MASTER

Extending the Scope of Algebraic Kernelization for Constraint Satisfaction Problems

Beukers, Stijn

Award date:
2021

Department of Mathematics and Computer Science
Algorithms Geometry and Applications

Extending the Scope of Algebraic Kernelization for Constraint Satisfaction Problems

Master Thesis

Stijn Beukers
0993791

Supervisor:
dr. B.M.P. Jansen

Assessment committee:
dr. B.M.P. Jansen
prof. dr. M.T. de Berg
dr. J.J.A. Keiren

30 August 2021

Abstract

The constraint satisfaction problem (CSP) is a problem which asks whether it is possible to find a satisfying assignment for a number of variables such that a set of constraints is satisfied. While the complexity of any CSP depends on the types of allowed constraints, many have been observed to be NP-hard. Some well-known instances include q-Graph Coloring and d-CNF-SAT for $d > 2$. One way to more efficiently solve NP-hard CSPs is through kernelizations. A kernelization is a formalization of preprocessing the original problem to a simpler one without changing the solutions. We will explore how the size of the set of suitably preprocessed constraints for CSP depends on the number of variables and the types of allowed constraints. The work by Chen, Jansen and Pieterse showed how to construct a kernel with a constraint set of linear size in terms of the number of variables for CSPs over a Boolean domains where the types of allowed constraints are so-called “balanced”. Their approach relied on representing the constraints by low-degree polynomials. By computing a basis for these polynomials it was then possible to sparsify the set of constraints. Earlier work only considered applying this method on Boolean domain CSPs however.

In this thesis we will expand upon the method provided by Chen, Jansen and Pieterse over non-Boolean finite domains, and show how to identify which CSPs have a kernel of size $\mathcal{O}(n^t)$ for $t > 1$, thus going beyond linear kernelization. For Boolean CSPs we show that when the allowed types of constraints are so-called t-*balanced*, it is possible to construct polynomials of degree t which represent the constraints of the CSP. These polynomials can then be used to sparsify said CSP as mentioned before. A similar approach can be applied for non-Boolean CSP by first rewriting the constraints of such a CSP to an equivalent binary constraint and then applying a framework similar to that of the CSP over a Boolean domain. For a specific type of CSP, known as 3-Uniform Hypergarph 3-Coloring, we will show that it is unlikely for a kernel with $\mathcal{O}(n^{3-\varepsilon})$ constraints for $\varepsilon > 0$ to exist. This type of CSP is similar to 3-Graph Coloring except that the graph is structured differently. In a 3-uniform hypergraph, each edge consists of 3 vertices instead of 2. In addition, a coloring is said to be a proper coloring if each edge contains at least 2 uniquely colored vertices. This can be thought of as a CSP where each constraint consists of 3 variables and each variable can be assigned a value from 1 to 3.

Contents

1 Introduction

Background and motivation. An important aspect of computer science is the search for efficient algorithms which can be used to solve complex problems. Although for many problems it is possible to find algorithms that find solutions efficiently, there are also many problems for which this does not seem to be feasible. An algorithm is called efficient if it is capable of computing a solution for a problem in time polynomial to the size of the input. Such problems are also referred to as *tractable* and the set containing all such problems is denoted by P. Problems for which it is believed that they cannot be solved in time polynomial in the size of the input are called *intractable*. Many intractable problems are also often referred to as NP-hard. The set of problems NP is characterized by the fact that a solution for a problem contained in NP can be checked on correctness in polynomial time. The definition NP-hard stems from the fact that an NP-hard problem is at least as hard as the hardest problems in NP, which are believed to be intractable. In addition to a problem being *NP-hard*, a problem is called *NP-complete* if it is both NP-hard and contained within NP. The definition of NP trivially leads to $P \subseteq NP$ however, $NP \subseteq P$ has not been proven and is generally thought to be false. This leads to the well-known conjecture $P \neq NP$.

One specific a class of NP-complete problems is the *constraint satisfaction problem* (CSP) which, given a set of variables and constraints, requires one to compute an assignment to said variables in order to satisfy all constraints. Although general CSP is NP-complete, some variations of CSP are tractable. A well-known example of a tractable variant of CSP is 2-CNF-SAT [1, 2, 3]. This problem requires multiple disjunctive clauses of two literals, which are either variables or the negations of variables, to be assigned a Boolean value. A solution would require each clause to evaluate to true. Now, if each clause were to contain more than two variables, also referred to as d-CNF-SAT for $d \geqslant 3$, this problem becomes NP-complete [4]. Whether a variant of CSP is NP-complete depends on the types of allowed constraints. More generally we can refer to a set of allowed constraint types by a so-called constraint language Γ. A constraint type contained in a language is also often referred to as a relation $R \in \Gamma$. A more formal definition will be presented in Section 2. If the constraint language Γ is fixed for CSP, then the problem is known as *non-uniform constraint satisfaction problem* and is denoted by $CSP(\Gamma)$. For a long time the conjecture whether any $CSP(\Gamma)$ is either tractable or NP-complete depending on the language Γ was an open problem. It was eventually proven that this conjecture is correct [5, 6, 7]. In this thesis we will not focus on tractable instances of $CSP(\Gamma)$ and instead focus on NP-complete instances.

To get around this computational complexity, new ways of dealing with NP-complete problems had to be invented. A well-known way to do so is by looking at so-called *parameterized problems*. These are problems where instead of only consisting of an input instance I, an integer k is also provided which is known as the *parameter*. Some examples are VERTEX COVER parameterized by the size of the vertex cover or GRAPH q-COLORING parameterized by the size of the largest clique.

