 2$ for sufficiently large x. Then

$$(Q,\kappa) \text{ has a polynomial kernelization iff } (Q,g) \text{ has a polynomial kernelization.}$$

Proof: Clearly, every polynomial kernelization of (Q,κ) is a polynomial kernelization of (Q,g). Conversely, let $\mathbb{K}$ be a polynomial kernelization of (Q,g). Choose $c, c' \in \mathbb{N}$ and an algorithm $\mathbb{A}$ solving $x \in Q$ in time $2^{\kappa(x)^c}|x|^{c'}$. We define a polynomial kernelization $\mathbb{K}'$ for (Q,κ).

Fix $x_+ \in Q$ and $x_- \notin Q$. (If Q is trivial, that is, $Q = \emptyset$ or $Q = \{0,1\}^*$, we let $\mathbb{K}'$ always output the empty string.) Let $x \in \{0,1\}^*$. If $\kappa(x) < (\log |x|)^{1/c}$, the algorithm $\mathbb{A}$ on input x needs at most $|x|^{c'+1}$ steps. In this case we let $\mathbb{K}'(x)$ be x_+ or x_- according to the answer of $\mathbb{A}$. Otherwise $\kappa(x)^c \geq \log |x|$. Then $|\mathbb{K}(x)| = (\kappa(x) \cdot \log h(x))^{O(1)} = (\kappa(x) \cdot \log |x|)^{O(1)} = \kappa(x)^{O(1)}$, so we can set $\mathbb{K}'(x) := \mathbb{K}(x)$. □

The reader familiar with [10] will realize that this result shows that any parameterized problem (Q,κ) in EXPT has a polynomial kernelization if and only if the problem Q is self-compressible.

4. Excluding strong kernelizations

In this section we exemplify how self-reducibility can be used to rule out *strong* polynomial kernelizations. This method is very simple and works under the assumption that P $\neq$ NP. We use it to give two natural examples of problems in EPT that do not have *strong* polynomial kernelizations.

We will revisit these examples in section 5. There we will see that these problems do not even have polynomial kernelizations using the stronger assumption that the polynomial hierarchy does not collapse to its third level.

Lemma 12. *Let (Q,κ) be a parameterized problem and assume that the 0th slice $Q(0) := \{x \in Q \mid \kappa(x) = 0\}$ is in* PTIME. *If there is a polynomial (subexponential) kernelization $\mathbb{K}$ such that for all $x \notin Q(0)$*

$$\kappa(\mathbb{K}(x)) < \kappa(x), \tag{5}$$

*then $Q \in$ PTIME $\quad ((Q,\kappa) \in$ SUBEPT).

Proof: Let $\mathbb{K}$ be a kernelization satisfying (5). The following algorithm $\mathbb{A}$ decides Q (using a polynomial time decision procedure $\mathbb{B}$ for $Q(0)$). Given an instance x of Q, the algorithm $\mathbb{A}$ computes $\mathbb{K}(x), \mathbb{K}(\mathbb{K}(x)), \ldots$; by (5) after at most $\kappa(x)$ steps we obtain an instance y with $\kappa(y) = 0$; hence $(x \in Q \iff y \in Q(0))$; now $\mathbb{A}$ simulates $\mathbb{B}$ on y.

If $\mathbb{K}$ was a polynomial kernelization, say, $|\mathbb{K}(x)| \leq \kappa(x)^c$, then, again by (5), all of $|\mathbb{K}(\mathbb{K}(x))|, |\mathbb{K}(\mathbb{K}(\mathbb{K}(x)))|, \ldots$ are bounded by $\kappa(x)^c$. Recall that parameterizations are computable in polynomial time even if the result is encoded in unary. Hence $\kappa(x) = |x|^{O(1)}$. It follows that $\mathbb{A}$ runs in polynomial time.

If $\mathbb{K}$ was a subexponential kernelization, say, $|\mathbb{K}(x)| \leq 2^{\kappa(x)/\iota(\kappa(x))}$ with computable, nondecreasing and unbounded ι and $\mathbb{K}(x)$ is computable in time $|x|^d$, then to compute the equivalent instance y algorithm $\mathbb{A}$ needs at most

$$|x|^d + 2^{d\cdot\kappa(x)/\iota(\kappa(x))} + 2^{d\cdot(\kappa(x)-1)/\iota(\kappa(x)-1)} + 2^{d\cdot(\kappa(x)-2)/\iota(\kappa(x)-2)} + \ldots + 2^{d\cdot 1/\iota(1)},$$

many steps. As we can assume that the function $j \mapsto j/\iota(j)$ is increasing, this number of steps is bounded by $|x|^d + \kappa(x) \cdot 2^{d\cdot\kappa(x)/\iota(\kappa(x))}$, which shows that $(Q,\kappa) \in$ SUBEPT. □

Corollary 13. *Let (Q,κ) be a parameterized problem with $Q(0) \in$ PTIME. Assume that there is a polynomial reduction R from Q to itself which is* parameter decreasing, *that is, for all $x \notin Q(0)$,*

$$\kappa(R(x)) < \kappa(x).$$

- *If (Q,κ) has a strong polynomial kernelization, then $Q \in$ PTIME.*
- *If (Q,κ) has a strong subexponential kernelization, then $(Q,\kappa) \in$ SUBEPT.*

Proof: Let R be as in the statement and let $\mathbb{K}$ be a strong polynomial (subexponential) kernelization of (Q, κ). Then the composition $\mathbb{K} \circ R$, that is, the mapping $x \mapsto \mathbb{K}(R(x))$, is a polynomial (subexponential) kernelization of (Q, κ) satisfying (5); hence, by the previous lemma, we get $Q \in$ PTIME ($Q \in$ SUBEPT). □

Examples 14. The classical problems underlying

$$p\text{-SAT} \text{ and } p\text{-POINTED-PATH}$$

have parameter-decreasing polynomial reductions to themselves, where

p-POINTED-PATH	
Instance:	A graph $G = (V, E)$, a vertex $v \in V$, and $k \in \mathbb{N}$.
Parameter:	k.
Question:	Does G have a path of length k starting at v?

Proof: p-SAT: We define a parameter-decreasing polynomial reduction R from p-SAT to itself as follows: Let α be a CNF formula. If α has no variables, we set $R(\alpha) := \alpha$. Otherwise let X be the first variable in α. We let $R(\alpha)$ be a formula in CNF equivalent to

$$(\alpha\frac{\text{TRUE}}{X} \vee \alpha\frac{\text{FALSE}}{X}),$$

where, for example, $\alpha\frac{\text{TRUE}}{X}$ is the formula obtained from α by replacing X by TRUE everywhere. Clearly $R(\alpha)$ can be computed from α in polynomial time.

p-POINTED-PATH: We define a parameter-decreasing polynomial reduction R from p-POINTED-PATH to itself as follows: Let (G, v, k) be an instance of p-POINTED-PATH and assume $k \geq 3$. For any path $P : v, v_1(P), v_2(P)$ of length 2 starting from v let G_P be the graph obtained from G by deleting the two vertices $v, v_1(P)$ (and all the edges incident with one of these vertices). Let H be the graph obtained from the disjoint union of all the graphs G_P (where P ranges over all paths of length 2 starting in v) by adding a new vertex w and all edges $\{w, v_2(P)\}$. Then H has a path of length $(k-1)$ starting at w if and only if G has a path of length k starting at v. Hence we can set $R((G, v, k)) := (H, w, k-1)$. □

Corollary 15. (1) If P $\neq$ NP, then p-SAT has no strong polynomial kernelization.

(2) If ETH holds, then p-SAT has no strong subexponential kernelization.

(3) If P $\neq$ NP, then p-POINTED-PATH has no strong polynomial kernelization.

(4) If ETH holds, then p-POINTED-PATH has no strong subexponential kernelization.

Proof: Part (1) and (3) are immediate by Corollary 13. Moreover, we know by this corollary that if one of the two problems has a strong subexponential kernelization, then it is in SUBEPT. However then ETH would fail in the case of p-SAT by [11] and in the case of p-POINTED-PATH by [2]. □

5. Excluding polynomial kernelizations

The following type of reductions that preserve polynomial kernels was introduced in [6] (based on a notion of [10]) under the name "W-reductions."

