A Completeness Theory for Polynomial (Turing) Kernelization

Danny Hermelin*
Ben Gurion University of the Negev, Be'er Sheva, Israel
hermelin@bgu.ac.il

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

Karolina Sołtys
Max Planck Institute for Informatics, Saarbrücken, Germany
ksoltys@mpi-inf.mpg.de

Magnus Wahlström*
Royal Holloway, University of London, England
Magnus.Wahlstrom@rhul.ac.uk

Xi Wu
University of Wisconsin Madison, Madison, USA
xiwu@cs.wisc.edu

March 2, 2015

Abstract

The framework of Bodlaender et al. (ICALP 2008, JCSS 2009) and Fortnow and Santhanam (STOC 2008, JCSS 2011) allows us to exclude the existence of polynomial kernels for a range of problems under reasonable complexity-theoretical assumptions. However, some issues are not addressed by this framework, including the existence of Turing kernels such as the "kernelization" of LEAF OUT BRANCHING(k) that outputs n instances each of size poly(k). Observing that Turing kernels are preserved by polynomial parametric transformations (PPTs), we define two *kernelization hardness* hierarchies by the PPT-closure of problems that seem fundamentally unlikely to admit efficient Turing kernelizations. This gives rise to the MK- and WK-hierarchies which are akin to the M- and W-hierarchies of parameterized complexity. We find that several previously considered problems are complete for the fundamental hardness class WK[1], including MIN ONES d-SAT(k), BINARY NDTM HALTING(k), CONNECTED VERTEX COVER(k), and CLIQUE parameterized by $k \log n$. We conjecture that no WK[1]-hard problem admits a polynomial Turing kernel. Our hierarchy subsumes an earlier hierarchy of Harnik and Naor (FOCS 2006, SICOMP 2010) that, from a parameterized perspective, is restricted to classical problems parameterized by witness size. Our results provide the first natural complete problems for, e.g., their class VC_1; this had been left open.

1 Introduction

Kernelization, or data reduction, is a central concept in parameterized complexity, and has important applications outside this field as well. Roughly speaking, a kernelization algorithm reduces an instance

*Main work done while the author was at the Max Planck Institute for Informatics.

†Main work done while supported by the Netherlands Organization for Scientific Research (NWO), project "KERNELS: Combinatorial Analysis of Data Reduction".

of a given parameterized problem to an equivalent instance of size $f(k)$, where k is the parameter of the input instance. Appropriately, the function $f()$ is referred to as the *size* of the kernel. A kernel with a good size guarantee is very useful – whether one wants to solve a problem exactly, or apply heuristics, or compute an approximation, it never hurts to first apply the kernelization procedure.[1] It can also be seen more directly as instance compression, e.g., for storing a problem instance for the future use; see Harnik and Naor [21]. The common milestone for an efficient kernelization is a *polynomial kernel*, i.e., a kernel with a polynomial size guarantee. Several significant kernelization results can be found in the literature, sometimes using non-trivial mathematical tools; see, e.g., the $2k$-vertex kernel for VERTEX COVER [32], the $\mathcal{O}(k^2)$ kernel for FEEDBACK VERTEX SET [33], and the recent randomized polynomial kernel for ODD CYCLE TRANSVERSAL [29].

Fairly recently, work by Bodlaender et al. [6] together with a result of Fortnow and Santhanam [20] provided the first technique to rule out the existence of any polynomial kernel for certain problems, assuming that NP $\not\subseteq$ coNP/poly (and PH does not collapse [34]). A series of further papers have applied this framework to concrete problems and developed it further, e.g., [16, 15, 8, 14, 23, 26, 17].

However, there are relaxed notions of efficient kernelization which are not ruled out by any existing work, but which would still be useful in practice and interesting from a theoretical point of view. Almost immediately after the appearance of the above lower bounds framework, the question was raised whether there were notions of "cheating" kernels, for example for the problem k-PATH, which could circumvent the above lower bounds by producing Turing kernels instead of standard many-one kernelizations [5]. Not long afterwards, the first example of such a cheating kernel appeared: Binkele-Raible et al. [4] showed that the k-LEAF OUT-BRANCHING problem (given a directed graph G and an integer k, does G contain a directed tree with at least k leaves?) does not admit a polynomial kernel unless NP $\not\subseteq$ coNP/poly, but does admit one (with $\mathcal{O}(k^3)$ vertices) if the root of the tree is fixed, implying a Turing kernel in the form of a disjunction over n instances, each of size polynomial in k. There are also simpler problems sharing the same behavior; for example, the problem of CLIQUE parameterized by maximum degree is trivially compatible with the lower-bound frameworks, implying that it has no polynomial many-one kernel unless NP $\not\subseteq$ coNP/poly, but admits a very simple Turing kernel into n instances of k vertices (by taking the neighborhood of each vertex). We will call such a disjunctive Turing kernel an *OR-kernel*. We observe that many of the positive aspects of standard (many-one) polynomial kernels are preserved by OR-kernels, or even generally Turing kernels; in particular the algorithmic consequences (e.g., a Turing kernel with polynomial individual instance sizes for a problem in NP implies an algorithm with a running time of $2^{k^{\mathcal{O}(1)}}n^{\mathcal{O}(1)}$, same as for a polynomial many-one kernel).

The question of the extent to which such Turing kernels exist is theoretically very interesting and one of the most important problems in the field. Some restricted forms of Turing kernels, e.g., polynomial AND-kernels, can already be excluded by the existing framework, as they are special cases of polynomial kernels which may use co-nondeterministic polynomial time (cf. [15, 26, 29]). However, for OR-kernels or Turing kernels the current framework does not apply (as also witnessed by the k-LEAF OUT-BRANCHING and CLIQUE by max degree problems). It is also unclear if the framework could be adapted to deal with them. Instead, we take an approach common in complexity theory, namely that of defining an appropriate notion of hardness, and studying problems that are complete under this notion. We start from a set of problems for which we conjecture that none of them has a polynomial Turing kernel, and show that they are equivalent under PPT-reductions (which preserve existence of polynomial Turing kernels). The result is a robust class of hardness of Turing kernelization, dubbed WK[1], whose complete problems include central problems from different areas of theoretical computer science. While we have no concrete evidence that our conjecture holds, we feel that the abundance of WK[1]-complete problems, where Turing-kernelization is found for none of them, might suggest its validity.

[1] This is assuming that the kernel preserves solution values, which most do.

WK[1]-hard problems. A cornerstone problem of WK[1] is the k-step halting problem for nondeterministic Turing machines parameterized by $k \log n$. To see why this is a powerful problem, and why an efficient Turing kernel would seem unlikely, consider the k-clique problem. Given a graph G with n vertices and an integer k, it is easy to construct a Turing machine which checks in poly(k) nondeterministic steps whether G contains a k-clique (by using a number of states polynomial in n). On the other hand, an OR-kernel for the problem (or more generally, a polynomial Turing kernel) would require reducing k-clique to a polynomial number of questions of size polynomial in $k \log n$ (e.g., poly(n) induced subgraphs of G, each of size poly$(k \log n)$, which are guaranteed to cover any k-clique of G). In fact, the above-mentioned halting problem captures not only clique, but all problems where a witness of t bits can be verified in poly$(t, \log n)$ time (e.g., SUBGRAPH ISOMORPHISM).

Other WK[1]-complete problems include MIN ONES d-SAT, the problem of finding a satisfying assignment with at most k true variables for a d-CNF formula, parameterized by the solution size k; HITTING SET parameterized by the number of sets or hyperedges m; and CONNECTED VERTEX COVER parameterized by the solution size k. Of these, we in particular want to single out MIN ONES d-SAT, which captures all minimization problems for which the consistency of a solution can be locally verified (by looking at combinations of d values at a time). For example, this includes the $\mathcal{H}$-FREE EDGE MODIFICATION(k) problems, where $\mathcal{H}$ is a finite, fixed set of forbidden induced subgraphs, and the goal is to remove or add k edges in the input graph in order to obtain a graph with no induced subgraph in $\mathcal{H}$ [11].

Extending the hardness class WK[1], we also define a hierarchy of hardness classes WK[t] and MK[t] for $t \geq 1$, mirroring the W- and M-hierarchies of traditional parameterized complexity; see [19]. We note that there are also strong similarities to the work of Harnik and Naor [21], in particular to the VC-hierarchy (which is defined around the notion of *witness length* for problems in NP). However, the notion of a parameter seems more general and robust than witness length; consider for example the volume of work in FPT on structural parameters such as treewidth. We also feel that the connections to the traditional FPT hardness classes (see Section 3) flesh out and put Harnik and Naor's work into context, and the link to the Turing kernel question adds interest to the separation question. Still, the main focus of our work is the hardness class WK[1].

We hope that our conjecture, that WK[1]-hard problems do not have polynomial Turing kernels, will inspire other researchers to revisit the kernelization properties of problems which have been shown not to admit standard polynomial kernels unless PH collapses, but for which WK[1]-hardness is less obvious. In particular, we leave open the WK[1]-hardness of k-PATH, the problem for which the existence of Turing kernels was originally asked in [5].

2 Preliminaries

We begin our discussion by formally defining some of the main concepts used in this paper, and by introducing some terminology and notation that will be used throughout. A list of problem definitions can be found in Section A of the appendix. We use $[n]$ to denote the set of integers $\{1, \ldots, n\}$.

Definition 1 (Kernelization)**.** *A* kernelization algorithm*, or, in short, a* kernel *for a parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is a polynomial-time algorithm that on a given input $(x, k) \in \Sigma^* \times \mathbb{N}$ outputs a pair $(x', k') \in \Sigma^* \times \mathbb{N}$ such that $(x, k) \in L \Leftrightarrow (x', k') \in L$, and $|x'| + k' \leq f(k)$ for some function f. The function f above is referred to as the* size *of the kernel.*

In other words, a kernel is a polynomial-time reduction from a problem to itself that compresses the problem instance to a size depending only on the parameter. If the size of a kernel for L is polynomial, we say that L has a *polynomial kernel*. In the interest of robustness and ease of presentation, we relax the notion of kernelization to allow the output to be an instance of a different problem. This has been referred

to as a generalized kernelization [6] or bikernelization [2]. The class of all parameterized problems with polynomial kernels in this relaxed sense is denoted by PK.

Definition 2 (Turing Kernelization)**.** *A* Turing kernelization *for a parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is a polynomial-time algorithm with oracle access to a parameterized problem L' that can decide whether an input (x, k) is in L using queries of size bounded by $f(k)$, for some computable function f. The function f is referred to as the* size *of the kernel.*

If the size is polynomial, we say that L has a *polynomial Turing kernel*. The class of all parameterized problems with polynomial Turing kernels is denoted by Turing-PK.

Definition 3 (Polynomial Parametric Transformations [9])**.** *Let L_1 and L_2 be two parameterized problems. We write $L_1 \leq_{ppt} L_2$ if there exists a polynomial time computable function $\Psi : \{0,1\}^* \times \mathbb{N} \to \{0,1\}^* \times \mathbb{N}$ and a constant $c \in \mathbb{N}$, such that for all $(x, k) \in \Sigma^* \times \mathbb{N}$, if $(x', k') = \Psi(x, k)$ then $(x, k) \in L_1 \iff (x', k') \in L_2$, and $k' \leq ck^c$. The function Ψ is called a* polynomial parameter transformation *(*PPT *for short). If $L_1 \leq_{ppt} L_2$ and $L_2 \leq_{ppt} L_1$ we write $L_1 \equiv_{ppt} L_2$.*

Proposition 1. *Let L_1, L_2, and L_3 be three parameterized problems.*

- *If $L_1 \leq_{ppt} L_2$ and $L_2 \leq_{ppt} L_3$ then $L_1 \leq_{ppt} L_3$.*
- *If $L_1 \leq_{ppt} L_2$ and $L_2 \in$* PK *(resp.* Turing-PK*) then $L_1 \in$* PK *(resp.* Turing-PK*).*

We denote parameterizations in parentheses after the problem name, for example, CLIQUE$(k \log n)$. In this example, k is the solution size, and n the size of the input. (Recall that CLIQUE(k) is one of the fundamental hard problems for parameterized complexity, and unlikely to admit a kernel of any size [19]; however, under a parameter $p = k \log n$ it has a trivial kernel of size 2^p.)

