The Computational Complexity Column

by

V. Arvind

Institute of Mathematical Sciences, CIT Campus, Taramani
Chennai 600113, India
arvind@imsc.res.in
http://www.imsc.res.in/~arvind

A fundamental technique in the design of parameterized algorithms is *kernelization*: Given a problem instance I with parameter k, the basic idea is to try and preprocess the instance I of length n by applying efficient "reduction rules" in order to simplify it and reduce it to a kernel instance of the same problem that is of size a polynomial in k. A brute-force/exponential-time algorithm can then be used to solve the kernel instance. Smaller kernels often lead to faster algorithms. How small, as a function of k, can kernels be made? There is a nice hardness theory, based on the complexity theoretic assumption coNP $\not\subseteq$ NP/poly, which can be used to prove lower bounds for kernel size.

Kernelization is a flourishing area of parameterized complexity with many recent results (both upper and lower bounds). Stefan Kratsch shares with us some of the latest developments in the field. His very readable survey article, with illustrative examples, invites the non-expert to this exciting area of complexity theory.

Recent developments in kernelization: A survey

Stefan Kratsch*
Technical University Berlin, Germany
stefan.kratsch@tu-berlin.de

Abstract

Kernelization is a formalization of efficient preprocessing, aimed mainly at combinatorially hard problems. Empirically, preprocessing is highly successful in practice, e.g., in state-of-the-art SAT and ILP solvers. The notion of kernelization from parameterized complexity makes it possible to rigorously prove upper and lower bounds on, e.g., the maximum output size of a preprocessing in terms of one or more problem-specific parameters. This avoids the often-raised issue that we should not expect an efficient algorithm that provably shrinks every instance of any NP-hard problem.

In this survey, we give a general introduction to the area of kernelization and then discuss some recent developments. After the introductory material we attempt a reasonably self-contained update and introduction on the following topics: (1) Lower bounds for kernelization, taking into account the recent progress on the AND-conjecture. (2) The use of matroids and representative sets for kernelization. (3) Turing kernelization, i.e., understanding preprocessing that adaptively or non-adaptively creates a large number of small outputs.

1 Introduction

Kernelization is a theoretical formalization of efficient preprocessing for (NP-)hard problems. By efficient preprocessing we mean any polynomial-time algorithm that given a problem instance outputs an equivalent instance that is, if possible, simpler than the initial one. Mainly, we are interested in data reduction where the obtained instance is as small as possible (but we will

*Supported by the Emmy Noether-program of the German Research Foundation (DFG), KR 4286/1.

avoid the term data reduction for its name clash with reductions). Empirically, preprocessing is very successful in practice, e.g., within the well-known ILP solver CPLEX, which motivates a mathematically rigorous study.

Before giving formal definitions and further background, let us begin with a simple and well-known example. Consider the Vertex Cover problem where we are given as input a graph $G = (V, E)$ and a value $k \in \mathbb{N}$ and we need to determine whether there exists a set S of at most k vertices such that every edge is incident with at least one vertex in S. Due to the NP-hardness of the problem we do not expect that *every instance* can be efficiently reduced in size. Indeed, any polynomial-time algorithm that guarantees a size reduction of at least one bit for *all instances* of Vertex Cover could be iterated to also solve Vertex Cover in polynomial time, implying $\mathsf{P} = \mathsf{NP}$. Despite this obstacle to efficient preprocessing there are simple reduction rules that can be seen to yield a provable size bound; how does that fit together?

Rule 1. Delete any isolated vertex v of G, i.e., return $(G-v, k)$. *Correctness: We never need v in any solution since it covers no edges.*

Rule 2. If a vertex v has degree greater than k in G then we (are forced to) select the vertex for the solution, which is expressed by returning $(G-v, k-1)$. *Correctness: Not selecting v would require selecting the neighborhood $N(v)$ of v which is of size greater than our budget k.*

Rule 3. If Rule 2 does not apply and the graph G has more than k^2 edges then answer no. *Correctness: Covering more than k^2 edges with at most k vertices would require at least one vertex of degree greater than k.*

It is not hard to see that all three rules can be applied in polynomial time and that when no rule is applicable we have an equivalent instance with a graph that has at most k^2 edges and $2k^2$ vertices; this instance can be encoded in $\mathcal{O}(k^2 \log k)$ bits. (By more sophisticated arguments this can be improved to at most $2k$ vertices and $\mathcal{O}(k^2)$ total size [15].)

We see that by relating the output guarantee of our preprocessing to the value k, we avoided the issue of not being able to shrink every instance. Intuitively, the solution size k in a vertex cover instance is a good measure of its complexity, since it is not hard to find, e.g., a $\mathcal{O}(2^k nm)$ time branching algorithm for it; if k is constant or at least $k \in \mathcal{O}(\log n)$ then this runtime is even polynomial in the input size. Similarly, our simple preprocessing has showed us that a comparatively small value of k implies that the size of our instance can be reduced. If, otherwise, k is large (compared to n) then the bound of $n \leq 2k^2$ does not guarantee any simplification, which is consistent with the observed obstacle to general efficient size reductions.

Generally, the field of *parameterized complexity* studies the influence of so-called *parameters*, like k for Vertex Cover, on problem complexity. We will adopt the naming convention of including the parameter choice into the problem name, e.g., Vertex Cover(k) stands for Vertex Cover with parameter k and Vertex Cover(Δ) stands for parameterization by maximum degree. A kernelization for a parameterized problem can then be simply formalized as any efficient algorithm that gives an equivalent instance of size (and parameter value) bounded by a function in the input parameter (see Section 3 for formal definitions). It should come as no surprise that the achievable output guarantees depend greatly on the choice of parameter.

2 A brief history and overview of kernelization

The use of reduction rules to simplify problems is often traced back to the work of Quine [66] from 1952 on simplifying truth functions, e.g., by unit-clause propagation and elimination of pure literals. It was recognized early that efficient reduction rules are not only empirically useful but could also be used to improve theoretical performance guarantees of exhaustive search algorithms by ensuring structural restrictions (like degree-bounds); see, e.g., [68]. The study of provable performance guarantees for preprocessing by reduction rules (or any other means) regarding the achievable output size, rather than achievable structure, took much longer to develop.

Kernelization originated as one of many techniques in the toolbox of parameterized complexity (see [24, 25]) and is a successful theoretical formalization of efficient preprocessing with provable performance guarantees. In its early stages kernelization was mostly about coming up with clever reduction rules and combining them with combinatorial arguments to prove that exhaustively reduced instances (to which no more rule could be applied) have size bounded by some function in the initial parameter value. A 2007 survey of Guo and Niedermeier [40] nowadays provides a nice overview on these "early days of kernelization"[1] and in particular asked to develop techniques for kernelization lower bounds. Two other influential works from that time are the linear kernel for Planar Dominating Set by Alber et al. [3] and a programmatic paper of Estivill-Castro et al. [29] that amongst others was perhaps the first to explicitly ask for Turing kernelizations.

The field of kernelization matured, in a sense, when in 2008 Bodlaender et al. [9] came up with a framework for ruling out polynomial sized kernels for many parameterized problems, and, shortly afterwards, this was followed by the first paper on meta kernelization by Bodlaender et al. [10] that gave

[1] The field of kernelization is still in its twenties.

general kernelization results for a wealth of problems on planar and bounded genus graphs (see also the 2009 survey of Bodlaender [7]). Since then, the field of kernelization has been growing rapidly and many new techniques for upper and lower bounds were invented in short succession, apart, of course, from a wealth of results for concrete problems. The survey of Lokshtanov et al. [58] on the occasion of Mike Fellows' 60th birthday in 2012 (see also [8]) gives an excellent account of these developments.

In the present survey we want to focus mainly on recent developments that have taken place since 2012, but also provide a fair introduction for readers new to the field. To this end, the core part of the survey singles out three topics and attempts a (as far as possible) self-contained and detailed presentation. Concretely, we will discuss the use of matroids and representative sets for kernelization (based on [56, 57]), and review the current knowledge about Turing kernelization (motivated by recent progress [69, 49]). Furthermore, since the lower bound framework initiated by Bodlaender et al. [9] holds a central place in kernelization, we explain one complete set of tools for proving such lower bounds. This is, of course, also motivated by the breakthrough work of Drucker [26] that (among other results) settled the so-called AND-distillation conjecture.[2] But, first things first, let us begin by giving an overview of all the interesting things that could not be fitted into this survey for the sake of length and focus.[3]

Overview. The "bread and butter", so to speak, in the kernelization business lies in studying a given parameterized problem, deriving efficient reduction rules for it, and analyzing the obtained rules, that is, analyzing the structure and size of reduced instances. Unfortunately, such rules are of course problem dependent and there does not appear to be *the single general recipe* for them. That said, two frequently used approaches are the following: (1) Begin with an approximation of the desired object or a dual structure. If this is sufficiently large then the instance is trivially YES or trivially NO. If not then there must be large parts that do not contribute to the solution (or do not incur any cost), or that are obstructed by a small set of objects/vertices/etc. Often, a careful analysis can devise "high-degree rules" (as for the simple example of VERTEX COVER(k)) that resolve or simplify these cases. (2) Another frequently used tool is the Sunflower Lemma of Erdős and Rado [28], particularly for covering or packing objects or sets of bounded size. Effectively, the Sunflower Lemma states that a sufficiently

[2] Very recently, Dell [20] announced a simpler proof for the AND-distillation conjecture.

[3] Conveniently, and not entirely by chance, these topics are covered in detail by Lokshtanov et al. [58].

large family of bounded size objects either involves a large packing (giving trivial YES for packing and trivial NO for covering) or it contains a so-called sunflower formed by objects that are pairwise obstructing in the same way; often, we can safely delete on of these obstructing objects (and repeat).

To get a more detailed understanding of reduction rule based kernelization results it is probably best to read some of them in detail; see, e.g., [50, 11, 52].

Above-guarantee parameterization. Many maximization problems have the property that, perhaps after some simple reduction rules, the optimum value OPT for an instance x is at least $\frac{1}{c} \cdot |x|$. This entails that, if $|x| \geq ck$ then the question whether $OPT \geq k$ is trivially YES, and otherwise we have $|x| < ck$; this is a (trivial) kernelization for the problem. As an example, consider the MAX CUT(k) problem where given a graph $G = (V, E)$ and $k \in \mathbb{N}$ we ask whether there is a bipartition of the vertex set such that at least k edges have endpoints on both sides. It is well known that OPT equals at least half the number m of the edges. Thus, $m \geq 2k$ gives an immediate YES and $m < 2k$ gives a linear kernelization (after discarding isolated vertices). More generally, if we know that $OPT \in \Omega(|x|^{-c})$ then we get a trivial kernelization to size $\mathcal{O}(k^c)$.

Motivated by these trivial kernelizations and the fact that the parameter needs to be large to have a nontrivial instance, Mahajan and Raman [61] initiated the study of problems parameterized *above lower bounds*. For example, they considered the MAX CUT($k - \frac{m}{2}$) problem asking whether there is a cut with at least $k = \frac{m}{2} + k'$ edges, parameterized by $k' = k - \frac{m}{2}$, and showed that this problem remains fixed-parameter tractable. Gutin et al. [43] (and follow-up work of Alon et al. [4]) made an important contribution to this direction by introducing the use of the probabilistic method. At high level, they prove that a random solution will exceed the lower bound by at least k with nonzero probability, provided that the instance is sufficiently large compared to k; again (though no longer trivial) this yields either a direct YES or the instance is sufficiently small. Among the further results in this direction let us point out Crowston et al. [17, 16] who obtain further kernelization results.

Meta kernelization. The term *meta kernelization* refers to a series of (positive) kernelization results that apply to a large variety of graph problems when the input graphs are restricted to (in most cases) sparse graph classes such as planar, bounded genus, or H-minor-free graphs [10, 32, 36, 37, 51, 38]. "Meta" here means that the results apply assuming that the problem in question fulfills an appropriate set of technical but rather general properties, obviating the need for any problem-specific reduction rules. A key necessity (but far from sufficient) is, thus, that the problem in question can be formalized in some general language, e.g., monadic second order logic. The first

result of this type was obtained by Bodlaender et al. [10], namely linear and polynomial kernelizations for a wealth of problems when restricted to planar or bounded genus graphs. Important predecessors of this work are the linear kernelization for DOMINATING SET in planar graphs by Alber et al. [3] and a more general planar kernelization result, still using problem-specific rules, by Guo and Niedermeier [41].

Most meta kernelization results are based on the following intuition: The central notion is that of a *protrusion*, which refers to a subgraph (of the input graph) that is structurally simple and has a limited interaction with the rest of the graph. More concretely, a protrusion has a constant size *boundary* of vertices that are adjacent to the rest of the graph. Furthermore, it has bounded treewidth, which, for the considered problems, implies that we have an efficient dynamic programming routine to solve the problem on the protrusion subgraph (or any other graph of bounded treewidth). The outcome of this dynamic programming is a set of partial solutions relative to the boundary vertices alone. Intuitively, if the problem in question has a bounded number of partial solutions relative to any constant-size boundary, then many protrusions must give rise to the same partial solutions; this is, roughly, captured by the notion of the problem being *finite integer index*. Thus, if we can manage to compute a smaller protrusion with the same partial solutions then this can replace the original protrusion, shrinking the overall instance size. Thus, modulo a significant amount of technical heavy lifting (which we omit), this yields a *protrusion replacement rule* that can be used to replace large protrusions by smaller ones. Apart from this well-behaved interaction with dynamic programming it is required that YES- or NO-instances of the problem in question admit a small set of vertices whose deletion leaves a graph of bounded treewidth. (This holds trivially, for example, for VERTEX COVER(k) or for the FEEDBACK VERTEX SET(k) problem of deleting at most k vertices to obtain a forest.) This can be combined with the topological properties of the input graph class under consideration to prove that the graph can be decomposed into a small number of protrusions, the so-called *protrusion decomposition*.