Definition 16. Let (Q, κ) and (Q', κ') be parameterized problems. A *polynomial reduction* from (Q, κ) to (Q', κ') is a polynomial reduction R from Q to Q' such that

$$\kappa'(R(x)) = \kappa(x)^{O(1)}.$$

We then write $R : (Q, \kappa) \leq^p (Q', \kappa')$. Furthermore $(Q, \kappa) \leq^p (Q', \kappa')$ means that there is a polynomial reduction from (Q, κ) to (Q', κ').

Example 17. *uni*-PATH $\leq^p$ p-SAT.

Proof: Let (G, k) with $G = (V, E)$ be an instance of *uni*-PATH. We may assume that $V = [0, n-1]$ and (by adding isolated points if necessary) that n is a power of 2. We will assign to (G, k) a formula α in CNF containing variables $X_{s,i}$ with $s \in [\log\ n]$ and $i \in [k]$ with the intended meaning "the sth bit of the ith vertex of a path of

length k is 1." For $i, j \in [k]$, $i \neq j$ one has to express by a clause that the selected vertices as ith and jth point of the path are distinct and for $i \in [k-1]$ that the ith and the $(i+1)$th selected vertices are related by an edge. For example the second one may be expressed by letting be for every $i \in [k-1]$ and every $u, v \in V$ with $\{u, v\} \notin E$

$$\bigvee_{s \in [\log\, n]} \neg X_{s,i}^{\mathrm{bit}(s,u)} \vee \bigvee_{s \in [\log\, n]} \neg X_{s,i+1}^{\mathrm{bit}(s,v)},$$

a clause of α, where $\mathrm{bit}(s, u)$ denotes the sth bit in the binary representation of u of length $\log\ n$ and where $X^1 := X$ and $X^0 := \neg X$ for every variable X.

Then G has a path of length k if and only if α is satisfiable. As α has $k \cdot \log\ |V|$ variables, the mapping $(G, k) \mapsto \alpha$ is a polynomial reduction. □

Example 18 ([10]). p-SAT $\leq^p$ *uni*-DOMINATING-SET.

Polynomial reductions preserve polynomial kernelizations in the following sense:

Lemma 19. *Let (Q, κ) and (Q', κ') be parameterized problems. with*

$$(Q, \kappa) \leq^p (Q', \kappa') \quad \textit{and} \quad Q' \leq^p Q.$$

If (Q', κ') has a polynomial kernelization, then (Q, κ) has a polynomial kernelization.

Note that $Q' \leq^p Q$ is always satisfied for NP-complete problems Q and Q'.

Proof of Lemma 19: Let $R : (Q, \kappa) \leq^p (Q', \kappa')$ and $S : Q' \leq^p Q$. Assume that $\mathbb{K}$ is a polynomial kernelization for (Q', κ'). Then $S \circ \mathbb{K} \circ R$ is a polynomial kernelization for (Q, κ), as for all $x \in \{0,1\}^*$

$$|S(\mathbb{K}(R(x)))| = |\mathbb{K}(R(x))|^{O(1)} = \kappa'(R(x))^{O(1)} = \kappa(x)^{O(1)}.$$

□

In order to exclude polynomial kernelizations using the previous lemma one needs a primal problem without a polynomial kernelization. A central ingredient needed to obtain such problems was provided by Fortnow and Santhanam [6]. It is contained in the following theorem.

Definition 20 ([4]). Let $Q, Q' \subseteq \{0,1\}^*$ be classical problems. A *distillation from Q in Q'* is a polynomial time algorithm $\mathbb{D}$ that receives as inputs finite sequences $\bar{x} = (x_1, \ldots, x_t)$ with $x_i \in \{0,1\}^*$ for $i \in [t]$ and outputs a string $\mathbb{D}(\bar{x}) \in \{0,1\}^*$ such that

(1) $|\mathbb{D}(\bar{x})| = \left(\max_{i \in [t]} |x_i|\right)^{O(1)}$;

(2) $\mathbb{D}(\bar{x}) \in Q'$ if and only if for some $i \in [t]$: $x_i \in Q$.

If $Q' = Q$ we speak of a *self-distillation*. We say that Q *has a distillation* if there is a distillation from Q in Q' for some Q'.

"Self-distillations" without property (1) has been called OR_ω functions in [1]. Their importance for classical complexity has been studied in various papers (see [1] and its references). Every NP-complete problem Q has an OR_ω function: Take a polynomial time reduction of the problem $\{(x_1, \ldots, x_t) \mid t \in \mathbb{N} \text{ and } x_i \in Q \text{ for some } i \in [t]\}$ to Q. However:

Theorem 21 ([6]). *No* NP*-hard problem has a distillation unless* PH $= \Sigma_3^{\mathrm{P}}$ *(that is, unless the polynomial hierarchy* PH *collapses to its third level Σ_3^{P}).*

To see how this result (and the polynomial reductions) can be used to exclude polynomial kernelizations we include applications from [4] and [6].

Corollary 22 ([4]). p-PATH *has no polynomial kernelization unless* PH $= \Sigma_3^{\mathrm{P}}$.

Proof: We assume that p-PATH has a polynomial kernelization $\mathbb{K}$ and show that then the (classical) problem PATH has a self-distillation to itself. In fact, let $(G_1, k_1), \ldots, (G_t, k_t)$ be instances of PATH. Let $k := 1 + 2 \cdot \max_{i \in [t]} k_i$. Let $i \in [t]$. By adding to G_i a path of length $k - k_i - 1$ with one endpoint connected to all vertices of G_i we obtain a graph G_i' such that the instance (G_i', k) of p-Path is equivalent to (G_i, k_i). Let G be the disjoint union of all the graphs G_i'. Clearly, G has a path of length k if and only if there exists an $i \in [t]$ such that G_i' has a path of length k and hence, if and only if there exists an $i \in [t]$ such that G_i has a path of length k_i As $|\mathbb{K}((G, k))|$ is polynomially bounded in k and hence in $\max_{i \in [t]} \|(G_i, k_i)\|$, the mapping $(G_1, k_1), \ldots, (G_t, k_t) \mapsto \mathbb{K}((G, k))$ is a self-distillation of PATH. □

Corollary 23 ([6]). *The problems*

$$p\text{-SAT} \text{ and } \textit{uni}\text{-DOMINATING-SET}$$

have no polynomial kernelization unless PH $= \Sigma_3^{\mathrm{p}}$.

Proof: Assume PH $\neq \Sigma_3^{\mathrm{p}}$. By the previous corollary we know that p-PATH has no polynomial kernelization. Hence, as p-PATH $\in$ EPT, its canonical reparametrization *uni*-PATH has no polynomial kernelization by Proposition 11. The claims follow from Examples 17 and 18 by Lemma 19. □

The proof of Corollary 22 consists of two parts. Let $(G_1, k_1), \ldots, (G_t, k_t)$ and (G, k) be as there. In the first part we show that $\mathbb{O}$ with $\mathbb{O}((G_1, k_1), \ldots, (G_t, k_t)) := (G, k)$ is an OR for p-PATH in the sense of the following definition.

Definition 24. Let (Q, κ) be a parameterized problem. An *OR for* (Q, κ) is a polynomial time algorithm $\mathbb{O}$ that for every finite tuple $\bar{x} = (x_1, \ldots, x_t)$ of instances of Q outputs an instance $\mathbb{O}(\bar{x})$ of Q such that

(1) $\kappa(\mathbb{O}(\bar{x})) = (\max_{i\in[t]}|x_i|)^{O(1)}$;

(2) $\mathbb{O}(\bar{x}) \in Q$ if and only if for some $i \in [t]$: $x_i \in Q$.

The second part of the proof of Corollary 22 shows the following lemma (there the argument is presented for $(Q, \kappa) := p$-PATH).

Lemma 25. *Assume that* (Q, κ) *has an* OR $\mathbb{O}$ *and a polynomial kernelization* $\mathbb{K}$*. Then* $\mathbb{D}$ *with*

$$\mathbb{D}(x_1, \ldots, x_t) := \mathbb{K}(\mathbb{O}((x_1, \ldots, x_t))$$

is a self-distillation of Q.

Hence by Theorem 21:

Corollary 26. *Assume that* (Q, κ) *has an* OR $\mathbb{O}$ *and that* Q *is* NP*-hard. Then* (Q, κ) *has no polynomial kernelization unless* PH $= \Sigma_3^{\mathrm{p}}$.