Note that if a problem Q is solvable in $2^{k^{\mathcal{O}(1)}} n^{\mathcal{O}(1)}$ time, then $\mathrm{Q}(k) \equiv_{ppt} \mathrm{Q}(k \log n)$.

3 The WK- and MK-Hierarchies

In this section we introduce our hierarchies of inefficient kernelizability, the MK- and WK-hierarchies. Relations to the so-called *VC-hierarchy* of Harnik and Naor [21] are discussed in Section 3.3. To begin with, for $t \geq 0$ and $d \geq 1$, we inductively define the following classes $\Gamma_{t,d}$ and $\Delta_{t,d}$ of formulas following [19]:

$$\begin{array}{lcl}
\Gamma_{0,d} & := & \{\lambda_1 \wedge \cdots \wedge \lambda_c : c \in [d] \text{ and } \lambda_1, \ldots, \lambda_c \text{ are literals}\}, \\
\Delta_{0,d} & := & \{\lambda_1 \vee \cdots \vee \lambda_c : c \in [d] \text{ and } \lambda_1, \ldots, \lambda_c \text{ are literals}\}, \\
\Gamma_{t+1,d} & := & \{\bigwedge_{i \in I} \delta_i : I \text{ is a finite non-empty set and } \delta_i \in \Delta_{t,d} \text{ for all } i \in I\}, \\
\Delta_{t+1,d} & := & \{\bigvee_{i \in I} \gamma_i : I \text{ is a finite non-empty set and } \gamma_i \in \Gamma_{t,d} \text{ for all } i \in I\}.
\end{array}$$

Thus, $\Gamma_{1,3}$ is the set of all 3-CNF formulas, and $\Gamma_{2,1}$ is the set of all CNF formulas. Given a class Φ of propositional formulas, we let $\Phi^+, \Phi^- \subseteq \Phi$ denote the restrictions of Φ to formulas containing only positive and negative literals, respectively. We will refer to formulas belonging to Φ^+ and Φ^- as *monotone* and *anti-monotone* formulas, respectively.

For any given Φ, we define two parameterized problems:

- Φ-WSAT$(k \log n)$ is the problem of determining whether a formula $\phi \in \Phi$ with n variables has a satisfying assignment of Hamming weight exactly k, parameterized by $k \log n$.
- Φ-SAT(n) is the problem of determining whether a formula $\phi \in \Phi$ with n variables is satisfiable, parameterized by n.

In particular, we will be interested in $\Gamma_{t,d}$-WSAT$(k \log n)$ and $\Gamma_{t,d}$-SAT(n).

We now reach our class definitions. For a parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$, we let $[L]_{\leq_{ppt}}$ denote the closure of L under polynomial parametric transformations. That is, $[L]_{\leq_{ppt}} := \{L' \subseteq \Sigma^* \times \mathbb{N} : L' \leq_{ppt} L\}$.

Definition 4. *Let $t \geq 1$ be an integer. The classes* WK$[t]$ *and* MK$[t]$ *are defined by*

- WK$[t] := \bigcup_{d \in \mathbb{N}} [\Gamma_{t,d}\text{-WSAT}(k \log n)]_{\leq_{ppt}}$.
- MK$[t] := \bigcup_{d \in \mathbb{N}} [\Gamma_{t,d}\text{-SAT}(n)]_{\leq_{ppt}}$.

The naming of the classes in our hierarchies comes from the close relationship of the MK- and WK-hierarchies to the M- and W-hierarchies of traditional parameterized complexity [19]. Roughly speaking, WK$[t]$ and MK$[t]$ are reparameterizations by a factor of $\log n$ (or log of the instance size) of the traditional parameterized complexity classes W$[t]$ and M$[t]$ (although W$[t]$ and M$[t]$ are closed under FPT reductions, which may use superpolynomial time in k).

There are also close connections to the so-called *subexponential time* S-hierarchy (see [19, Chapter 16]); specifically, S$[t]$ and MK$[t]$ are defined from the same starting problems, using closures under different types of reduction; see also Cygan et al. [13]. We further note that [13] asked as an open problem whether there is a subexponential-time reduction from CNF-SAT(n) to HITTING SET(m). We will find the former to be MK[2]-complete and the latter to be WK[1]-complete, which may explain why such a reduction has been hard to find.

3.1 Canonical complete problems

We show the following complete problems for our hierarchy.

Theorem 1. *Let $t \geq 1$. The following hold.*

- $\Gamma^-_{1,2}$-WSAT$(k \log n)$ *is* WK$[1]$*-complete.*
- $\Gamma^-_{t,1}$-WSAT$(k \log n)$ *is* WK$[t]$*-complete for odd $t > 1$.*
- $\Gamma^+_{t,1}$-WSAT$(k \log n)$ *is* WK$[t]$*-complete for even $t > 1$.*
- $\Gamma_{1,d}$-SAT(n) *is* MK$[1]$*-complete for every $d \geq 3$.*
- $\Gamma_{t,1}$-SAT(n) *is* MK$[t]$*-complete for $t \geq 2$.*

Theorem 1 above shows that the traditional problems used for showing completeness in the W- and M-hierarchies have reparameterized counterparts which are complete for our hierarchy. The theorem is proven using a set of PPTs from the specific class-defining problems to the corresponding target problem in the theorem. Our main contribution is a PPT for the first item, for which previous proofs used FPT-time reductions. The remaining items are either easy or well-known.

Lemma 1. *Let $d \geq 1$. Then $\Gamma_{1,d}$-WSAT$(k \log n) \leq_{ppt} \Gamma^-_{1,2}$-WSAT$(k \log n)$.*

Proof. The lemma is trivial for $d = 1$, so assume $d \geq 2$. We show the proof in four steps. First we transform our input formula into an anti-monotone $\Gamma^-_{1,d}$ formula. The anti-monotone formula is then transformed into a multicolored $\Gamma^-_{1,d}$ formula, then to a multicolored $\Gamma^-_{1,2}$ formula, and finally to an uncolored $\Gamma^-_{1,2}$ formula. Since the transformation at each step will be a PPT, this will prove the lemma.

$\Gamma_{1,d}$-**WSAT**$(k \log n) \leq_{ppt} \Gamma^-_{1,d}$-**WSAT**$(k \log n)$**:** To transform to an anti-monotone formula, we adapt a trick used in [19]. Let ϕ be a $\Gamma_{1,d}$-formula on variables $X = \{x_1, \ldots, x_n\}$. We introduce a new set of variables $y_{i,j,j'}$, with the interpretation "the i'th true variable is x_j and the $(i+1)$'th true variable is $x_{j'}$", taken over the ordering of X assumed above. We will convert the formula ϕ into a formula ϕ' using only the y-variables. By a combination of the Hamming weight condition and negative 2-clauses, we can enforce consistency among the y-variables. Specifically, it is easy to enforce the following structure on the solutions of the formula:

- At most one y-variable is true for every $1 \leq i < k$.
- If y_{i,j_1,j_2} and y_{i+1,j_3,j_4} are both true, then $j_1 < j_2 = j_3 < j_4$.
- If y_{i,j_1,j_2} and y_{i',j_3,j_4} are both true, and $i' > i+1$, then $j_1 < j_2 < j_3 < j_4$.

Thus, if there are $k-1$ true y-variables, then these correspond to an ordered sequence of k variables x_j. Let ϕ'' be the formula containing these clauses. We can now replace every clause of ϕ by a conjunction of negative clauses, as follows. First, we may replace every literal x_j or $\neg x_j$ by a conjunction of negative literals of y-variables:

$$x_j = \bigwedge_{\substack{j_1 > j \\ j_2 > j}} \neg y_{1,j_1,j_2} \wedge \bigwedge_{\substack{i \\ j_1 < j \\ j_2 > j}} \neg y_{i,j_1,j_2} \wedge \bigwedge_{\substack{j_1 < j \\ j_2 < j}} \neg y_{k-1,j_1,j_2},$$

$$\neg x_j = \bigwedge_{i,j'} \neg y_{i,j,j'} \wedge \bigwedge_{j'} \neg y_{k-1,j',j}.$$

To see why these equalities hold, negate both sides of the equations; the result is an obvious description of x_j and $\neg x_j$ as disjunctions over positive y-variables. Second, since every clause in ϕ has bounded size, we may multiply out these conjunctions (using the distributive law of conjunctions) into individual d-clauses over literals $\neg y_{i,j,j'}$. Let ϕ' be the resulting formula. Then $\phi' \wedge \phi''$ is a $\Gamma^-_{1,d}$-formula and has a satisfying assignment of weight $k' = k-1$ if and only if ϕ has a satisfying assignment of weight k.

$\Gamma^-_{1,d}$-**WSAT**$(k \log n) \leq_{ppt}$ **MULTICOLORED** $\Gamma^-_{1,d}$-**WSAT**$(k \log n)$**:** Let (ϕ, k) be an instance of $\Gamma^-_{1,d}$-WSAT$(k \log n)$. We will produce an equivalent multicolored instance, where variables come in one of k colors, and the solution is required to contain exactly one variable of each color. This is easy since ϕ is anti-monotone: Create k copies of the entire variable set, each colored in a different color. For each variable x_i in ϕ, let $x_{i,c}$ be the copy of x_i of color c. For every x_i and every pair of colors $c \neq c'$, add a clause $(\neg x_{i,c} \vee \neg x_{i,c'})$, ensuring that k distinct variables are chosen. Then, for every clause $(\neg x_1 \vee \ldots \vee \neg x_d)$ in ϕ, replace it by one clause $(\neg x_{1,c_1} \vee \ldots \vee \neg x_{d,c_d})$ for every set of distinct colors $c_1, \ldots, c_d \in [k]$. Let ϕ' be the resulting formula. Note that each clause in ϕ excludes only a specific combination $x_1 \wedge \ldots \wedge x_d$, while the set of replacing clauses in ϕ' collectively excludes every possible selection of $x_1, \ldots, x_d$ from different colors; thus ϕ' has a multicolored satisfying assignment of weight k iff ϕ has a satisfying assignment of weight k.

MULTICOLORED $\Gamma^-_{1,d}$-**WSAT**$(k \log n) \leq_{ppt}$ **MULTICOLORED** $\Gamma^-_{1,2}$-**WSAT**$(k \log n)$**:** Let ϕ be a multicolored $\Gamma^-_{1,d}$-formula on variables X, $|X| = n$, and k colors. We create a multicolored $\Gamma^-_{1,2}$-formula ϕ' as follows: Let the colors of ϕ' correspond to d-tuples of colors from ϕ; thus ϕ' has $k' = \mathcal{O}(k^d)$ colors. For every color C of ϕ', corresponding to colors $(c_1, \ldots, c_d)$ of ϕ, create a set of variables as follows. Start with the set $X_C = \{(x_1, \ldots, x_d) : x_i \in X, x_i \text{ has color } c_i, 1 \leq i \leq d\}$. Then, remove from X_C every tuple $(x_1, \ldots, x_d)$ which explicitly falsifies a clause of ϕ. Since ϕ is anti-monotone, this is simple, e.g., if all clauses of ϕ are d-ary, then $(x_1, \ldots, x_d)$ is removed from X_C if and only if ϕ

contains a clause $(\neg x_1 \vee \ldots \vee \neg x_d)$. Let the remaining set of tuples be X^*_C. Now, for every pair of color-tuples C and C' which have at least one color in common, enforce consistency by excluding pairs of variable-tuples which disagree on some common color, using a conjunction of clauses with two negative literals each. Let the resulting instance be ϕ'. It is clear that a multicolored satisfying assignment to ϕ' corresponds to a multicolored satisfying assignment for ϕ. Furthermore, since ϕ is anti-monotone, every clause has been "verified" in the construction of some set X^*_C. Thus ϕ' has a multicolored satisfying assignment of weight k' if and only if ϕ has a multicolored satisfying assignment of weight k.