Let us conclude this part by highlighting recent papers on meta kernelization: Kim et al. [51] recently extended the range of applicable sparse graph classes to classes excluding any fixed graph H as a *topological* minor. Gajarský et al. [36] extended this even further to the larger classes of graphs of bounded expansion, locally bounded expansion, and nowhere dense graphs. This, however, comes at the price that the kernelization bounds are no longer (implicitly) in terms of vertex-deletion distance to bounded treewidth, but instead by distance to bounded *treedepth* (which cannot be avoided [36]). Note also, that, unlike previous work where a low vertex-deletion distance to

bounded treewidth is a consequence of other problem properties, Gajarský et al. [36] directly consider the deletion distance to bounded treedepth as the parameter. Independently, Ganian et al. [37] also initiated a study of meta kernelization with respect to structural parameters. Their results apply to problems on *general graphs* and do not require finite integer index. Very recently, Garnero et al. [38] revisited the meta kernelization framework and initiated research into making the obtained kernelization results more explicit. At high level, this is achieved by working more closely on the intuitive connection between meta kernelization and dynamic programming. For an overview on earlier meta kernelization results and a more detailed explanation thereof we refer to the survey of Lokshtanov et al. [58].

Further new results. Last year, Wahlström [70] came up with an intriguing polynomial compression for the Steiner Cycle(k) problem of finding a cycle (of unbounded length) through a given set of k terminals in a graph. Crucially, the result makes use of the Tutte matrix (and randomization) and, while it obtains an equivalent instance of bounded size, it is not known whether this can be turned into a polynomial kernelization because the output language is not known to be in NP (the connection between compressions and kernelizations will be discussed later).

Fomin et al. [34] proved that Dominating Set(k) and Connected Dominating Set(k) admit linear kernels when restricted to input graphs excluding any fixed graph H as a topological minor. This continues a sequence of results [44, 65, 59, 63, 33, 34] on kernels for (Connected) Dominating Set(k) in restricted graph classes. Note that both problems are W[2]-hard on general graphs and thus do not even admit exponential kernels unless FPT = W[2].

A recent work of Kratsch et al. [54] settled the question of whether the so-called Point Line Cover(k) problem of covering a point set in the plane by at most k lines admits an efficient reduction to significantly less than $\mathcal{O}(k^2)$ *points*. (The reader is invited to rediscover a simple reduction to k^2 points that is in the spirit of the Vertex Cover(k) example.) Crucially, the result that no reduction to $\mathcal{O}(k^{2-\varepsilon})$ points is possible unless the polynomial hierarchy collapses used the full generality of Dell and van Melkebeek's [22] lower bound framework that applies also to oracle communication protocols. While we will discuss at length the existing lower bound techniques (see Section 4), a discussion of the latter is beyond the scope of this survey.

3 Formal definitions

Formally, a parameterized problem is any language $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$, where Σ is any finite alphabet and $\mathbb{N}$ denotes the non-negative integers. The second component k of any instance $(x, k) \in \Sigma^* \times \mathbb{N}$ is called the *parameter*. The problem $\mathcal{Q}$ is *fixed-parameter tractable* (FPT) if there is an algorithm A, a computable function $f\colon \mathbb{N} \to \mathbb{N}$, and a constant c such that A correctly decides $(x, k) \in \mathcal{Q}$ for all $(x, k) \in \Sigma^* \times \mathbb{N}$ in time $f(k) \cdot |x|^c$. We omit in this survey a detailed discussion of *fixed-parameter intractability*, e.g., regarding fpt-reductions and the W-hierarchy. It suffices to know that intractability is typically established by proving W[1]- or W[2]-hardness;[4] note that FPT $\subseteq$ W[1] $\subseteq$ W[2] and it is believed that the inclusions are strict.

A kernelization for a parameterized problem $\mathcal{Q}$ is a polynomial-time algorithm K that given any instance $(x, k) \in \Sigma^* \times \mathbb{N}$ returns an instance (x', k') such that $(x, k) \in \mathcal{Q}$ if and only if $(x', k') \in \mathcal{Q}$ and with $|x'|, k' \leq f(k)$ for some computable function $f\colon \mathbb{N} \to \mathbb{N}$. The function f is called the *size of the kernelization* K and K is a *polynomial (linear) kernelization* if $f(k)$ is polynomially (linearly) bounded in k. For simplicity, we allow a kernelization to outright answer YES or NO, understanding that it could instead return any hard-wired YES- or NO-instance of $\mathcal{Q}$ (of constant size). It is known that a parameterized problem is fixed-parameter tractable if and only if it is decidable and admits a kernelization (see Theorem 1 below).

In the literature there exist two relaxed variants of kernelization: A *generalized kernelization* (or *bikernel*) returns an output instance (x', k') that is with respect to a, possibly different, parameterized problem $\mathcal{Q}'$. More general, a *compression* may return an instance with respect to any (also unparameterized) language $L \subseteq \Sigma^*$. All kernelization lower bound tools in this survey, and almost all lower bounds in the literature, imply also the same lower bounds for compressions. We will see later (in Section 4) that lower bounds for compressions are slightly preferable, due to greater ease of transferring them by appropriate reductions.

Theorem 1. *A parameterized problem $\mathcal{Q}$ is fixed-parameter tractable if and only if it is decidable and has a kernelization.*

Proof. Assume that we have a kernelization for $\mathcal{Q}$ that reduces any instance (x, k) to an equivalent instance (x', k') of size at most $f(k)$. We can then apply an arbitrary algorithm for $\mathcal{Q}$ (guaranteed by decidability) to solve (x', k') and thereby also (x, k). If $g\colon \mathbb{N} \to \mathbb{N}$ bounds the runtime of the

[4] E.g., CLIQUE(k) is W[1]-complete and HITTING SET(k) is W[2]-complete.

assumed algorithm then the total time investment is $|x|^{\mathcal{O}(1)}$ for the kernelization plus $g(f(k))$ for the algorithm. This is bounded by $f'(k)|x|^{\mathcal{O}(1)}$ where $f'(k) := g(f(k))$, implying fixed-parameter tractability.

For the converse, assume that we have an algorithm that solves all instances (x, k) of $\mathcal{Q}$ in time $f(k)|x|^c$. Now run this assumed algorithm for $|x|^{c+1}$ steps. If it finishes then we have the correct YES or NO answer. Otherwise, it did not finish cause $f(k)|x|^c > |x|^{c+1}$. This, however, implies $|x| < f(k)$. Thus, either way, in polynomial time $\mathcal{O}(|x|^{c+1})$ we get an equivalent instance of size at most $f(k)$. □

Note that the kernelizations implied by this theorem are not very useful cause the size bound $f(k)$ is the same $f(k)$ as in the FPT runtime, which is usually exponential in k. Nevertheless, the existence of exponential kernelizations for many problems further motivates the question which of them also have polynomial kernelizations. Conversely, if a problem is W[1]-hard and thus not FPT unless FPT = W[1] then we also expect no kernelization.

4 Lower bounds for kernelization

The goal of this section is to explain the basic intuition underlying known techniques for lower bounds for kernelization and to give *one* complete set of tools for proving them. To this end, we will formally define so-called *cross-compositions* and *polynomial parameter transformations* as these appear very convenient to use. Cross-composition is a unifying front end to various insightful tools, and complexity theorists might prefer to directly employ these underlying results of, e.g., Dell and van Melkebeek [22] and Drucker [26].

At high level, there are two prevalent forms of kernelization lower bounds known so far: First, and dominantly, for a wealth of problems it has been shown that they admit no polynomial kernelization unless NP ⊆ coNP/poly. Second, for a smaller list of problems that do have polynomial kernels, it is known that no kernels of size $\mathcal{O}(k^{c-\varepsilon})$ are possible, where k is the parameter and c is some constant, unless NP ⊆ coNP/poly. The assumption that NP ⊈ coNP/poly (or, equivalently, coNP ⊈ NP/poly) is clearly stronger than P ≠ NP and NP ⊈ coNP but, since its failure would imply a collapse of the polynomial hierarchy [71, 14], it is still widely believed.

Intuition for ruling out polynomial kernels. Let us consider the NP-hard PATH(k) problem where we are given a graph $G = (V, E)$ and $k \in \mathbb{N}$ with the question of whether G contains a simple path on at least k vertices. If we combine t instances $(G_1, k), \ldots, (G_t, k)$ into a single one (G', k) by

letting G' be the disjoint union of the graphs G_i then, clearly, (G', k) is YES if and only if at least one (G_i, k) is YES. Intuitively, for t large but polynomial in k, a kernelization applied to (G', k) would have to determine some graphs G_i that are less likely to be YES and remove the corresponding components from G'. More concretely, if we assume a kernelization to size k^c and take $t = k^{c+1}$ then the output of the kernelization applied to G' has less than one bit per instance (G_i, k). On the other hand, the total input size is polynomial in the largest instance (G_i, k) and, hence, we do not expect that (in general) the time would suffice to solve any of the instances.

More generally, we do not expect an efficient algorithm that for $s \in \mathbb{N}$ takes t instances of any NP-hard problem, each of size at most s, and returns a single instance of size polynomial in s that is YES if at least one of the inputs is YES. Such an algorithm is called an OR-*distillation* in the breakthrough lower bound framework of Bodlaender et al. [9]; and they conjectured that no NP-hard problem admits an OR-distillation. The conjecture was proved shortly after by Fortnow and Santhanam [35] modulo the assumption that $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$. The analogous conjecture for the natural variant called AND-*distillation* was made as well, but it remained an open problem for five years until it was settled by an impressive work of Drucker [26]; amongst a wide range of results on both deterministic and probabilistic compression (in fact also for quantum compression) Drucker proved that the AND-distillation conjecture holds under $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$ as well.

The framework of Bodlaender et al. [9] introduced so-called OR- *and* AND-*composition* algorithms that, essentially, generalize the above example for PATH(k) to any efficient mapping (not just disjoint union and not just for graph problems) that encodes the OR or AND of t instances with parameter value k into a single instance of the same problem with parameter value k' polynomially bounded in k. I.e., given t instances the obtained instance is YES if and only if at *least one* respectively *all* given instances are YES. Similarly to the example, such a composition together with a polynomial kernelization gives an OR- or AND-distillation. Since proving existence of a particular algorithm (the composition) is typically easier than ruling out an algorithm (the polynomial kernelization) proving compositions became a very successful way of ruling out polynomial kernels. Curiously, even before Drucker's result [26], most lower bounds used OR-compositions and only very few proofs had to rely on the then unproven AND-distillation conjecture.

Cross-composition. We will now review an extension to the composition-based framework that was introduced by Bodlaender at al. [12]. In a so-called OR- resp. AND-cross-composition the input consists of instances of *any* NP-

hard problem, while the output is an instance of the target parameterized problem for which we desire a lower bound. Essentially, the parameter of the output instance must be polynomially bounded in the largest size among input instances, which often makes the proofs easier. In addition, there is the straightforward notion of a so-called *polynomial equivalence relation* that simplifies arguments for why inputs to a (cross-)composition may be assumed to be fairly similar (e.g., you may have wondered why we tacitly assumed that all PATH(k) inputs have the same parameter).

Despite these extensions to the composition-based framework [9, 35, 26] the underlying ideas go through in the same way. Nevertheless, several fairly ad-hoc tricks needed for compositions are no longer required for cross-compositions and this front end has seen wide adoption.

Definition 1 (polynomial equivalence relation [12])**.** An equivalence relation $\mathcal{R}$ on Σ^* is called a *polynomial equivalence relation* if the following two conditions hold:

1. There is an algorithm that given two strings $x, y \in \Sigma^*$ takes time polynomial in $|x| + |y|$ and decides whether x and y belong to the same equivalence class.

2. For any finite set $S \subseteq \Sigma^*$ the equivalence relation $\mathcal{R}$ partitions the elements of S into a number of classes that is polynomially bounded in the size of the largest element of S.

A simple example usage of a polynomial equivalence relation for PATH(k) instances (G_i, k_i) would be to declare instances (G_i, k_i) and (G_j, k_j) equivalent if $k_i = k_j$. (As a technical remark, if k is given in binary then this would formally allow an exponential number of equivalence classes. Thus, one usually resorts to a dummy class containing "ill-posed" or otherwise infeasible inputs. E.g., for PATH(k) we can make one class for all instances where k exceeds the number of vertices since these are trivially NO.)

Definition 2 (AND/OR-cross-composition [12])**.** Let $L \subseteq \Sigma^*$ be a language, let $\mathcal{R}$ be a polynomial equivalence relation on Σ^*, and let $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. An OR-*cross-composition of* L *into* $\mathcal{Q}$ (with respect to $\mathcal{R}$) is an algorithm that, given t instances $x_1, x_2, \ldots, x_t \in \Sigma^*$ of L belonging to the same equivalence class of $\mathcal{R}$, takes time polynomial in $\sum_{i=1}^{t} |x_i|$ and outputs an instance $(y, k) \in \Sigma^* \times \mathbb{N}$ such that:

"PB": The parameter value k is polynomially bounded in $\max_i |x_i| + \log t$.

"OR": The instance (y, k) is YES for $\mathcal{Q}$ if and only if *at least one* instance x_i is YES for L.