One way in which such parameterized problems can be applied is by creating so-called *kernels*. A parameterized problem $Q \subseteq \{0,1\}^* \times \mathbb{N}$ is the set of pairs (I, k) with I being the encoding of a decision problem with answer yes, also referred to as a yes-instance, and integer parameter k. A kernel for Q is an algorithm $\mathcal{A} : \{0,1\}^* \times \mathbb{N} \to \{0,1\}^* \times \mathbb{N}$ which satisfies that any instance $(I, k) \in Q$ can be efficiently reduced to an instance $(I', k') \in Q$ such that the size of (I', k') is bounded by some function $g(k)$, and such that $(I, k) \in Q$ if and only if $(I', k') \in Q$. Thus the size of the kernelized instance (I', k') directly depends on k. A more formal definition will be provided in Section 2. In general, the size of I can be reduced when it is large with respect to k. If this is not the case then it is possible that no smaller equivalent instance for (I, k) exists. It is thus not possible to indefinitely reduce a parameterized problem instance.

When it comes to non-uniform CSP, finding a kernel is especially relevant as many computational problems can be formulated as variants of $CSP(\Gamma)$ with an appropriate language Γ. Some examples are d-CNF-SAT, which was previously introduced, and Graph q-Coloring. However, for many non-uniform CSP variants, it has been proven that the best obtainable kernels are so-called *trivial kernels*. A kernel is trivial when it is obtained by simply removing duplicate constraints. An example would be for d-CNF-SAT for which it was proven by Dell and Van Melkebeek that there is no kernel of size $\mathcal{O}(n^{d-\varepsilon})$, where n denotes the number of variables, d the size of all clauses and $\varepsilon > 0$, unless $NP \subseteq coNP/poly$ [8]. The conjecture that $NP \subseteq coNP/poly$ is false is a generally accepted hypothesis for proving kernel lower bounds. A kernel of size $\mathcal{O}(n^d)$ is trivial since for any instance of d-CNF-SAT there can be at most $2^d\binom{n}{d} = \mathcal{O}(n^d)$ distinct constraints. In addition, Jansen and Pieterse have shown that Graph q-Coloring when parameterized by the number of vertices does not admit a nontrivial kernel of size $\mathcal{O}(n^{2-\varepsilon})$ for $\varepsilon > 0$, unless $NP \subseteq coNP/poly$, since a trivial kernel has size $\mathcal{O}(n^2)$ [9]. It was later shown that for a different instance of non-uniform CSP there exists a nontrivial kernel. Jansen and Pieterse showed that for d-Not All Equal SAT (d-NAE-SAT) there exists a kernel with $\mathcal{O}(n^{d-1})$ constraints [10]. The problem of d-NAE-SAT is similar to d-CNF-SAT in that each constraint consists of exactly d literals, but a constraint in d-NAE-SAT is only satisfied when not all literals in a clause get assigned the same value. Their approach models the constraints of a d-NAE-SAT instance as polynomials, which can be sparsified by computing a basis for all such polynomials and removing the polynomials which are not in the basis.

In addition, using an algebraic interpretation of $CSP(\Gamma)$, Lagerkvist and Wahlström showed that it is possible to identify instances of $CSP(\Gamma)$ that have linear sized kernel, i.e. of at most $\mathcal{O}(n)$ constraints, by checking if the language admits a so-called *Maltsev embedding* [11]. This result is significant as it gives a generic way to identify $CSP(\Gamma)$ instances that admit a linear kernel. Chen, Jansen and Pieterse later showed that for so-called *balanced* constraint languages [12] it is possible to construct a kernel of $\mathcal{O}(n)$ constraints. A constraint language is balanced when for each of the relations in the language, each unsatisfying assignment of a relation cannot be expressed as an integer linear combination whose coefficients sum up to 1 of the satisfying assignments of the same relation. Constructing a linear kernel for $CSP(\Gamma)$ defined over a balanced Γ is done by creating low-degree polynomials which represent the constraints of a $CSP(\Gamma)$ and then computing a basis for these polynomials, similarly to Jansen and Pieterse [10].

During this thesis we will investigate which $CSP(\Gamma)$ variants have a non-trivial kernel and which do not. We will show that there is a generic scheme that, for any $CSP(\Gamma)$ defined over a finite domain, identifies whether $CSP(\Gamma)$ admits a kernel of $\mathcal{O}(n^t)$ constraints for some $t > 0$ and n being the number of variables. Values of t which are not trivial are especially of interest.