MULTICOLORED $\Gamma^-_{1,2}$-WSAT$(k \log n) \leq_{ppt} \Gamma^-_{1,2}$-WSAT$(k \log n)$: Given a multicolored $\Gamma^-_{1,2}$ formula ϕ, we transform ϕ into an "uncolored" formula ϕ' by adding a conjunction of negative 2-clauses over every color class of ϕ'. These ensure that exactly one variable of each class is set to true in any satisfying assignment of ϕ'. Thus, ϕ' has a weight k satisfying assignment if and only if ϕ' has a weight k satisfying assignment. □

The following follows from Flum and Grohe [19]. It can be readily verified that every reduction used to prove [19, Lemma 7.5] is a PPT.

Lemma 2 ([19, Lemma 7.5])**.** *Let $d \geq 1$. Then:*

- $\Gamma_{t,d}$-WSAT$(k \log n) \equiv_{ppt} \Gamma^-_{t,1}$-WSAT$(k \log n)$ *for all odd $t > 1$.*
- $\Gamma_{t,d}$-WSAT$(k \log n) \equiv_{ppt} \Gamma^+_{t,1}$-WSAT$(k \log n)$ *for all even $t > 1$.*

Next, we give the reductions for the MK-hierarchy.

Lemma 3. *Let $d \geq 3$. Then $\Gamma_{1,d}$-SAT$(n) \leq_{ppt} \Gamma_{1,3}$-SAT$(n)$.*

Proof. Let ϕ be a $\Gamma_{1,d}$ formula, i.e., a d-CNF formula. By removing duplicate clauses, the total length of ϕ is at most $\mathcal{O}(n^d)$. Since d is a constant, the classical reduction to 3-CNF of Karp [25] (via clause splitting) is a PPT. □

Lemma 4. *Let $t \geq 2$ and $d \geq 1$. Then $\Gamma_{t,d}$-SAT$(n) \leq_{ppt} \Gamma_{t,1}$-SAT$(n)$.*

Proof. Let ϕ be a $\Gamma_{t,d}$-formula. Introduce a new variable for every bottom-level disjunction or conjunction in ϕ of arity at most d. It is simple to enforce the values of the newly introduced variables using a $\Gamma_{2,1}$-formula. Let ϕ' be such a formula, and let ϕ'' be ϕ with every bounded-arity bottom-level conjunction or disjunction replaced by the corresponding new variable. Then $\phi \equiv \phi' \wedge \phi''$ and the reduction creates at most $\mathcal{O}(n^d)$ variables. □

This finishes the set of reductions needed to prove Theorem 1.

3.2 Class containments

We now proceed to show the class containments in our hierarchy. The main containments are as follows.

Theorem 2. MK[1] $\subseteq$ WK[1] $\subseteq$ MK[2] $\subseteq$ WK[2] $\subseteq$ MK[3] $\subseteq \cdots \subseteq$ EXPT.

The proof is split into three lemmas.

Lemma 5. MK$[t] \subseteq$ WK$[t]$ *for any integer $t \geq 1$.*

Proof. Let $t \geq 1$. By Proposition 1 and Definition 4, to prove the lemma it suffices to show that $\Gamma_{t,d}$-SAT$(n) \leq_{ppt} \Gamma_{t,d+1}$-WSAT$(k \log n)$ for every $d \in \mathbb{N}$. Let d be fixed, and let α be an $\Gamma_{t,d}$ formula on n variables given as an input to $\Gamma_{t,d}$-SAT(n). Let $x_1, \ldots, x_n$ denote the variables of α. We construct a formula over the variables $x_1^0, x_1^1, \ldots, x_n^0, x_n^1$. Let α' denote the formula obtained after replacing in α each positive occurrence of x_i with x_i^1, and each negative occurrence of x_i with x_i^0. Then $\alpha' \in \Gamma_{t,d}$. For each $i \in [n]$, define β_i to be the formula $\beta_i := (x_i^0 \vee x_i^1) \wedge (\neg x_i^0 \vee \neg x_i^1)$, i.e., $\beta_i \equiv x_i^0 \neq x_i^1$, and consider the formula $\alpha' \wedge \left(\bigwedge_i \beta_i\right)$. Then, as $\alpha' \in \Gamma_{t,d}$ and $\beta_i \in \Gamma_{1,2}$ for all $i \in [n]$, this formula can be written as a $\Gamma_{t,d'}$ formula β, where $d' = \max\{d, 2\} \leq d + 1$. Moreover, β has a satisfying assignment of weight weight $k := n$ iff α is satisfiable. The lemma follows. □

Lemma 6. WK$[t] \subseteq$ MK$[t+1]$ *for any integer* $t \geq 1$.

Proof. The lemma can be shown using the "$k \log n$ trick" introduced by Abrahamson et al. [1]. Introduce k groups of $\log n$ variables, each group determining via binary expansion the identity of one selected variable. Since MK[t+1] may use formulas of depth one larger than WK[t], it is possible to essentially replace the literals in an input formula by expressions over the $k \log n$ new variables. See [19, Theorem 16.42] for a detailed construction. □

Lemma 7. MK$[t]$, WK$[t] \subseteq$ EXPT *for all* $t \geq 1$.

Proof. Membership in EXPT is preserved by PPTs. Thus by Lemma 6 it suffices that $\Gamma_{t,d}$-SAT$(n) \in$ EXPT for every t and d, which is trivial by brute force. □

We also study a few further particular classes. First, let PK_{NP} denote the class of parameterized problems with polynomial kernels whose output problem lies in NP. We have the following relationship.

Lemma 8. MK$[1] = \text{PK}_{\text{NP}}$.

Proof. If a problem has a generalized polynomial kernel within NP, then we may kernelize any input instance to an equivalent instance of size polynomial in the parameter. Since the output is with respect to an NP-problem, we can then invoke a Karp reduction to 3-SAT by the NP-completeness of the latter. The size of the obtained equivalent 3-SAT instance is polynomial in the initial parameter value. The same is true for the number of variables, since it cannot exceed the size of the formula. Altogether we get a PPT from our initial problem to 3-SAT(n), proving membership in MK[1].

Conversely, any problem in MK[1] has a PPT to d-SAT(n) for some constant d. This directly constitutes also a generalized polynomial kernelization within NP since the PPT guarantees a polynomial bound for the number of variables and the size of a d-SAT is $\mathcal{O}(n^d)$ bits (assuming no duplicate clauses, but those can be safely discarded). □

Next, for a problem L, we define a parameterized problem OR$(L)(\ell)$ where the input is a set of instances $x_1, \ldots, x_t$ of L, each of length at most ℓ, and the task is to decide whether $x_i \in L$ for at least one instance x_i.

Lemma 9. *Let* L *be an* NP*-complete language. Then* MK$[1] \subseteq$ [OR$(L)(\ell)]_{\leq_{ppt}} \subseteq$ WK$[1]$*, where the first inclusion is strict unless* NP $\subseteq$ coNP/poly*, and the second is strict unless every problem in* WK$[1]$ *has a polynomial OR-kernel.*

Proof. The first containment is trivial. For the second containment, first note that OR$(L)(\ell)$ defines the same class of problems for any NP-complete language L, by applying the corresponding Karp reduction to every instance x_i of the input. Thus it suffices to show a single NP-complete problem L for which OR$(L)(\ell)$ is in WK[1], and it is not difficult to construct a PPT from OR(INDEPENDENT SET)(ℓ) to INDEPENDENT SET$(k \log n)$ (essentially by taking the join of all graphs in the input instance).

The first consequence is a direct application of the kernelization lower bound framework [6, 20]; the second follows since OR(L) has an OR-kernel, and OR-kernels are preserved by PPTs. □

For AND(L), and problems with Turing kernels more generally, no similar containment is known (or likely, as such a containment would imply a short witness for AND(L), which seems highly unlikely). However, our hierarchy can still be useful for AND-compositional problems, in showing them to be *hard* for some level.

3.3 Comparison with the VC-hierarchy

We now discuss the relations between the VC-hierarchy of Harnik and Naor [21] and the MK- and WK-hierarchies defined in this paper. Let us review some definitions. An NP-*language* L is for the purposes of this section defined by a pair (R_L, k), where $R_L(\cdot,\cdot)$ is a polynomial-time computable relation and $k(x) = |x|^{\mathcal{O}(1)}$ a polynomial-time computable function, and $x \in L$ for an instance x if and only if there is some string y with $|y| \leq k(x)$ such that $R_L(x, y)$ holds. The string y is called the *witness* for x. This naturally defines a parameterization of L, with parameter $k(x)$; the resulting problem is FPT (with running time $O^*(2^k)$). We refer to this parameterized problem as the *direct parameterization* of (R_L, k). Harnik and Naor define a notion of compressibility of an NP-language L based on the corresponding pair (R_L, k): Given an instance x, the compression takes polynomial time and returns an equivalent instance x' for some language L'; the length of x' is bounded polynomially in $k(x) + \log|x|$. For problems in NP the $\log|x|$ term makes no difference (cf., e.g., [21, 20]). Thus, for problems in NP this coincides with our relaxed notion of polynomial kernelization.

Before we proceed with the technical results, let us raise two points in comparison. First, Harnik and Naor deal solely with problems where the parameter is the length $k(x)$ of a witness. Although as we have seen this always defines a valid parameter, not every reasonable parameterization of an NP-problem can be interpreted as a witness in this sense (*e.g.* the treewidth or maximum degree parameterizations for graph problems). Second, although Harnik and Naor are rather lax with the choice of witness (frequently letting it go undefined), we want to stress that the choice of witness can have a big impact on the kernelization complexity of a problem. Consider as an example the case of HITTING SET (treated later in this paper). Let n denote the number of vertices of an instance, m the number of edges, and k the upper bound on solution size; note that the total coding size is $\mathcal{O}(mn)$. We will find that the problem is WK[1]-complete when parameterized by m, MK[2]-complete under the parameter n, and WK[2]-complete under the parameter $k \log n$. The latter two represent plausible choices of witnesses; a witness of length m is less obvious (or natural), but there is a simple witness of length $m \log k \leq m \log m$ obtained by describing a partition of the edges into k sets. One may also consider structural parameters such as treewidth (e.g., of the bipartite vertex/edge incidence graph), for which a corresponding short witness seems highly unlikely (but which can be shown to be MK[2]-hard).

That said, on a formal level we will find that the VC-hierarchy has much in common with the WK- and MK-hierarchies. Having already seen that our notions of kernelization/compression agree, we review their notion of problem reductions.

Definition 5 ([21])**.** *Let L and L' be* NP*-languages. We say that L* W-reduces *to L' if there exist polynomials p_1 and p_2 and a polynomial-time computable function f that takes an instance x for L and outputs an instance $f(x)$ for L' such that*

1. *$f(x) \in L'$ if and only if $x \in L$, and*
2. *if x is of length n with witness length k, then $f(x)$ is of length at most $p_1(n)$ with witness length at most $p_2(k, \log n)$.*

The following simple lemma establishes that our notions of reduction agree.

