Kernelizations for Parameterized Counting Problems

Marc Thurley*

Institut für Informatik, Humboldt-Universität zu Berlin
thurley@informatik.hu-berlin.de

Abstract. Kernelizations are an important tool in designing fixed parameter algorithms for parameterized decision problems. We introduce an analogous notion for counting problems, to wit, *counting kernelizations* which turn out to be equivalent to the fixed parameter tractability of counting problems. Furthermore, we study the application of well-known kernelization techniques to counting problems. Among these are the Buss Kernelization and the Crown Rule Reduction for the vertex cover problem. Furthermore, we show how to adapt kernelizations for the hitting set problem on hypergraphs with hyperedges of bounded cardinality and the unique hitting set problem to their counting analogs.

1 Introduction

In the last decade, *parameterized complexity* matured as a field of refined complexity analyses of algorithms and decision problems. It replaces the classical notion of tractability with *fixed parameter tractability* [12], requiring tractable problems to be solvable by a deterministic algorithm in time $f(k) \cdot n^{O(1)}$ for some (small) parameter k, a computable function $f : \mathbb{N} \to \mathbb{N}$ and n the input size. This notion is tightly connected to the concept of kernelization. Intuitively, a kernelization is a polynomial time computable function mapping problem instances to instances whose size is bounded effectively in terms of the parameter. Ever since its introduction in [11] this concept has found may applications with the result that today most fixed parameter algorithms use techniques that are inspired by this notion [15,8,19,20].

Notably, all of the work mentioned is devoted to decision problems. In contrast, the study of parameterized *counting* problems has not yet matured as far. Some work has been done on carrying over fixed parameter algorithm design to counting problems. For example, the best known algorithms following this approach solve the counting analog of the vertex cover problem in time $O(1.62^k n)$ (due to Fernau [16] - another, yet unpublished, result by Rossmanith [24] establishes an even lower bound of $O(1.47^k n)$). For some other counting problems fixed parameter algorithms were developed [10,17,22] as well. A variety of more

* This research was supported in part by the Deutsche Forschungsgemeinschaft within the research training group 'Methods for Discrete Structures' (GRK 1408).

J.-Y. Cai, S.B. Cooper, and H. Zhu (Eds.): TAMC 2007, LNCS 4484, pp. 703–714, 2007.

general results states the fixed parameter tractability of large classes of counting problems [2,7,18].

However, a notion of kernelization that is applicable to counting problems has not yet been developed, although there has been some recent interest in forming a notion of this kind. Nishimura et al. [22] suggested *compactor enumeration*, which performs a reduction to an enumeration problem on an instance with size depending only on the parameter. However, this notion has not yet been thoroughly formalized. In particular, the informal definition of a compactor from [22] is not applicable to counting problems in general.

We will remedy this deficiency by formally describing kernelizations of counting problems. Moreover, we will show that this notion is equivalent to the notion of fixed parameter tractability for counting problems and we give some examples, how this notion can be applied in practice. These examples will include vertex cover and hitting set problems.

There are at least three important kernelizations of the vertex cover decision problem: the Buss kernelization [11], the Crown Rule Reduction [15] and the kernel based on the Nemhauser-Trotter theorem [5]. We will apply the former two to the counting problem (the Nemhauser-Trotter kernel can be applied to counting in a way analogous to that of the Crown Rule [26]). Furthermore, we will adapt the Sunflower kernel [17] for the hitting set problem on hypergraphs with edges of bounded cardinality. Additionally, for the unique hitting set problem [13] we will develop an adaptation of a well-known kernelization of the decision problem to counting.

This paper is organized as follows. Section 1 gives the basic definitions and the concept of counting kernelization. In Section 2 we give kernelizations of vertex cover and hitting set counting problems. Section 3 is devoted to the Crown Rule Reduction. Then, in section 4, we will kernelize the counting unique hitting set problem and the last section will reveal the equivalence of counting kernelizations and fixed parameter tractable counting problems.

2 Definitions and the Concept of Counting Kernelizations

We will briefly sketch the basic notions of fixed parameter tractability. For a more thorough introduction to the field the reader is refered to one of [13,21,17].

Parameterized problems are formed from classical problems by adding a *parameterization of the input alphabet* Σ, that is, a polynomial time computable function $\kappa : \Sigma^* \to \mathbb{N}$. Thus for classical decision problems $Q \subseteq \Sigma^*$ and counting problems $F : \Sigma^* \to \mathbb{N}$ we obtain parameterized analogs (Q, κ) and (F, κ). We denote the class of all fixed parameter tractable decision problems by FPT and that of fixed parameter tractable counting problems by $FFPT$.