Our results. We show that it is possible for any variant of $CSP(\Gamma)$, given a so called *t-balanced* language Γ, to construct a kernel of size $\mathcal{O}(n^t)$. The definition of t-balancedness will be introduced in Section 3.1 for Boolean domains and will be enriched for finite domains in Section 3.2. This approach is based on the work by Chen, Jansen and Pieterse [12]. They showed that when a language is balanced, it is possible for each relation to create a set of *capturing* polynomials. A polynomial is said to capture a relation $R \in \Gamma$ with respect to some unsatisfying assignment when all satisfying assignments of R evaluate to 0 under the polynomial and the unsatisfying assignment evaluates to some nonzero value. Thus testing if an assignment is satisfying is equivalent to checking if all capturing polynomials evaluate to 0 when given the same assignment. The approach by Chen, Jansen en Pieterse showed that it is possible for each balanced constraint language to construct a set of linear capturing polynomials [12]. Then by computing a basis over these polynomials, they found that it was possible to construct a kernel for $CSP(\Gamma)$ with $\mathcal{O}(n)$ constraints. Our work expands upon this framework with the notion of t-balanced constraint languages. Given such a language, it is possible to construct a set of capturing polynomials of degree t which then allows us to compute a basis, resulting in a kernel for $CSP(\Gamma)$ of $\mathcal{O}(n^t)$ constraints. For $CSP(\Gamma)$ over finite domains the relations are first transformed to a binary representation called the *choice representation.* A choice representation introduces for each entry in the original relation, a new entry for each possible value of the domain which can be assigned to the entry in the original relation. An entry in the choice representation is set to 1 if and only if the related entry in the original relation was given the related value of the domain. Thus representing the value assigned in the original relation as a binary "choice", after which an approach similar to $CSP(\Gamma)$ over Binary domains is applied to find a kernel of $\mathcal{O}(n^t)$ constraints.

We also show that, while in many cases it is possible to find for each relation $R \in \Gamma$ a single polynomial which captures R, using our approach there is no simple way to find such a polynomial. Being able to find such a polynomial would be of interest as having a single polynomial for every relation instead of a set of polynomials simplifies the process greatly. This does not mean that no such polynomial exists, just that our approach cannot easily be applied to find such polynomials. It is, however, possible to combine two capturing polynomials which each capture a single unsatisfying assignment with respect to R into a single capturing polynomial.

Finally, it would be desirable to show that the kernels our method finds are tight. This means that whenever a t-balanced $CSP(\Gamma)$ has a kernel of $\mathcal{O}(n^t)$ constraints, then there does not exist a kernel for the same $CSP(\Gamma)$ with $O(n^{t-\varepsilon})$ constraints for any $\varepsilon > 0$. While proving this for all $CSP(\Gamma)$ in general is a daunting task, we provide further evidence that this seems to be the case. We say "further" evidence as when we tested our method on problems with known tight upper bounds, such as Graph q-Coloring and d-Not All Equal SAT, our method gave the same result as an upper bound. To this end we show that 3-Uniform Hypergraph 3-Coloring has no kernel with $\mathcal{O}(n^{3-\varepsilon})$ edges for any $\varepsilon > 0$ unless $NP \subseteq coNP/poly$. 3-Uniform Hypergraph 3-Coloring asks whether a 3-Hypergraph, which is a graph where each edge contains 3 vertices, can be colored such that each edge is colored using at least 2 distinct colors. This problem trivially has a kernel of $O(n^3)$ edges, thus implying that this problem does not have a nontrivial kernel. To prove this lower bound, we combine a set of specific graph coloring problem instances into a single 3-uniform hypergraph which can only be 3-colored if at least one of the original instances can be properly colored. We note that this specific graph coloring problem is a subproblem of graph coloring called 2-3-Coloring with Triangle Split Decomposition for which the definition is provided in Section 4. It is also interesting to note that both Graph q-Coloring and 3-Uniform Hypergraph 3-Coloring do not admit a nontrivial kernel. In addition, since a specific subset of graph coloring problems was used to construct the kernel for 3-Uniform Hypergraph 3-Coloring, it is clear that the complexity of these problems is tightly related.

Related work. Graph q-Coloring is a well-known instance of $CSP(\Gamma)$. The goal is, given a graph G, to check whether it is possible to assign each vertex in $V(G)$ a color from the set $\{1, ..., q\}$ such that each edge in $E(G)$ has endpoints in differently colored vertices. It was shown by Jansen and Pieterse that is possible to find a kernel of size $\mathcal{O}(k^{q-1})$ for Graph q-coloring with k being the size of the vertex cover [9]. A more general definition of Graph q-coloring is List H-Coloring. Here the goal is to map each vertex in a graph G to a vertex in another graph H such that for each edge in G, the end points are mapped to vertices in H which also share an edge. In addition, each vertex in G is only allowed to be mapped to a subset of vertices from H which is referred to as a list. For List H-Coloring it was proven by by Feder, Hell and Huang showing that based on H, the problem is either tractable or NP-complete [13]. It was later found by Chen, Jansen, Okrasa, Pietese and Rzążewski that it is not possible to find a kernel of size $\mathcal{O}(n^{2-\varepsilon})$ with $\varepsilon > 0$ unless $NP \subseteq coNP/poly$ for the NP-complete instances of List H-Coloring [14]. As $\mathcal{O}(n^2)$ is a trivial kernel for List H-Coloring, this result implies that it is unlikely that a non-trivial kernel exists for List H-Coloring.

It is also noteworthy how the complexity of parameterized CSP changes depending on the chosen parameter. First, we introduce the notion of *fixed parameter tractable* problems for which the set is denoted by FPT. Parameterized problems $Q \subseteq \{0,1\}^* \times \mathbb{N}$ in FPT are those for which a solution can be found in time $\mathcal{O}(f(k)n^{\mathcal{O}(1)})$, where f is some computable function, n represents the size of a problem instance and k is the parameter. Essentially this states that a solution to the problem must be found in time polynomial in terms of the input size and time $f(k)$ on k. Note that no restrictions are put on what kind of function f can be, so it can even be superpolynomial, meaning that f does not have to be bounded by a polynomial. It is important to note that a decidable problem Q is part of FPT if and only if Q has a kernelization, as shown in the book by Cygan et al. [15]. Not all problems are in FPT however, and these problems are higher in the complexity hierarchy. Similar to NP-hard problems in classical complexity theory, parameterized problems which are believed not to be fixed parameter tractable are called $W[1]$*-hard.* Majdoddin has proven that uniform CSP parameterized by the size of the solution is $W[1]$-hard [16]. This result implies that no kernelization exists for uniform CSP parameterized by the solution size. Another example of how complexity differs can be seen by looking at the Clique problem, which asks if a graph contains a clique of a certain size. When the Clique problem is parameterized by the size of a clique, it is known to be $W[1]$-hard. However, when parameterized by the maximum degree of the graph, it is known to be in FPT, which implies that it has a kernel as well. This shows that the choice of parameter is equally as important to the choice of the problem when trying to construct a kernel for any parameterized problem.