An AND-*cross-composition of* L *into* $\mathcal{Q}$ (with respect to $\mathcal{R}$) is an algorithm that, instead, fulfills Properties "PB" and "AND".

"AND": The instance (y, k) is YES for $\mathcal{Q}$ if and only if *all* instances x_i are YES for L.

We say that L OR-cross-composes, respectively AND-cross-composes, into $\mathcal{Q}$ if a cross-composition algorithm of the relevant type exists for a suitable relation $\mathcal{R}$.

Note that the use of a polynomial equivalence relation in the definition is, effectively, optional since $\mathcal{R} = \Sigma^* \times \Sigma^*$ is a valid choice and simply makes all inputs equivalent. The *intended use* of polynomial equivalence relations, however, is to group inputs for a cross-composition such that it need only be applied to groups of instances that are somewhat similar, thereby simplifying the necessary constructions and gadgets.

Similar to compositions, any AND- or OR-cross-composition combined with a polynomial kernelization creates an AND- or OR-distillation. Thus, using the results of Fortnow and Santhanam [35] and Drucker [26] we can use them to rule out polynomial kernelizations.

Theorem 2 ([12])**.** *If an* NP*-hard language* L AND/OR*-cross-composes into the parameterized problem* $\mathcal{Q}$*, then* $\mathcal{Q}$ *does not admit a polynomial kernelization or polynomial compression unless* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ *and the polynomial hierarchy collapses.*

Note that the theorem also rules out polynomial compressions, which relax polynomial kernelizations by allowing the output to be an instance (a string) with respect to *any language*; in the same way this holds also for lower bounds via AND- and OR-compositions. This simplifies transferring lower bounds via appropriate reductions (as we will see later).

An example for AND-cross-composition. We will now sketch an AND-cross-composition for the EDGE CLIQUE COVER(k) problem. The question about existence of a polynomial kernelization for EDGE CLIQUE COVER(k) was a frequently posed open problem (see, e.g., Guo and Niedermeier [40]) until being settled negatively by Cygan et al. [19].

EDGE CLIQUE COVER(k)
Input: A graph $G = (V, E)$ and $k \in \mathbb{N}$.
Parameter: k.
Question: Is there a collection of at most k cliques in G such that each edge is contained in at least one of them?

We give an AND-cross-composition from EDGE CLIQUE COVER to EDGE CLIQUE COVER(k) following in spirit the construction of Cygan et al. [19]. (Note that EDGE CLIQUE COVER has the same problem definition as EDGE CLIQUE COVER(k), including the value $k \in \mathbb{N}$, except for not specifying k as the parameter.) We begin by choosing a polynomial equivalence relation. We make one equivalence class for all instances that are trivially YES because k exceeds the number of edges. Among the rest, let any two instances (G_i, k_i) and (G_j, k_j) be equivalent if G_i and G_j have the same number of vertices and furthermore $k_i = k_j$. Finally, since we are careful theoreticians, we devote one class to all inputs that are not valid encodings of a graph and integer k (and which are thus NO instances). Of course, in the following it suffices to discuss the interesting case of inputs that are not trivially YES or NO.

Let t instances from the same (nontrivial) equivalence class be given, e.g., $(G_1, k), \ldots, (G_t, k)$. Let n be the number of vertices in each graph and, for convenience, assume that the vertices of each graph G_i are numbered arbitrarily, say $V_i = \{v_{i,1}, \ldots, v_{i,n}\}$.

The basic idea is to start with a disjoint union of the graphs and add all edges between different graphs (i.e., we take the join of the graphs). Then, if all instances are YES, we may combine the t times k cliques used for the graphs into k cliques that cover all edges in graphs G_i. Concretely, say that for $i \in \{1, \ldots, t\}$ the edges of G_i can be covered by cliques $C_{i,1}, \ldots, C_{i,k}$. Then for $j \in \{1, \ldots, k\}$ each set $\widehat{C}_j := \bigcup_i C_{i,j}$ induces a clique (using join edges), and together these k cliques cover all edges *inside* each graph G_i.

The caveat, however, is that the combination of the cliques does not necessarily cover *all* join edges that we introduced between different graphs G_i. We handle this situation by increasing the budget and forcing inclusion of additional $\mathcal{O}(n \log t)$ cliques that cover all join edges *but do not contain any edge in any graph* G_i. If we can ensure this, then the remaining budget of k will allow only k further cliques, like, e.g., $\widehat{C}_1, \ldots, \widehat{C}_k$, that must induce a k-clique cover in each graph G_i.

The idea is to add auxiliary vertices that will each be adjacent to exactly one vertex $v_{i,\ell}$ per graph G_i. To ensure that we cover all edges between any graphs G_i and G_j the exact choice for each auxiliary vertex depends on the binary expansion of i and j (using that different numbers differ in at least one position, but avoiding the use of $\mathcal{O}(t)$, or worse, many extra vertices/cliques).

We introduce auxiliary vertices $w_{a,b,p}$ for all $a, b \in \{1, \ldots, n\}$ and $p \in \{1, \ldots, \log t\}$. We connect a vertex $w_{a,b,p}$ to vertex $v_{i,a}$ of graph G_i if the pth bit in the binary expansion of i is even, and to $v_{i,b}$ otherwise (if the bit is odd). We call the obtained graph (of G_i's and auxiliary vertices) G' and let the budget be $k' := k + n^2 \cdot \log t$. Since we already excluded instances with k

exceeding the number of edges, which is less than n^2, the value k' is indeed polynomially bounded in the largest input instance plus $\log t$.

Let us briefly check that the obtained instance behaves as intended. Crucially, the auxiliary vertices form an independent set and none of them is isolated. Thus, we need to include at least one separate clique for each of them. Clearly, the closed neighborhood of any $w_{a,b,p}$ is a clique since all neighbors are adjacent by join edges. Thus, a single clique per $w_{a,b,p}$ is necessary and sufficient. For any join edge from, say, $v_{i,a}$ to $v_{j,b}$, we find that both vertices are contained in the neighborhood of $w_{a,b,p}$ or $w_{b,a,p}$ for all positions p where the binary expansions of i and j differ (the choice of $w_{a,b,p}$ or $w_{b,a,p}$ depends on the respective parities in position p). At this point, all join edges are covered and all edges inside graphs G_i still need to be covered by the remaining k cliques (which can be combined over all t graphs). Thus, the instance (G', k') correctly encodes the AND and by Theorem 2 this rules out polynomial kernels and compressions for Edge Clique Cover(k).

Polynomial parameter transformations. Before the framework of Bodlaender et al. [9] the question for lower bounds for kernelization was frequently posed as an open problem. It is surprising, in hindsight, that this never led to a reduction-based study of polynomial kernels akin to the collective evidence created by NP-complete problems. In contrast, shortly after the framework was published, it was recognized that compositions are by no means always as easy as for Path(k) and may sometimes be outright impossible.[5]

It was soon recognized that having a Karp reduction from one parameterized problem to another with the additional restriction that the output parameter is polynomially bounded in the input parameter essentially preserves kernelization properties (we will formalize this in a moment). This was first, implicitly, used by Binkele-Raible [6], first made formal by Bodlaender et al. [13], and first heavily used by Dom et al. [23]. We introduce these reductions under the widely adopted name of *polynomial parameter transformations*.

Definition 3 (polynomial parameter transformation)**.** Let $\mathcal{Q}, \mathcal{Q}' \subseteq \Sigma^* \times \mathbb{N}$ be parameterized problems. A *polynomial parameter transformation* (PPT) from $\mathcal{Q}$ to $\mathcal{Q}'$ is a polynomial-time computable mapping $\pi\colon \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}\colon (x,k) \mapsto (x',k')$ such that $(x,k) \in \mathcal{Q}$ if and only if $(x',k') \in \mathcal{Q}'$ and $k' \leq p(k)$ for all $(x,k) \in \Sigma^* \times \mathbb{N}$, where $p\colon \mathbb{N} \to \mathbb{N}$ is some fixed polynomial. If there is such a reduction from $\mathcal{Q}$ to $\mathcal{Q}'$ then we write $\mathcal{Q} \leq_{ppt} \mathcal{Q}'$.

[5]This problem was mainly with the original notion of compositions, where source and target problem needed to be the same.

If $\mathcal{Q} \leq_{ppt} \mathcal{Q}'$ and $\mathcal{Q}'$ has a polynomial kernelization (or compression) then we can take any instance (x, k) for $\mathcal{Q}$, compute an equivalent instance (x', k') of $\mathcal{Q}'$ with k' polynomially bounded in k, and then apply the kernelization/compression of $\mathcal{Q}'$. The obtained instance, say (x'', k'') of $\mathcal{Q}'$ is YES if and only if (x, k) is YES for $\mathcal{Q}$ and its size is polynomially bounded in k. Thus, the combined algorithm of PPT plus polynomial kernelization/compression constitutes a polynomial compression for $\mathcal{Q}$. This yields the following simple but useful lemma for proving lower bounds.

Lemma 1. *If $\mathcal{Q} \leq_{ppt} \mathcal{Q}'$ and $\mathcal{Q}$ admits no polynomial compression (possibly modulo some complexity assumption) then $\mathcal{Q}'$ admits no polynomial kernel or compression (under the same assumption).*

Note that to combine a PPT from $\mathcal{Q}$ to $\mathcal{Q}'$ and a polynomial kernelization for $\mathcal{Q}'$ into a polynomial *kernelization* for $\mathcal{Q}$ we still need to convert the output, which is a *poly*(k)-sized instance for $\mathcal{Q}'$, into an instance for $\mathcal{Q}$ without blowing up size and parameter more than polynomially. If $\mathcal{Q}$ is NP-hard and $\mathcal{Q}' \in \mathsf{NP}$ then we can use the implied Karp reduction from $\mathcal{Q}'$ to $\mathcal{Q}$; a technicality, however, is that we need NP-hardness of $\mathcal{Q}$ for polynomially bounded value of its parameter (or, equivalently, with parameter value encoded in unary) to ensure that there is a Karp reduction that also implies a polynomial bound for the parameter (see Bodlaender et al. [13]).

We will make further use of PPTs in Section 6. Let us anyway copy a nice example from [58]: In the 2-PATH(k) problem, given (G, k) we need to find two vertex-disjoint simple paths of length k each. The disjoint union composition fails, since we might have two input graphs *with only one k-path each*. There is, however, a simple PPT from PATH(k) to 2-PATH(k): Given a PATH(k) instance (G, k), simply return (G', k) where G' is obtained from the disjoint union of G and a k-path. Clearly, G has a k-path if and only if G' contains two vertex-disjoint k-paths.

Let us add to the example that there is also a simple OR-cross-composition from PATH(k) to 2-PATH(k), either by disjoint union with *two copies* of each input graph or by similarly adding one additional disjoint k-path.

Polynomial lower bounds for kernelization. So far we have discussed how to rule out polynomial kernels for certain parameterized problems. An insightful work of Dell and van Melkebeek [22] was the first to open up the possibility of proving polynomial lower bounds for problems that do admit some polynomial kernelization. E.g., they showed that d-HITTING SET(k) admits no kernelization to size $\mathcal{O}(k^{d-\varepsilon})$ for any fixed $\varepsilon > 0$ unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. In fact, their bounds are more general and apply also to

compressions and, interestingly, to a form of oracle communication protocol. For reasons of space (and focus) we restrict ourselves to the goal of discussing polynomial lower bounds, but strongly suggest a follow-up reading of [22].

The key step for getting to polynomial lower bounds was a closer inspection of Fortnow and Santhanam's [35] proof of the OR-distillation conjecture [22]. This revealed that, roughly speaking, an efficient algorithm that encodes the OR of any t instances for L into an equivalent instance of L' of length $\mathcal{O}(t \log t)$ implies $L \in \mathsf{coNP/poly}$. More concretely, we need such an algorithm that works when given $t := t(n)$ instances of size at most n each for any value of n, where t is any polynomially bounded function. A similar statement follows for encoding the AND of t instances of L (see Theorem 4) as one of many consequences of Drucker's work [26].

To sketch how this gives polynomial lower bounds let us first see how it works for ruling out all polynomial kernels. If we have an OR-cross-composition of some L into a parameterized problem that yields parameter $k \in \mathcal{O}(n^c)$ then applying *any polynomial kernelization* yields a total size of $\mathcal{O}(k^d) \subseteq \mathcal{O}(n^{cd})$. If we apply the combined algorithm to $t = n^{cd}$ instances then this makes the total size $\mathcal{O}(n^{cd}) \subseteq \mathcal{O}(t)$. Hence, for any assumed polynomial kernelization we can choose $t\colon \mathbb{N} \to \mathbb{N}$ such that we get "OR of t instances into $\mathcal{O}(t)$ bits", implying $L \in \mathsf{coNP/poly}$.

Now, assume instead that we can encode the OR of t instances of L of size n each into one instance with parameter $k \in \mathcal{O}(t^{1/2}n^c)$. Using *any kernelization with size guarantee* $\mathcal{O}(k^{2-\varepsilon})$ would now give total size $\mathcal{O}(t^{1-\varepsilon'}n^{c'})$. This again, for an appropriate function $t\colon \mathbb{N} \to \mathbb{N}$, suffices to get "OR of t instances into $\mathcal{O}(t)$ bits" and, hence, $L \in \mathsf{coNP/poly}$.

We will next define an extension of AND/OR-cross-composition that allows for such larger contributions of the number t of instances in the parameter obtained by the compositions. Again, this is a front end to very insightful works [22, 26], and, hopefully, motivates more applications of their results.