Lemma 10. *Let L and L' be* NP*-languages in the above sense. Let $\mathcal{Q}$ resp. $\mathcal{Q}'$ be the direct parameterization of R_L resp. $R_{L'}$. Then L W-reduces to L' if and only if there is a PPT from $\mathcal{Q}$ to $\mathcal{Q}'$.*

Proof. On the one hand, let f be the function guaranteed by the W-reduction, and let x (with bound $k = k(x)$ on witness length) be an instance of L. Define a mapping Ψ as follows: if $k \leq \log n$, solve the problem by iterating over all 2^k possible witness strings, and output a corresponding dummy instance; otherwise, output $(f(x), k'(f(x))$. It is clear that this function is polynomial-time computable and preserves problem membership; furthermore, the output parameter is bounded by $p_2(k, \log n) \leq p_2(k, k) = k^{O(1)}$. This proves that Ψ is a PPT.

On the other hand, let (x, k) be an instance of $\mathcal{Q}$, and let $\Psi(x, k) = (x', k')$. Let $f(x)$ output x'. Again, it is clear that the mapping is polynomial-time and preserves problem membership. Furthermore, by definition of $\mathcal{Q}'$, the value k' is a correct upper bound on the length of a witness for x' in L', and since Ψ is a PPT we have $k' = k^{O(1)} = (k + \log n)^{O(1)}$. This completes the proof. □

We now show the class correspondences. This result also answers a question left open in [21], of finding a natural problem complete for $\mathcal{VC}_1$ (as well as some minor questions about specific problem placements).

Theorem 3. *Let L be an* NP*-language defined by (R_L, k), and let $\mathcal{Q}$ be its direct parameterization. The following hold.*

1. *L is contained in (hard for, complete for) $\mathcal{VC}_{\text{or}}$ if and only if $\mathcal{Q}$ is contained in (hard for, complete for) the PPT-closure of* OR(3-SAT).
2. *L is contained in $\mathcal{VC}_1$ ($\mathcal{VC}_1$-hard, $\mathcal{VC}_1$-complete) if and only if $\mathcal{Q}$ is contained in* WK[1] *(*WK[1]*-hard,* WK[1]*-complete).*
3. *L is contained in $\mathcal{VC}_t$ ($\mathcal{VC}_t$-hard, $\mathcal{VC}_t$-complete) for $t > 1$ if and only if $\mathcal{Q}$ is contained in* MK[t] *(*MK[t]*-hard,* MK[t]*-complete).*

Proof. The class $\mathcal{VC}_1$ is defined as the closure under W-reductions of LOCAL CIRCUIT SAT, which under parameter $k \log n$ (i.e., the witness size) is WK[1]-complete by Theorem 8. Thus by Lemma 10 a problem is contained in $\mathcal{VC}_1$ if and only if its direct parameterization is PPT-reducible to a problem in WK[1], i.e., contained in WK[1]. Similarly, an NP-problem L is $\mathcal{VC}_1$-hard if and only if there is a PPT from a WK[1]-complete problem to the direct parameterization $\mathcal{Q}$ of L, i.e., if and only if $\mathcal{Q}$ is WK[1]-hard.

For $t \geq 2$, the class $\mathcal{VC}_t$ is defined via the DEPTH t CIRCUIT SAT problem with witness length n. It is a standard observation that for constant depth, we can reduce circuits to formulas (i.e., from a DAG-structure to a tree-structure) with only a polynomial blowup of size (and with inputs preserved). Thus, DEPTH t CIRCUIT SAT(n) PPT-reduces to $\Gamma_{t,1}$-SAT(n), and hence by Theorem 1 is MK[t]-complete. The results on containment, hardness, and completeness now follow as above.

Finally, $\mathcal{VC}_{\text{or}}$ is defined in [21] as the closure under W-reductions of OR(CIRCUIT SAT), under a witness length equal to the *largest total size* of an individual circuit in the input. It is easy to see that under such a large witness, the inner problem CIRCUIT SAT can be replaced by any NP-complete problem; we pick 3-SAT to remove confusions of instance length versus witness length. □

Let us also mention the following, to address the other direction of class comparison.

Lemma 11. *Let $\mathcal{Q}$ be a parameterized problem contained in* WK*[t] or* MK*[t] for some $t \geq 0$. Then the unparameterized version of $\mathcal{Q}$ can be given as an* NP*-language with a witness length polynomially bounded in the parameter value.*

Proof. Let $\mathcal{Q}$ be contained in MK[t] (without loss of generality, by the class containments). Then there is a PPT whose output is a problem with a witness of polynomially bounded length; we may define a verifier for $\mathcal{Q}$ by first running the PPT, then checking the witness against the output problem. □

In fact, it is possible to strengthen this to add that the direct parameterization of the resulting NP-language is again PPT-equivalent to $\mathcal{Q}$, implying that every "intermediate" class between MK[t] and WK[t] also has a corresponding class inside the $\mathcal{VC}$-hierarchy, but we omit the details of this extension.

4 Complete Problems for WK[1]

In this section we show that several natural problems are complete for our fundamental hardness class WK$[1]$ and thus exemplify its robustness. Our starting point will be CLIQUE$(k \log n)$ which is clearly equivalent to $\Gamma_{1,2}^{-}$-WSAT$(k \log n)$; the latter is WK$[1]$-complete by Theorem 1.

Theorem 4. CLIQUE$(k \log n)$ *is complete for* WK$[1]$.

The section is organized as follows. We will first address several standard problems that have already appeared as source problems for ruling out polynomial kernelizations via PPTs; we expect that these will be equally useful for proving WK$[1]$-hardness and completeness (see Subsection 4.1). Next, in Subsection 4.2, we consider several path- and cycle-related problems. In Subsection 4.3, we prove WK$[1]$-completeness of LOCAL CIRCUIT SAT$(k \log n)$, which is used by Harnik and Naor [21] to define their class $\mathcal{VC}_1$. Finally, in Subsection 4.4, we prove WK$[1]$-completeness for various problems that have kernelization lower bounds due to Dom et al. [16].

4.1 Basic Problems

This section establishes the following theorem that covers some basic problems which will be convenient for showing WK[1]-hardness and completeness for other problems. Standard many-one polynomial kernels for these problems were excluded in previous work [6, 30, 28, 16].

Theorem 5. *The following problems are all complete for* WK$[1]$*:*

- BINARY NDTM HALTING(k) *and* NDTM HALTING$(k \log n)$.
- MIN ONES d-SAT(k) *for* $d \geq 3$*, with at most* k *true variables.*
- HITTING SET(m) *and* EXACT HITTING SET(m)*, with* m *sets.*
- SET COVER(n) *and* EXACT SET COVER(n)*, with* n *elements.*

The following colorful variants are helpful for our reductions.

Lemma 12 ([18, 16])**.** *The following equivalences hold.*

- MULTICOLORED CLIQUE$(k \log n) \equiv_{ppt}$ CLIQUE$(k \log n)$.
- MULTICOLORED HITTING SET$(m) \equiv_{ppt}$ HITTING SET(m).

We now proceed with the reductions. For many problems, we will find it convenient to show hardness by reduction from EXACT HITTING SET(m) or HITTING SET(m); hence we begin by showing the completeness of these problems. We give a chain of reductions from CLIQUE$(k \log n)$, via EXACT HITTING SET(m) and NDTM HALTING$(k \log n)$, and back to CLIQUE$(k \log n)$, closing the cycle. After this we will treat the HITTING SET(m) problem. Note that NDTM HALTING$(k \log n)$ and BINARY NDTM HALTING(k) (the problem restricted to machines with a binary tape alphabet) are easily PPT-equivalent.

Lemma 13. MULTICOLORED CLIQUE$(k \log n) \leq_{ppt}$ EXACT HITTING SET(m).

Proof. Let G be a graph on n vertices in an instance of MULTICOLORED CLIQUE$(k \log n)$, and let $c : V(G) \to [k]$ be the coloring function of G. We assume that $V(G) = [n]$, and we let $b_\ell(v)$ denote the ℓ'th bit in the binary expansion of $v \in V(G)$. The instance $(U, \mathcal{F})$ of EXACT HITTING SET(m) is constructed by taking $U = E(G)$ and defining $\mathcal{F}$ to be the following subsets of U:

- $F_{i,j} := \{uv \in E(G) : c(u) = i,\ c(v) = j\}$ for all $1 \leq i < j \leq k$.
- $F_{i,j,j',\ell} := \{uv \in E(G) : c(u) = i, c(v) = j, b_\ell(u) = 1\} \cup \{uv \in E(G) : c(u) = i, c(v) = j', b_\ell(u) = 0\}$ for every pair of color pairs (i, j) and (i, j') with $j \neq j'$, and all $1 \leq \ell \leq \log n$.

Observe that any exact hitting set $U^* \subseteq U$ for $(U, \mathcal{F})$ consists of $\binom{k}{2}$ edges, one for each pair of distinct colors $i, j \in [k]$. The sets $F_{i,j,j',\ell}$, $1 \leq \ell \leq \log n$, ensure that any pair of edges $e \neq e' \in U^*$ with one endpoint colored i are incident with the same vertex $v \in V(G)$ with $c(v) = i$. Otherwise, if e is incident with an i-colored vertex v, and e' is incident with an i-colored vertex $v' \neq v$, then $b_\ell(v) \neq b_\ell(v')$ for some $\ell \in \{1, \ldots, \log n\}$, and e and e' are both in some $F_{i,j,j',\ell}$ for this specific ℓ. Thus, any exact hitting set of $(U, \mathcal{F})$ corresponds to a multicolored clique in G. Conversely, the edge-set of any multicolored clique in G is an exact hitting set of $(U, \mathcal{F})$. As $m := |\mathcal{S}| = O(k^2 + k \log n)$, our construction is a polynomial parametric transformation, and so MULTICOLORED CLIQUE$(k \log n) \leq_{ppt}$ EXACT HITTING SET(m). □

Lemma 14. EXACT HITTING SET$(m) \leq_{ppt}$ NDTM HALTING$(k \log n)$.

Proof. Let $(U, \mathcal{F})$ be an instance of EXACT HITTING SET(m), with $U = [n]$ and $\mathcal{F} = \{F_1, \ldots, F_m\}$. By identifying vertices which are included in the same set of edges, we may assume that $n \leq 2^m$. We create a Turing machine M with alphabet $[n]$, which writes down m values specifying the selected member of each set, followed by simple poly(m)-time verification. Specifically, let the m selections be $u_1, \ldots, u_m$. Then the machine has to verify that $u_i \in F_i$ for each $i \in [m]$, and that for every pair $i, j \in [m]$, either $u_i = u_j$ or $u_i \in (F_i \setminus F_j)$ and $u_j \in (F_j \setminus F_i)$. By encoding this information in the state space of M, we get a machine of size n' which is polynomial in $n + m$, that can nondeterministically verify in $k = O(m^2)$ steps whether $(U, \mathcal{F})$ has an exact hitting set. Since $\log n' = O(m)$, this is indeed a PPT. □

Lemma 15. BINARY NDTM HALTING$(k) \leq_{ppt}$ MIN ONES 3-SAT(k).

Proof. Let (M, k) be the input to BINARY NDTM HALTING(k), where M is a Turing machine of size n with s states and ℓ edges in its transition diagram. We construct a 3-CNF formula ϕ such that ϕ has a satisfying assignment of Hamming weight k', value to be chosen later, iff M accepts the empty string in k steps. For this we create the following variable groups:

- $S_{i,t}$, $i \in [s]$ and $t \in \{0\} \cup [k]$, signifying that the machine is in state number i after t steps.
- $M_{e,t}$, $e \in [\ell]$ and $t \in [k]$, signifying that transition e of the machine is followed as step t.
- $H_{p,t}$, $0 \leq p, t \leq k$, signifying that the machine's tape head is in position p after t steps.
- $T_{p,t}$, $0 \leq p, t \leq k$, signifying that position p of the tape contains value 1 after t steps.
- $\overline{T}_{p,t}$, $0 \leq p, t \leq k$. These are constrained so that $T_{p,t} \neq \overline{T}_{p,t}$ for all values of p and t, allowing us to control the weight of any satisfying assignment for ϕ.