Organization. In Section 2 we go over all the preliminaries required to understand this paper. In Section 3 we show our kernelization algorithm for both Boolean and finite domain constraint languages. In addition we show that it is not trivial to construct a single polynomial for any $CSP(\Gamma)$. In Section 4 we show the construction for the lower bound for 3-Uniform Hypergraph 3-List-Coloring. Finally we will summarize and discuss our results in Section 5.

2 Preliminaries

2.1 CSP

First, we define what $CSP(\Gamma)$ is. To get there, we introduce the formal definition of a relation. We define a finite set D to be the domain and V to be the set of variables over which an instance of $CSP(\Gamma)$ is specified. A relation is a set of tuples $R \subseteq D^k$ of arity k containing some satisfying assignments. A constraint language Γ is then the set of relations, which specifies all valid constraint types for $CSP(\Gamma)$. A constraint defined over a relation R is given by $R(x_1, ..., x_k)$, with $x_1, ..., x_k \in V$. A constraint $R(x_1, ..., x_k)$ is satisfied by an assignment $f : V \to D$ when it holds that $(f(x_1), ..., f(x_k)) \in R$ holds. This leads to the formal definition of $CSP(\Gamma)$:

$CSP(\Gamma)$
Input: A tuple (C, V) where C is a finite set of constraints, V is a finite set of variables, and each constraint is of the form $R(x_1, ..., x_k)$ for $R \in \Gamma$ and $x_1, ..., x_k \in V$.
Question: Does there exists a *satisfying assignment*, that is, an assignment $f : V \to D$ such that for each $R(x_1, ..., x_k) \in C$ it holds that $(f(x_1), ..., f(x_k)) \in R$?

A mapping f for which a clause $R(x_1, ..., x_k)$ satisfies $(f(x_1), ..., f(x_k)) \in R$ is called a satisfying assignment for $R(x_1, ..., x_k)$. Since we will be referring to the relations of a $CSP(\Gamma)$ instance quite extensively, we introduce the following notation for simplicity.

Definition 2.1. *For a positive integer n, we let $[n]$ denote the set of all positive integers $\{1, 2, ..., n\}$.*

Definition 2.2. *For a tuple $t \in D^k$ of arity k, we define $t[i]$ for $i \in [k]$ as the ith element of t.*

Since we have defined a relation $R \subseteq D^k$ as a set of tuples of size k over the domain D, we can thus also denote all elements of some $r \in R$ by using $r[i]$. For completeness, we introduce the following notation which will be used later on for the construction of certain relations.

Definition 2.3. *For a finite set S, $\binom{S}{i}$ denotes the collection of subsets of S of size i and $\binom{S}{\leqslant i}$ the collection of all subsets of S of size at most i.*

2.2 Sparsification

It was proven that, depending on the constraint language Γ, $CSP(\Gamma)$ is either polynomial time solvable or NP-complete [5, 6, 7]. While the polynomial time solvable instances provide few problems in terms of running time, many NP-complete instances are still of interest and we would thus like to find a way to solve them more efficiently. It is unlikely that we can find a polynomial time algorithm which can find a solution for these NP-complete problems as this would imply $NP = P$. One way to solve NP-complete problem instances more efficiently is to reduce the input size in polynomial time before trying to solve the problem. This is where kernelization comes in, which is a kind of sparsification. A kernelization algorithm is a type of preprocessing algorithm that tries to reduce an instance I of some NP-hard problem until the irreducible core is reached (just like the kernel of a seed). We also say that such an irreducible core is a sparsified instance. First off, a kernelization algorithm uses so-called parameterized instances for which the definition is given in Definition 2.4.

Definition 2.4 (Parameterized problem)**.** *A parameterized problem is a language $L \subseteq (\Sigma^* \times \mathbb{N})$ with Σ being a fixed finite alphabet . For an instance $(x, k) \in (\Sigma^* \times \mathbb{N})$, k is called the parameter.*

We note that without loss of generality we can represent any alphabet Σ by a binary interpretation $\{0,1\}^*$. The parameter k of a parameterized problem can be defined as anything and will greatly aid in the construction of a kernelization algorithm. An example of such an instance is a graph coloring (G,k) where G is some undirected graph and k the size of the largest clique in G. The key idea of a kernelization algorithm is to efficiently reduce the size of the original problem as much as possible before applying an algorithm with an exponential worst case running time. This allows the instance to be solved in much less time than it would normally take. For defining a kernelization algorithm $\mathcal{A} : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ we first need the following notion:

$$size_{\mathcal{A}}(k) = \sup\{|I'| + k' : (I',k') = \mathcal{A}(I,k), I \in \Sigma^*\}$$

This means that the size of some natural number k with respect to a kernelization algorithm $\mathcal{A}$ is defined for all $I \in \Sigma^*$ as the largest sum of $|I'|$ and k' where (I',k') is the output of $\mathcal{A}$ over instance (I,k). Let $Q, Q' \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. We say that the instances $(I,k) \in Q$ and $(I',k') \in Q'$ are equivalent if it holds that $(I,k) \in Q$ if and only if $(I',k') \in Q'$. This essentially means that (I,k) is a yes-instance if and only if (I',k') is a yes-instance. Using this notation, the definition of a kernelization algorithm is given in Definition 2.5.