Perhaps the reader might object that the proof of Corollary 22 is algorithmically not convincing, as the OR function used in the first part essentially yields the disjoint union of given graphs, while probably any reasonable algorithm for determining whether a graph has a path of a given length will first compute its connected components and then check these components for such a path. Hence the question arises whether the path problem for the class of *connected* graphs has a polynomial kernelization. We deny this, we even show that the path problem for the class PLAN-CONN of planar connected graphs has no polynomial kernelization:

Proposition 27. p-PATH(PLAN-CONN) *has no polynomial kernelization unless* PH $= \Sigma_3^{\mathrm{p}}$.

To show this claim we show in a first step:

Lemma 28. p-POINTED-PATH(PLAN-CONN) *has no polynomial kernelization unless* PH $= \Sigma_3^{\mathrm{p}}$.

Proof: We show p-POINTED-PATH(PLAN-CONN) has an OR (then our claim follows from Corollary 26). Let $(G_1, v_1, k_1) \ldots, (G_t, v_t, k_t)$ be instances of the problem. First let us assume that for every $i \in [t]$, we take a drawing of G_i[2] such that v_i lies on the boundary of its outer face. Let $k := \max_{i\in[t]} k_i$. By adding to every G_i a path of length $k - k_i$ starting in v_i and ending in a vertex v_i' we obtain an equivalent instance (G_i', v_i', k). Let G be the planar and connected graph obtained from the disjoint union of the G_i's by adding a new vertex v and edges from v to all v_i'. It is easy to verify that

$$\begin{aligned} &G \text{ has a path of length } k+1 \text{ starting at } v \\ &\qquad\qquad \iff \text{there exists an } i \in [t] \text{ such that } G_i \text{ has a path of length } k \text{ starting at } v_i. \end{aligned}$$

Hence we can set $\mathbb{O}((G_1, v_1, k_1) \ldots, (G_t, v_t, k_t)) := (G, v, k+1)$. □

Proof of Proposition 27: We show that there is a polynomial reduction from p-POINTED-PATH(PLAN-CONN) to p-PATH(PLAN-CONN). Then our claim follows from the previous lemma by Lemma 19.

[2]Note that we actually do not need to compute the drawing of G_i. It is only needed to show that the graph G we construct is planar.

Let (G, v, k) be an instance of p-POINTED-PATH(PLAN-CONN). Using the connectedness of G one easily verifies:

$$\text{if } G \text{ contains a path of length } 2k-1, \text{ then } G \text{ contains a path of length } k \text{ starting at } v. \tag{6}$$

We add to G in v a path P of length $k-1$ of new vertices, thereby obtaining the planar and connected graph G'. We show that

$$(G, v, k) \in p\text{-POINTED-PATH(PLAN-CONN)} \iff (G', 2k-1) \in p\text{-PATH(PLAN-CONN)}.$$

Then $(G, v, k) \mapsto (G', 2k-1)$ is the desired reduction.

Assume first that G has a path of length k starting at v. Clearly, then G' has a path of length $2k-1$. Conversely, let P' be a path of length $2k-1$ in G'. If v is a vertex of P', then the vertices of P' contained in G constitute a path of G of length at least k starting at v. If v is not a vertex of P', then P' is a path in G and by (6) the graph G contains a path of length k starting at v. □

We know that no NP-hard problem has a self-distillation (unless PH $= \Sigma_3^{\mathrm{P}}$). Clearly each problem in PTIME has a self-distillation.

Proposition 29. *If* NE $\neq$ E*, then there is a problem in* NP $\setminus$ P *that has a self-distillation.*

By E and NE we denote the class of problems Q such that $x \in Q$ is solvable by a deterministic algorithm and a nondeterministic algorithm, respectively, in time $2^{O(|x|)}$.

Proof of Proposition 29: Let $Q_0 \subseteq \{0,1\}^*$ be a language in NE $\setminus$ E. We assume that each yes instance of Q_0 starts with a 1, and can thus be viewed as a natural number in binary. For $n \in \mathbb{N}$ let $\mathit{bin}(n)$ denote its binary representation. We set

$$Q := \{1^n \mid \mathit{bin}(n) \in Q_0\}.$$

It is easy to see that $Q \in$ NP $\setminus$ P. Now let Q' be the "OR-closure" of Q, that is

$$Q' := \{(x_1, \ldots, x_m) \mid m \geq 1 \text{ and } x_i \in Q \text{ for some } i \in [m]\}.$$

Again it is easy to see that $Q' \in$ NP $\setminus$ P. We claim that Q' has a self-distillation.

Let $(x_{11}, \ldots, x_{1m_1}), \ldots, (x_{t1}, \ldots, x_{tm_t})$ be a sequence of instances of Q'. We can assume that all x_{ij} are sequences of 1s (otherwise we simply ignore those which are not). Let n be the maximal length of the x_{ij}. Then

$$\{x_{11}, \ldots, x_{1m_1}, \ldots, x_{t1}, \ldots, x_{tm_t}\} = \{y_1, \ldots, y_q\}$$

for some $q \leq n$. Thus $(y_1, \ldots, y_q)$ has length $O(n^2)$. Clearly $(y_1, \ldots, y_q)$ is in Q' if and only if $(x_{i1}, \ldots, x_{im_i}) \in Q'$ for some $i \in [t]$. □

6. Strong lower bounds

In this section and the next one, by a careful analysis of the proof of Theorem 21, we obtain improvements, which yield better lower bounds for kernelizations. In particular for the path problem we will show:

Theorem 30. *Let $\varepsilon > 0$ and assume* PH $\neq \Sigma_3^{\mathrm{P}}$*. Then there is* no *polynomial reduction from* PATH *to itself computing for each instance (G, k) of* PATH *an instance (G', k') with*

$$\|G'\| = k^{O(1)} \cdot \|G\|^{1-\varepsilon}.$$

We define:

Definition 31. Let $\varepsilon > 0$. A parameterized problem (Q, κ) *has an ε self-reduction* if there is a polynomial reduction from Q to itself that assigns to every instance x of Q an instance y with

$$|y| = \kappa(x)^{O(1)} \cdot |x|^{1-\varepsilon}.$$

Note that it is not required that the parameter of y is bounded in terms of the parameter of x.

Clearly, if (Q, κ) has a polynomial kernelization, then (Q, κ) has an ε self-reduction for every $\varepsilon > 0$. Now we can rephrase Theorem 30 by saying that, unless PH $= \Sigma_3^{\mathrm{p}}$, for every $\varepsilon > 0$ the problem p-PATH has no ε self-reduction. This result will be a special instance of a more general result stating similar lower bounds for problems with a linear OR.

Definition 32. Let (Q, κ) be a parameterized problem. A *linear OR for* (Q, κ) is a polynomial time algorithm $\mathbb{O}$ that for every finite tuple $\bar{x} = (x_1, \ldots, x_t)$ of instances of Q outputs an instance $\mathbb{O}(\bar{x})$ of Q such that

(1) $|\mathbb{O}(\bar{x})| = t \cdot \big(\max_{i\in[t]}|x_i|\big)^{O(1)}$;

(2) $\kappa(\mathbb{O}(\bar{x})) = \big(\max_{i\in[t]}|x_i|\big)^{O(1)}$;

(3) $\mathbb{O}(\bar{x}) \in Q$ if and only if for some $i \in [t]$: $x_i \in Q$.

Hence a linear OR is an OR with the additional property (1).

Examples 33. (a) The parameterized problems p-PATH and p-POINTED-PATH(PLAN-CONN) have a linear OR. In fact, the ORs defined in the proofs of Corollary 22 and of Lemma 28 are linear ones.

(b) The parameterized problem p-SAT has a linear OR.

Proof: We define a linear OR $\mathbb{O}$. Let $\alpha_1, \ldots, \alpha_t$ be CNF formulas, say, α_i a formula with n_i variables. We set

$$n := \max_{i\in[t]} n_i \quad \text{and} \quad m := \max_{i\in[t]} |\alpha_i|.$$

We may assume that all α_i have variables in $\{X_1, \ldots, X_n\}$ and that $\log t$ is a natural number (if t is not a power of two we duplicate one of the formulas for an appropriate number of times).

If $t \geq 2^n$, the algorithm $\mathbb{O}$ proves whether one of the α_is is satisfiable (by systematically checking all assignments) and outputs a CNF formula $\mathbb{O}(\alpha_1, \ldots, \alpha_t)$ satisfying condition (3) of the preceding definition.

Assume $t < 2^n$. We introduce $\log t$ new variables $Y_1, \ldots, Y_{\log t}$. For $i \in [t]$ we set

$$\beta_i := \bigwedge_{s\in[\log t]} Y_s^{\mathrm{bit}(s,i)}$$

(recall that $\mathrm{bit}(s, i)$ denotes the sth bit in the binary representation of i and that $X^1 = X$ and $X^0 = \neg X$ for every variable X).