We add two groups of clauses. The first ensures that for every $0 \leq t \leq k$ exactly one variable $S_{i,t}$ is true (and similarly for variable groups $M_{e,t}$ and $H_{p,t}$). The second group ensures that these assignments are consistent with an execution of M.

For the first group of clauses, we use the *log-cost selection formula* construction of [28]. Specifically, we employ [28, Lemma 5, item 1] with a relation $R(x,y,z) = \{(0,0,0),(1,1,0),(1,0,1)\}$; note that such a relation R can easily be implemented by 3-clauses. Given a set of variables $\{x_1,\ldots,x_n\}$, this creates a formula F_n of polynomial size in n, with additional (internal) variables y_i, which has a satisfying assignment if and only if exactly one variable $x_i, i \in [n]$ is true; furthermore, every satisfying assignment to F_n has Hamming weight exactly $w_n = \mathcal{O}(\log n)$ among the variables y_i. We create one such formula F_n for each group of variables $\{S_{i,t} : i \in [s]\}$, $0 \leq t \leq k$; $\{M_{e,t} : e \in [\ell]\}$, $t \in [k]$; and $\{H_{p,t} : 0 \leq p \leq k\}$, $0 \leq t \leq k$, in each case keeping the internal variables distinct. We note that every satisfying assignment to this part of the formula has a specific known weight of $W(k,\ell,s) = \mathcal{O}(k \log \ell)$ among all internal variables y (assuming that $s \leq \ell$, e.g., that every state i is reachable in the diagram).

For the second group, we first add clauses of size 2 to ensure that $T_{p,t} \neq \overline{T}_{p,t}$ for every value of p and t. We additionally enforce, by negative clauses of size 1, that the final state of the machine is an accepting state. We then add clauses enforcing consistency of states, head, tape, and transition variables. For example, if transition e moves from state i to state j, reading a 0, writing a 1, and moving right, then we have constraints: $(M_{e,t} \wedge H_{p,t-1} \to \neg T_{p,t-1})$, $(M_{e,t} \wedge H_{p,t-1} \to T_{p,t})$, $(M_{e,t} \wedge H_{p,t-1} \to H_{p+1,t})$, $(M_{e,t} \to S_{i,t-1})$, and $(M_{e,t} \to S_{j,t})$, for all possible values of p and t. We complete the construction of ϕ by adding clauses encoding $(\neg H_{p,t} \to (T_{p,t} = T_{p,t+1}))$ for all p and t, and clauses for the initial setup of the machine (e.g., head position 0, state 1, all tape entries 0). This finishes the construction of ϕ.

As a final step of our reduction we set $k' = W(k,\ell,s) + 2(k+1) + (k+1)^2 + k$. This accounts for the weight of the variables y in the first group, as well as the fact that exactly one variable among the S-, H-, and M-variables is set to true, and that the total Hamming weight of the T- and $\overline{T}$-variables is $(k+1)^2$. The formula ϕ constructed above is satisfiable by an assignment of Hamming weight k' if and only if M accepts the empty string in k steps. As $W(k,\ell,s) = O(k \log \ell)$, we again have two cases. If $\log \ell > k$, then we may solve the problem exactly in polynomial time; otherwise, we have $k' = \mathcal{O}(k^2)$. As the reduction runs in polynomial time, we have a PPT. □

The following lemma is a direct consequence of Theorem 1 in Section 3.

Lemma 16. MIN ONES d-SAT$(k) \leq_{ppt}$ CLIQUE$(k \log n)$ *for every fixed* d.

The remaining problems in Theorem 5 for which we need to show completeness are HITTING SET(m), SET COVER(n), and EXACT SET COVER(n). It is well known that HITTING SET$(m) \equiv_{ppt}$ SET COVER(n) and EXACT HITTING SET$(m) \equiv_{ppt}$ EXACT SET COVER(n), thus we finish the proof of Theorem 5 by proving WK[1]-completeness for HITTING SET(m).

Lemma 17. HITTING SET(m) *is* WK[1]*-complete.*

Proof. HITTING SET(m) can be shown to be in WK[1] by a similar argument used in Lemma 14. To show WK[1]-hardness, we reduce from EXACT HITTING SET(m). Let $(U,\mathcal{F})$ be an input instance to EXACT HITTING SET(m), with $\mathcal{F} = \{F_1,\ldots,F_m\}$ and $U := [n]$. We can assume $\log n \leq k$ by identifying identical vertices. Let $V = \bigcup_{i\in[m]}\{v_{i,j} : j \in F_i\}$; this will be the vertex set of our HITTING SET(m) instance. To ensure consistency between the selections in different sets, we consider the binary expansions of the vertices in U. Let $i,i' \in [m]$, $i \neq i'$, and $\ell \in [\log n]$. Let $A_{(i,i',\ell)}$ consist of vertices $v_{i,j}$ for every $j \in F_i \cap F_{i'}$ such that the ℓ'th bit of the binary expansion of j is 0, and $B_{(i,i',\ell)}$ consist of vertices $v_{i',j}$ for every $j \in F_i \cap F_{i'}$ such that the ℓ'th bit of the binary expansion of j is 1. Let $C_{(i,i')} = \{v_{i,j} : j \in F_i \setminus F_{i'}\}$. We add the edge $E_{(i,i',\ell)} = A_{(i,i',\ell)} \cup B_{(i,i',\ell)} \cup C_{(i,'i)}$ to the HITTING SET(m) instance for all $i,i' \in [m]$, $i \neq i'$, and $\ell \in [\log n]$. Finally, for every $i \in [m]$, we add the edge $E_i = \{v_{i,j} : j \in F_i\}$ to our output instance.

Let $\mathcal{E}$ denote the resulting edge set, and set the desired solution size to m. If $(U,\mathcal{F})$ has an exact hitting set, then this immediately gives a hitting set for $(V,\mathcal{E})$ of size m: For every pair of edges F_i and $F_{i'}$, either their intersection is hit, in which case exactly one of $A_{(i,i',\ell)}$ and $B_{(i,i',\ell)}$ is hit for every ℓ,

or the symmetric difference is hit in both sets, in which case $C_{(i,'i)}$ is hit. On the other hand, assume that $(V, \mathcal{E})$ has a hitting set of size m; then the edges E_i ensure that this hitting set corresponds to selecting one vertex from each edge $F_i \in \mathcal{F}$. We argue that these vertices must form an exact hitting set for $(U, \mathcal{F})$. Consider thus a pair of sets F_i and $F_{i'}$. Clearly, one vertex is selected in each edge. If both selected vertices are in $F_i \cap F_{i'}$, but the vertices differ, then one of the edges $E_{(i,i',\ell)}$ or $E_{(i',i,\ell)}$ is not hit, where ℓ is a bit distinguishing the binary expansions of these two vertices. If the vertex selected from F_i lies in $F_i \cap F_{i'}$, but the vertex selected from $F_{i'}$ lies in the symmetric difference of F_i and $F_{i'}$, let ℓ be a bit equalling 1 in the binary expansion of the vertex selected from F_i (this exists by construction). Then the edge $E_{(i,i',\ell)}$ is not hit: Clearly, $B_{(i,i',\ell)}$ and $C_{(i,i')}$ are not hit, and by choice neither is $A_{(i,i',\ell)}$. Thus, if the two selected vertices are two distinct vertices, both these vertices must lie in the symmetric difference of F_i and $F_{i'}$. This implies that for every pair of sets, the vertices selected from these sets are either identical vertices in the intersection, or two distinct vertices in the symmetric difference. It follows that the set of all selected vertices forms a valid exact hitting set for $(U, \mathcal{F})$. □

4.2 Path and cycle problems

In this section we prove WK[1]-hardness and completeness of several problems regarding existence of paths and cycles in a given graph. Concretely, we address DISJOINT PATHS(k) and DISJOINT CYCLES(k), and multicolored variants of finding a path or cycle on k vertices (i.e, PATH(k) and CYCLE(k)) in directed or undirected graphs. Our results are as follows.

Theorem 6. *The* DISJOINT PATHS(k) *and* DISJOINT CYCLES(k) *problems are* WK[1]*-hard.*

Theorem 7. *The following problems are complete for* WK[1]*:*

- MULTICOLORED PATH(k) *and* DIRECTED MULTICOLORED PATH(k).
- MULTICOLORED CYCLE(k) *and* DIRECTED MULTICOLORED CYCLE(k).

We note that the "uncolored" versions of both problems in Theorem 7 are in WK[1], but we do not know whether they are complete or not. Nevertheless, the above four problems are interesting in their own right, and algorithms for them are used as subroutines in the classical color-coding algorithms for their uncolored counterparts [3].

To prove both theorems above we go through an intermediate problem known as the DISJOINT FACTORS(k) problem, and mimic the construction used in [9] to show that this problem is WK[1]-complete. The input to DISJOINT FACTORS(k) is a string S of length n over the alphabet $[k]$. The goal is to determine whether there exists a set of k disjoint substrings $S_1, \ldots, S_k$ of S, where S_i of the form $i \cdots i$ (i.e., a factor) for each $i \in [k]$. Bodlaender et al. [9] show that this problem is solvable in $2^{\mathcal{O}(k)} \cdot n$ time, and thus is in EXPT.

Lemma 18. *The* DISJOINT FACTORS(k) *problem is* WK[1]*-hard.*

Proof. We construct a PPT from MULTICOLORED HITTING SET(m) which is WK[1]-hard according to Theorem 5. Given an instance $(V, \mathcal{E}, c, k)$ of MULTICOLORED HITTING SET(m), with $|V| = n$, $|\mathcal{E}| = m$, and $c : V \to [k]$, we create a string S of size polynomial in $n + \bigcup_{E \in \mathcal{E}} |E|$ as an instance of DISJOINT FACTORS as follows: Our alphabet will consist of one symbol $\mathbf{e}$ for every edge $e \in \mathcal{E}$, and of at most $k \cdot \lceil \log n \rceil$ auxiliary symbols: For every color $i \in [k]$, we have $\lceil \log |V_i| \rceil$ symbols $\mathbf{a}^i_1, \ldots \mathbf{a}^i_{\lceil \log |V_i| \rceil}$, where $V_i \subseteq V$ is the subset of vertices with $c(v) = i$. Clearly we can identify vertices which are included in the same set of edges, and therefore we can assume that $\log n = \mathcal{O}(m)$, which makes our reduction a PPT. Our string S will contain substrings corresponding to vertices; to a vertex $v \in V$ contained in the edges $E_1, \ldots E_\ell \in \mathcal{E}$, we assign the substring $S_v = \mathbf{e}_1 \mathbf{e}_1 \ldots \mathbf{e}_\ell \mathbf{e}_\ell$. The symbols corresponding to edges will appear only inside such substrings $S(v)$. For every color $i \in [k]$, we will

build a "selection gadget" for choosing a vertex of the given color analogous to the one described in [9, Lemma 2]. The gadget will ensure that we are able to pick factors contained in $S(v)$ if and only if v is the chosen vertex from its color class.