Definition 4 (AND/OR-cross-composition of bounded cost [12])**.** An AND/OR-*cross-composition of L into $\mathcal{Q}$* (with respect to $\mathcal{R}$) *of cost $f(t)$ for t instances* is an AND/OR-cross-composition algorithm as described in Definition 2 that satisfies "CB" instead of "PB".

"CB": The parameter k is bounded by $\mathcal{O}(f(t) \cdot (\max_i |x_i|)^c)$, where c is some constant independent of t.

The following theorem formalizes the intuition of how the dependence on t in an AND/OR-cross-composition relates to polynomial lower bounds.

Theorem 3 ([12]). *Let $L \subseteq \Sigma^*$ be a language, let $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem, and let d, ε be positive reals. If L has an* AND/OR*-cross-composition into $\mathcal{Q}$ with cost $f(t) = t^{1/d+o(1)}$, where t denotes the number of instances, and $\mathcal{Q}$ has a polynomial compression into an arbitrary language L' with size bound $\mathcal{O}(k^{d-\varepsilon})$, then $L \in \mathsf{coNP/poly}$. If, additionally, L is NP-hard, then $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.*

The statement for OR-cross-composition was proved in [12] building on [22]. The analogous proof for AND-cross-compositions is given here for the first time. Modulo swapping of AND and OR and avoiding the use of the oracle communication protocol this proof is fully analogous to the OR-cross-composition case. Crucially, however, the proof depends on having a proven consequence of encoding the AND of t instances of any L into $\mathcal{O}(t \log t)$ bits, which follows as a consequence of a more powerful result of Drucker [26, 27].[6]

Theorem 4 (Consequence of [27, Theorem 7.1]). *Let L, L' be any languages, let $d > 0$, and let $t \colon \mathbb{N} \to \mathbb{N}$ be polynomially bounded. Suppose that there exists a polynomial-time mapping that on input of $t := t(n)$ instances $x_1, \ldots, x_t$ for L each of size n computes a single instance x of size at most $d \cdot t \log t$ such that $x \in L'$ if and only if $x_i \in L$ for all i. Then $L \in \mathsf{coNP/poly}$.*

Proof. This follows as an application of the more general [27, Theorem 7.1]. First, we need to swap the role of AND and OR by complementation to match [27, Theorem 7.1]: Assume a mapping that given $x_1, \ldots, x_t$ returns x with $x \in L'$ if and only if $x_i \in L$ for all i. If we consider $\overline{L}$ and $\overline{L'}$ instead then we get $x \in \overline{L'}$ if and only $x_i \in \overline{L}$ for at least one i. Once we have chosen all other parameters we can thus apply our mapping as an OR for $\overline{L}$ in [27, Theorem 7.1] which implies $\overline{L} \in \mathsf{NP/poly}$ and $L \in \mathsf{coNP/poly}$.

We use the following choices for $t_1(n)$, $t_2(n)$, $\widehat{\delta}$, and $\xi(n)$: We have an error-free mapping and, thus, use error bound $\xi(n) = 0$. We set $t_1(n) := t(n)$ and $t_2(n) := d \cdot t(n) \log t(n)$. Using the definition of $\widehat{\delta}$ in [27, Theorem 7.1], this yields $\widehat{\delta} \leq 1 - \frac{1}{8}(t(n))^{-d}$. Since t is polynomially bounded, there are constants a, b such that $t(n) \leq a \cdot n^b$ for sufficiently large n. Our parameters fulfill the requirement of $1 - 2\xi(n) - \widehat{\delta} \geq \frac{1}{n^c}$ in [27, Theorem 7.1] for $c = bd+1$:

$$1 - 2\xi(n) - \widehat{\delta} \geq \frac{1}{8 \cdot (t(n))^d} \geq \frac{1}{8 \cdot a \cdot n^{bd}} \geq \frac{1}{n^c},$$

for sufficiently large n. □

[6] The author is indebted to Andrew Drucker for clarifying how this follows from his work.

Now we can explain the proof of Theorem 3. It follows the basic intuition given earlier and is analogous to the OR case in Bodlaender et al. [12].

Proof of Theorem 3 for AND*-cross-compositions.* Let $\mathcal{R}$ denote a polynomial equivalence relation on Σ^* which partitions any set of strings of length at most s into at most $\mathcal{O}(s^b)$ equivalence classes. Let $f(t) = t^{1/d+o(1)}$ for some constant d. Let C be an AND-cross-composition from L into $\mathcal{Q}$, which maps t instances of size at most s and from the same $\mathcal{R}$-equivalence class to an output instance with parameter value bounded by $\mathcal{O}(f(t)s^c)$. Finally, let K be a polynomial compression for $\mathcal{Q}$ into some language L' that given an instance with parameter k outputs an equivalent string (with respect to L') of size bounded by $h(k) = \mathcal{O}(k^{d-\varepsilon})$.

We define a polynomially bounded function t by $t(s) := s^{(b+cd)\cdot\frac{d}{\varepsilon}}$. By Theorem 4 it suffices to provide an appropriate encoding of the AND of t instances of L. As the target language we will use $\text{AND}(L') := \{(x_1, \ldots, x_r) \mid r \in \mathbb{N} \wedge x_1, \ldots, x_r \in L'\}$. Fixing s and $t := t(s)$, let t instances $x_1, \ldots, x_t$ of L each of length at most s be given.

As a first step, we partition the strings x_i according to equivalence under $\mathcal{R}$, obtaining $r \leq \mathcal{O}(s^b)$ groups. Then we apply the AND-cross-composition C to each group, obtaining r instances $(y_1, k_1), \ldots, (y_r, k_r)$. The parameter values k_i are bounded by $\mathcal{O}(f(t)s^c)$. Now we apply the assumed polynomial compression K to each instance (y_i, k_i), obtaining instances $z_1, \ldots, z_r$ of the language L'. We return the instance $(z_1, \ldots, z_r)$.

Each compressed instance z_i has size at most

$$h(k_i) = \mathcal{O}((k_i)^{d-\varepsilon}) = \mathcal{O}((f(t)s^c)^{d-\varepsilon}).$$

Thus we can bound the output size, i.e., the size of $(z_1, \ldots, z_r)$, as follows:

$$\mathcal{O}\left(r\left(f(t)s^c\right)^{d-\varepsilon}\right) = \mathcal{O}\left(s^b\left(t^{\frac{1}{d}+o(1)}s^c\right)^{d-\varepsilon}\right) = \mathcal{O}\left(s^{b+c(d-\varepsilon)}t^{1-\frac{\varepsilon}{d}+o(1)}\right) = \mathcal{O}(t),$$

using that $r \leq \mathcal{O}(s^b)$ and the following bound for $s^{b+c(d-\varepsilon)}$:

$$s^{b+c(d-\varepsilon)} = s^{b+cd} \cdot s^{-c\varepsilon} = t^{\frac{\varepsilon}{d}} \cdot s^{-c\varepsilon} = t^{\frac{\varepsilon}{d}-\delta},$$

where $\delta = \frac{c\varepsilon^2}{(b+cd)d} > 0$. (Note that $t^{1-\delta+o(1)} = \mathcal{O}(t)$, for any $\delta > 0$.)

Correctness. It remains to show that the returned instance $(z_1, \ldots, z_r)$ is indeed an encoding of the AND of the instances $x_1, \ldots, x_t$. Assume first that at least one input instance x_i is a NO-instance (requiring the output to be NO for $\text{AND}(L')$). It follows that the corresponding instance (y_j, k_j) that is

created by C from all instances $\mathcal{R}$-equivalent to x_i must be NO for $\mathcal{Q}$. Accordingly, the polynomial compression K transforms (y_j, k_j) to a NO-instance z_j for the language L'. Hence, the output instance $(z_1, \ldots, z_r)$ is NO for AND(L').

In the remaining case all input instances $x_1, \ldots, x_t$ are YES for L. The AND-cross-composition C will therefore create r YES-instances (y_i, k_i) for $\mathcal{Q}$. These are converted to r YES-instances z_i for L'. Hence, the returned instance $(z_1, \ldots, z_r)$ is YES for AND(L'). Thus, we get a polynomial-time mapping fulfilling the requirement of Theorem 4. It follows that $L \in \mathsf{coNP/poly}$, as claimed. If L is NP-hard then $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. □

To conclude the section on lower bounds for kernelization, let us illustrate a successful "design-paradigm" for proving polynomial lower bounds that has been identified through results of Dell and van Melkebeek [22] and Dell and Marx [21]. The idea is to use a source problem that is d-partite in a sense. More strongly, similar to, for example, problems on bipartite graphs, all the relevant information needs to be encoded in the adjacency (or other structure) between the partite sets; the partite sets themselves should be isomorphic over all input instances (here polynomial equivalence relations can be of help). Thus, one can tightly encode t instances of a bipartite problem by using only $\sqrt{t}$ copies each of both partite sets and choosing a different pair for each instance. Let us perhaps make this more concrete in the following example.

Example of a polynomial lower bound. As an illustration let us sketch an $\mathcal{O}(n^{d-\varepsilon})$ lower bound for the d-Hitting Set(n) problem for any fixed $d \geq 3$. We give an OR-cross-composition from Hitting Set restricted to d-partite d-uniform hypergraphs, which is NP-hard for $d \geq 3$ (cf. [42]). In that problem we have a given partition of the ground set U into d color classes, say $U = C_1 \cup \ldots \cup C_d$ with each hyperedge containing exactly one vertex from each set C_i, and the task is to find k elements of U that intersect all edges (if possible).

Let t instances $(U_i, \mathcal{F}_i, k)$ of Hitting Set on d-partite d-uniform hypergraphs be given. For simplicity, skipping over padding arguments and choice of polynomial equivalence relation, assume that the ground set U_i of each instance is partitioned into d color classes, each containing exactly n vertices. As a first step, rename the instances from $i \in \{1, \ldots, t\}$ to $\mathbf{i} \in \{(i_1, \ldots, i_d) \mid i_j \in \{1, \ldots, t^{1/d}\}$; a simple counting argument shows that this allows an injective renaming.

Now, rather than taking simply the disjoint union of the instances we carefully identify the color classes of different instances. Concretely, for $p \in \{1, \ldots, d\}$ and $q \in \{1, \ldots, t^{1/d}\}$ identify, vertex by vertex, the pth color class

of all instances with number $\mathbf{i} = (i_1, \ldots, i_d)$ with $i_p = q$. In this way, for each color $p \in \{1, \ldots, d\}$ we end up with $t^{1/d}$ color classes (each with n vertices) that are shared by several instances. Let $C_{p,q}$ for $p \in \{1, \ldots, d\}$ and $q \in \{1, \ldots, t^{1/d}\}$ denote the obtained color classes.

Now, for all colors p and any two vertices u and v in different color classes $C_{p,q}$ (i.e., with different values of q) we add a new edge $\{u, v\}$. Thus, any hitting set for the instance has to *completely contain all but one color class $C_{p,q}$ for each color p*. Let us see what happens if, taking this into account, we ask for a hitting set of total size at most $k' = d(t^{1/d} - 1)n + k$ for the combined instance of d-Hitting Set(n).

As just observed any k'-hitting set, say S, must contain all but one color class $C_{p,q}$ for each color p. Let $q_1, \ldots, q_d \in \{1, \ldots, t^{1/d}\}$ such that $C_{p,q_p} \not\subseteq S$ for all p, i.e., each q_i corresponds to the color class that is not fully contained in S. Since $|S| \leq k'$ we find that the intersection of S with $C_{1,q_1} \cup \ldots \cup C_{d,q_d}$ is of size at most k; let S' denote the intersection. It follows that S' is a k-hitting set for *all edges that are fully contained in* $C_{1,q_1} \cup \ldots \cup C_{d,q_d}$. Note that, during our identification process, all color classes of instance $\mathbf{i}$ with $\mathbf{i} = (q_1, \ldots, q_d)$ have been identified with $C_{1,q_1}, \ldots, C_{d,q_d}$ and all its hyperedges are, therefore, contained in $C_{1,q_1} \cup \ldots \cup C_{d,q_d}$. Thus, S' is a k-hitting set for instance $\mathbf{i}$, proving that at least one input is YES.

For the converse, if some instance $(U_i, \mathcal{F}_i, k)$ is YES then begin by letting S' a k-hitting set for that instance. Let $\mathbf{i} = (i_1, \ldots, i_d)$ be the assigned renaming of i. Now, let S contain S' as well as all color classes $C_{p,q}$ with $q \neq i_p$, i.e., all color classes not used for instance $\mathbf{i}$. Clearly, this covers all additional edges between color classes $C_{p,q}$ and $C_{p,q'}$ with $q \neq q'$. Furthermore, for every instance $\mathbf{i}' = (i'_1, \ldots, i'_r) \neq (i_1, \ldots, i_r) = \mathbf{i}$ at least one position must differ, e.g., $i'_p \neq i_p$. But then S already includes all vertices of C_{p,i'_p} covering all hyperedges of instance $\mathbf{i}'$. Thus, the constructed instance is YES.

To wrap up, note that the combined instance has exactly $n' = d \cdot t^{1/d} \cdot n$ vertices, which is bounded by $t^{1/d}$ times a polynomial in the largest instance size. Thus, we have an OR-cross-composition with cost $t^{1/d}$ implying that d-Hitting Set(n) has no kernelization with size $\mathcal{O}(n^{d-\varepsilon})$ for any $\varepsilon > 0$ unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. As in [22] the analogous bound for d-Hitting Set(k) follows immediately by noting that all nontrivial instances have $k \leq n$.