We bring each $(\beta_i \to \alpha_i)$ into conjunctive normal form: Assume $\alpha_i = \bigwedge_\ell \bigvee_{\ell'} \lambda_{\ell\ell'}$ with literals $\lambda_{\ell\ell'}$, then $(\beta_i \to \alpha_i)$ is equivalent to

$$\gamma_i := \bigwedge_\ell \Big(\bigvee_{s\in[\log t]} Y_s^{1-\mathrm{bit}(s,i)} \vee \bigvee_{\ell'} \lambda_{\ell\ell'} \Big).$$

We let γ be the CNF formula $\gamma := \bigwedge_{i\in[t]} \gamma_i$. We set $\mathbb{O}(\alpha_1, \ldots, \alpha_t) := \gamma$.

Clearly $\mathbb{O}$ is computable in polynomial time. Furthermore, by construction the formula $\mathbb{O}(\alpha_1, \ldots, \alpha_t)$ is equivalent to $\bigwedge_{i\in[t]}(\beta_i \to \alpha_i)$. Because any assignment to $Y_1, \ldots, Y_{\log t}$ satisfies exactly one of the β_is, the formula $\mathbb{O}(\alpha_1, \ldots, \alpha_t)$ is satisfiable if and only if there is an $i \in [t]$ such that α_i is satisfiable; hence condition (3) of Definition 32 is satisfied . Furthermore, $\mathbb{O}$ also satisfies the conditions (1) and (2). For (2) note that γ has $n + \log t$ variables. By our assumption on t, we have $n+\log t \leq 2n \leq 2m$. For (1) note that each γ_i has length $O(m\cdot(m+\log t))$ and hence, $\mathbb{O}(\alpha_1, \ldots, \alpha_t)$ has length $O(m^3)$. □

(c) The parameterized problem *uni*-CLIQUE has a linear OR.

Proof: Let $(G_1, k_1), \ldots, (G_t, k_t)$ be instances of *uni*-CLIQUE. Of course, we can assume that $k_i \leq |V_i|$, where V_i is the set of vertices of G_i. Let $k := \max_{i\in[t]} k_i$. By adding a clique of $k-k_i$ new vertices to G_i and connecting all new vertices to all old vertices in V_i we can pass to an instance (G_i', k) equivalent to (G_i, k_i). Let $m := \max_{i\in[t]}|V_i'|$ $(\leq 2 \cdot \max_{i\in[t]}|V_i|)$.

If $t \geq 2^m$, by exhaustive search the algorithm $\mathbb{O}$ checks whether one of the G_i's has a clique of size k; if this is the case $\mathbb{O}$ outputs (G_i, k_i) for such a G_i' and otherwise it outputs, say, (G_1, k_1).

Assume that $t < 2^m$. We set $\mathbb{O}((G_1, k_1), \ldots, (G_t, k_t)) := (G, k)$, where G denotes the disjoint union of the graphs G_i'. Clearly, $\mathbb{O}$ is computable in polynomial time and condition (3) is satisfied. For condition (1) note that we have for the set V of vertices of G the inequality $|V| \leq t \cdot m$. The parameter of $\mathbb{O}((G_1, k_1), \ldots, (G_t, k_t))$ is $k \cdot \log |V| \leq k \cdot \log (t \cdot m) \leq k \cdot (m + \log m) = O(m^2)$. □

(d) The parameterized problem *uni*-DOMINATING-SET has a linear OR.

Proof: Let $(G_1, k_1), \ldots, (G_t, k_t)$ be instances of *uni*-DOMINATING-SET. Let $k := \max_{i\in[t]} k_i$. By adding $k - k_i$ isolated vertices, we can pass to equivalent instances $(G'_1, k), \ldots, (G'_t, k)$. Let $G'_i = (V'_i, E'_i)$. We may assume that $t > k$ and that the vertex sets V'_i are pairwise disjoint.

If $t \geq 2^m$, where $m := \max_{i\in[t]} |V'_i|$, the algorithm $\mathbb{O}$ checks by exhaustive search whether one of the G'_is has a dominating set of size k; if so $\mathbb{O}$ outputs (G_i, k_i) for such a G'_i and otherwise it outputs (G_1, k_1).

Assume that $t < 2^m$. For $i \in [t]$ and $j \in [0, k] := \{0, 1, \ldots, k\}$ let $V'_i(j)$ be a copy of V'_i, say,

$$V'_i(j) := \{(v, j) \mid v \in V'_i\}.$$

Let $G = (V, E)$ be the graph with vertex set

$$V := \bigcup_{s\in[\log\, t]} \{s(-), s(0), s(1)\} \cup \bigcup_{i\in[t], j\in[0,k]} V'_i(j).$$

The edge set E contains

- edges that make $\{s(-), s(0), s(1)\}$ a clique for $s \in [\log\, t]$;
- for $s \in [\log\, t]$ and $i \in [t]$ edges from $s(1)$ to all vertices in $V'_i(0)$ if $\text{bit}(s, i) = 0$ and edges from $s(0)$ to all vertices in $V'_i(0)$ if $\text{bit}(s, i) = 1$;
- for $i, i' \in [t]$, $v \in V'_i$, $w \in V'_{i'}$, and $j, j' \in [0, k]$ the edge $\{(v, j), (w, j')\}$ if
  - $i \neq i'$ and $j = j' > 0$ or
  - $i = i'$ and $\{v, w\} \in E_i$ or
  - $i = i'$, $j \neq j'$ and $v = w$.

We claim that

$$(G, k + \log\, t) \in \textit{uni}\text{-DOMINATING-SET} \iff \text{there is an } i \in [t]\text{: } (G'_i, k) \in \textit{uni}\text{-DOMINATING-SET}. \quad (7)$$

For the backward direction assume for $i \in [t]$ that $\{v_1, \ldots, v_k\}$ is a dominating set in G'_i. Then

$$\{(v_1, 1), \ldots, (v_k, k)\} \cup \{s(\text{bit}(s, i)) \mid s \in [\log\, t]\}$$

is a dominating set of G.

For the forward direction let X be a dominating set of G of size $k + \log\, t$. For $s \in [\log\, t]$ in order to dominate the point $s(-)$ we see that at least one point of the clique $\{s(-), s(0), s(1)\}$ has to be contained in X.

Clearly, as $k < t$, there is an $i_0 \in [t]$ such that

$$X \cap \bigcup_{j\in[0,k]} V'_{i_0}(j) = \emptyset.$$

For $j \in [k]$ (in particular $j \neq 0$), in order to dominate the elements of $V'_{i_0}(j)$, the set X must contain an element of the form (v_j, j) with $v_j \in V'_{i_j}$ for some $i_j \neq i_0$. Moreover, as X only contains $k + \log\, t$ elements, the vertex v_j (and hence i_j) are uniquely determined by j. Then it is not hard to see that the set $\{v_j \mid j \in [k] \text{ and } i_j = i_1\}$ is a dominating set in G'_{i_1}. This finishes the proof of the equivalence (7).

We set $\mathbb{O}((G_1, k_1), \ldots, (G_t, k_t)) := (G, k)$. That $\mathbb{O}$ also satisfies condition (2) of a linear OR is shown as in the case of *uni*-CLIQUE. □

(e) The problem *alpha*-LCS has a linear OR. Here *alpha*-LCS denotes the canonical parameterization of the longest common subsequence problem:

alpha-LCS	
Instance:	An alphabet Σ, strings $X_1, \ldots, X_\ell \in \Sigma^*$, and $m \in \mathbb{N}$.
Parameter:	$m \cdot \log\, \lvert\Sigma\rvert$.
Question:	Is there a common subsequence of $X_1, \ldots, X_\ell$ of length m?

Proof: Let $(\Sigma_1, X_{11}, \ldots, X_{1\ell_1}, m_1) \ldots (\Sigma_t, X_{t1}, \ldots, X_{t\ell_t}, m_t)$ be instances of *alpha*-LCS. We can assume that $\ell_1 = \cdots = \ell_t = \ell$ (by repeating a sequence if necessary) and that $m_1 = \cdots = m_t = m$ (by adding $c_i^{m-m_i}$ to each X_{ij} for some new letter c_i). Moreover we can assume that the alphabets Σ_i are disjoint. Now we consider the ℓ strings over $\Sigma_1 \cup \ldots \cup \Sigma_t$

$$X_{11}X_{21}\ldots X_{t1}, \quad X_{12}X_{22}\ldots X_{t2}, \quad \ldots \quad X_{1\ell}X_{2\ell}\ldots X_{t\ell}$$

and the string $X_{t1}X_{(t-1)1}\ldots X_{11}$.