For parameterized decision problems, a *kernelization* is a mapping $K : \Sigma^* \to \Sigma^*$ satisfying the following conditions.

(K1) K is polynomial time computable and there is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that for all $x \in \Sigma^*$ we have

$$|K(x)| \leq g(\kappa(x))$$

(K2) $x \in Q \Leftrightarrow K(x) \in Q$, for all $x \in \Sigma^*$

Hence, if a problem (Q, κ) has a kernelization it is fixed parameter tractable.

This concept has indeed led to the creation of some of the most efficient fpt-algorithms and it is one of the most frequently applied approaches in parameterized algorithm design. Considering this popularity of kernelization in the field of decision problems one would naturally ask, if this notion can be applied to counting problems as well. We will discuss this question now.

2.1 Counting Kernelization

Note that in this purely theoretical context we regard computations always as computations of Turing machines. We begin with some preliminary definitions.

Definition 1. *Let $\kappa : \Sigma^* \to \mathbb{N}$ be a parameterization of Σ^*.*

(I) A mapping $K : \Sigma^* \to \Sigma^*$ is κ-bounded *if there is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that for all $x \in \Sigma^*$:*

$$|K(x)| \leq g(\kappa(x)).$$

(II) A relation $Y \subseteq \Sigma^* \times \{0,1\}^*$ is K-aware *for a κ-bounded mapping K if there are computable functions $h, f : \mathbb{N} \to \mathbb{N}$ such that for every $x \in \Sigma^*$ and every $y \in \{0,1\}^*$:*

(1) Given $K(x)$, it can be decided in time $f(\kappa(x))$ whether $(K(x), y) \in Y$ holds.

(2) if $(K(x), y) \in Y$ then $|y| \leq h(\kappa(x))$.

Note that the notion of *κ-boundedness* is simply derived from the notion of kernelization by omitting (K2), i.e. the part that deals explicitly with decision problems. Furthermore K-aware relations Y will be used to characterize the "solutions" of the reduced instance $K(x)$ that have to be found by search algorithms. Unfortunately, the formal definition of a counting problem does not mention what a solution might be. Therefore we will formalize this by the well-known notion of *witnesses*:

Let $Y \subseteq \Sigma^* \times \{0,1\}^*$ be a relation. For $x \in \Sigma^*$ we define $w_Y(x) := \{y \,|\, (x,y) \in Y\}$ as the set of *witnesses* of x in Y. Note that K-aware relations are compatible with the notions of *P-relations* from classical complexity (see e.g. [23]) in such a way that every P-relation is a K-aware relation.

A notion of counting kernelization needs an additional ingredient. This will become clear by an example. Consider the following problem:

p-#VertexCover

Instance: A graph $\mathcal{G} = (V, E)$ and $k \in \mathbb{N}$
Parameter: k
Problem: Compute the number of vertex covers of size k in $\mathcal{G}$

Let $\#vc(\mathcal{G}, k)$ denote the number of size k vertex covers in $\mathcal{G}$. Let $K : \Sigma^* \to \Sigma^*$ be κ-bounded, mapping instances of p-#VertexCover to instances of the same problem s.t. for all graphs $\mathcal{H}$ we have $|K(\mathcal{H}, k)| \leq g(k)$, for some computable function $g : \mathbb{N} \to \mathbb{N}$.

Example 1. Let $\mathcal{G} = (V, E)$ be a graph and $k \in \mathbb{N}$. Suppose that $\#vc(K(\mathcal{G}, k)) = \#vc(\mathcal{G}, k)$. Let $K(\mathcal{G}, k) =: (\mathcal{G}', k')$ for a graph $\mathcal{G}' = (V', E')$ and $k' \in \mathbb{N}$. As $\|\mathcal{G}'\| \leq g(k)$, we know that there can be no more than $\binom{g(k)}{k'}$ vertex covers of cardinality k' in $\mathcal{G}'$.

Let $\mathcal{H} = (W, \emptyset)$ be a graph with $|W| = g(k) + k$. It is easy to see that

$$\#vc(\mathcal{H}, k) = \binom{g(k)+k}{k} > \binom{g(k)}{k} \geq \#vc(K(\mathcal{H}, k)).$$

This example illustrates, that the reduced instance $K(x)$ may have less solutions than x itself. Hence we need a function, say μ, that for each solution y of the reduced instance $K(x)$ computes the number of solutions of an instance x that are "represented" by y. Moreover, the function μ has to depend on the original instance x to be able to create the link between $K(x)$ and x.