Further reading. We point out some more results regarding polynomial lower bounds for concrete problems since, unlike ruling out polynomial kernels altogether, this is not yet in common use. Independently from Dell and Marx [21], Hermelin and Wu [47] formalized a form of composition algorithms with larger dependence on the number t of composed instances,

which they called *weak compositions*. Both papers prove polynomial lower bounds for several standard problems when restricted to families of sets of bounded size or graphs of bounded degree, respectively. A recent work of Cygan et al. [18] obtains kernelization lower bounds for several problems when restricted to graphs of bounded degeneracy that almost exactly match known upper bounds. Jansen [48] used the polynomial lower bound framework to rule out sparsification for computing the treewidth of a graph by proving that the problem admits no polynomial compression to size $\mathcal{O}(n^{2-\varepsilon})$, which would, for example, be implied by any nontrivial reduction to the number of edges. Generally, also the initial results of Dell and van Melkebeek [22] had sparsification lower bounds as one of their goals.

5 Representative sets and matroids

In this section we give an introduction to using representative sets and matroids for kernelization. As a warm-up, we will begin by introducing representative sets for set families and using them to reproduce two "classic" kernelization results, namely polynomial kernels for d-Hitting Set(k) and d-Set Packing(k). (See below for problem definitions.) It is known that kernels for these two problems can also be obtained via the Sunflower Lemma of Erdős and Rado [28]; see, e.g., [30, 21]. The best known kernelizations for both problems are due to Abu-Khzam [2, 1], with a slightly smaller ground set of $\mathcal{O}(k^{d-1})$ but same asymptotic total size of $\mathcal{O}(k^d \log k)$. It is known, by work of Dell and van Melkebeek [22] and Dell and Marx [21], that neither result can be improved to size $\mathcal{O}(k^{d-\varepsilon})$ unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

In the second part we move on to using representative sets on families of independent sets of a given matroid. A 1977 result of Lovász [60] states that such sets, of modest size, exist for every linear matroid, i.e., for every matroid that can be represented as the column matroid of a matrix. Marx [62] observed that Lovász' proof in fact also gives rise to an efficient algorithm. Since then, representative sets, both for set families (or, equivalently, uniform matroids) but also for gammoids and graphic matroids, have found various applications in parameterized complexity for kernelization [57] and faster algorithms [31]. In particular, Fomin et al. [31] also gave faster algorithms for finding representative sets for both linear matroids and the special case of uniform matroids. To illustrate the use for kernelization, we will give a fairly detailed description of the polynomial kernelization for Deletable Terminal Multiway Cut(k) obtained in [57].

Representative sets for set families. Let us jump right in and give a definition of q-representativeness for the case of set families.

Definition 5 (q-representative set family)**.** Let $\mathcal{A}$ be a family sets and let $q \in \mathbb{N}$. A subset $\mathcal{A}' \subseteq \mathcal{A}$ is *q-representative for $\mathcal{A}$* if for every set B of size at most q there is a set $A \in \mathcal{A}$ with $A \cap B = \emptyset$ if and only if there is a set $A' \in \mathcal{A}'$ with $A' \cap B = \emptyset$.

We will later give a similar definition for representative independent sets in a specified matroid (see Definition 6) that additionally requires $A \cup B$ and $A' \cup B$ to be independent sets of the matroid. The present definition can then be seen as a special case by using so-called uniform matroids where all sets up to some prescribed size are independent, but this is not at all required for understanding. Nevertheless, the general efficient algorithm of Lovász [60] and Marx [62] (see also Theorem 5 below) implies the following lemma.

Lemma 2. *Let $\mathcal{A}$ be a family of sets of size p each and let $q \in \mathbb{N}$. In time polynomial in $\binom{p+q}{p} + |\mathcal{A}|$ one can compute a q-representative subset $\mathcal{A}' \subseteq \mathcal{A}$ of size at most $\binom{p+q}{p}$.*

While the guaranteed size bound of $\binom{p+q}{p}$ might seem somewhat arbitrary at first, it is in fact tight: Consider the family $\mathcal{A}$ containing all $\binom{p+q}{p}$ subsets of size p of the set $\{1, \ldots, p+q\}$. Then, going over all sets B that are size q subsets of $\{1, \ldots, p+q\}$, we always find a unique set $A \in \mathcal{A}$ that is disjoint from B, namely $A = \{1, \ldots, p+q\} \setminus B$. Thus, all sets in $\mathcal{A}$ must be included and the lemma is tight. We will later make more use of the implicit observation that sets A that are unique "partners" for some set B must be included in any q-representative subset.

Let us now see that even this simple form of using representative sets, i.e., without the full power of specialized matroids, already suffices to reproduce "classic" kernelization results. We begin with the d-Hitting Set(k) problem, defined as follows.

> d-Hitting Set(k)
> **Input:** A universe U, a family $\mathcal{A}$ of subsets of U each of size at most d, and $k \in \mathbb{N}$.
> **Parameter:** k.
> **Question:** Is there a set of at most k elements of U that intersects all sets in $\mathcal{A}$?

We sketch a kernelization; let an instance $(U, \mathcal{A}, k)$ be given. Using Lemma 2 with $p = d$ and $q = k$ compute a k-representative subset $\mathcal{A}' \subseteq \mathcal{A}$ of

size at most $\binom{k+d}{d} \in \mathcal{O}(k^d)$. If $(U, \mathcal{A}, k)$ is YES then also $(U, \mathcal{A}', k)$ must be YES since $\mathcal{A}' \subseteq \mathcal{A}$. If, however, $(U, \mathcal{A}, k)$ is NO then, in particular, no set $B \subseteq U$ of size at most k can be a solution for $(U, \mathcal{A}, k)$. In other words, for each such set B there is at least one set $A \in \mathcal{A}$ that avoids B, i.e., $A \cap B = \emptyset$. Since $\mathcal{A}'$ is k-representative for $\mathcal{A}$, for each choice of B we also find a set $A' \in \mathcal{A}'$ with $A' \cap B = \emptyset$, implying that $(U, \mathcal{A}', k)$ is NO, too.

We remark that the reduction to $|\mathcal{A}'| \in \mathcal{O}(k^d)$ allows an encoding in $\mathcal{O}(k^d \log d)$ bits, which is essentially optimal due to the mentioned result of Dell and van Melkebeek [22] that rules out efficient reduction to bit size $\mathcal{O}(k^{d-\varepsilon})$ unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. It is possible, however, to improve the size of the ground set to $\mathcal{O}(k^{d-1})$, rather than the implicit $\mathcal{O}(d \cdot k^d) = \mathcal{O}(k^d)$, using the kernelization of Abu-Khzam [2]. (It is an interesting problem to close the wide gap between this result and the trivial lower bound of $\Omega(k)$ for the ground set size.)

Let us now consider d-Set Packing(k) where the argument is slightly more involved, though certainly comparable to the less obvious application of the Sunflower Lemma as compared to d-Hitting Set(k) (cf. [21]).

> d-Set Packing(k)
> **Input:** A universe U, a family $\mathcal{A}$ of subsets of U each of size at most d, and $k \in \mathbb{N}$.
> **Parameter:** k.
> **Question:** Is there a selection of k sets in $\mathcal{A}$ that are pairwise disjoint?

Again, representative sets can be used to obtain a polynomial kernelization whose size is essentially optimal. This time, given an instance $(U, \mathcal{A}, k)$ of d-Set Packing(k) we compute a $d(k-1)$-representative subset $\mathcal{A}'$ of $\mathcal{A}$. Let us see that this works correctly. Clearly, if $(U, \mathcal{A}, k)$ was NO in the first place then the obtained instance $(U, \mathcal{A}', k)$ will be NO too. Assume now that $(U, \mathcal{A}, k)$ is YES. Let $A_1, \ldots, A_k \in \mathcal{A}$ be a selection of k pairwise disjoint sets such that as many sets A_i as possible are also contained in $\mathcal{A}'$. If $A_1, \ldots, A_k \in \mathcal{A}'$ then we are done, so assume w.l.o.g. that $A_1 \notin \mathcal{A}'$. Then, letting $B := A_2 \cup \ldots \cup A_k$ we note that $A_1 \cap B = \emptyset$ and that $|B| \leq d(k-1)$. It follows, since $\mathcal{A}'$ is $d(k-1)$-representative for $\mathcal{A}$, that there exists $A_1' \in \mathcal{A}'$ with $A_1' \cap B = \emptyset$. Then, however, we immediately see that $A_1' \in \mathcal{A}$ and $A_1', A_2, \ldots, A_k$ is also a selection of k pairwise disjoint sets but with more sets also contained in $\mathcal{A}'$; a contradiction. Thus, we must have $A_1, \ldots, A_k \in \mathcal{A}'$, and, therefore, the obtained instance $(U, \mathcal{A}', k)$ is indeed equivalent to $(U, \mathcal{A}, k)$.

Representative sets for matroids. We will now introduce representative sets for families of independent sets of a given matroid. Since all further known kernelizations via representative sets [57] make use of a particular type of matroid called *gammoid* we will mainly focus on those. Let us recall that a matroid $M = (U, \mathcal{I})$ consists of a finite set U and a family $\mathcal{I}$ of subsets of U, called *independent sets*, fulfilling the following properties:

1. $\emptyset \in \mathcal{I}$.
2. If $X \subseteq Y$ and $Y \in \mathcal{I}$ then also $X \in \mathcal{I}$.
3. If $X, Y \in \mathcal{I}$ with $|X| < |Y|$ then there exists $y \in Y \setminus X$ such that $X \cup \{y\} \in \mathcal{I}$.

We can now give the full definition of q-representative sets for families of independent sets in a matroid. For ease of writing, let us say that an *independent set A extends an independent set B* if $A \cap B = \emptyset$ and $A \cup B$ is independent. Note that independence of $A \cup B$ requires independence of both A and B due to the second matroid property.

Definition 6 (q-representativeness for families of independent sets)**.** Let $M = (U, \mathcal{I})$ be a matroid. Let $\mathcal{A} \subseteq \mathcal{I}$ be a collection of independent sets of M and let $q \in \mathbb{N}$. We call a set $\mathcal{A}' \subseteq \mathcal{A}$ *q-representative for $\mathcal{A}$* if for every independent set B of size at most q there is an $A \in \mathcal{A}$ that extends B if and only if there is also an $A' \in \mathcal{A}'$ that extends B.

It should not come as a surprise that with the addition of matroid independence this opens up a much bigger world of applications. The, so far, most interesting matroids regarding kernelization applications are the gammoids (defined below). Their independence notion is strongly related to Menger's Theorem, and the proof that they are indeed matroids is due to Perfect [64].

Let $G = (V, E)$ be a graph that may have both directed and undirected edges, and let $S \subseteq V$. Say that a set $T \subseteq V$ is *linked to S* if there exist $|T|$ vertex-disjoint paths from S to T, i.e., each vertex in T is endpoint of a different path from S. Then the set system $M = (V, \mathcal{I})$ where $\mathcal{I}$ contains all sets T that are linked to S is a matroid. We say that M is the *gammoid on G with sources S*. (We note that often the roles of S and T are switched, which makes no difference regarding what matroids are gammoids. Furthermore, restricting $\mathcal{I}$ to any subset $V' \subseteq V$ still yields a gammoid, and the case of $V' = V$ is also called a *strict gammoid*.)

It is known that every gammoid can be represented as the (linear) independence of column vectors of a matrix, making them *linear matroids* (cf. [62]). The construction of the matrix over an appropriately large field

can be made constructive by an efficient, randomized algorithm but it is a big open problem whether a deterministic construction exists. For simplicity, we hide these details in the following theorem, noting that the general version [60, 62] holds for any linear matroid when given the matrix representation (and without further use of randomization).

Theorem 5 (simplified version of result by Lovász [60] and Marx [62])**.** *Let M be a gammoid and let $\mathcal{A} = \{A_1, \ldots, A_m\}$ be a collection of independent sets, each of size p. We can find in randomized polynomial time a set $\mathcal{A}' \subseteq \mathcal{A}$ of size at most $\binom{p+q}{p}$ that is q-representative for $\mathcal{A}$.*

A highly useful property of representative sets is that they can be employed for actually finding particular objects (e.g., vertices) rather than just "blindly" discarding sets (or other objects) as we did for d-Hitting Set(k) and d-Set Packing(k). For $\mathcal{A}'$ to be q-representative for $\mathcal{A}$ it is required that *every set* B that can be extended by some $A \in \mathcal{A}$ can also be extended by some $A' \in \mathcal{A}'$. This entails, however, that if a given $A \in \mathcal{A}$ is *unique in extending some given* B then this enforces that $A \in \mathcal{A}'$; else, no set in $\mathcal{A}'$ could extend B. We will return to this trick soon.

Example application. Let us now discuss an application of Theorem 5, namely a polynomial kernelization for the following variant of Multiway Cut(k), called Deletable Terminal Multiway Cut(k):

> Deletable Terminal Multiway Cut(k)
> **Input:** A graph $G = (V, E)$, a set of terminals $S \subseteq V$, and $k \in \mathbb{N}$.
> **Parameter:** k.
> **Question:** Is there a set X of at most k vertices such that in $G - X$ no two terminals $t_1, t_2 \in S \setminus X$ are in the same connected component?

The problem can be easily seen to be NP-hard, since using terminal set $S = V$ requires finding a vertex cover of size at most k. Note also, that all instances with $|S| \leq k + 1$ are trivial since this would allow deletion of all but one terminal. Finally, unlike Multiway Cut, which is hard already for three terminals, for any *fixed size* of S we have a trivial solution if $k \geq |S| - 1$ or else can enumerate and test all $\mathcal{O}(|V|^k) \subseteq \mathcal{O}(|V|^{|S|-1})$ solution candidates in polynomial time.