One easily verifies that these $(\ell+1)$ strings have a common subsequence of length m if and only if for some $i \in [t]$ the strings $X_{i1}, \ldots, X_{i\ell_i}$ have one (for the forward direction note that a common subsequence of $X_{11}X_{21}\ldots X_{t1}$ and $X_{t1}X_{(t-1)1}\ldots X_{11}$ is a sequence over Σ_i for some $i \in [t]$). Now, if $t \geq \max_{i\in[t]}|\Sigma_i|^m$ we determine the value of $\mathbb{O}$ by exhaustive search and otherwise, we use the set of strings just constructed. □

Even though we could add further examples of parameterized problems with a linear OR, there are also many problems where we do not know whether they have a linear OR. We just mention one example, the problem *uni*-RED/BLUE-NONBLOCKER, the canonical reparametrization of the problem p-RED/BLUE-NONBLOCKER.

As we have seen that p-PATH has a linear OR, Theorem 30 follows from:

Theorem 34. *Let $\varepsilon > 0$. Let (Q, κ) be a parameterized problem with a linear* OR *and with* NP*-hard Q. Unless* $\text{PH} = \Sigma_3^{\text{p}}$*, the problem (Q, κ) has no ε self-reduction.*

As we have seen that p-PATH has a linear OR, Theorem 30 is a special instance of Theorem 34. It will be convenient to reformulate Theorem 34. For this purpose we need some further notions.

Definition 35. A function $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is *pseudo-linear* if there is some $c \in \mathbb{N}$ and some $\varepsilon \in \mathbb{R}$ with $\varepsilon > 0$ such that for all $t \in \mathbb{N}$

$$f(t) \leq c \cdot t^{1-\varepsilon}.$$

The property that we need of pseudo-linear functions is contained in the following lemma. It is easy to prove.

Lemma 36. *Let $\varepsilon > 0$ and $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ be a pseudo-linear function. Then for every $c \in \mathbb{N}$ there exists a $d \in \mathbb{N}$ such that for sufficiently large n we have*

$$f(n^d) \cdot n^c + 1 \leq n^d.$$

Remark 37. As f will determine the lower bound stated in Theorem 34, it is worthwhile to note that a weak converse of the above lemma holds: Let f satisfy the conclusion of Lemma 36. Then there is some $\varepsilon > 0$ such that $f(t) < t^{1-\varepsilon}$ for infinitely many t.

To see this write $f(t) = t^{g(t)}$ for some g. Then for $c = 1$ there are $d, n_0 \in \mathbb{N}$ such that $n^{d \cdot g(n^d)} < n^{d-1}$ for all $n \geq n_0$. Thus $g(t) < 1 - 1/d$, i.e. $f(t) \leq t^{1-1/d}$, for $t = n_0^d, (n_0+1)^d, (n_0+2)^d \ldots$. ⊣

For a parameterized problem (Q, κ), a constant $c \in \mathbb{N}$, and a function $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ consider the preparameterized problem

$(Q, \kappa^c \times f)$	
Instance:	$x \in \{0,1\}^*$.
Parameter:	$\kappa(x)^c \cdot f(\lvert x\rvert)$.
Question:	$x \in Q$?

Theorem 34 follows from:

Lemma 38. *Let $c \in \mathbb{N}$ and $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ be pseudo-linear. Let (Q, κ) be a parameterized problem with a linear* OR *and with* NP*-hard Q. Then $(Q, \kappa^c \times f)$ has no linear kernelization, unless* $\text{PH} = \Sigma_3^{\text{p}}$.

We prove this lemma by generalizing Theorem 21.

Definition 39. Let $Q, Q' \subseteq \{0,1\}^*$ be classical problems and let $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ be a function. A *linear f-distillation from Q in Q'* is a polynomial time algorithm $\mathbb{D}$ that receives as inputs finite sequences $\bar{x} = (x_1, \ldots, x_t)$ with $x_i \in \{0,1\}^*$ for $i \in [t]$ and outputs a string $\mathbb{D}(\bar{x}) \in \{0,1\}^*$ such that

(1) $|\mathbb{D}(\bar{x})| = f(t) \cdot (\max_{i\in[t]}|x_i|)^{O(1)}$;

(2) $\mathbb{D}(\bar{x}) \in Q'$ if and only if for some $i \in [t]$: $x_i \in Q$.

We say that Q *has a linear f-distillation* if there is a linear f-distillation from Q in Q' for some problem Q'.

Lemma 40. *Let $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ be pseudo-linear. No* NP*-hard problem has a linear f-distillation unless* PH $= \Sigma_3^{\mathrm{P}}$.

Proof: Let $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ be pseudo-linear and $Q \subseteq \{0,1\}^*$ be NP-hard. Assume that $\mathbb{D}$ is an f-distillation from Q in some problem Q'. We choose a constant $c \in \mathbb{N}$ such that

$$|\mathbb{D}(\bar{x})| \leq f(t) \cdot \Big(\max_{i\in[t]} |x_i|\Big)^c \tag{8}$$

for all $t \in \mathbb{N}$ and all sequences $\bar{x}$ of t instances of Q.

Let $\overline{Q} := \{0,1\}^* \setminus Q$ be the complement of Q and similarly $\overline{Q'}$ the complement of Q'. Clearly $\overline{Q}$ is coNP-hard. We show that $\overline{Q} \in \mathrm{NP/poly}$ and hence, coNP $\subseteq$ NP/poly. This yields our claim, as then PH $= \Sigma_3^{\mathrm{P}}$ by a result of Yap [13, Theorem 2]. Note that for all $\bar{x} = (x_1, \ldots, x_t)$ we have

$$\mathbb{D}(\bar{x}) \in \overline{Q'} \iff \text{for all } i \in [t] : \ x_i \in \overline{Q}. \tag{9}$$

To prove $\overline{Q} \in \mathrm{NP/poly}$ it suffices to show that for sufficiently large $n \in \mathbb{N}$ there is a $t = n^{O(1)}$ and a set S of strings with $\|S\| := \sum_{x\in S} |x| = n^{O(1)}$ such that for all $x \in \{0,1\}^n$

$$x \in \overline{Q} \iff \exists x_1, \ldots, x_t \in \{0,1\}^n : \quad \Big(x \in \{x_1, \ldots, x_t\} \text{ and } \mathbb{D}(x_1, \ldots, x_t) \in S\Big).$$

In other words, S can be viewed as a polynomial size advice string for instances of length n. As we will see, the elements of S are strings in $\overline{Q'}$, more precisely, we will choose $\mathbb{D}$-values "with many preimages."

For every $m \in \mathbb{N}$, we have $|\{0,1\}^{\leq m}| \leq 2^{m+1}$, in particular,

$$|\{0,1\}^{\leq f(m)\cdot n^c}| \leq 2^{f(m)\cdot n^c + 1} \tag{10}$$

As f is pseudo-linear, by Lemma 36 there is a constant $d \in \mathbb{N}$ such that for all sufficiently large $n \in \mathbb{N}$

$$\frac{f(n^d) \cdot n^c + 1}{n^d} \leq 1. \tag{11}$$

For $n \geq 1$ we set

$$t := n^d.$$

Then (10) and (11) imply for $Y := \overline{Q'} \cap \{0,1\}^{\leq f(t)\cdot n^c}$ that

$$|Y|^{1/t} \leq 2. \tag{12}$$

Recall that $\overline{Q}_{=n} := \overline{Q} \cap \{0,1\}^n$. By (8) we can define a function $g : (\overline{Q}_{=n})^t \to Y$ by

$$g(\bar{x}) := \mathbb{D}(\bar{x}).$$

We construct the advice string S inductively. First we let $X_0 := \overline{Q}_{=n}$. Choose $y_0 \in Y$ such that

$$g^{-1}(y_0) := \big\{\bar{x} \in X_0^t \mid g(\bar{x}) = y_0\big\}$$

contains at least $|X_0|^t/|Y|$ many tuples. Let *string*$(g^{-1}(y_0))$ be the set components of tuples in $g^{-1}(y_0)$, that is,

$$\textit{string}(g^{-1}(y_0)) := \big\{x \in X_0 \mid \text{there exists some } (x_1, \ldots, x_t) \in g^{-1}(y_0) \text{ such that } x \in \{x_1, \ldots, x_t\}\big\}.$$