Let us describe the selection gadget for color $i \in [k]$ more precisely. We assume that the number of vertices n_i of this color is a power of 2 (otherwise we can just add isolated vertices), and we let $V_i = \{v_1, \ldots, v_{n_i}\}$. If we had only two vertices v_1 and v_2 to pick from, we could implement the gadget in the following manner: $S[1,2] = \mathbf{a}_1^i S(v_1) \mathbf{a}_1^i S(v_2) \mathbf{a}_1^i$, where $\mathbf{a}_1^i$ is some symbol which does not appear inside $S(v_1)$ nor $S(v_2)$. By choosing the factor for $\mathbf{a}_1^i$, we prevent selecting factors from either $S(v_1)$ or $S(v_2)$, which corresponds to selecting whether to hit all sets which include v_1, or all sets which include v_2. We can apply this construction in a recursive manner; if the substring $S[1, 2^j]$ implements the selection of a vertex from $\{v_1, \ldots, v_{2^j}\}$, and if $S[2^j + 1, 2^{j+1}]$ implements the selection of a vertex from $\{v_{2^j+1}, \ldots, v_{2^{j+1}}\}$, then the substring $\mathbf{a}_{j+1}^i S[1, 2^j] \mathbf{a}_{j+1}^i S[2^j + 1, 2^{j+1}] \mathbf{a}_{j+1}^i$ selects a single vertex from $\{v_1, \ldots, v_{2^{i+1}}\}$ (note that we need only one auxiliary symbol per level of the recursion; symbols $\mathbf{a}_1^i, \ldots, \mathbf{a}_j^i$ appear inside both $S[1, 2^j]$ and $S[2^j + 1, 2^{j+1}]$). It is easy to check that the only way of selecting factors for every edge symbol $\mathbf{e}$ is selecting in the described gadgets substrings $S(v)$ corresponding to vertices in a multicolored hitting set of size k for $(V, \mathcal{E}, c)$. The lemma thus follows. □

Bodlaender et al. [9] provide polynomial parametric transformations from DISJOINT FACTORS(k) to DISJOINT CYCLES(k) and DISJOINT PATHS(k). This, in combination with the lemma above provides the proof for Theorem 6. To prove Theorem 7, we show that DIRECTED MULTICOLORED PATH(k) and MULTICOLORED PATH(k) are WK[1]-complete. The corresponding cycle problems in Theorem 7 follow immediately from this, and are thus omitted.

Lemma 19. *The* DIRECTED MULTICOLORED PATH(k) *is* WK[1]*-complete.*

Proof. It is easy to see that DIRECTED MULTICOLORED PATH$(k) \in$ WK[1], by reducing this problem to NDTM HALTING$(k \log n)$. Let G be a directed k-colored graph on n vertices given as input to DIRECTED MULTICOLORED PATH(k). First note that as DIRECTED MULTICOLORED PATH(k) can be solved in $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time [3], we can assume $\log n = \mathcal{O}(k)$. We construct a Turing machine M that encodes within its state-space the adjacency matrix of G. It is easy to see that such a Turing machine can be programmed to determine non-deterministically in $\mathcal{O}(k)$ steps whether G has a multicolored path on k vertices, by guessing k vertices of different colors in G, and then checking whether these vertices form a path. Since $|M| = \mathcal{O}(n^2)$ and $\log n = \mathcal{O}(k)$, this gives the desired PPT.

To show hardness, we reduce from DISJOINT FACTORS(k). Given an input string S to DISJOINT FACTORS(k) over the alphabet $[k]$, we construct a directed k-colored graph G which has a vertex corresponding to each factor of S. Each vertex is colored according to the starting (or ending) letter of its corresponding factor, and there is an edge (u, v) in G if the factor corresponding to u is strictly to the left of the factor corresponding to v in S. It is easy to verify that G has a multicolored path of length k iff S has k disjoint factors. Thus, DISJOINT FACTORS$(k) \leq_{ppt}$ DIRECTED MULTICOLORED PATH(k), and the lemma is proven. □

Lemma 20. MULTICOLORED PATH(k) *is* WK[1]*-complete.*

Proof. The argument showing that MULTICOLORED PATH$(k) \in$ WK[1] is very similar to the one used for DIRECTED MULTICOLORED PATH(k). To show hardness, we provide a PPT from DIRECTED MULTICOLORED PATH(k). Let G be a directed k-colored graph on n vertices given as input to DIRECTED MULTICOLORED PATH(k). We construct a k'-colored graph G' on $\mathcal{O}(n)$ vertices with $k' = 3k + 4$. First, we split each vertex v of color c in G into three vertices: v_{in}, v_{mid}, and v_{out}, of colors c_{in}, c_{mid} and c_{out} respectively, forming a path v_{in}, v_{mid}, v_{out} in G'. A directed edge (u, v) in G will be transformed to an edge $\{u_{out}, v_{in}\}$ in G'. We add to G' four additional vertices s_{in}, s_{out}, t_{in} and t_{out}, and assign

to them four unique colors. The source vertex s_{in} will be connected only to s_{out}, and analogously the sink vertex t_{out} will be connected only to t_{in}. We connect s_{out} to every vertex v_{in}, $v \in V(G)$, and we connect t_{in} to every v_{out}, $v \in V(G)$.

Note that a multicolored path of length k' in G' must have s_{in} and t_{out} as its endpoints: These vertices are the only representatives of their color classes, and therefore have to appear on the path, and both have degree exactly one, which means that they can only be endpoints of the path. Another property of G' is the fact that any multicolored path which visits one of the vertices v_{in}, v_{mid} or v_{out} has to visit the other two vertices as well in a consecutive manner, which can be proven by a straightforward case analysis. A path starting at s_{in} will therefore always follow the directed edges of G in the "right direction": When arriving at $v_i n$ via some incoming edge it will proceed to v_{mid} and v_{out} and then take some outgoing edge. This observation shows that G has a path of length k iff G' has a path of length k'. Since our construction is clearly a PPT, the lemma is proven. □

4.3 Local Circuit SAT

In this section we focus on the LOCAL CIRCUIT SAT$(k \log n)$ problem, which was introduced by Harnik and Naor [21] and plays a crucial role in their incompressibility hierarchy. The input to the LOCAL CIRCUIT SAT$(k \log n)$ problem is a string of length n and a circuit C with $k + k \log n$ inputs, of size $k + k \log n$. An instance is positive if there are k positions $i_1, \ldots, i_k$ in the string such that feeding the contents of the positions to the first k inputs, and the binary expansions of $i_1, \ldots, i_k$ to the remaining inputs, causes the circuit to accept. As Harnik and Naor note, we may equivalently assume the circuit to have size polynomially bounded in $k \log n$, rather than exactly $k + k \log n$.

Theorem 8. LOCAL CIRCUIT SAT$(k \log n)$ *is complete for* WK[1].

Proof. To show WK[1]-hardness, we reduce from CLIQUE$(k \log n)$. Let an instance (G, k) be given. Assume that $n = 2^\ell$, or else pad the instance with isolated vertices (at most doubling the size). The input to the circuit is the adjacency matrix written row by row as a string of length $n' = n^2$, modified to have a 1 in each diagonal entry (for ease of presentation). The circuit gets input from k^2 positions, with each position coded in $\log n' = 2 \log n$ bits; here $n = 2^\ell$ ensures that there is a trivial way of converting between matrix positions and places in the string. The first part of the string thus provides the circuit with k^2 entries of the adjacency matrix, and it simply checks that these are all ones. The second part contains the position and the circuit checks that all the positions can be obtained by taking all k^2 combinations of concatenating the numbers of two vertices (size $\log n$). For this we may fix any desired ordering of the positions, hence the checking comes down to hardwired equalities in an appropriate way (e.g. we might decide that the first k positions correspond to the first vertex hence the first $\log n$ bit-positions of the first k positions should be bitwise the same). It is easy to check the correspondence between a k-clique in G and a choice of variable assignments such that the circuit accepts.

To show membership in WK[1], we reduce to BINARY NDTM HALTING(k) which is WK[1]-complete according to Theorem 5. The number of steps will be polynomial in $k \log n$. It is well known that a Turing machine can be constructed to simulate a fixed circuit in a number of steps that is polynomial in the circuit size. To simulate an instance of LOCAL CIRCUIT SAT we additionally encode the input string of length n into the Turing machine. The machine guesses the k positions in $k \log n$ steps, writes the according contents of the string onto the tape, followed by the positions, and then simulates the circuit. □

4.4 Further problems

In this final section regarding problems that are complete or hard for WK[1] we address several problems for which (many-one) kernelization lower bounds were obtained in previous work of Dom et al. [16].

From their work we immediately get WK[1]-hardness for four of the following problems. To complete the theorem, we mainly need to provide WK[1]-membership proofs.

Theorem 9. *The following problems are complete for* WK[1]*:*

- CONNECTED VERTEX COVER(k).
- CAPACITATED VERTEX COVER(k).
- STEINER TREE$(k+t)$.
- SMALL SUBSET SUM(k).
- UNIQUE COVERAGE(k).

The first four problems in the theorem above all have polynomial parametric transformations from HITTING SET(m) [16, Theorems 2 and 7], and are thus WK[1]-hard by Lemma 17. To see that UNIQUE COVERAGE(k) is also WK[1]-hard, consider the following easy PPT from EXACT HITTING SET(m). Let $(V, \mathcal{E})$ be the input instance with $|\mathcal{E}| = m$, and recall that we can assume as usual that $\log n \leq m$. For each $v \in V$ make a set F_v containing all sets $E \in \mathcal{E}$ with $v \in E$. It is easy to see that any set $\mathcal{F}' \subseteq \{F_v : v \in V\}$ which uniquely covers $k = m$ elements of $\mathcal{F}$ directly corresponds to an exact hitting set for $(V, \mathcal{E})$. Thus, all problems in Theorem 9 are WK[1]-hard, and so to complete the proof of the theorem we show that all these problems are contained in WK[1].

For all five problems, membership in WK[1] is shown by PPTs to NDTM HALTING$(k \log n)$. In all cases the input will be included in the description of the Turing machine to allow for a fast verification of guessed solutions. For STEINER TREE$(k+t)$ and SMALL SUBSET SUM(k) the PPTs are straightforward. The PPT for UNIQUE COVERAGE(k) is also straightforward once we realize that it sufficient to consider a solution of size at most k, and that the $\mathcal{O}(4^k)$-size kernel of Misra et al. [31] lets us assume that the logarithm of the input size is polynomially bounded in k.

Lemma 21. STEINER TREE$(k+t)$*,* SMALL SUBSET SUM(k)*, and* UNIQUE COVERAGE(k) *all have* PPT*s to the* NDTM HALTING$(k \log n)$ *problem.*

For the two remaining vertex cover variants in Theorem 9 the reductions are a bit more subtle. Both of them utilize the so-called Buss rule used in the classical $\mathcal{O}(k^2)$ kernel for VERTEX COVER(k) [10]. This allows the output Turing machine in the reduction to quickly verify solutions that it guesses. More details are given in the proofs of the two lemmas below.

Lemma 22. CONNECTED VERTEX COVER$(k) \leq_{ppt}$ NDTM HALTING$(k \log n)$.

Proof. Given an instance (G, k) of CONNECTED VERTEX COVER(k), we first identify the set T of all vertices of degree greater than k, and output a Turing machine that never halts if $|T| > k$ since the vertices of T must be in every vertex cover of size k for G. The remaining budget $k' = k - |T|$ must be spend on a set N of vertices which covers all edges not incident on T and such that $G[T \cup N]$ is connected. Since all vertices outside T have degree at most k there can be at most $\binom{k}{2}$ uncovered edges, or we may output a machine that never halts (as vertices in N cover at most $k|N|$ edges altogether). We construct a Turing machine M by encoding in its state-space the set of at most $\binom{k}{2}$ uncovered edges, the set T, the graph G, and the budget k'. It is now not difficult to see that by properly programming M, the machine will determine in $k^{\mathcal{O}(1)}$ non-deterministic steps whether G has a connected vertex cover of size k. □

Lemma 23. CAPACITATED VERTEX COVER$(k) \leq_{ppt}$ NDTM HALTING$(k \log n)$.