The kernelization proceeds as follows: (1) We show that if an instance is yes then there is always a solution X that allows a certain path packing from S to X. (2) We set up a gammoid based on a graph G' derived from G, and

with sources S. (3) We use Theorem 5 to find a superset of X of size $\mathcal{O}(k^3)$, using the path packing to distinguish vertices in V. (4) We briefly explain how to use this superset to shrink the input graph G to $\mathcal{O}(k^3)$ vertices.

Analyzing solutions. Let an instance (G, S, k) of DELETABLE TERMINAL MULTIWAY CUT(k) be given. Assume that the instance is YES and, for analysis, let X denote a solution for (G, S, k) that contains the maximum number of terminals from S (among solutions of size at most k). Clearly, vertices in $X \cap S$ correspond to outright deletions of terminals, whereas $X_0 := X \setminus S$ separates the remaining terminals $S_0 := S \setminus X$ from one another. We want to establish that X_0 is linked to S_0 in a strong sense, by using Hall's theorem.

Note that each connected component of $G - X$ contains at most one terminal from S; for brevity, we will call C containing a terminal from $S_0 = S \setminus X$ a *terminal component*. Let us say that a vertex $x \in X_0$ *sees a terminal component* C if in G the vertex x is adjacent to a vertex of C. We extend this to sets $Y \subseteq X_0$ by saying that Y sees a terminal component C if at least one $x \in Y$ sees C. Intuitively, if a vertex of X_0 sees some terminal components, then "putting that vertex back" into $G - X$ reconnects those components and terminals; ditto for $Y \subseteq X_0$.

We set up for using Hall's Theorem: Assume that any nonempty set $Y \subseteq X_0$ sees at most $|Y| + 1$ terminal components. It follows that in $G - (X \setminus Y)$ the set Y together with these terminal components (and possibly terminal-free components) forms a larger component with up to $|Y| + 1$ terminals. All other terminal components not seen by Y are unaffected. Observe that this allows an alternative solution by deleting any $|Y|$ of the $|Y| + 1$ terminals, say a set $Y' \subseteq S_0$. This, however, contradicts our choice of X since $(X \setminus Y) \cup Y'$ would be a solution with larger intersection with S. Thus, every $Y \subseteq X_0$ sees at least $|Y| + 2$ terminal components C.

Using Hall's Theorem it can now be checked that we can find a matching of $|X_0| + 2$ terminal components to vertices in X_0 such that:

- Each component is matched to a vertex $x \in X_0$ that sees it.
- For any fixed vertex $x \in X_0$ we get three components matched to x.

Now, we "trade" matched components for disjoint paths from S_0 to X_0: Notice that in each component with a terminal t that is seen by some $x \in X_0$ we can freely choose a path from t to x with all vertices but x contained in the component. Thus, for all $|X_0| + 2$ components we can find disjoint paths to the matched vertices in X_0. Hence, we get a path packing with $|X_0| + 2$ paths from S_0 to X_0 with three paths ending in any chosen vertex $x \in X_0$.

Setting up the gammoid. For the gammoid M we use a graph G' that is obtained from $G = (V, E)$ by adding two so-called *sink-only copies* v', v'' for each vertex $v \in V$. A sink-only copy v' (or v'') for v shares all in-neighbors with v but has no out-neighbors (i.e., if $\{u, v\}$ is an edge then we only add a directed edge (u, v')). Thus, adding such vertices does not affect, e.g., the existence of paths between any terminals, since they can only act as endpoints (sinks) of paths. Using the sink-only copies, we can formalize the informal statement of three paths ending in any $x \in X_0$ to three paths ending in $\{x, x', x''\}$. Let us also point out that the gammoid setting allows trivial paths consisting of just one vertex, e.g., we have such paths from $S \cap X$ to $S \cap X$. Overall, together with the above path packing we get that in G' there must exist a path packing of $|X| + 2$ paths from S to $X \cup \{x', x''\}$ for every choice of $x \in X_0$.

Applying representative sets. Now we will apply the idea that representative sets can be used to identify particular objects. We will use Theorem 5 to compute a $k - 1$ representative subset $\mathcal{T}'$ of $\mathcal{T}$ where $\mathcal{T} := \{\{v, v', v''\} \mid v \in V\}$. Our goal is to show that for all $x \in X_0$ we must have $\{x, x', x''\} \in \mathcal{T}'$. Note that the theorem guarantees $|\mathcal{T}'| \in \mathcal{O}(k^3)$.

Our argument now depends crucially on the trick that we outlined previously: If there exists an independent set I of M of size/rank at most $k-1$ such that $\{x, x', x''\}$ uniquely extends I then this directly implies that $\{x, x', x''\}$ is contained in every $k - 1$-representative subset $\mathcal{T}'$ of $\mathcal{T}$. Recall that we already know that $X \cup \{x', x''\}$ is linked to S in G' and thus it is independent, for all $x \in X_0$. It follows directly that $\{x, x', x''\}$ extends the independent set $X - x$ for all $x \in X_0$. It remains to prove that no other set $\{v, v', v''\} \in \mathcal{T}$ extends $X - x$.

Consider first any $v \in X - x$. In this case we have $\{v, v', v''\} \cap (X - x) = \{v\} \neq \emptyset$, implying that the set $\{v, v', v''\}$ does not extend $X - x$. The more interesting case is for $\{v, v', v''\}$ with $v \in V \setminus X$. First, note that for $\{v, v', v''\}$ to extend $X - x$ requires for $(X - x) \cup \{v, v', v''\}$ to be linked to S in G'. A (weaker) requirement is that $\{v, v', v''\}$ is linked to S in $G' - (X - x)$, since any paths from S to $X - x$ definitely block at least $X - x$ from being used in paths from S to $\{v, v', v''\}$.

Let us see that there cannot be three disjoint paths from S to $\{v, v', v''\}$ in $G' - (X - x)$: Recall that paths cannot have sink-only copies as interior vertices, so apart from v' and v'' we can use that X is a solution in graph G. At most one of the paths can come from a terminal in the terminal component of v, and one more path can include the vertex x. No third path is possible. Thus, we find that no other set $\{v, v', v''\}$ can extend $X - x$.

Since for each $x \in X_0$ the set $\{x, x', x''\}$ uniquely extends $X - x$ we get that for all vertices $x \in X_0$ we must have $\{x, x', x''\} \in \mathcal{T}'$. Hence, letting

$V(\mathcal{T}')$ stand for $\{v \mid \{v, v', v''\} \in \mathcal{T}'\}$, it is guaranteed that $X_0 \subseteq V(\mathcal{T}')$. In extension this implies $X = X_0 \cup (X \cap S) \subseteq V(\mathcal{T}') \cup S$. There is a reduction rule that ensures $|S| = \mathcal{O}(k)$ (see [39]), but let us omit this detail and directly assume that we have a set of $\mathcal{O}(k^3)$ vertices containing all terminals S as well as at least one solution X (if one exists).

Shrinking the input graph to $\mathcal{O}(k^3)$ vertices. We can now complete the kernelization. Let W denote the established set of $\mathcal{O}(k^3)$ vertices that is guaranteed to completely contain at least one solution (as well as all terminals). Using this guarantee, there is no harm in making all vertices of $V \setminus W$ *undeletable*: For any vertex $v \in V \setminus W$ simply make the neighbors of v a clique and remove v from the graph; this captures the intention that deleting v does not remove any connectivity while also shrinking the graph. (Note that doing this for all vertices of $V \setminus W$ at once corresponds to the so-called *torso* operation applied to W.) We obtain an equivalent instance $(\hat{G}, S, k)$ where $\hat{G}$ is a graph on vertex set W of size at most $\mathcal{O}(k^3)$.

Further results kernelization results based on matroids. Prior to the application of representative sets for kernelization [57], the fact that gammoids admit an efficient representation as column matroids of matrices over (sufficiently large) finite fields (cf. [62]) was used to find a (randomized) polynomial kernelization for Odd Cycle Transversal(k) [56], settling a well-known problem in kernelization. At high level, a represented gammoid is used to fairly succinctly encode a family of two-way cut queries that are sufficient to determine the status of the input instance. In the follow-up work [57] representative set tools were used, amongst others, to obtain somewhat more combinatorial[7] kernel results based on irrelevant vertex arguments.

Theorem 6 ([57]). *The following kernelizations are possible:* Almost 2-SAT(k)*, with $\mathcal{O}(k^6)$ variables;* s-Multiway Cut(k)*, with $\mathcal{O}(k^{s+1})$ vertices;* s-Multicut(k)*, with $\mathcal{O}(k^{\lceil\sqrt{2s}\rceil+1})$ vertices;* Group Feedback Vertex Set(k)*, for a group of s elements, with $\mathcal{O}(k^{2s+2})$ vertices. All results are randomized, with failure probability exponentially small in n.*

Note that, Almost 2-SAT(k), i.e., the task of making a 2-CNF formula satisfiable by deleting at most k variables, is a pivotal problem since several other problems have PPTs to it, e.g., e.g., Vertex Cover Above Matching, Vertex Cover Above LP, and RHorn-Backdoor Deletion Set. It also directly generalizes Odd Cycle Transversal(k). All these problems have polynomial kernelizations due to this connection.

[7] The underlying result of Lovász [60] is proved via exterior algebra, and derived algorithms [62, 31] still use linear algebra tools.

Furthermore, the techniques were also used to obtain results called *cut covering sets*, which guarantee to include an optimal cut for each one of a (possibly exponentially large) set of cut queries. We recall the statement for the two-way cut setting and direct the reader to [57] for an s-multiway cut variant of the theorem.

Theorem 7 ([57]). *Let $G = (V, E)$ be a digraph and let $S, T \subseteq V$. Let r denote the size of a minimum (S, T)-vertex cut (which may intersect S and T). There exists a set $Z \subseteq V$, $|Z| = \mathcal{O}(|S| \cdot |T| \cdot r)$, such that for any $A \subseteq S$ and $B \subseteq T$, it holds that Z contains a minimum (A, B)-vertex cut. We can find such a set in randomized polynomial time with failure probability $\mathcal{O}(2^{-n})$.*

Further reading. The already mentioned recent paper of Fomin et al. [31] is a recommended follow-up read. Fomin et al. obtain faster algorithms for finding representative sets for linear matroids and for the special case of uniform matroids; in particular the second does not require a matrix representation. Furthermore, they explain several algorithmic applications and obtain, amongst others, the so far fastest deterministic algorithm for Path(k), running in time $\mathcal{O}(2.851^k m \log^2 n)$.

6 Turing kernelization

Already before the kernelization lower bound framework [9] several authors had suggested the possibility of preprocessing into many independent small instances rather than just one [29, 40]. After the framework appeared, it was noted that the obtained lower bounds do not apply to this relaxed form of kernelization, which makes it a possible option for avoiding lower bounds.

A Turing kernel for Leaf Out-Tree(k). A first example was soon discovered by Binkele-Raible et al. [6]: Say that an *out-tree* is any directed tree with a unique vertex of in-degree zero, called the *root*, and with vertices of out-degree zero called the *leaves*. The Leaf Out-Tree(k) problem asks whether a given digraph $D = (V, A)$ contains an out-tree with at least k leaves. Binkele-Raible et al. [6] showed that this problem admits no polynomial kernelization unless NP $\subseteq$ coNP/poly (using the then new framework of Bodlaender et al. [9]). In contrast, they proved that a variant called Rooted Leaf Out-Tree(k), where in addition to $D = (V, A)$ and k we are given a fixed vertex $v \in V$ to use as the root of the out-tree, does admit a kernelization to $\mathcal{O}(k^3)$ vertices (and, hence, polynomial total size). They concluded that, since a given instance $(D = (V, A), k)$ of Leaf Out-Tree(k) has only

$|V|$ choices for a root v, one may preprocess the instance by returning $|V|$ instances (D, v, k) of Rooted Leaf Out-Tree(k), one for each choice of $v \in V$. Since the latter admits a polynomial kernelization, this yields $|V|$ instances on $\mathcal{O}(k^3)$ vertices each. Furthermore, (D, k) is yes for Leaf Out-Tree(k) if and only if at least one instance (D, v, k) is yes for Rooted Leaf Out-Tree(k). Altogether, the reduction of one instance of Leaf Out-Tree(k) to $|V|$ instances of Rooted Leaf Out-Tree(k) combined with a polynomial kernelization for the latter gave the first example[8] of what is now called a (polynomial) Turing kernelization. More specifically, it is a polynomial disjunctive kernelization since the status of the input instance is equivalent to the disjunction (or) of the outcomes of the $|V|$ reduced instances.

Turing kernelization and other variants. Given the success of the lower bound framework and the wealth of obtained results, a notion of preprocessing that avoids these lower bounds is of course highly interesting. Note that, from a practical perspective, a sequence of small, independent instances might also be easier to handle (e.g., by parallelization) than a single large instance. This aspect applies of course only to the case that the reduced instances are created in parallel, rather than adaptively. Theoretically, also an adaptive creation of inputs is interesting; in particular, lower bounds against adaptive (i.e., Turing) kernelization would be very powerful. Note that this necessitates a slightly more involved definition, since the "kernelization" needs to know the answers to already created instances before outputting the next one. It is thus natural to formalize a Turing kernelization for $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$ as an efficient algorithm that given $(x, k) \in \Sigma^* \times \mathbb{N}$ *correctly decides whether* $(x, k) \in \mathcal{Q}$ provided that it gets the answers to all (adaptively) created small instances. The traditional way in computer science to formalize this is by means of an oracle; we recall the definition given by Binkele-Raible et al. [6].