It follows that $g^{-1}(y_0) \subseteq \big(\textit{string}(g^{-1}(y_0))\big)^t$ and hence

$$\big|\textit{string}(g^{-1}(y_0))\big| \geq |g^{-1}(y_0)|^{1/t} \geq \left(\frac{|X_0|^t}{|Y|}\right)^{1/t} \geq \frac{|X_0|}{2},$$

the last inequality holding by (12). If $X_0 \neq \textit{string}(g^{-1}(y_0))$, then let $X_1 := X_0 \setminus \textit{string}(g^{-1}(y_0))$. Now, we view g as a function of X_1 to Y and, by the same argument as above, we choose $y_1 \in Y$ such that $|\textit{string}(g^{-1}(y_1))| \geq |X_1|/2$. We iterate this process until we reach the first $\ell \in \mathbb{N}$ with $X_\ell = \textit{string}(g^{-1}(y_\ell))$. We let

$$S := \{y_0, \ldots, y_\ell\}.$$

Then $S \subseteq Y \subseteq \overline{Q'}$ and $|S| = \ell \leq \log |X_0| \leq n$ and thus $\|S\| \leq n \cdot f(t) \cdot n^c \leq n^{d+1}$ (by (11)). Hence $\|S\|$ is polynomially bounded in n.

We show the equivalence (10). Let $x \in \{0,1\}^n$. If $x \in \overline{Q}$, by our construction of S, there is a tuple $\bar{x}$ containing x as a component such that $g(\bar{x}) = \mathbb{D}(\bar{x}) \in S$.

Conversely, assume $x \notin \overline{Q}$. Then for every $\bar{x} := (x_1, \ldots, x_t)$ with $x_1, \ldots, x_t \in \{0,1\}^n$ and $x \in \{x_1, \ldots, x_t\}$, we have, by (9), that $\mathbb{D}(\bar{x}) \notin \overline{Q'}$ and hence $\mathbb{D}(\bar{x}) \notin S \subseteq \overline{Q'}$. □

Proof of Lemma 38: Let $c \in \mathbb{N}$ and f be pseudo-linear, say $f(t) = O(t^{1-\varepsilon})$. Assume that (Q, κ) is a parameterized problem with a linear OR $\mathbb{O}$ and NP-hard Q. Assume $\Sigma_3^{\mathrm{p}} \neq$ PH. For the sake of contradiction assume that $(Q, \kappa^c \times f)$ has a linear kernelization $\mathbb{K}$. By Lemma 40 it suffices to show that Q has a linear f-distillation $\mathbb{D}$.

We define $\mathbb{D}$ on finite sequences $\bar{x} = (x_1, \ldots, x_t)$ by

$$\mathbb{D}(\bar{x}) := \mathbb{K}(\mathbb{O}(\bar{x})).$$

It is clear that

$$\mathbb{D}(\bar{x}) \in Q \iff \text{for some } i \in [t]: \ \ x_i \in Q.$$

Write $n := \max_{i \in [t]} |x_i|$. Then, because $\mathbb{K}$ is a linear kernelization for $(Q, \kappa^c \times f)$,

$$|\mathbb{D}(\bar{x})| = O\Big(\kappa(\mathbb{O}(\bar{x}))^c \cdot f(|\mathbb{O}(\bar{x})|)\Big) = O(n^{O(1)} \cdot |\mathbb{O}(\bar{x})|^{1-\varepsilon}) = n^{O(1)} \cdot |\mathbb{O}(\bar{x})|^{1-\varepsilon},$$

where the second equality follow from Definition 32 (2). Now, by Definition 32 (1) we know $|\mathbb{O}(\bar{x})| = t \cdot n^{O(1)}$. Hence $|\mathbb{D}(\bar{x})| = t^{1-\varepsilon} \cdot n^{O(1)}$ and therefore $\mathbb{D}$ is a linear f-distillation from Q in itself. □

In particular the problems mentioned in Examples 33 do not have an ε self-reduction unless PH $= \Sigma_3^{\mathrm{p}}$.

7. Lower bounds for problems with an OR for instances with constant parameter

We consider the parameterized problem

p-CYCLE	
Instance:	A graph G and $k \in \mathbb{N}$.
Parameter:	k.
Question:	Does G have a cycle of length k?

Let $(G_1, k_1), \ldots, (G_t, k_t)$ be instances of p-CYCLE. If $k_1 = \ldots = k_t =: k$, then for the disjoint union G of the G_is we have $(G, k) \in p$-CYCLE if and only if $(G_i, k_i) \in p$-CYCLE for some $i \in [t]$. However, it is not clear how to define such an instance (G, k) if $k_1, \ldots, k_t$ are distinct, more precisely, we do not know whether p-CYCLE has an OR. The following concept is tailored for such situations.

Definition 41. Let (Q, κ) be a parameterized problem and let λ be a further parameterization. An *OR for λ-constant instances of* (Q, κ) is a polynomial time algorithm $\mathbb{O}$ that for every finite tuple $\bar{x} = (x_1, \ldots, x_t)$ of instances of Q with $\lambda(x_1) = \ldots = \lambda(x_t)$ outputs an instance $\mathbb{O}(\bar{x})$ of Q such that

(1) $\kappa(\mathbb{O}(\bar{x})) = (\max_{i \in [t]} |x_i|)^{O(1)}$;

(2) $\mathbb{O}(\bar{x}) \in Q$ if and only if for some $i \in [t]$: $x_i \in Q$.

Examples 42. The instances of the following problems are pairs (G, k), where G is a graph and $k \in \mathbb{N}$. We let λ always be the function with $\lambda(G, k) := k$. In all examples we get the claimed OR for λ-constant instances by setting $\mathbb{O}((G_1, k), \ldots, (G_t, k)) := (G, k)$, where the graph G is the disjoint union of the G_is. In all cases we do not know whether the corresponding problem has an OR.

(a) The problem p-CYCLE has an OR for λ-constant instances.

(b) The problems *uni*-CHORDLESS-PATH and *uni*-CHORDLESS-CYCLE have an OR for λ-constant instances. Here, for example,

uni-CHORDLESS-CYCLE	
Instance:	A graph $G = (V, E)$ and $k \in \mathbb{N}$.
Parameter:	$k \cdot \log |V|$.
Question:	Does G have a chordless cycle of length k?

Note that in the last example $\lambda(G, k) = k$ is not the parameter of (G, k) as instance of *uni*-CHORDLESS-CYCLE.

For problems with an OR for constant instances we get a slightly weaker result than that in Theorem 34 for problems with a linear OR. To state the result we first define:

Definition 43. Let (Q, κ) be a parameterized problem. A *subexponential self-reduction of* (Q, κ) is a polynomial reduction from Q to itself that assigns to every instance x of Q an instance y with

$$|y| = \kappa(x)^{O(1)} \cdot |x|^{o(1)}.$$

Clearly if (Q, κ) has a subexponential self-reduction, then it has an ε self-reduction for every $\varepsilon > 0$.

Theorem 44. *Let (Q, κ) be a parameterized problem with* NP*-complete Q. Furthermore assume that (Q, κ) has an* OR *for λ-constant instances, where λ is a further parameterization. Unless* PH $= \Sigma_3^{\mathrm{P}}$*, there is* no *subexponential self-reduction of (Q, κ).*

In particular, (Q, κ) has no polynomial kernelization (a result shown in [4]).

Recall the reparameterization $(Q, \kappa^c \times f)$ of (Q, κ) for $c \in \mathbb{N}$ and $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$. Clearly $(Q, \kappa^c \times f)$ has a polynomial kernelization if and only if $(Q, \kappa \times f)$, the problem for $c = 1$, has one.

For the purposes of the proof of Theorem 44 we call a function $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ *good* if $f(t) = t^{o(1)}$ for all $t \in \mathbb{N}$ (that is, if we can write $f(t) = t^{1/h(t)}$ for some function $h : \mathbb{N} \to \mathbb{R}_{\geq 0}$ with $\lim_{t\to\infty} h(t) = \infty$).