Definition 2 (Counting Kernelization). *Let (F, κ) be a parameterized counting problem over Σ. A* counting kernelization of (F, κ) *is a pair (μ, K) of polynomial time computable functions $K : \Sigma^* \to \Sigma^*$ and $\mu : \Sigma^* \times \Sigma^* \times \{0,1\}^* \to \mathbb{N}$ such that for all $x \in \Sigma^*$ the following is satisfied:*

(1) K is κ-bounded
(2) There is a K-aware relation $Y \subseteq \Sigma^* \times \{0,1\}^*$ such that

$$F(x) = \sum_{y \in w_Y(K(x))} \mu(x, K(x), y) \tag{1}$$

We call $|K(x)|$ the size of the kernel.

Note that, on an informal basis, this notion follows the intuition given in [22], to wit, a counting kernelization reduces a counting problem to an enumeration problem on an instance that is bounded in terms of the parameter. This is done in such a way that we can associate solutions of the kernel to (sets of) solutions of the original instance by an efficiently computable function μ.

Nevertheless, this definition still seems somewhat arbitrary. Its motivation will become clear if we know how to apply it to some counting problems. We have to note that in the course of this, we will sloppily define K-aware relations Y that do not directly satisfy $Y \subseteq \Sigma^* \times \{0,1\}^*$, however, it will always be clear that they can be represented straightforwardly in such a way.

3 p-#VertexCover and p-Card-#Hittingset

We start with two tightly connected problems. Recall that a *hitting set* in a hypergraph $\mathcal{H} = (V, E)$ is a set $S \subseteq V$ such that $S \cap e \neq \emptyset$ for all $e \in E$.

Furthermore, if we regard graphs as hypergraphs, vertex covers are simply a special case of hitting sets.

In the analysis of all algorithms considered here we will use the uniform cost measure. This implies that all basic arithmetic computations can be carried out in constant time.

p-card-#HittingSet

Instance: A hypergraph $\mathcal{H} = (V, E)$ and $k \in \mathbb{N}$
Parameter: $k + d$ with $d := max_{e \in E}|e|$
Problem: Compute the number of hitting sets of cardinality k in $\mathcal{H}$

We define $\#hs(\mathcal{H}, k)$ to denote the number of hitting sets of size k in the hypergraph $\mathcal{H}$.

The kernelization of the decision problem p-card-HittingSet given in [17] utilizes the well known *Sunflower Lemma*. Given a hypergraph $\mathcal{H} = (V, F)$, a *sunflower* in $\mathcal{H}$ is a family $\mathbf{S} = \{S_1, \ldots, S_k\} \subseteq F$ of hyperedges such that there is a set $C \subseteq V$ satisfying

$$S_i \cap S_j = C \qquad \text{for all } 1 \leq i < j \leq k \tag{2}$$

C is the *core* of the sunflower $\mathbf{S}$ and for all $i \in [k]$ the set $S_i \setminus C$ is called a *petal* if it is nonempty.

Lemma 3 (Sunflower Lemma). *Let $k, d \in \mathbb{N}$ and be $\mathcal{H} = (V, F)$ a d-uniform hypergraph with more than $(k-1)^d \cdot d!$ hyperedges. Then there is a sunflower S of cardinality k in $\mathcal{H}$.*

The Sunflower Lemma can be applied to counting in a straightforward manner:

Lemma 4. *1. There is an algorithm* FindSunflower *that computes a sunflower S according to the Sunflower Lemma in time $O(d\|\mathcal{H}\|)$.*

2. Let S with $|S| = k+1$ be a sunflower in $\mathcal{H}$ with core C. Define $\mathcal{H}' = (V, E')$ with

$$E' := E \setminus S \cup \{C\}$$

then $\#hs(\mathcal{H}, k) = \#hs(\mathcal{H}', k)$.

The kernelization mapping $K_{Sunflower}$ is given in algorithm 1. Observe that the algorithm deletes vertices that are isolated after the graph has been reduced (line 1). The following Lemma shows how to compute the correct hitting set count from the output of $K_{Sunflower}$.