Proof. Let (G, α, k) be an instance of CAPACITATED VERTEX COVER(k), where $G = (V, E)$ is a graph on n vertices. We again identify the set T of vertices with degree exceeding k in G, and output a non-halting machine in case $|T| > k$. We also output a non-halting machine if the capacity of some vertex in T is lower than the number of its incident edges by more than k; conversely we may delete a vertex of T if its capacity suffices for all incident edges (decreasing k by one). By selecting a set N of $k - |T|$ further vertices at most k^2 edges can be covered; this includes edges not incident with T but also edges incident with some vertex of T for which the budget does not suffice. The output Turing machine M is hardwired with an encoding of the set T along with the input graph, including the degrees and capacity of all vertices, to essentially allow random access to all these values. The machine M proceeds as follows:

1. Guess a set N of at most $k' = k - |T|$ vertices. Verify that all edges have at least one endpoint in $T \cup N$ (this only has to be done for the at most k^2 edges not incident to T).
2. For every edge with both endpoints in $T \cup N$, guesses the vertex which covers it. Register on the tape for every vertex in $T \cup N$ the number of incident edges that the vertex does not have to cover (due to it being covered at its other endpoint).
3. Verify for every vertex in $T \cup N$ that its capacity suffices to cover all remaining edges, that is, that its capacity is at least its degree minus the number of edges covered by another vertex.

It can be verified that the required number of steps can be bounded by a polynomial in k and that the total machine size and construction time are polynomial in n. □

5 Problems in Higher Levels

In this section we investigate the second level of the MK- and WK-hierarchies, and present some complete and hard problems for these classes.

5.1 MK[2]-complete Problems

According to Theorem 1, MK[2] is the PPT-closure of the classical CNF satisfiability problem where the parameter is taken to be the number of variables in the input formula. The PPT-equivalence of this problem to HITTING SET(n) and SET COVER(m) is well known.

Theorem 10. HITTING SET(n) *and* SET COVER(m) *are complete for* MK[2].

Proof. CNF-SAT(n) is equivalent to $\Gamma_{2,1}$-SAT(n), which is MK[2]-complete by Theorem 1. HITTING SET(n) is equivalent to CNF-SAT(n) by well-known reductions: On the one hand, we can reduce from CNF-SAT to HITTING SET by introducing two vertices $v, \bar{v}$ for every variable v, in addition to sets $\{v, \bar{v}\}$, and letting $k = n$. On the other hand, a HITTING SET instance can be directly interpreted as a purely positive CNF-SAT formula, to which we only need to add a (polynomial-sized) counting formula to restrict the number of true variables to k (which is involved, but not difficult to do). □

Corollary 1. *The following problems are hard for* MK[2]*:*

- DOMINATING SET(k, c) *where* c *is the size of a minimum vertex cover.*
- HITTING SET(k, d) *where* d *is the maximum size of sets.*
- DOMINATING SET(k, d) *where* d *is the degeneracy.*
- DOMINATING SET$(k, |H|)$ *on* H*-minor free graphs.*

Proof. This follows from [16, Theorems 5 and 6]. The first follows by a PPT from HITTING SET(n), the others follow by direct consideration of parameters. (Respectively: $k, d \leq n$, thus the HITTING SET problem is hard; the degeneracy is bounded by the vertex cover size; and vertex cover size c excludes a K_{c+2}-minor.) □

5.2 WK[2]-complete Problems

Theorem 11. *The following problems are complete for* WK[2]*:*

- HITTING SET$(k \log n)$ *and* SET COVER$(k \log m)$.
- DOMINATING SET$(k \log n)$ *and* INDEPENDENT DOMINATING SET$(k \log n)$.
- STEINER TREE$(k \log n)$.

For the first four problems in Theorem 11, the results follow easily. The PPT-equivalence between $\Gamma_{2,1}$-WSAT$(k \log n)$, HITTING SET$(k \log n)$, SET COVER$(k \log m)$, and DOMINATING SET$(k \log n)$ are well known, and for INDEPENDENT DOMINATING SET$(k \log n)$, a PPT to $\Gamma_{2,1}$-WSAT$(k \log n)$ is trivial and a PPT from DOMINATING SET$(k \log n)$ can be produced by standard methods.

The story is different with STEINER TREE$(k \log n)$. While WK[2]-hardness for this problem follows immediately from, e.g., the PPT from HITTING SET$(k \log n)$ given in [16], showing membership in WK[2] is more challenging. To facilitate this and other non-trivial membership proofs, we consider the issue of a machine characterization of WK[2], similarly to the WK[1]-complete NDTM HALTING$(k \log n)$ problem. The natural candidate would be MULTI-TAPE NDTM HALTING$(k \log n)$, as this same problem with parameter k is W[2]-complete [12]. However, while the problem with parameter $k \log n$ is easily shown to be WK[2]-hard, we were so far unable to show WK[2]-membership. On the other hand, the following extension of a single-tape non-deterministic Turing machine leads to a WK[2]-complete problem, which we name NDTM HALTING WITH FLAGS.

Definition 6. *A* (single-tape, non-deterministic) Turing machine with flags *is a standard (single-tape, non-deterministic) Turing machine which in addition to its working tape has access to a set F of* flags*. Each state transition of the Turing machine has the ability to read and/or write a subset of the flags. A transition that* reads *a set $S \subseteq F$ of flags is only applicable if all flags in S are active. A transition that* writes *a set $S \subseteq F$ of flags causes every flag in S to become active. In the initial state, all flags are inactive. Note that there is no operation to deactivate a flag.*

Theorem 12. NDTM HALTING WITH FLAGS$(k \log n)$ *is* WK[2]*-complete.*

Proof. Showing WK[2]-hardness is easy by reduction from HITTING SET$(k \log n)$. In fact, the hitting set instance can be coded directly into the flags, without any motion of the tape head – simply construct a machine that non-deterministically makes k non-writing transitions, each corresponding to including a vertex in the hitting set, followed by one verification step. The machine has m flags, one for every set in the instance, and a step corresponding to selecting a vertex v activates all flags corresponding to sets containing v. Finally, the step to the accepting state may only be taken if all flags are active. By assuming $\log m \leq k \log n$ (or else solving the instance exactly) we get a PPT.

Showing membership in WK[2] can be done by translation to $\Gamma_{2,1}$-WSAT$(k \log n)$. The transition is similar to that in Lemma 15. The only complication is to enforce consistency of transitions which read and write sets of flags, but this is easily handled. Let $M_{e,t}$ signify that step number t of the machine follows edge e of the state diagram (as in Lemma 15). If transition e has a flag f as a precondition, then we simply add a clause

$$(\neg M_{e,t} \vee M_{e_{i_1},1} \vee \ldots \vee M_{e_{i_m},1} \vee \ldots \vee M_{e_{i_1},t-1} \vee \ldots \vee M_{e_{i_m},t-1}),$$

where $e_{i_1}, \ldots, e_{i_m}$ is an enumeration of all transitions in the state diagram which activate flag f. The rest of the reduction proceeds without difficulty. □

Lemma 24. STEINER TREE$(k \log n) \leq_{ppt}$NDTM HALTING WITH FLAGS$(k \log n)$.

Proof. As mentioned above, WK[2]-hardness for STEINER TREE$(k \log n)$ follows from the PPT from HITTING SET$(k \log n)$ given in [16]. We show membership in WK[2] by a reduction to the Turing machine problem with flags. Let (G, T, k) be an instance of STEINER TREE. We make the following observations.

1. If two terminals $t, t' \in T$ are neighbors in G, they may be identified.
2. Assume that no two terminals are neighbors in G. Let G' be G with $N(t)$ replaced by a clique for every $t \in T$. Then, for any solution S, the graph $G'[S]$ must be connected.

In fact, a set S is a Steiner tree if and only if $G'[S]$ is connected and every $t \in T$ is neighbor to a vertex in S. Thus, perform the reduction to G' as described. The Turing machine then goes in two phases. First, it guesses the solution S consisting of k vertices, and checks (in poly(k) time) the connectivity of $G'[S]$. Second, using the flags as in Theorem 12, it goes through the vertices of S and verifies that every terminal is neighbor to at least one vertex. It accepts if both tests pass. □

Finally, we point out that existing W[2]-hardness proofs in the literature can often be seen to also yield WK[2]-hardness for the reparameterized versions of the problems. As an example, Heggernes et al. [22] proved W[2]-hardness of PERFECT DELETION(k) and WEAKLY CHORDAL DELETION(k) by reductions from HITTING SET(k); the reductions can be seen to also be PPTs from HITTING SET$(k \log n)$ to PERFECT DELETION$(k \log n)$ and WEAKLY CHORDAL DELETION$(k \log n)$, respectively.

Corollary 2. *The following problems are hard for* WK[2]*:*

- PERFECT DELETION$(k \log n)$.
- WEAKLY CHORDAL DELETION$(k \log n)$.

6 Discussion

We have defined a hierarchy of PPT-closed classes, akin to the M- and W-hierarchy of parameterized intractability, in order to build up a completeness theory for polynomial (Turing) kernelization. The fundamental hardness class is called WK[1] and we conjecture that no WK[1]-hard problem admits a polynomial Turing kernelization. At present, the state of the art in lower bounds for kernelization does not seem to provide a way to connect this conjecture to standard complexity assumptions. However, there is collective evidence in favor of the conjecture by the wealth of natural problems that are complete for WK[1] and for which polynomial Turing kernels seem unlikely. (Recall that the admittance of Turing kernels is preserved by PPTs and hence a single polynomial Turing kernel would transfer to all WK[1] problems.) Of course, our examples provide only a partial image of the WK[1] landscape. For example, the various kernelizability dichotomies that have been shown for CSP problems [28, 27] can be shown to imply dichotomies between problems with polynomial kernels and WK[1]-complete problems (and in some cases the third class of W[1]-hard problems). We take this as further evidence of the naturalness of the class.

On the more structural side, we have discussed the relation to the earlier VC-hierarchy of Harnik and Naor [21] which, from our perspective, is restricted to NP-problems parameterized by witness size.

Under this interpretation their hierarchy folds into ours, with the levels of their hierarchy mapping to a subset of the levels of our hierarchy.

Many questions remain. One is the WK[1]-hardness of PATH(k) and CYCLE(k); for these problems, we have only lower bound proofs in the framework of Bodlaender et al. [6], leaving the question of Turing kernels open. (Very recently, Jansen [24] showed polynomial Turing kernels for PATH(k) and CYCLE(k) for inputs in restricted graph classes, including in particular planar graphs and claw-free graphs.) There are also several problems, including the work on structural graph parameters by Bodlaender, Jansen, and Kratsch, e.g. [7], which we have not investigated. It is also unknown whether MULTI-TAPE NDTM HALTING($k \log n$) is in WK[2]. Furthermore, it would be interesting to know some natural parameterized problems which are WK[2]-complete under a standard parameter (e.g., k rather than $k \log n$).

Still, the main open problem is to provide (classical or parameterized) complexity theoretical implications of polynomial Turing kernelizations for WK[1]. More modest variants of this goal include excluding only OR-kernels, and/or considering the general problem of Turing machine acceptance parameterized by witness length.

References

[1] K. A. Abrahamson, R. G. Downey, and M. R. Fellows. Fixed-parameter tractability and completeness IV: On completeness for W[P] and PSPACE analogues. *Annals of Pure and Applied Logic*, 73(3):235–276, 1995.

[2] N. Alon, G. Gutin, E. J. Kim, S. Szeider, and A. Yeo. Solving MAX-*r*-SAT above a tight lower bound. *Algorithmica*, 61(3):638–655, 2011.

[3] N. Alon, R. Yuster, and U. Zwick. Color coding. *Journal of the ACM*, 42(4):844–856, 1995.

[4] D. Binkele-Raible, H. Fernau, F. V. Fomin, D. Lokshtanov, S. Saurabh, and Y. Villanger. Kernel(s) for problems with no kernel: On out-trees with many leaves. *ACM Transactions on Algorithms*, 8(4):38, 2012.

[5] H. L. Bodlaender, E. D. Demaine, M. R. Fellows, J. Guo, D. Hermelin, D. Lokshtanov, M. Müller, V. Raman, J. van Rooij, and F. A. Rosamond. Open problems in parameterized and exact computation – IWPEC 2008. Technical Report UU-CS-2008-017, Dept. of Informatics and Computing Sciences, Utrecht University, 2008.

[6] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009.