The statement of this theorem can be equivalently formulated as:

Lemma 45. *Let (Q, κ) be a parameterized problem with* NP*-complete Q. Furthermore assume that (Q, κ) has an* OR *for λ-constant instances, where λ is a further parameterization. Then, unless* PH $= \Sigma_3^{\mathrm{P}}$*, for every good $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ the problem $(Q, \kappa \times f)$ has no polynomial kernelization.*

Proof: Assume PH $\neq \Sigma_3^{\mathrm{P}}$. Furthermore, we choose for (Q, κ) an OR $\mathbb{O}$ for λ-constant instances.

Let $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ be good. One easily sees that there is a good increasing function $f' : \mathbb{N} \to \mathbb{R}_{\geq 0}$ of the form

$$f'(t) = 2^{\log t/\iota(\log t)} \tag{13}$$

with a nondecreasing and unbounded function $\iota : \mathbb{N} \to \mathbb{R}_{\geq 0}$ such that $f(t) \leq f'(t)$ for all (sufficiently large) t.

For the sake of contradiction assume also that $(Q, \kappa \times f)$ has a polynomial kernelization. Of course, then $(Q, \kappa \times f')$ has a polynomial kernelization $\mathbb{K}$. We show that then Q has a linear g-distillation $\mathbb{D}$ for some pseudo-linear g, which contradicts Lemma 40.

Let $x_1, \ldots, x_t$ be instances of Q. We let $n := \max_{i\in[t]}|x_i|$ and $\ell := \max_{i\in[t]}\lambda(x_i)$. Then $\ell = n^{O(1)}$. For $j \leq \ell$ let

$$y_j := \mathbb{K}(\mathbb{O}(\bar{x}_j)),$$

where $\bar{x}_j$ stands for the subsequence of $x_1, \ldots, x_t$ consisting of the instances with λ-value j.

We show that for some good function f_1 and all $j \leq \ell$

$$|y_j| = f_1(t) \cdot n^{O(1)}. \tag{14}$$

In fact, as $\mathbb{K}$ is a polynomial kernelization of $(Q, \kappa \times f')$, we know

$$|y_j| = |\mathbb{K}(\mathbb{O}(\bar{x}_j))| = \Big(\kappa(\mathbb{O}(\bar{x}_j)) \cdot f'(|\mathbb{O}(\bar{x}_j)|)\Big)^{O(1)} = n^{O(1)} \cdot f'(|\mathbb{O}(\bar{x})|)^{O(1)},$$

where the last equality holds by Definition 41 (1). We show that $f'(|\mathbb{O}(\bar{x})|) = f'(t)^d \cdot n^d$ for some $d \in \mathbb{N}$. Then we get (14) for $f_1(t) := f'(t)^d$.

As $\mathbb{O}$ is polynomial time computable, we know $|\mathbb{O}(\bar{x}_j)| \leq t^c \cdot n^c$ for some constant $c \in \mathbb{N}$. Since f' is increasing, it is enough to show

$$f'(t^c \cdot n^c) \leq (f'(t) \cdot n)^{2c}.$$

By (13)

$$f'(t^c \cdot n^c) = 2^{\frac{c \cdot \log t + c \cdot \log n}{\iota(c \cdot \log t + c \cdot \log n)}}.$$

We distinguish two cases.

- If $t \geq n$, then, as ι is nondecreasing, we get

$$f'(t^c \cdot n^c) \leq 2^{\frac{2c \cdot \log t}{\iota(\log t)}} = f'(t)^{2c}.$$

- If $t < n$, then

$$f'(t^c \cdot n^c) \leq 2^{2c \cdot \log n} = n^{2c}.$$

This finishes the proof of (14).

As by assumption Q is NP-complete, there are polynomial time reductions R and S from Q to SAT and from SAT to Q, respectively. Applying the reduction $R : Q \leq^p \text{SAT}$ we have $|R(y_j)| = f_1(t)^{O(1)} \cdot n^{O(1)}$. Hence

$$\Big| \bigvee_{j \in [\ell]} R(y_j) \Big| = O\big(\ell \cdot f_1(t)^{O(1)} \cdot n^{O(1)}\big) = f_1(t)^{O(1)} \cdot n^{O(1)},$$

because $\ell = n^{O(1)}$. As $S : \text{SAT} \leq^p Q$, there is a $e \in \mathbb{N}$ such that

$$\big|S\big(\bigvee_{j \in [\ell]} R(y_j)\big)\big| \leq f_1(t)^e \cdot n^e$$

However $g(t) := f_1(t)^e$ is good and in particular, pseudo-linear. Therefore $\mathbb{D}(x_1, \ldots, x_t) := S\big(\bigvee_{j \in [\ell]} R(y_j)\big)$ defines a linear g-distillation from Q in itself. □

In particular, we can apply Theorem 44 to the problems listed in Examples 42.

Clearly, every parameterized problem (Q, κ) with an OR has an OR for κ-constant instances. In this case we do not need the reduction to the problem SAT in the previous proof. Hence, we get the following improvement of Corollary 26:

Theorem 46. *Let (Q, κ) be a parameterized problem with an* OR *and with* NP*-hard Q. Unless* PH $= \Sigma_3^{\text{p}}$*, the problem (Q, κ) has no subexponential self-reduction.*

We omit the proof of the following lemma, which is simple and similar to that of Lemma 19.

Lemma 47. *Let (Q, κ) and (Q', κ') be parameterized problems. with*

$$(Q, \kappa) \leq^p (Q', \kappa') \quad \textit{and} \quad Q' \leq^p Q.$$

If (Q', κ') has a subexponential self-reduction, then (Q, κ) has a subexponential self-reduction.

We finish this section with an example.

Example 48. Unless PH $= \Sigma_3^{\text{p}}$, the problem p-PATH(PLAN-CONN) has no subexponential self-reduction.

Proof: We know that the problem p-POINTED-PATH(PLAN-CONN) has an OR and hence no subexponential self-reduction. In the proof of Proposition 27 we showed that there is a polynomial reduction from p-POINTED-PATH(PLAN-CONN) to p-PATH(PLAN-CONN). Hence, the claim follows from the previous lemma.

8. Concluding remarks

8.1. Comparing the different notions of OR. From Theorem 21, Corollary 26, and Theorem 34 we know:

Proposition 49. *Assume that* PH $\neq \Sigma_3^{\text{p}}$*. Then:*

(1) No NP*-complete problem has a self-distillation.*

(2) No parameterized problem (Q, κ) with polynomial kernelization and with NP*-complete Q has an* OR*.*

(3) No parameterized problem (Q, κ) with polynomial kernelization and with NP*-complete Q has a linear* OR*.*

We do not know whether one of the three conclusions holds under weaker assumptions, say, under P $\neq$ NP. In this context it might be interesting to be aware of:

Proposition 50. *The conclusions (1), (2), and (3) of Proposition 49 are mutually equivalent.*

Proof: The implication (2) $\Rightarrow$ (3) is trivial. For (3) $\Rightarrow$ (1) assume, by contradiction, that Q is NP-complete and has a self-distillation $\mathbb{D}$. Define $\kappa(x) := |x|$. Then $x \mapsto x$ is a polynomial kernelization of (Q, κ) and $\mathbb{D}$ is a linear OR of (Q, κ), the desired contradiction to (3).

For the implication (1) $\Rightarrow$ (2) assume that (Q, κ) with NP-complete Q has a polynomial kernelization $\mathbb{K}$ and an OR $\mathbb{O}$. Then $\mathbb{K} \circ \mathbb{O}$ is a self-distillation, as

$$\mathbb{K}(\mathbb{O}(\bar{x})) = \kappa(\mathbb{O}(\bar{x}))^{O(1)} = (\max_i |x_i|)^{O(1)}.$$

□

The next result shows in particular that every parameterized problem (Q, κ) with polynomial kernelization and NP-complete Q already has no OR if it has no linear OR. For example, p-VC has no linear OR if and only if it nas no OR.

Proposition 51. *Assume that (Q, κ) and (Q', κ') are parameterized problems with* NP*-complete Q and Q' and that (Q', κ') has a polynomial kernelization. If (Q, κ) has no linear OR, then (Q', κ') has no* OR*.*

Proof: Let $R : Q \to Q'$ and $S : Q' \to Q$ be polynomial reductions and $\mathbb{K}$ a polynomial kernelization of (Q', κ') and assume that $\mathbb{O}$ is a OR of Q', then

$$x_1, \ldots, x_t \mapsto S(\mathbb{K}(\mathbb{O}(R(x_1), \ldots, R(x_t))))$$

is a linear OR of (Q, κ). □

8.2. Comparing the different notions of self-reduction. Clearly, every parameterized problem with a polynomial kernelization has a subexponential self-reduction, and every parameterized problem with a subexponential self-reduction has an ε self-reduction for every $\varepsilon > 0$. Proposition 53 and Proposition 52 show that the reverse of the first implication and of the second implication fail, respectively.