Definition 2.5 (Kernelization, Kernel). *A kernelization algorithm, or simply a kernel, for a parameterized problem Q is an algorithm $\mathcal{A}$ that, given an instance $(I,k) \in \Sigma^* \times \mathbb{N}$, works in polynomial time and returns an equivalent instance $(I',k') \in \Sigma^* \times \mathbb{N}$ such that $(I,k) \in Q$ if and only if $(I',k') \in Q$. Moreover, we require that $size_{\mathcal{A}}(k) \leqslant g(k)$ for some computable function $g : \mathbb{N} \to \mathbb{N}$.*

Note that this final requirement of kernelization gives an upper bound on the size of any kernel of a parameterized problems (I,k) depending on k. We call a kernel a *polynomial kernel* if the function g in Definition 2.5 is a polynomial. During the course of this paper we will construct a kernel for variants of $CSP(\Gamma)$. While kernelization algorithms are very powerful tools, it is also good to know whether or not we can improve upon an established kernel. To do so we would like to make a statement regarding the lower bound of a kernel. To this end, a construct called a *cross-composition* is used which requires a so-called *polynomial equivalence relation.* These concepts are defined in Definitions 2.7 and 2.6 respectively.

Definition 2.6 (Polynomial equivalence relation [17, Definition 3.1]). *An equivalence relation $\mathcal{R}$ on Σ^* is called a polynomial equivalence relation if the following conditions hold.*

- *There is an algorithm that, given two strings $x, y \in \Sigma^*$, decides whether x and y belong to the same equivalence class in time polynomial in $|x| + |y|$.*
- *For any finite $S \subseteq \Sigma^*$ the equivalence relation $\mathcal{R}$ partitions the elements in S into a number of classes that is polynomially bounded in the size of the largest element of S.*

Definition 2.7 (Cross-composition [17, Definition 3.3]). *Let $L \subseteq \Sigma^*$ be a language, let $\mathcal{R}$ be a polynomial equivalence relation on Σ^*, let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem, and let $f : \mathbb{N} \to \mathbb{N}$ be a function. An* OR*-cross-composition of L into Q (with respect to $\mathcal{R}$) of cost $f(t)$ is an algorithm that, given t instances $x_1, x_2, ..., x_t \in \Sigma^*$ of L belonging to the same equivalence class $\mathcal{R}$, takes time polynomial in $\Sigma_{i=1}^{t}|x_i|$ and outputs an instance $(y,k) \in \Sigma^* \times \mathbb{N}$ such that:*

- *the parameter k is bounded by $\mathcal{O}(f(t) \cdot (\max_i |x_i|)^c)$, where c is some constant independent of t, and*

- *instance $(y, k) \in Q$ if and only if there is an $i \in [t]$ such that $x_i \in L$.*

This finally leads to Theorem 2.8.

Theorem 2.8 ([17, Theorem 3.8]). *Let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem, let Σ be an alphabet, and let d, ε be positive reals. If L is NP-hard under Karp reductions, has an* OR*-cross-composition into Q with cost $f(t) = t^{1/d+\mathcal{O}(1)}$, where t denotes the number of instances, and Q has a polynomial (generalized) kernelization with size bound $\mathcal{O}(k^{d-\varepsilon})$, then $NP \subseteq coNP/poly$.*

Here $NP \subseteq coNP/poly$ is a conjecture which is believed to be false and is generally used for proving conditional kernel lower bounds. We will call an OR-cross-composition with cost function $f(t) = t^{1/d+\mathcal{O}(1)}$ a degree-d cross-composition. Such a cross-composition in combination with Theorem 2.8 will be used to prove lower bounds for some parameterized problem Q. More specifically, if there is a degree-d OR-cross-composition from an NP-hard problem L into a parameterized problem Q, then Q does not admit a kernel of size $\mathcal{O}(k^{d-\varepsilon})$ with k being the parameter of Q and $\varepsilon > 0$ unless $NP \subseteq coNP/poly$. Note that during this thesis we will generally consider k to be the number of variables, given by n.

2.3 Balanced operations

Next we introduce the notions of balanced relations and operations. These properties form the basis of our sparsification method. First we introduce what a partial operation is. A partial operation is a function $f : X \to Y$, where X, Y are some sets, which is only defined over some subset $X' \subseteq X$. This means that for any $x \in X'$, $f(x) \in Y$ and for all $x' \in X \setminus X'$, $f(x')$ is undefined. One such a function is the square root operation $f(x) = \sqrt{x}$ defined over the integers. It can only be applied to positive numbers which are also perfect squares. So $0, 1, 4, 9, ...$ are in the domain and $2, -3, 17$ are not in the domain of f defined over the integers. Since an operation is always a type of function, we will use the notions of operation and function interchangeably. With this definition and the definition of a relation, we give the definition of a relation being preserved by a partial operation in Definition 2.9.