Lemma 5. *Let $(\mathcal{H}, k)$ be an instance of* p-card-#HittingSet *with $\mathcal{H} = (V, E)$ and let $\mathcal{H}' \leftarrow K_{Sunflower}(\mathcal{H}, k)$ with $\mathcal{H}' = (V', E')$. Define $I := V \setminus V'$, then*

$$\#hs(\mathcal{H}, k) = \textstyle\sum_{i=0}^{k} \#hs(\mathcal{H}', i) \cdot \binom{|I|}{k-i}$$

The proof of this Lemma follows from the easily observable fact that any size k hitting set C satisfies $|C \cap V'| = i$ and $|C \cap I| = k - i$ for some suitable $0 \leq i \leq k$. Conversely, for any size i hitting set C' of $\mathcal{H}'$ and for every set $I' \subseteq I$ of cardinality $|I'| = k - i$ the union $C' \cup I'$ is a size k hitting set of $\mathcal{H}$.

```
   K_Sunflower(H, k) // H = (V, E) a hypergraph, k ∈ ℕ
 1 d ← max_{e∈E}|e|;
   // Begin kernelization
 2 for i ← 1 to d do
 3     E_i ← {e ∈ E : |e| = i };
 4     V_i ← ⋃_{e∈E_i} e;
 5     while FindSunflower((V_i, E_i), k + 1) ≠ (∅, ∅) do
 6         Let (S, C) be the sunflower returned by FindSunflower ;
 7         E_i ← (E_i \ S) ∪ {C};
 8         V_i ← (V_i \ ⋃_{X∈S} X) ∪ C;
 9     end
10 end
   // End kernelization
11 V' ← V_1 ∪ V_2 ∪ ... ∪ V_d;
12 E' ← E_1 ∪ E_2 ∪ ... ∪ E_d;
13 return (V', E');
```

Algorithm 1. The Kernelization algorithm based on the Sunflower Lemma

Note that a run of $K_{Sunflower}(\mathcal{H}, k)$ takes time at most $O(d^2 \cdot |E| \cdot \|\mathcal{H}\|)$ and for the hypergraph $\mathcal{H}' = (V', E')$ the Sunflower Lemma implies $|E'| \leq k^d \cdot d! \cdot d$ and $|V'| \leq k^d \cdot d! \cdot d^2$. Thus we have a kernelization of size $\|\mathcal{H}'\| \leq 2 \cdot k^d \cdot d! \cdot d^2$.

To complete the definition of the counting kernelization, we define a relation Y such that for all instances $(\mathcal{H}, k)$ of p-card-#HittingSet and $C \subseteq V$ we have $((\mathcal{H}, k), C) \in Y$ if and only if C is a hitting set of size k in $\mathcal{H}$. Hence, $w_Y((\mathcal{H}', k))$ can be enumerated in time $h(k)$ for some computable function h and we are done by defining

$$\mu((\mathcal{H}, k), (\mathcal{H}', k), C) := \binom{|I|}{k - |C|}.$$

For Vertex Cover we can easily form a slightly better kernel. Observe that, for graphs, the sunflower kernel implies $|E'| \leq 2k^2$ and $|V'| \leq 4k^2$. A simple but powerful kernelization of p-VertexCover is known as *Buss' Kernelization*. Its idea is very similar to that of of the sunflower kernel:

Lemma 6. *Let $\mathcal{G} = (V, E)$ be a graph and $k \in \mathbb{N}$ then:*

1. *Any $v \in V$ with $d(v) > k$ is contained in every k-element vertex cover of $\mathcal{G}$.*
2. *If $\Delta(\mathcal{G}) \leq k$ and $\mathcal{G}$ has a k-element vertex cover, then $|E| \leq k^2$.*

From this we derive a kernel of half the size of the sunflower kernel, as for the kernel $\mathcal{G}' = (V', E')$ the Lemma implies $|E'| \leq k^2$ and hence $|V'| \leq 2k^2$. Note that the counting kernelization according to Buss' Lemma can be build completely analogously to that of the sunflower kernel.

Observe furthermore that we can always combine these kernelizations with fixed parameter algorithms, based on the method of bounded search trees. In this

way, for example, for the p-#VertexCover problem, using the algorithm by Fernau [16], we obtain an algorithm which runs in time $O(1.62^k k^2 + poly(n))$ [26].

4 Crown Rule Reduction for p-#VertexCover

One of the most efficient kernelization techniques known in parameterized algorithm design is the application of the so-called *Crown Rule Reduction* [15]. We will see now, how to apply this kernelization to counting vertex covers. This will, somewhat surprisingly, result in a counting kernelization with size exponential in k. Nevertheless it still may guide the way to counting kernelizations of problems for which no kernel is known at all.