Proposition 52. *Let $Q \subseteq \mathbb{N}$ be a classical problem such that every $x \in Q$ is a power of 2 with an odd exponent and is written in unary. We define the parameterized problem p-Q by*

p-Q	
Instance:	$m, k \in \mathbb{N}$ in unary with $\log k \geq \frac{\log m}{\log \log m}$.
Parameter:	k.
Question:	Is $(\log m) \cdot (\log k) \in Q$?

Then:

(1) If Q is decidable, then p-Q is fixed-parameter tractable.

(2) For every $\varepsilon > 0$ the problem p-Q has an ε self-reduction.

(3) If $Q \notin$ E, then p-Q has no subexponential reduction.

Proof: (1) As for yes-instances (m, k) of p-Q, we have $\log\ k \geq \log\ m/\log\ \log\ m$, the problem p-Q has a kernelization and hence is fixed-parameter tractable by Proposition 2.

(2) Let $t \in \mathbb{N}$. We show that there is an $1/d$ self-reduction of p-Q for $d := 2^t$.

Let (m, k) be an instance of p-Q. We can assume that $m = 2^{2^u}$ and $k = 2^{2^v}$ (otherwise, (m, k) is a no-instance of p-Q).

We set

$$m' := 2^{2^{u-t}}\ (= (2^{2^u})^{1/d}) \qquad \text{and} \qquad k' := 2^{2^{v+t}}\ (= (2^{2^v})^d).$$

Clearly, $(m,k) \in p\text{-}Q$ if and only if $(m',k') \in p\text{-}Q$. Moreover, $|m'| = |m|^{1/d}$ and $|k'| = |k|^d$ and hence, $|(m',k')| = O(k^d \cdot m^{1/d})$. Altogether, $(m,k) \mapsto (m',k')$ is an $1/d$ self-reduction of $p\text{-}Q$.

(3) We assume that $p\text{-}Q$ has a subexponential self-reduction $(m,k) \mapsto (m',k')$. Then

$$|(m',k')| = k^c \cdot (m+k)^{o(1)} = k^c \cdot m^{o(1)}$$

for some $c \in \mathbb{N}$. We can assume that c is a power of 2. We show that $Q \in \mathrm{E}$.

Let x be an instance of Q with $x \geq d \geq 2^{4c^2}$, where $d \in \mathbb{N}$ will be fixed later. We assume that x is an odd power of 2 (otherwise, $x \notin Q$). We set

$$u := \sqrt{2c^2 \cdot x} \qquad \text{and} \qquad v := \frac{u}{2c^2}.$$

Then, u and v are powers of 2 (note that $v = \sqrt{x/2c^2}$) and $u \cdot v = x$. Moreover, $v \geq u/\log u$ by our assumption $x \geq 2^{4c^2}$. Hence, $(2^u, 2^v) \in p\text{-}Q$ if and only if $x \in Q$. We apply the subexponential self-reduction to $(2^u, 2^v)$ obtaining an equivalent instance (m',k') of $p\text{-}Q$ with

$$m', k' \leq 2^{v \cdot c} \cdot (2^u)^{o(1)} = 2^{v \cdot c + u \cdot o(1)}.$$

If d has been chosen big enough, we have

$$x' := (\log m') \cdot (\log k') \leq (v \cdot c)^2 + v \cdot u \cdot o(1) + u^2 \cdot o(1) \leq (u/2c)^2 + u^2 \cdot o(1) < u^2/2c^2 = uv = x.$$

Thus, $x' < x$. If $k' < m'/\log m'$, then $(m',k') \notin p\text{-}Q$ and hence, $x \notin Q$. Otherwise, ($x' \in Q \iff x \in Q$). We continue this way and obtain equivalent instances $x'', x''', \ldots$ of Q till we get an instance $\leq d$, which is decided directly. Altogether, we have a single exponential decision procedure for Q. □

Proposition 53. *Let $Q \subseteq \mathbb{N}$ be a classical problem such that every $x \in Q$ is represented in unary and has the form*

$$x = 2^{2^t} \tag{15}$$

for some $t \in \mathbb{N}$. We define the parameterized problem $p\text{-}\mathrm{EXP}(Q)$ by

$p\text{-}\mathrm{EXP}(Q)$	
Instance:	$m, k \in \mathbb{N}$ in unary with $k \geq \log\log m$.
Parameter:	k.
Question:	Is $m^k \in Q$?

Then:

(1) If Q is decidable, then $p\text{-}\mathrm{EXP}(Q)$ is fixed-parameter tractable.

(2) The problem $p\text{-}\mathrm{EXP}(Q)$ has a subexponential self-reduction.

(3) If $Q \notin$ PTIME, then $p\text{-}\mathrm{EXP}(Q)$ has no polynomial kernelization.

Proof: (1) As for yes-instances (m,k) of $p\text{-}\mathrm{EXP}(Q)$, we have $k \geq \log\log m$, the problem $p\text{-}\mathrm{EXP}(Q)$ has a kernelization and hence is fixed-parameter tractable by Proposition 2.

(2) Let (m,k) be an instance of $p\text{-}\mathrm{EXP}(Q)$. By (15), we can assume that $m = 2^{2^t}$ for some $t \in \mathbb{N}$ (otherwise, (m,k) is a no-instance of $p\text{-}\mathrm{EXP}(Q)$). Then

$$(m,k) \in p\text{-}\textsc{Exp}(Q) \iff 2^{k \cdot 2^t} \in Q \iff (2, k \cdot 2^t) \in p\text{-}\textsc{Exp}(Q).$$

Therefore the mapping $(m,k) \mapsto (2, k \cdot \log m)$ is the desired reduction.

(3) We assume that $\mathbb{K}$ is a polynomial kernelization of $p\text{-}\textsc{Exp}(Q)$ and show that $Q \in$ PTIME.

Let $x = 2^{2^t}$ be an instance of Q. We let t' be the minimum power of 2 with $t' \geq t$. Thus, $2t \geq t' \geq t$. Clearly

$$x \in Q \iff (2^{2^t/t'}, t') \in p\text{-}\textsc{Exp}(Q).$$

Furthermore we set $(m,k) := \mathbb{K}(2^{2^t/t'}, t')$. We know that

$$|(m,k)| = t'^{O(1)} = t^{O(1)}$$

and that $x \in Q$ if and only if $m^k \in Q$. As

$$m^k = t^{O(t^{O(1)})} = 2^{t^{O(1)}}$$

we see that this is strictly smaller than x if x is sufficiently large. □

References

[1] R. Chang and Y. Kadin. On computing boolean connectives of characteristic functions. *Math. Systems Theory*, 28:173 – 198, 1995.

[2] Y. Chen and J. Flum. On parameterized path and chordless path problems. In *Proceedings of the 22nd IEEE Conference on Computational Complexity (CCC'07)*, page 250 – 263, 2007

[3] Y. Chen and J. Flum. Subexponential time and fixed-parameter tractability: exploiting the miniaturization mapping. In *Proceedings of the 21st International Workshop on Computer Science Logic (CSL'07)*, Lecture Notes in Computer Science 4646, page 389 – 404, 2007.

[4] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. Submitted, 2007.

[5] J. Flum and M. Grohe. *Parameterized Complexity Theory*, Springer, 2006.

[6] L. Fortnow and Santhanam. Infeasibility of instance compression and succinct PCPs for NP. Available at `http://lance.fortnow.com/papers/`

[7] M. Frick and M. Grohe. Deciding first-order properties of locally tree-decomposable structures. *Journal of the ACM*, 48:1184-1206, 2001.

[8] M. Frick and M. Grohe. The complexity of first-order and monadic second-order logic revisited. *Annals of Pure and Applied Logic*, 130:3 – 31, 2004.

[9] J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, Vol. 38, No. 1, 2007.

[10] D. Harnik and M. Naor. On the compressibility of NP instances and cryptographic applications, In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, page 719 – 728, 2006. Full version appears as TR06-022 in ECCC Reports 2006, available at `http://eccc.hpi-web.de/eccc-local/Lists/TR-2006.html`

[11] R. Impagliazzo, R.Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63:512 – 530, 2001.

[12] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[13] C. K. Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science* 26, page 287 – 300, 1983.