Definition 2.9. *Let $f : \{0,1\}^n \to \{0,1\}$ be some partial operation over the binary domain and let R be some relation. We say that R is preserved by f when for each $r_1 = (r_1[1], ..., r_1[k]), ..., r_n = (r_n[1], ..., r_n[k]) \in R$, if the entries of the tuple $(f(r_1[1], ..., r_n[1]), ..., f(r_1[k], ..., r_n[k]))$ are defined, then this tuple is an element of R. If all relations in constraint language Γ are preserved by f then we say that Γ is preserved by f.*

During this thesis we will mainly focus on a subset of partial operations defined over a binary domain, in particular on partial functions which are considered *balanced*. The definition of a balanced partial operation is given in Definition 2.10.

Definition 2.10. *A partial Boolean function $f : \{0,1\}^n \to \{0,1\}$ is called balanced if there exist integer coefficients $\alpha_1, ..., \alpha_n$ such that:*

- $\sum_{i=1}^{n} \alpha_i = 1$*,*
- *$(x_1, ..., x_n)$ is in the domain of f if and only if $\sum_{i=1}^{n} \alpha_i x_i \in \{0,1\}$*
- *$f(x_1, ..., x_n) = \sum_{i=1}^{n} \alpha_i x_i$ for all tuples in the domain of f.*

Definition 2.11. *A Boolean relation R is balanced if it is preserved by all balanced operations. A Boolean constraint language Γ is balanced if all relations $R \in \Gamma$ are balanced.*

A specific balanced partial operation which will be important later on is called the *alternating operation*, which is defined in Definition 2.12.

Definition 2.12. *For each odd $n \geqslant 1$, the alternating operation is defined to be the balanced operation $a_n : \{0,1\}^n \to \{0,1\}$ such that the coefficients alternate between $+1, -1$ such that:*

$$\alpha_i = \begin{cases} +1 & i \text{ is odd} \\ -1 & i \text{ is even} \end{cases} \tag{1}$$

Using this definition, Proposition 2.13 was proven by Chen, Jansen and Pieterse [12].

Proposition 2.13 ([12, Proposition 2.8])**.** *A Boolean relation R is balanced if and only if for all odd $k \geqslant 1$, the relation R is preserved by the alternating operation of arity k.*

2.4 Polynomials

Polynomials are a powerful tool which can be utilized effectively to sparsify constraint satisfaction problems. This was first shown by Jansen and Pieterse [10] which showed how to construct polynomials for Not-All-Equal-SAT (NAE-SAT) and thus allowed for the construction of a kernel of size $\mathcal{O}(n^{k-1})$ with k being the arity of all relations and n being the number of variables over which the instance of NAE-SAT is defined. The main tool they used is to construct polynomials that capture an unsatisfying assignment. By computing a basis for these polynomials a kernel can then be constructed. We call a polynomial d-variate if it is defined over d variables. The definition of a polynomial capturing some unsatisfying assignment is given in Definition 2.14.

Definition 2.14. *For a k-ary Boolean relation $R \subseteq \{0,1\}^k$ and a k-variate polynomial p_u defined over some ring E_u, we say that $u \in \{0,1\}^d \setminus R$ is captured by p_u with respect to R if the following two conditions hold over E_u.*

- $p_u(x) = 0$ *for all* $x \in R$ *and*
- $p_u(u) \neq 0$

Using such polynomials, a system of linear equations can be constructed which can then be used to reduce to a kernel of size $\mathcal{O}(n^{k-1})$ for k-NAE-SAT [10].

For 3-NAE-SAT this is done as follows. First we note that a clause in 3-NAE-SAT is only satisfied when not all literals in a constraint evaluate to the same value. For example, if we have a constraint $R(x_1, x_2, x_3) = (x_1, x_2, \neg x_3)$ of 3-NAE-SAT then the assignment $\gamma : V \to \{0,1\}$ with $\gamma(x_1) = 1, \gamma(x_2) = 1, \gamma(x_3) = 0$ would not be a satisfying assignment. By applying γ to the constraint we get $(\gamma(x_1), \gamma(x_2), \neg\gamma(x_3)) = (1, 1, \neg 0) = (1, 1, 1)$ from which it follows that γ is not satisfying as all literals evaluate to 1. However, an assignment $\gamma' : V \to \{0,1\}$ with $\gamma'(x_1) = 1, \gamma(x_2) = 1, \gamma(x_3) = 1$ does satisfy the clause as this results in $(\gamma'(x_1), \gamma'(x_2), \neg\gamma'(x_3)) = (1, 1, \neg 1) = (1, 1, 0)$ and thus clearly not all literals evaluate to these same value. With this in mind, a 3-variate polynomial which captures a constraint $R(x_1, x_2, x_3)$ defined over some relations should be constructed. Since all relations are similar up to negation, the polynomial in Equation 2 can be used to capture the constraints of 3-NAE-SAT. Here $\ell(x_i) = x_i$ if x_i is not negated in $R(x_1, x_2, x_3)$ and $\ell(x_1) = 1 - x_i$ if x_i is negated in $R(x_1, x_2, x_3)$.

$$p(x_1, x_2, x_3) = 1 - \ell(x_1) - \ell(x_2) - \ell(x_3) + \ell(x_1)\ell(x_2) + \ell(x_1)\ell(x_3) + \ell(x_2)\ell(x_3) \tag{2}$$

So for example, the clause $(x_1, x_2, \neg x_3)$ of 3-NAE-SAT would be captured by the polynomial shown in Equation 3.

$$\begin{aligned} p(x_1, x_2, x_3) &= 1 - x_1 - x_2 - (1 - x_3) + x_1x_2 + x_1(1 - x_3) + x_2(1 - x_3) \\ &= x_3 + x_1x_2 - x_1x_3 - x_2x_3 \end{aligned} \tag{3}$$

It is simple to confirm that this polynomial does capture $(x_1, x_2, \neg x_3)$. By creating these polynomials it is then possible to find a kernel of size $\mathcal{O}(n^2)$ for 3-NAE-SAT as shown by Jansen and Pieterse [10].