Definition 7 ([6])**.** A *t-oracle* for a parameterized problem $\mathcal{Q}$ is an oracle that takes as input (x, k) with $|x|, k \leq t$ and decides whether $(x, k) \in \mathcal{Q}$ in constant time.

Definition 8 ([6])**.** A parameterized problem $\mathcal{Q}$ is said to have a *$g(k)$-sized Turing kernelization* if there is an algorithm which given an input (x, k) together with a $g(k)$-oracle for $\mathcal{Q}$ decides whether $(x, k) \in \mathcal{Q}$ in time polynomial in $|x| + k$.

[8] Binkele-Raible et al. [6] also proved analogous results for Rooted Leaf Out-Branching(k) and Leaf Out-Branching(k) where the out-tree is required to span the input graph D.

Naturally, by letting the oracle queries be to any other parameterized problem $\mathcal{Q}'$ or to any (classical) language L we could define variants such as generalized Turing kernelization or Turing compression. Note, however, that using Karp reductions we can easily translate oracle questions, which probably makes the distinction meaningless. In the following we will not insist on a concrete definition and simply allow the most relaxed variant of t-sized queries to any language L.

Let us informally state also the following restricted variants of Turing kernelization:

Disjunctive kernels: Like the example for Leaf Out-Tree(k), given an input (x, k), create $|x|^{\mathcal{O}(1)}$ instances of size bounded in k such that (x, k) is YES if and only if *at least one output instance is* YES.

Conjunctive kernels: Given an input (x, k), create $|x|^{\mathcal{O}(1)}$ instances of size bounded in k such that (x, k) is YES if and only if *all output instances are* YES. Surprisingly perhaps, we are already able to rule out polynomial conjunctive kernels for most problems with lower bounds against polynomial kernelization. We will recall this briefly later in this section.

Truth-table kernels: Generalizing conjunctive and disjunctive kernels one may simply define any Boolean function (or a family thereof, one for each arity) and demand that the input is YES if and only if the function applied to the outcomes for all output instances (treating YES as true and NO as false) evaluates to true.

Initially, only few examples of polynomial Turing kernels were found for problems without polynomial kernels and all of them are in fact disjunctive kernels [6, 5, 67]. A few more simple examples have been observed throughout the community. As an example, the reader is invited to consider the Clique(Δ) problem where we seek a k-clique in a given graph G, parameterized by the maximum degree of G. It is not hard to give both an OR-(cross-)composition and a disjunctive polynomial kernelization.

Recently discovered Turing kernels. Last year, Thomassé et al. [69] found a polynomial Turing kernelization for Independent Set on bull-free graphs[9], where the oracle questions are used in a dynamic programming fashion on a decomposition of the bull-free input graphs. In this case, the full power of Turing kernelizations as opposed to truth-table kernelization (or

[9]The so-called bull graph is obtained from a triangle by attaching a leaf each to two of its vertices. Bull-free graphs are exactly those graphs that contain no induced subgraph (on five vertices) that is isomorphic to the bull.

others) seems required. A similar form of Turing kernelization was independently found by Jansen [49] more recently for the PATH(k) problem restricted to planar graphs (and related cases). We describe a simplified version of the approach taken by Jansen [49], since this requires less preliminaries.

1. We are given a planar graph $G = (V, E)$ and an integer k, and want to find out whether G contains a simple path on at least k vertices. We will efficiently solve the instance by making a polynomial in $|V|$ number of oracle queries of size polynomial in k each.

2. We apply a tree-like decomposition of the graph into its three-connected components (attributed to Tutte). Any two incident components overlap in at most two vertices. Roughly, this can be obtained by recursing on vertex-separators of size at most two, until reaching a three-connected component.

3. Any three-connected component of a planar graph on at least $\Omega(k^c)$ vertices must contain a path of length at least k, for some known constant c (cf. [49]). Thus, if the graph has a three-connected component that has size $\Omega(k^c)$, then we can safely answer YES. Otherwise, and henceforth, all three-connected components have size $\mathcal{O}(k^c)$.

4. If we take a leaf component then this is of size $\mathcal{O}(k^c)$ and we can afford an oracle question for the longest path therein. If this returns a path of length at least k then we can answer YES and stop. Else, we ask for the longest paths ending in the component or passing through it. Concretely, if, e.g., the component has vertices p and q shared with its parent component, then we also perform oracle questions for (1) the longest p,q-path; (2) the maximum total length of two disjoint paths starting in p and q; (3) the longest path starting in p; (4) the longest path starting in p and avoiding q; (5) the longest path starting in q; (6) the longest path starting in q and avoiding p.

5. If the computation on a component does not lead to an immediate YES answer, then we encode the gained information from questions (1-6) using annotations in the parent component, delete the present component, and continue. Note that, in this simplified version, we tacitly used oracle questions for finding longest paths in some form of annotated graph. With a bit more work (cf. [49]), we can avoid annotations and employ self-reduction to find longest paths.

Jansen [49] also proved a polynomial Turing kernelization for CYCLE(k) on planar graphs, and generalized his ideas to work also on bounded degree graphs, claw-free graphs, and $K_{3,t}$ minor graphs (for both problems).

Note also that all mentioned cases of PATH(k) and CYCLE(k) remain NP-hard and have trivial OR-(cross-)compositions by disjoint union that rule out polynomial kernels (cf. [49]). While the Tutte decomposition works on general graphs, it is crucial that the considered graph class has an inverse polynomial lower bound on the length of simple paths inside three-connected components (i.e., a component of size ℓ must be known to contain a path of length at least ℓ^{-c}).

Ruling out polynomial conjunctive kernels. Consider a polynomial conjunctive kernelization for a problem $\mathcal{Q}$. On input (x, k) it will create $|x|^{\mathcal{O}(1)}$ instances of size polynomial in k such that the input is YES if and only if all output instances are YES. (Note that, again, this will work just fine independently of whether the outputs are for $\mathcal{Q}$, another problem $\mathcal{Q}'$, or any classical language L.) Let us modify the kernelization to arbitrarily (i.e., nondeterministically) output *only one of its created instances.* Clearly, if the input is YES then all outputs are YES and it returns any one of them. If the input is NO then at least one created output is NO. Thus, by nondeterministically selecting one output, it may falsely return a YES instance but at least one possible computation leads to the output of a NO instance. Generally, such kernelizations have been called *co-nondeterministic kernelizations* [53] for their similarity to Turing machines for coNP. (Note that those are in general more powerful because they are not restricted to "just" $|x|^{\mathcal{O}(1)}$ instances but may in fact have $2^{|x|^{\mathcal{O}(1)}}$ computation paths, each with different output.)

It has been observed[10] that the proof of Fortnow and Santhanam [35] for the OR-distillation conjecture applies also if the OR-distillation behaves, similarly to above, in a co-nondeterministic fashion. In the work of Dell and van Melkebeek [22] the so-called "complementary witness lemma" holds explicitly also for the co-nondeterministic setting. Long story short, both OR-(cross-)compositions and polynomial kernelizations/compressions may behave co-nondeterministically without any harm to the lower bound implications. Thus, any (possibly co-nondeterministic) OR-(cross-)composition rules out co-nondeterministic polynomial kernelizations and compressions; in particular, this rules out the more restricted case of polynomial conjunctive kernels for the problem in question [53]. (For more applications of co-nondeterminism we refer to [53, 55].)

Lower bounds for Turing kernels. Unlike for normal (many-one) kernelization, there is yet no technique for ruling out polynomial Turing kernels for any FPT problem (modulo any reasonable complexity hypothesis). The

[10]This is attributed to Chen and Müller by Harnik and Naor [45].

observation applied for polynomial conjunctive kernelizations should not be expected to generalize, in particular not to the seemingly powerful adaptive setting of Turing kernels. (Note that having any Turing kernelization again also implies fixed-parameter tractability, and thus W[1]-hardness rules out such kernels, assuming FPT $\neq$ W[1].)

Motivated by this state of the art, Hermelin et al. [46] initiated a completeness program centered around a newly introduced WK/MK-hierarchy of parameterized problems.[11] The starting point is the fact that results for polynomial kernelizations transfer, modulo technical details, by polynomial parameter transformations (see Bodlaender et al. [13]). If we relax to using generalized kernelizations or compressions then results transfer directly (see, e.g., Lemma 1). In the same way, this applies to the existence and non-existence of polynomial disjunctive, conjunctive, truth-table, and Turing kernelizations.

Arguably the most important class in [46] is WK[1]; it is the lowest hardness class in the hierarchy. Since a variety of problems were shown to be complete for WK[1] we will simply list some complete problems for WK[1], MK[2], and WK[2] below rather than giving formal definitions (and will not discuss further classes). At high level, all WK[i] and MK[i] classes are defined as closures of certain parameterized satisfiability-related problems under PPTs. These defining problems are reparameterizations of problems used to define the W[i] and M[i] classes from the parameterized hierarchy of intractability (see, e.g., [30]). Motivated by the variety of problems that could be classified as WK[1]-complete, Hermelin et al. [46] conjectured that no WK[1]-hard problem admits a polynomial Turing kernelization. Similarly to an efficient algorithm for any NP-hard problem (but maybe not as surprising) a polynomial Turing kernelization for any WK[1]-hard problem would be a breakthrough since none of the known hard problems (see below) seem particularly amenable to this (see also the discussion in [46]).

The Hitting Set problem (note the unrestricted set size) nicely showcases several levels of the hierarchy when taken under different parameterizations.

> Hitting Set
> **Input:** A universe U, a set family $\mathcal{F} \subseteq 2^U$, and $k \in \mathbb{N}$.
> **Question:** Is there a set of at most k elements of U that intersects every set in $\mathcal{F}$?

[11] The hierarchy is, in a sense, a reparameterization of the W[i]- and M[i]-hierarchies in parameterized intractability. It subsumes a strongly related hierarchy of Harnik and Naor [45] aimed at classical problems in relation to their witness size. A detailed discussion of the relation is given in Hermelin et al. [46].

Under its standard parameter k the problem is complete for W[2] under parameterized reductions and, thus, not even FPT unless FPT = W[2]. Using, however, parameters $n := |U|$, $m := |\mathcal{F}|$, or $k \log n$ it can be easily seen to be FPT. Nevertheless, for all three parameters it is possible to rule out polynomial kernelizations; for the first two results this follows from work of Dom et al. [23]. Curiously, all three parameterizations give problems that are complete for different levels of the WK- and MK-hierarchies.

- Hitting Set(m) is complete for WK[1] and equivalent (also under PPTs) to problems such as Capacitated Vertex Cover(k), Connected Vertex Cover(k), Steiner Tree($k + t$), Min Ones d-SAT(k), Clique($k \log n$), Set Cover(n), Multicolored Path(k), and Binary NDTM Halting(k). The latter problem asks whether a given nondeterministic Turing machine with binary alphabet stops within k steps.

  Disjoint Paths(k) and Disjoint Cycles(k) are WK[1]-hard.

- Hitting Set(n) is complete for MK[2] and equivalent to problems such as Set Cover(m) and CNF-SAT(n).

  Among hard problems for MK[2] there are, e.g., several structural parameterizations of Dominating Set(k).

- Hitting Set($k \log n$) is complete for WK[2] and equivalent to Set Cover($k \log m$), and Dominating Set($k \log n$).

We refer to Hermelin et al. [46] for a more extensive list of hard and complete problems, in particular also for MK[2] and WK[2]. The most interesting feature, perhaps, is the richness of complete problems for WK[1]. The fact that all these fairly different problems are equivalent for existence of polynomial Turing kernelizations supports the conjecture that no WK[1]-hard problem has such a kernelization. We also refer to Hermelin et al. [46] for a discussion of why these problems seem hard to Turing-kernelize.

A particular problem that has so far resisted a classification is Path(k), for which neither a polynomial Turing kernelization nor WK[1]-hardness are known. If we make the problem slightly richer by taking the input graph to be k-colored and asking for a k-path containing all k colors then it becomes WK[1]-complete [46]; Jansen [49] extended this to the special case of planar inputs, motivated by his Turing kernelization for the un-colored version. Apart from this it would, obviously, be of high interest to have any complexity-theoretic evidence for the correctness of the conjecture that WK[1]-hard problems have no polynomial Turing kernels.

7 Open problems

In this section we conclude the survey with some open problems. One of the central problems in kernelization research is certainly the understanding of possibilities and limitations of Turing kernelization. Furthermore, the Turing kernelization status of the Path(k) problem is of particular interest since it is not known to be hard for $\mathsf{WK}[1]$.

Open problem 1. Devise general upper and lower bound tools for Turing kernelization.

Open problem 2. Prove or disprove the conjecture that no $\mathsf{WK}[1]$-hard problem admits a polynomial Turing kernelization.

Open problem 3. Prove or disprove the existence of a polynomial Turing kernelization for Path(k).

The randomized polynomial kernelizations for, e.g., Deletable Terminal Multiway Cut(k) and Odd Cycle Transversal(k) [57], bring up the question of whether there are also deterministic polynomial kernels for these problems. This could be either by a derandomization of the existing approach or by completely new methods. Note that the exponentially small error in the kernelizations makes a lower bound against deterministic kernelizations unlikely (at least within the current framework).

Open problem 4. Are there deterministic polynomial kernelizations for the problems covered by the matroid-based kernelization results in [57]?

Finally, we mention (and recall) two concrete parameterized problems that have so far resisted classification into admitting or not admitting (e.g., modulo $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$) a polynomial kernelization.