Definition 7. *Let $\mathcal{G} = (V, E)$ be a graph. A* crown *in $\mathcal{G}$ is a bipartite subgraph $\mathcal{C} = (I, N(I), F)$ of $\mathcal{G}$ satisfying three conditions:*

(1) I is an independent set in $\mathcal{G}$ and $N(I)$ is the set of all neighbors of vertices from I in $\mathcal{G}$.
(2) F contains all edges from E that connect vertices in $N(I)$ to vertices in I.
(3) $\mathcal{C}$ has a matching of cardinality $|N(I)|$.

Let $vc(\mathcal{G})$ denote the minimum size of a vertex cover in $\mathcal{G}$.

Let $\mathcal{G} = (V, E)$ be a graph and $k \in \mathbb{N}$. For the time being, let us assume, that $\mathcal{G}$ contains no isolated vertices. The case of isolated vertices will be considered later. We follow the construction of a crown as given in [17], this construction algorithm produces three different results:

(A) it determines $vc(\mathcal{G}) > k$
(B) $|V| \leq 3k$ holds
(C) a crown $\mathcal{C} = (I, N(I), F)$ on at least $|V| - 3k$ vertices is constructed

In the case of p-VertexCover the crown can simply be deleted. In counting we have to do some additional work to obtain an appropriate kernel.

Applying a crown in a counting algorithm. Let $\mathcal{G} \setminus \mathcal{C}$ be the graph obtained from $\mathcal{G}$ by deleting all vertices in $\mathcal{C}$ and all edges incident to vertices in $\mathcal{C}$.

Let $S_\mathcal{C}$ be a vertex cover of $\mathcal{C}$ and let $\mathcal{G}' = (V', E')$ be the graph obtained from $\mathcal{G} \setminus \mathcal{C}$ by deleting all edges covered by vertices in $S_\mathcal{C}$. One can easily see that $\#vc(\mathcal{G}', k - |S_\mathcal{C}|)$ equals the number of size k vertex covers S in $\mathcal{G}$ with $S \supseteq S_\mathcal{C}$. As $|V'| \leq 3k$, $\#vc(\mathcal{G}', k - |S_\mathcal{C}|)$ can be computed in time depending only on k. Therefore, to show how to compute the number of size k vertex covers in $\mathcal{G}$ it remains to show, how to enumerate the vertex covers of $\mathcal{C}$.

Recall that $|N(I)| < k$ and note that, with respect to the edges in $\mathcal{G}$ (and hence in $\mathcal{C}$ as well), all vertices in I have neighbors only in $N(I)$. Therefore, we can define an equivalency relation with at most 2^k equivalency classes by defining for all $v, w \in I$:

$$v \sim w \;:\Leftrightarrow\; N(v) = N(w) \tag{3}$$

For every $v \in I$ define $[v] := \{w \in I \mid v \sim w\}$, i.e. the equivalence class of v.

Claim 1. Given a vertex cover C of $\mathcal{C}$ such that there is a vertex $y \in N(I) \setminus C$. Let $v \in I$ be a vertex with $y \in N(v)$, then $[v] \subseteq C$.

Proof. Assume that there is a $w \in [v]$ with $w \notin C$. Then, by the definition of $[v]$ there is an uncovered edge $\{y, w\}$ in $\mathcal{C}$. Contradiction.

Consider a vertex cover C of $\mathcal{C}$ and let $S = N(I) \cap C$ and $X = I \cap C$. Note that S and X form a partition of C. Let

$$\mathbf{A}(S) := \{[v] \mid v \in I,\ \exists\, y \in N(I) \setminus S \text{ such that } y \in N(v)\}.$$

and define

$$L(S) := \bigcup_{[v] \in \mathbf{A}(S)} [v] \tag{4}$$

Now, claim 1 implies that $L(S) \subseteq X$, that is $\mathbf{A}(S)$ is the (unique) minimal family of equivalency classes that is necessary such that $S \cup L(S)$ is a vertex cover in $\mathcal{C}$. Hence, we finally have to move from minimal (w.r.t. inclusion) vertex covers to arbitrary vertex covers of $\mathcal{C}$. In fact this means to consider the sets $R := X \setminus L(S)$. As $R \subseteq I \setminus L(S)$, for each $l \in \{0, \ldots, k - |S \cup L(S)|\}$ the number of such sets with $|R| = l$ is exactly

$$\binom{|I \setminus L(S)|}{l}$$

Hence, we are able to define our counting kernelization. The relation Y contains a pair $((\mathcal{G}, \mathcal{C}, k), C)$ iff $\mathcal{C} = (W, F)$ is a subgraph of $\mathcal{G}$ and C is a vertex cover of $\mathcal{G}$ with $|C| \leq k$ such that $C \cap W$ is minimal (w.r.t. inclusion) in $\mathcal{C}$.