This was later expanded upon by Chen, Jansen and Pieterse to find a kernel [12] for the more general problem definition $CSP(\Gamma)$ in Theorem 2.15. Here the definition $\mathbb{Z}/q\mathbb{Z}$ is used. This denotes the field of integers modulo q. Here two integers i, i' are equivalent if $i \equiv i' \mod q$. This is abbreviated by $i \equiv_q i'$. One important notion which they use is that of the span of a set of vectors as given in Definition 2.16.

Theorem 2.15 ([12, Theorem 3.5]). *Let $R \subseteq \{0,1\}^k$ be a fixed k-ary relation, such that for every $u \in \{0,1\}^k \setminus R$ there exists a ring $E_u \in \{\mathbb{Q}\} \cup \{\mathbb{Z}/q_u\mathbb{Z} | q_u > 1$ and $q_u \in \mathbb{N}\}$ and polynomial p_u over E_u of degree at most d that captures u with respect to R. Then there exists a polynomial-time algorithm, given a set of constraints C over R over n variables, outputs $C' \subseteq C$ with $|C'| = \mathcal{O}(n^d)$, such that any Boolean assignment satisfies all constraints in C if and only if it satisfies all constraints in C'.*

Definition 2.16 (Span). *Given a set of k-ary vectors $S = \{s_1, ..., s_m\}$ in $\mathbb{Z}^k$, $Span_{\mathbb{Z}}(S)$ is defined as the set of all vectors $y \in \mathbb{Z}^k$ such that there exist integers $\alpha_1, ..., \alpha_m \in \mathbb{Z}$ such that $y = \sum_{i\in[m]} \alpha_i s_i$. Similarly, $Span_q(S)$ defines the set of all k-ary vectors y over $\mathbb{Z}/q\mathbb{Z}$ such that there exist integers $\alpha_1, ..., \alpha_m$ such that $y \equiv_q \sum_{i\in[n]} \alpha_i s_i$.*

For some $m \times n$ matrix A we let a_i for $i \in [m]$ denote the ith row vector of A. In this thesis we will denote for such an $m \times n$ matrix A the span of the row vectors a_i by $Span_{\mathbb{Z}}(A)$ and $Span_q(A)$ as defined in Definition 2.16. Using this definition, Chen, Jansen and Pieterse proved, among others, Lemmas 2.17 and 2.18 which are used to construct polynomials p_u that capture unsatisfying $u \in U = \{0,1\}^k \setminus R$ assignments with respect to the respective relation R over the ring $\mathbb{Z}/q\mathbb{Z}$.

Lemma 2.17 ([12, Lemma 4.3]). *Let S be an $m \times n$ integer matrix. Let $u \in \mathbb{Z}^n$ be a row vector. If $u \notin Span_{\mathbb{Z}}(S)$, then there exist a prime power q such that $u \notin Span_q(S)$. Furthermore, there is a polynomial-time algorithm that computes a (possibly composite) integer q' for which $u \notin Span_{q'}(S)$.*

Lemma 2.18 ([12, Lemma 4.6]). *Let $q > 1$ be an integer. Let A be an $m \times n$ matrix over $\mathbb{Z}_q$. Suppose $a_m \notin Span_q(\{a_1, ..., a_{m-1}\})$. Then there exists a constant $c \not\equiv_q 0$ for which the system $Ax \equiv_q b$ has a solution, where $b = (0, ..., 0, c)^T$ is the vector with c on the last position and zeros in all other position. Furthermore, x and c can be computed in polynomial time.*

3 Kernels for non-uniform CSP

In this section we will be constructing a kernel for $CSP(\Gamma)$. In Section 3.1 we will show how to construct a kernel of $\mathcal{O}(n^t)$ for $CSP(\Gamma)$ defined over a so-called t-balanced Boolean constraint language Γ. This will be done by constructing polynomials of degree t which capture the unsatisfying assignments of the relations in Γ. In section 3.2 we will enrich this definition for $CSP(\Gamma)$ defined over a finite domain D of size $|D| \geqslant 2$. In Section 3.3 and 3.4 we show the difficulties of constructing a single polynomial for a relation $R \in \Gamma$ which captures all unsatisfying assignments with respect to R. Finally in Section 3.5 we show that for any two polynomials which each capture a different unsatisfying assignment with respect to R, it is possible to construct a single polynomial that captures both unsatisfying assignments with respect to R.

3.1 Generic Kernel for Boolean non-uniform CSP

First we introduce the notion of a t-extended form of a relation R over a binary domain, denoted by R_t. To do so properly we first introduce the $\delta(k,t)$ function.

Definition 3.1. *For non-negative integers k, t, the function $\delta(k,t)$ is defined by $\delta(k,t) = \sum_{i=0}^{t} \binom{k}{i}$.*

Using Definition 3.1, we give the definition of the t-extended form of a relation R in Definition 3.2.