Open problem 5. In the Multiway Cut(k) problem we are given an undirected graph $G = (V, E)$, a set of terminal vertices T, and $k \in \mathbb{N}$ with the task of deleting at most k non-terminal vertices to disconnect all terminals. Does this problem have a polynomial kernelization?

Recall that the restricted variant with only a fixed number s of terminals has a kernelization to an equivalent instance with $\mathcal{O}(k^{s+1})$ vertices [57]. It is interesting whether the occurrence of s in the exponent is necessary and, if so, whether it is asymptotically optimal.

Open problem 6. In the Directed Feedback Vertex Set(k) problem we are given a directed graph $G = (V, A)$ and $k \in \mathbb{N}$ with the task to delete at most k vertices to make the graph acyclic (if possible). Does this problem have a polynomial kernelization?

This problem has survived, so far, the development of various upper and lower bound techniques, and is probably the longest-standing open problem in kernelization (and holding a solid place among established open problems in parameterized complexity overall).

Acknowledgements

The author is indebted to Andrew Drucker and Magnus Wahlström for several discussions and comments that greatly improved this survey. Furthermore, detailed comments of Bart Jansen and Somnath Sikdar on their work are gratefully acknowledged.

References

[1] Faisal N. Abu-Khzam. An improved kernelization algorithm for r-Set Packing. *Inf. Process. Lett.*, 110(16):621–624, 2010.

[2] Faisal N. Abu-Khzam. A kernelization algorithm for d-Hitting Set. *J. Comput. Syst. Sci.*, 76(7):524–531, 2010.

[3] Jochen Alber, Michael R. Fellows, and Rolf Niedermeier. Polynomial-time data reduction for dominating set. *J. ACM*, 51(3):363–384, 2004.

[4] Noga Alon, Gregory Gutin, Eun Jung Kim, Stefan Szeider, and Anders Yeo. Solving MAX-r-SAT above a tight lower bound. *Algorithmica*, 61(3):638–655, 2011.

[5] Abhimanyu M. Ambalath, Radheshyam Balasundaram, Chintan Rao H., Venkata Koppula, Neeldhara Misra, Geevarghese Philip, and M. S. Ramanujan. On the kernelization complexity of colorful motifs. In *IPEC*, volume 6478 of *LNCS*, pages 14–25. Springer, 2010.

[6] Daniel Binkele-Raible, Henning Fernau, Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Yngve Villanger. Kernel(s) for problems with no kernel: On out-trees with many leaves. *ACM Transactions on Algorithms*, 8(4):38, 2012.

[7] Hans L. Bodlaender. Kernelization: New upper and lower bound techniques. In *IWPEC*, volume 5917 of *LNCS*, pages 17–37. Springer, 2009.

[8] Hans L. Bodlaender, Rod Downey, Fedor V. Fomin, and Dániel Marx, editors. *The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, volume 7370 of *LNCS*. Springer, 2012.

[9] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009.

[10] Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (Meta) Kernelization. In *FOCS*, pages 629–638. IEEE Computer Society, 2009.

[11] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Preprocessing for treewidth: A combinatorial analysis through kernelization. *SIAM J. Discrete Math.*, 27(4):2108–2142, 2013.

[12] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM J. Discrete Math.*, 28(1):277–305, 2014.

[13] Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theor. Comput. Sci.*, 412(35):4570–4578, 2011.

[14] Jin-yi Cai, Venkatesan T. Chakaravarthy, Lane A. Hemaspaandra, and Mitsunori Ogihara. Competing provers yield improved Karp-Lipton collapse results. *Inf. Comput.*, 198(1):1–23, 2005.

[15] Jianer Chen, Iyad A. Kanj, and Weijia Jia. Vertex cover: Further observations and further improvements. *J. Algorithms*, 41(2):280–301, 2001.

[16] Robert Crowston, Michael R. Fellows, Gregory Gutin, Mark Jones, E. J. Kim, Fran Rosamond, Imre Z. Ruzsa, Stéphan Thomassé, and Anders Yeo. Satisfying more than half of a system of linear equations over GF(2): A multivariate approach. *J. Comput. Syst. Sci.*, 80(4):687–696, 2014.

[17] Robert Crowston, Gregory Gutin, Mark Jones, and Gabriele Muciaccia. Maximum balanced subgraph problem parameterized above lower bound. *Theor. Comput. Sci.*, 513:53–64, 2013.

[18] Marek Cygan, Fabrizio Grandoni, and Danny Hermelin. Tight kernel bounds for problems on graphs with small degeneracy - (extended abstract). In *ESA*, volume 8125 of *LNCS*, pages 361–372. Springer, 2013.

[19] Marek Cygan, Stefan Kratsch, Marcin Pilipczuk, Michal Pilipczuk, and Magnus Wahlström. Clique cover and graph separation: New incompressibility results. In *ICALP (1)*, volume 7391 of *LNCS*, pages 254–265. Springer, 2012.

[20] Holger Dell. A simple proof that AND-compression of NP-complete problems is hard. *Electronic Colloquium on Computational Complexity (ECCC)*, 2014. Available at `http://eccc.hpi-web.de/report/2014/075/`.

[21] Holger Dell and Dániel Marx. Kernelization of packing problems. In *SODA*, pages 68–81. SIAM, 2012.

[22] Holger Dell and Dieter van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In *STOC*, pages 251–260. ACM, 2010.

[23] Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Incompressibility through colors and IDs. In *ICALP (1)*, volume 5555 of *LNCS*, pages 378–389. Springer, 2009.

[24] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity (Monographs in Computer Science)*. Springer, November 1998.

[25] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[26] Andrew Drucker. New limits to classical and quantum instance compression. In *FOCS*, pages 609–618. IEEE Computer Society, 2012.

[27] Andrew Drucker. New limits to classical and quantum instance compression. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:112, 2012.

[28] Paul Erdős and Richard Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, 35:85–90, 1960.

[29] Vladimir Estivill-Castro, Michael R. Fellows, Michael A. Langston, and Frances A. Rosamond. FPT is P-time extremal structure I. In *ACiD*, volume 4 of *Texts in Algorithmics*, pages 1–41. King's College, London, 2005.

[30] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*. Springer, March 2006.

[31] Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Efficient computation of representative sets with applications in parameterized and exact algorithms. In *SODA*, pages 142–151. SIAM, 2014.

[32] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Bidimensionality and kernels. In *SODA*, pages 503–510. SIAM, 2010.

[33] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Linear kernels for (connected) dominating set on H-minor-free graphs. In *SODA*, pages 82–93. SIAM, 2012.

[34] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Linear kernels for (connected) dominating set on graphs with excluded topological subgraphs. In *STACS*, volume 20 of *LIPIcs*, pages 92–103. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

[35] Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *J. Comput. Syst. Sci.*, 77(1):91–106, 2011.

[36] Jakub Gajarský, Petr Hlinený, Jan Obdrzálek, Sebastian Ordyniak, Felix Reidl, Peter Rossmanith, Fernando Sanchez Villaamil, and Somnath Sikdar. Kernelization using structural parameters on sparse graph classes. In *ESA*, volume 8125 of *LNCS*, pages 529–540. Springer, 2013.

[37] Robert Ganian, Friedrich Slivovsky, and Stefan Szeider. Meta-kernelization with structural parameters. In *MFCS*, volume 8087 of *LNCS*, pages 457–468. Springer, 2013.

[38] Valentin Garnero, Christophe Paul, Ignasi Sau, and Dimitrios M. Thilikos. Explicit linear kernels via dynamic programming. In *STACS*, volume 25 of *LIPIcs*, pages 312–324. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014.

[39] Sylvain Guillemot. FPT algorithms for path-transversal and cycle-transversal problems. *Discrete Optimization*, 8(1):61–71, 2011.

[40] Jiong Guo and Rolf Niedermeier. Invitation to data reduction and problem kernelization. *SIGACT News*, 38(1):31–45, 2007.

[41] Jiong Guo and Rolf Niedermeier. Linear problem kernels for NP-hard problems on planar graphs. In *ICALP*, volume 4596 of *LNCS*, pages 375–386. Springer, 2007.

[42] Venkatesan Guruswami and Rishi Saket. On the inapproximability of vertex cover on k-partite k-uniform hypergraphs. In *ICALP (1)*, volume 6198 of *LNCS*, pages 360–371. Springer, 2010.

[43] Gregory Gutin, Eun Jung Kim, Stefan Szeider, and Anders Yeo. A probabilistic approach to problems parameterized above or below tight bounds. *J. Comput. Syst. Sci.*, 77(2):422–429, 2011.

[44] Shai Gutner. Polynomial kernels and faster algorithms for the dominating set problem on graphs with an excluded minor. In *IWPEC*, volume 5917 of *LNCS*, pages 246–257. Springer, 2009.

[45] Danny Harnik and Moni Naor. On the compressibility of $\mathcal{NP}$ instances and cryptographic applications. *SIAM J. Comput.*, 39(5):1667–1713, 2010.

[46] Danny Hermelin, Stefan Kratsch, Karolina Soltys, Magnus Wahlström, and Xi Wu. A completeness theory for polynomial (Turing) kernelization. In *IPEC*, volume 8246 of *LNCS*, pages 202–215. Springer, 2013.

[47] Danny Hermelin and Xi Wu. Weak compositions and their applications to polynomial lower bounds for kernelization. In *SODA*, pages 104–113. SIAM, 2012.

[48] Bart M. P. Jansen. On sparsification for computing treewidth. In *IPEC*, volume 8246 of *LNCS*, pages 216–229. Springer, 2013.

[49] Bart M. P. Jansen. Turing kernelization for finding long paths and cycles in restricted graph classes. *CoRR*, abs/1402.4718, 2014.

[50] Bart M. P. Jansen and Hans L. Bodlaender. Vertex cover kernelization revisited - upper and lower bounds for a refined parameter. *Theory Comput. Syst.*, 53(2):263–299, 2013.

[51] Eun Jung Kim, Alexander Langer, Christophe Paul, Felix Reidl, Peter Rossmanith, Ignasi Sau, and Somnath Sikdar. Linear kernels and single-exponential algorithms via protrusion decompositions. In *ICALP (1)*, volume 7965 of *LNCS*, pages 613–624. Springer, 2013.

[52] Lukasz Kowalik, Marcin Pilipczuk, and Karol Suchan. Towards optimal kernel for connected vertex cover in planar graphs. *Discrete Applied Mathematics*, 161(7-8):1154–1161, 2013.

[53] Stefan Kratsch. Co-nondeterminism in compositions: a kernelization lower bound for a ramsey-type problem. In *SODA*, pages 114–122. SIAM, 2012.

[54] Stefan Kratsch, Geevarghese Philip, and Saurabh Ray. Point line cover: The easy kernel is essentially tight. In *SODA*, pages 1596–1606. SIAM, 2014.

[55] Stefan Kratsch, Marcin Pilipczuk, Ashutosh Rai, and Venkatesh Raman. Kernel lower bounds using co-nondeterminism: Finding induced hereditary subgraphs. In *SWAT*, volume 7357 of *LNCS*, pages 364–375. Springer, 2012.

[56] Stefan Kratsch and Magnus Wahlström. Compression via matroids: a randomized polynomial kernel for odd cycle transversal. In *SODA*, pages 94–103. SIAM, 2012.

[57] Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. In *FOCS*, pages 450–459. IEEE Computer Society, 2012.

[58] Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Kernelization - preprocessing with a guarantee. In Bodlaender et al. [8], pages 129–161.

[59] Daniel Lokshtanov, Matthias Mnich, and Saket Saurabh. A linear kernel for a planar connected dominating set. *Theor. Comput. Sci.*, 412(23):2536–2543, 2011.

[60] László Lovász. Flats in matroids and geometric graphs. In *Proc. Sixth British Combinatorial Conf.*, Combinatorial Surveys, pages 45–86, 1977.

[61] Meena Mahajan and Venkatesh Raman. Parameterizing above guaranteed values: MaxSat and MaxCut. *J. Algorithms*, 31(2):335–354, 1999.

[62] Dániel Marx. A parameterized view on matroid optimization problems. *Theor. Comput. Sci.*, 410(44):4471–4479, 2009.

[63] Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, and Saket Saurabh. The kernelization complexity of connected domination in graphs with (no) small cycles. *Algorithmica*, 68(2):504–530, 2014.

[64] Hazel Perfect. Applications of Menger's graph theorem. *J. Math. Anal. Appl.*, 22:96–111, 1968.

[65] Geevarghese Philip, Venkatesh Raman, and Somnath Sikdar. Polynomial kernels for dominating set in graphs of bounded degeneracy and beyond. *ACM Transactions on Algorithms*, 9(1):11, 2012.

[66] Willard Van Orman Quine. The problem of simplifying truth functions. *The American Mathematical Monthly*, 59(8):521–531, 1952.

[67] Alexander Schäfer, Christian Komusiewicz, Hannes Moser, and Rolf Niedermeier. Parameterized computational complexity of finding small-diameter subgraphs. *Optimization Letters*, 6(5):883–891, 2012.

[68] Robert Endre Tarjan and Anthony E. Trojanowski. Finding a maximum independent set. *SIAM J. Comput.*, 6(3):537–546, 1977.

[69] Stéphan Thomassé, Nicolas Trotignon, and Kristina Vuskovic. Parameterized algorithm for weighted independent set problem in bull-free graphs. *CoRR*, abs/1310.6205, 2013.

[70] Magnus Wahlström. Abusing the Tutte matrix: An algebraic instance compression for the K-set-cycle problem. In *STACS*, volume 20 of *LIPIcs*, pages 341–352. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

[71] Chee-Keng Yap. Some consequences of non-uniform conditions on uniform classes. *Theor. Comput. Sci.*, 26:287–300, 1983.