For the Kernelization mapping K reconsider the exit conditions (A),(B) and (C) of the crown construction. If (A) $vc(\mathcal{G}) > k$, then $K(\mathcal{G}, k) := (\mathcal{G}_0, \mathcal{C}_0, 0)$ with $\mathcal{G}_0 = (\{a, b\}, \{a, b\})$ and $\mathcal{C}_0$ the empty crown, i.e. trivial negative instance. In case (B) $K(\mathcal{G}, k) := (\mathcal{G}, \mathcal{C}_0, k)$ as $\mathcal{G}$ is small. If (C) holds, the construction algorithm returns a crown $\mathcal{C} = (I, N(I), F)$. Then $K(\mathcal{G}, k) := (\mathcal{G}', \mathcal{C}', k)$ where $\mathcal{C}' = (I', N(I'), F')$ is obtained from $\mathcal{C}$ by keeping one vertex from I per equivalence class and deleting the rest of the vertices in I. $\mathcal{G}' = (V', E')$ is obtained from $\mathcal{G}$ in the same manner. Thus, $|I'| \leq 2^k$, $|N(I)| \leq k$ which implies $|V'| \leq 2^k + 4k$, that is, K is a κ-bounded mapping.

Finally, the mapping μ is defined as follows. Let $(\mathcal{G}, k)$ be the input instance and $\mathcal{C} = (I, N(I), F)$ the crown in $\mathcal{G}$. Let the kernel be $K(\mathcal{G}, k) := (\mathcal{G}', \mathcal{C}', k)$ with $\mathcal{C}' = (I', N(I'), F')$ and $\mathcal{G}'$ defined as in the previous paragraph.

Consider a vertex cover C of $\mathcal{G}'$ with $|C| \leq k$ such that $C \cap (I' \cup N(I'))$ is minimal with respect to the vertices in $\mathcal{C}'$. Define $S' := N(I') \cap C$. We know that $C \cap I'$ is one-to-one with $\mathbf{A}(S')$ as I' contains one representative of each equivalency class, hence $|L(S')| = \sum_{v \in C \cap I'} |[v]|$. Thus defining

$$\mu((\mathcal{G}, k), K(\mathcal{G}, k), C) := \binom{|I \setminus L(S')|}{l}$$

with $l := k - |C \cap N(I')| - |L(S')|$ satisfies our needs, as the binomial coefficient is zero if $l < 0$ or $l > |I \setminus L(S')|$. Obviously, μ is polynomial time computable and the correctness of our kernelization follows from the above considerations.

For the case that the input graph $\mathcal{G}$ contains m isolated vertices, it is easy to see that it suffices to delete them from $\mathcal{G}$ and accommodate m in the computation of the binomial coefficient for the μ function.

5 A Kernelization for p-#UniqueHittingSet

The problem p-#UniqueHittingSet provides us with another nice example of applying well known kernelization techniques to counting problems. The corresponding decision problem is described in [13] (see exercise 3.2.5).

p-#UniqueHittingSet

Instance: A hypergraph $\mathcal{H} = (V, E)$
Parameter: $|E|$
Problem: Compute the number of *unique* hitting sets sets in $\mathcal{H}$ that is all sets $X \subseteq V$ satisfying $\forall\, e \in E : |e \cap X| = 1$

As with the previous problems we will outline the well known solution of the decision problem and show how it can be used to design an efficient algorithm for the counting problem.

Let $\mathcal{H} = (V, E)$ be a p-#UniqueHittingSet instance. Let $k := |E|$. For all $x, y \in V$ we define an equivalency relation by

$$x \sim y =_{def} \forall e \in E :\ x \in e \Leftrightarrow y \in e.$$

Clearly, this relation defines l nonempty equivalency classes for an $l \leq 2^k$. Let A_0 be the equivalency class of all isolated vertices and define $\tilde{\mathbf{A}} := \{A_1, \ldots, A_{l-1}\}$ as the family of all nonempty equivalency classes, except for A_0.