[7] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Kernel bounds for path and cycle problems. *Theoretical Computer Science*, 511:117–136, 2013.

[8] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Kernelization lower bounds by cross-composition. *SIAM Journal on Discrete Mathematics*, 28(1):277–305, 2014.

[9] H. L. Bodlaender, S. Thomassé, and A. Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theoretical Computer Science*, 412(35):4570–4578, 2011.

[10] J. Buss and J. Goldsmith. Nondeterminism within P. *SIAM Journal on Computing*, 22(3):560–572, 1993.

[11] L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996.

[12] M. Cesati. The Turing way to parameterized complexity. *Journal of Computer and System Sciences*, 67(4):654–685, 2003.

[13] M. Cygan, H. Dell, D. Lokshtanov, D. Marx, J. Nederlof, Y. Okamoto, R. Paturi, S. Saurabh, and M. Wahlström. On problems as hard as CNF-SAT. In *IEEE Conference on Computational Complexity*, pages 74–84, 2012.

[14] H. Dell and D. Marx. Kernelization of packing problems. In *Proc. of the 23rd annual ACM-SIAM Symposium On Discrete Algorithms (SODA)*, pages 68–81, 2012.

[15] H. Dell and D. van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In *Proc. of the 42th annual ACM Symposium on Theory Of Computing (STOC)*, pages 251–260, 2010.

[16] M. Dom, D. Lokshtanov, and S. Saurabh. Incompressibility through colors and IDs. In *Proc. of the 36th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 378–389, 2009.

[17] A. Drucker. New limits to classical and quantum instance compression. In *Proc. of the 53rd annual IEEE symposium on Foundations Of Computer Science (FOCS)*, pages 609–618. IEEE Computer Society, 2012.

[18] M. R. Fellows, D. Hermelin, F. A. Rosamond, and S. Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009.

[19] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer-Verlag New York, Inc., 2006.

[20] L. Fortnow and R. Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *Journal of Computer and System Sciences*, 77(1):91–106, 2011.

[21] D. Harnik and M. Naor. On the compressibility of NP instances and cryptographic applications. *SIAM Journal on Computing*, 39(5):1667–1713, 2010.

[22] P. Heggernes, P. van 't Hof, B. M. P. Jansen, S. Kratsch, and Y. Villanger. Parameterized complexity of vertex deletion into perfect graph classes. *Theoretical Computer Science*, 511:172–180, 2013.

[23] D. Hermelin and X. Wu. Weak compositions and their applications to polynomial lower bounds for kernelization. In *Proc. of the 23rd annual ACM-SIAM Symposium On Discrete Algorithms (SODA)*, pages 104–113, 2012.

[24] B. M. P. Jansen. Turing kernelization for finding long paths and cycles in restricted graph classes. arXiv preprint available at `http://arxiv.org/abs/1402.4718`, 2014.

[25] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1975.

[26] S. Kratsch. Co-nondeterminism in compositions: A kernelization lower bound for a Ramsey-type problem. In *Proc. of the 23rd annual ACM-SIAM Symposium On Discrete Algorithms (SODA)*, pages 114–122, 2012.

[27] S. Kratsch, D. Marx, and M. Wahlström. Parameterized complexity and kernelizability of max ones and exact ones problems. In *Proc. of the 35th international symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 489–500, 2010.

[28] S. Kratsch and M. Wahlström. Preprocessing of Min Ones problems: A dichotomy. In *Proc. of the 37th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 653–665, 2010.

[29] S. Kratsch and M. Wahlström. Compression via matroids: a randomized polynomial kernel for odd cycle transversal. In *Proc. of the 23rd annual ACM-SIAM Symposium On Discrete Algorithms (SODA)*, pages 94–103, 2012.

[30] S. Kratsch and M. Wahlström. Two edge modification problems without polynomial kernels. *Discrete Optimization*, 10(3):193–199, 2013.

[31] N. Misra, H. Moser, V. Raman, S. Saurabh, and S. Sikdar. The parameterized complexity of unique coverage and its variants. *Algorithmica*, 65(3):517–544, 2013.

[32] G. L. Nemhauser and L. E. Trotter, Jr. Vertex packings: Structural properties and algorithms. *Mathematical Programming*, 8(2):232–248, 1975.

[33] S. Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Transactions on Algorithms*, 6(2), 2010.

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

A Problem Zoo

Below we provide problem statements to all problems discussed in the paper. We adopt the notation that appends brackets at the end of problem names to specify the parameterization used for the specific problem. For instance, CONNECTED VERTEX COVER(k) denotes the CONNECTED VERTEX COVER problem parameterized by the number k of vertices in the solution.

Φ-SAT:
Input: A formula $\phi \in \Phi$ with n variables, and an integer k.
Task: Decide whether ϕ is satisfiable.

Φ-WSAT:
Input: A formula $\phi \in \Phi$ with n variables, and an integer k.
Task: Decide whether ϕ is satisfiable by an assignment of Hamming weight k (an assignment that assigns exactly k variables the boolean value 1).

BINARY NDTM HALTING:
Input: A Turing machine M of size n with a binary alphabet, and an integer k.
Task: Decide whether M halts on the empty string in k steps.

CLIQUE:
Input: A graph G with n vertices, and an integer k.
Task: Decide whether G has a clique of size k (a pairwise adjacent subset of k vertices).

CAPACITATED VERTEX COVER:
Input: A graph G with n vertices, a capacity function $\alpha : V(G) \to \mathbb{N}$, and an integer k.
Task: Decide whether G has a capacitated vertex cover of size k (a subset of k vertices S that are incident with each edge G, and such that each vertex $v \in S$ is incident with at most $\alpha(v)$ edges).

CONNECTED VERTEX COVER:
Input: A graph G with n vertices, and an integer k.
Task: Decide whether G has a connected vertex cover of size k (a connected subset of k vertices S that are incident with each edge of G).

DIRECTED MULTICOLORED CYCLE:
Input: A directed graph G, a coloring function $c : V \to [k]$, and an integer k.
Task: Decide whether G has a multicolored directed cycle of length k (a directed cycle which includes exactly one vertex from each color).

DIRECTED MULTICOLORED PATH:
Input: A directed graph G, a coloring function $c : V \to [k]$, and an integer k.
Task: Decide whether G has a multicolored directed path of length k (a directed path which includes exactly one vertex from each color).

DISJOINT CYCLES:
Input: A graph G with n vertices, and an integer k.
Task: Decide whether G contains k pairwise disjoint cycles.

DISJOINT FACTORS:
Input: A n-character string S over the alphabet $[k]$.
Task: Decide whether there exists a set of k non-overlapping substrings $S_1, \ldots, S_k$ of S such that S_i is of the form $i \cdots i$ for every alphabet symbol $i \in [k]$.

DISJOINT PATHS:
Input: A graph G with n vertices, and k pairs of vertices $(s_1, t_1), \ldots, (s_k, t_k)$.
Task: Decide whether G contains k pairwise disjoint paths connecting s_i to t_i for all $i \in [k]$.

DOMINATING SET:
Input: A graph G with n vertices, and an integer k.
Task: Decide whether G has a dominating set of size k (a set D of k vertices for which every vertex not in D has a neighbor in D).

EXACT HITTING SET:
Input: A hypergraph $(V, \mathcal{E})$ with $|V| = n$ and $|\mathcal{E}| = m$.
Task: Decide whether $(V, \mathcal{E})$ has an exact hitting set (a subset $S \subseteq V$ such that $|S \cap E| = 1$ for all $E \in \mathcal{E}$).

EXACT SET COVER:
Input: A hypergraph $(V, \mathcal{E})$ with $|V| = n$ and $|\mathcal{E}| = m$.
Task: Decide whether $(V, \mathcal{E})$ has an exact set cover (a subset $\mathcal{S} \subseteq \mathcal{E}$ of pairwise disjoint edges with $\bigcup \mathcal{S} = V$).

INDEPENDENT SET:
Input: A graph G with n vertices, and an integer k.
Task: Decide whether G has an independent set of size k (a pairwise non-adjacent subset of k vertices).

HITTING SET:
Input: A hypergraph $(V, \mathcal{E})$ with $|V| = n$ and $|\mathcal{E}| = m$, and an integer k.
Task: Decide whether G has a hitting set of size k (a subset $S \subseteq V$ of size k with $S \cap E \neq \emptyset$ for all $E \in \mathcal{E}$).

LOCAL CIRCUIT SAT:
Input: A circuit C over $k + k \log m$ variables and of size $k + k \log m$, and a string S.
Output: Decide whether there is a list of k positions $i_1, \ldots, i_k$ in S such that feeding the contents of the positions to the first k inputs, and the binary expansions of $i_1, \ldots, i_k$ to the remaining inputs, causes C to accept.

MIN ONES d-SAT:
Input: A formula $\phi \in \Gamma_{1,d}$ with n variables, and an integer k.
Task: Decide whether ϕ is satisfiable by an assignment of Hamming weight at most k.

MULTICOLORED Φ-WSAT:
Input: A formula $\phi \in \Phi$ over a variable set X of size n, a coloring function $c : X \to [k]$, and an integer k.
Task: Decide whether ϕ is satisfiable by an multicolored assignment of Hamming weight k (an assignment where no two variables of same color are assigned a 1).

MULTICOLORED CLIQUE:
Input: A graph $G = (V, E)$ with $|V| = n$, a coloring function $c : V \to [k]$, and an integer k.
Task: Decide whether G has a multicolored clique of size k (a clique containing exactly one vertex of each color).

MULTICOLORED CYCLE:
Input: A graph G, a coloring function $c : V \to [k]$, and an integer k.
Task: Decide whether G has a multicolored cycle of length k (a cycle which includes exactly one vertex from each color).

MULTICOLORED HITTING SET:
Input: A hypergraph $(V, \mathcal{E})$ with $|V| = n$ and $|\mathcal{E}| = m$, a coloring function $c : V \to [k]$, and an integer k.
Task: Decide whether G has a multicolored hitting set of size k (a hitting set which includes exactly one vertex from each color).

MULTICOLORED PATH:
Input: A graph G, a coloring function $c : V \to [k]$, and an integer k.
Task: Decide whether G has a multicolored path of length k (a path which includes exactly one vertex from each color).

NDTM HALTING:
Input: A Turing machine M of size n, and an integer k.
Task: Decide whether M halts on the empty string in k steps.

PERFECT DELETION:
Input: A graph G on n vertices, and an integer k.
Task: Decide whether G has at most k vertices S such that $G - S$ is perfect.

SET COVER:
Input: A hypergraph $(V, \mathcal{E})$ with $|V| = n$, $|\mathcal{E}| = m$, and $\max_{E \in \mathcal{E}} |E| = d$. Also, an integer k.
Task: Decide whether $(V, \mathcal{E})$ has a set cover of size k (a subset $\mathcal{S} \subseteq \mathcal{E}$ of k edges with $\bigcup \mathcal{S} = V$).

SMALL SUBSET SUM:
Input: An integer k, a set S of integers of size at most 2^k, and an integer t.
Task: Decide whether there are at most k distinct integers in S that sum up to t.

STEINER TREE:
Input: A graph $G = (V, E)$ with $|V| = n$, a set of t terminals $T \subseteq V$, a set of ℓ non-terminals $N \subseteq V$, and an integer k.
Task: Decide whether there is a subset of at most k non-terminals $N' \subseteq N$ such that $G[T \cup N']$ is connected.

UNIQUE COVERAGE:
Input: A hypergraph $(V, \mathcal{E})$ with $|V| = n$ and $|\mathcal{E}| = m$, and an integer k.
Task: Decide whether there exists a subset $\mathcal{E}' \subseteq E$ such that at least k vertices are contained in exactly one edge in $\mathcal{E}'$.

WEAKLY CHORDAL DELETION:
Input: A graph G on n vertices, and an integer k.
Task: Decide whether G has at most k vertices S such that $G - S$ is weakly chordal.