Furthermore, we define $\tilde{\mathcal{H}} := (\tilde{\mathbf{A}}, \tilde{E})$. This hypergraph will play the part of the kernel in our algorithm. Its set of hyperedges $\tilde{E}$ represents the hyperedges in E with respect to the equivalency classes in $\tilde{\mathbf{A}}$. To make this precise, we define two functions. Let $h : V \to \tilde{\mathbf{A}}$ be defined by $h(v) := A \in \tilde{\mathbf{A}}$ s.t. $v \in A$, i.e. we map vertices to their corresponding equivalency classes. A second function $f : 2^V \to 2^{\tilde{\mathbf{A}}}$ defined by $f(\{v_1, \ldots, v_b\}) := \{h(v_1), \ldots, h(v_b)\}$ does so analogously for sets of vertices. Note that h (and f) are undefined for isolated vertices (and sets containing them, respectively). The definition of $\tilde{E}$ is easy now:

$$\tilde{E} := \{f(e) \mid e \in E\}.$$

The following Lemma displays the correctness of our approach.

Lemma 8. *Let $\mathcal{H}, \tilde{\mathcal{H}}$ and $\tilde{\mathbf{A}}$ be defined as above.*
Let $\tilde{\mathbf{U}}$ be the set of all unique hitting sets in $\tilde{\mathcal{H}}$, then the number $\#uhs(\mathcal{H})$ of unique hitting sets in $\mathcal{H}$ satisfies:

$$\#uhs(\mathcal{H}) = \sum_{\tilde{S} \in \tilde{\mathbf{U}}} 2^{|A_0|} \prod_{A \in \tilde{S}} |A| \tag{5}$$

By this Lemma, we can develop a counting kernelization (μ, K). We simply define a relation Y s.t. for every hypergraph $\mathcal{H} = (V, E)$ and $S \subseteq V$, we have $(\mathcal{H}, S) \in Y$ iff S is a unique hitting set in $\mathcal{H}$.

We define $K(\mathcal{H}) := \tilde{\mathcal{H}}$ and for any hitting set $\tilde{S}$ of $\tilde{\mathcal{H}}$:

$$\mu(\mathcal{H}, \tilde{\mathcal{H}}, \tilde{S}) := 2^{|A_0|} \prod_{A \in \tilde{S}} |A|.$$

Obviously, by the construction of $\tilde{H}$ we have $|\tilde{E}| = |E| = k$ and $|\tilde{\mathbf{A}}| \leq 2^k$ and $\tilde{\mathcal{H}}$ can be constructed from $\mathcal{H}$ in polynomial time. Furthermore, μ can be computed in polynomial time, as well. Lemma 8 now implies that (μ, K) is a valid counting kernelization.

6 Characterizing FFPT by Counting Kernelizations

Up to now, we have seen that counting kernelizations can be used to describe certain fpt algorithms for counting problems. Theorem 9 diplays that the notion of counting kernelization is even more general in the sense that it provides a complete characterization of fixed parameter tractable counting problems.

Theorem 9. *Let (F, κ) be a parameterized counting problem. The following are equivalent:*

(1) $(F, \kappa) \in FFPT$
(2) (F, κ) has a counting kernelization.

Note that the direction (2) $\Rightarrow$ (1) is straightforward. The other direction of the proof uses as a main tool the following Lemma, which itself shows the relation between fixed parameter tractable counting problems and K-aware relations.

Lemma 10. *Let $(F, \kappa) \in FFPT$ be a parameterized counting problem over Σ. Let $K : \Sigma^* \to \Sigma^*$ be a κ-bounded polynomial time computable mapping.*
There is a K-aware relation $Y \subseteq \Sigma^* \times \{0,1\}^*$ such that for all $x \in \Sigma^*$:

$$F(K(x)) = |w_Y(K(x))|.$$

7 Concluding Remarks

We have introduced kernelizations of fixed parameter tractable counting problems which provide a characterization of fixed parameter tractability. As a line

of future research we would suggest scrutinizing kernelization techniques towards their applicability to counting problems. Although we demonstrated that, in principle the Crown Rule Reduction is applicable to counting problems, for other problems, which were originally kernelized by the Crown Rule, the kernelizations omit some cases that are trivial in decision problems but highly non-trivial for counting. These include, among others, p-SetSplitting [19], p-DisjointTriangles [20] and p-(n-k)-Coloring [6]. In fact it would not be very surprising if their counting analogs turn out to be $\#W[1]$-hard.

References

1. Susanne Albers and Tomasz Radzik, editors. *Algorithms - ESA 2004, 12th Annual European Symposium, Bergen, Norway, September 14-17, 2004, Proceedings*, volume 3221 of *Lecture Notes in Computer Science*. Springer, 2004.
2. Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *J. Algorithms*, 12(2):308–340, 1991.
3. Hans L. Bodlaender, editor. *Graph-Theoretic Concepts in Computer Science, 29th International Workshop, WG 2003, Elspeet, The Netherlands, June 19-21, 2003, Revised Papers*, volume 2880 of *Lecture Notes in Computer Science*. Springer, 2003.
4. Hajo Broersma, Matthew Johnson, and Stefan Szeider, editors. *Algorithms and Complexity in Durham 2005 - Proceedings of the First ACiD Workshop, 8-10 July 2005, Durham, UK*, volume 4 of *Texts in Algorithmics*. King's College, London, 2005.
5. Jianer Chen, Iyad A. Kanj, and Weijia Jia. Vertex cover: Further observations and further improvements. *J. Algorithms*, 41(2):280–301, 2001.
6. Benny Chor, Mike Fellows, and David W. Juedes. Linear kernels in linear time, or how to save k colors in $o(n^2)$ steps. In *WG*, pages 257–269, 2004.
7. Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discrete Applied Mathematics*, 108(1-2):23–52, 2001.
8. Frank K. H. A. Dehne, Michael R. Fellows, Frances A. Rosamond, and Peter Shaw. Greedy localization, iterative compression, modeled crown reductions: New fpt techniques, an improved algorithm for set splitting, and a novel 2k kernelization for vertex cover. In Downey et al. [14], pages 271–280.
9. Frank K. H. A. Dehne, Alejandro López-Ortiz, and Jörg-Rüdiger Sack, editors. *Algorithms and Data Structures, 9th International Workshop, WADS 2005, Waterloo, Canada, August 15-17, 2005, Proceedings*, volume 3608 of *Lecture Notes in Computer Science*. Springer, 2005.
10. Josep Díaz, Maria J. Serna, and Dimitrios M. Thilikos. Fixed parameter algorithms for counting and deciding bounded restrictive list h-colorings. In Albers and Radzik [1], pages 275–286.
11. R. G. Downey and M. R. Fellows. Parameterized computational feasibility. In *P. Clote, J. Remmel (eds.): Feasible Mathematics II*, pages 219–244. Boston: Birkhäuser, 1995.
12. Rodney G. Downey and Michael R. Fellows. Fixed parameter tractability and completeness. In *Complexity Theory: Current Research*, pages 191–225, 1992.
13. Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1999.

14. Rodney G. Downey, Michael R. Fellows, and Frank K. H. A. Dehne, editors. *Parameterized and Exact Computation, First International Workshop, IWPEC 2004, Bergen, Norway, September 14-17, 2004, Proceedings*, volume 3162 of *Lecture Notes in Computer Science*. Springer, 2004.
15. Michael R. Fellows. Blow-ups, win/win's, and crown rules: Some new directions in fpt. In Bodlaender [3], pages 1–12.
16. Henning Fernau. Parameterized algorithmics: A graph-theoretic approach. Habilitationsschrift, Universität Tübingen, 2005.
17. Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006.
18. M. Frick. *Easy Instances for Model Checking*. PhD thesis, Universität Freiburg, 2001.
19. Daniel Lokshtanov and Christian Sloper. Fixed parameter set splitting, linear kernel and improved running time. In Broersma et al. [4], pages 105–113.
20. Luke Mathieson, Elena Prieto, and Peter Shaw. Packing edge disjoint triangles: A parameterized view. In Downey et al. [14], pages 127–137.
21. Rolf Niedermeier. Invitation to fixed-parameter algorithms. Habilitationsschrift, Universität Tübingen, 2002.
22. Naomi Nishimura, Prabhakar Ragde, and Dimitrios M. Thilikos. Parameterized counting algorithms for general graph covering problems. In Dehne et al. [9], pages 99–109.
23. Christos H. Papadimitriou. *Computational Complexity*. Pearson, 1994.
24. Peter Rossmanith. (unpublished).
25. Dan Roth. On the hardness of approximate reasoning. *Artif. Intell.*, 82(1-2):273–302, 1996.
26. M. Thurley. Tractability and intractability of parameterized counting problems. diploma thesis. humboldt universität zu berlin, 2006.
27. Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.
28. Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979.