Definition 3.2. *For a k-ary Boolean relation R, the t-extended form of R is defined as the Boolean relation R_t of arity $\delta(k,t)$ which contains the extended tuples r', extended from $r \in R$, for which for all sets $S_1, S_2, ..., S_{\delta(k,t)} \in \binom{[k]}{\leqslant t}$ sorted in lexicographical order, it holds that $r'[j] = \prod_{l \in S_j} r[l]$.*

Note that in this definition, for all $r \in R$, $r'_i[1]$ is set to 1, since the set $S_1 = \emptyset$ for $S_1 \in \binom{[k]}{\leqslant t}$ and therefore $r'[j] = \prod_{l \in \emptyset} r_i[l] = 1$ as the product of the elements of $\emptyset$ is considered to be 1. A matrix representation can be seen for 3-NAE-SAT and 4-NAE-SAT in Equations 4 and 5 respectively. The matrix on the left represents the non-negated relation of the d-NAE-SAT problem and the matrix on the right represents the degree-$(d-1)$-extended form.

$$\begin{array}{cc}
\begin{array}{ccc} r[1] & r[2] & r[3] \end{array} & \begin{array}{c|ccc|ccc} S_1 = \emptyset & S_2 = \{1\} & S_3 = \{2\} & S_4 = \{3\} & S_5 = \{1,2\} & S_6 = \{1,3\} & S_7 = \{2,3\} \end{array} \\
\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} & \left(\begin{array}{c|ccc|ccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right)
\end{array} \tag{4}$$

$$
\begin{array}{cccc}
r[1] & r[2] & r[3] & r[4] \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 \\
0 & 1 & 1 & 1
\end{array}
\quad
\begin{array}{c|cccc|cccccc|cccc}
S_1 & S_2 & S_3 & S_4 & S_5 & S_6 & S_7 & S_8 & S_9 & S_{10} & S_{11} & S_{12} & S_{13} & S_{14} & S_{15} \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1
\end{array}
\tag{5}
$$

Using this definition we will define what it means when a relation is t-preserved and what it means to be t-balanced.

Definition 3.3. *Let $f : \{0,1\}^n \to \{0,1\}$ be some partial operation. Let $R \subseteq \{0,1\}^k$ be a Boolean relation of arity k and let $U = \{0,1\}^k \setminus R$ be the set of all Boolean assignments of arity k not satisfying R. Let R_t, U_t be the t-extended forms of R, U respectively, and let $I = \{0,1\}^{\delta(k,t)} \setminus (U_t \cup R_t)$ be the set of invalid assignments. A relation R is t-preserved by f if for any tuples $r'_1, ..., r'_n \in R_t$ for which the entries of the tuple $(f(r'_1[1], ..., r'_n[1]), ..., f(r'_1[\delta(k,t)], ..., r'_n[\delta(k,t)]))$ are defined and not contained in I, then the tuple is an element of R_t.*

Definition 3.4. *Let $R \subseteq \{0,1\}^k$ be a k-ary Boolean relation. A relation R is t-balanced if it is t-preserved by all balanced operations f. A constraint language Γ is t-balanced when all relations R in Γ are t-balanced.*

A good example of such a relation is 3-NAE-SAT shown in Equation 4. While it is not 1-balanced, it is 2-balanced. It is not 1-balanced as the balanced operation $f(x_1, x_2, x_3) = x_1 - x_2 + x_3$ does not preserve the 1-extended relation, which is the same as the matrix on the left with 1s appended to the left. If we were to take the tuples $(1,1,0,0), (1,1,1,0), (1,0,1,0)$ one can verify that applying f this will result in the tuple $(f(1,1,1), f(1,1,0), f(0,1,1), f(0,0,0)) = (1,0,0,0)$ which is not satisfying as this is the 1-extended form of $(0,0,0)$, which is an unsatisfying assignment. However, it is 2-balanced since we cannot construct a balanced operation which does not preserve the 2-extended form. It is easy to confirm that for the previous example, if we take the same f and the 2-extended forms of the same tuples, i.e $(1,1,0,0,0,0,0), (1,1,1,0,1,0,0), (1,0,1,0,0,0,0)$, the result of applying f on these tuples results in the tuple $(1,0,0,0,-1,0,0)$ which is not unsatisfying. It is, however, invalid since it does not represent an actual tuple, either satisfying or unsatisfying. This does therefore not make 3-NAE-SAT not 2-balanced by Definition 3.3.

We can see that Definition 3.3 is an extension of Definition 2.9 since the definitions are equivalent when $t = 1$. The same can be said for Definitions 3.4 and 2.10. We also define R_t to be the t-extended form of some arbitrary k-ary Boolean relation R and U_t to be the t-extended form of the unsatisfying assignments $U = \{0,1\}^k \setminus R$. Using these Definitions, we will prove Lemma 2.13 for a t-balanced relation.

Lemma 3.5. *Let $R \subseteq \{0,1\}^k$ be a Boolean relation of arity k. Then the following statements are equivalent:*

1. *R is t-balanced,*
2. *for all odd $n \geqslant 1$ the alternating operation of arity n t-preserves R,*
3. *for all $u' \in U_t$ it holds that $u' \notin Span_{\mathbb{Z}}(R_t)$.*

Proof. To prove this Lemma, we first prove the equivalence of 1 and 2. Since the alternating operation is a balanced operation, it must t-preserve R if R is t-balanced. Thus it remains to be proven that whenever for all odd $n \geqslant 1$ the alternating of arity n t-preserves R, then R is t-balanced. Let R be a non t-balanced relation. By Definition 3.3 this means that there is a partial operation $f : \{0,1\}^n \to \{0,1\}$ which is balanced and has the property

$$(f(r'_1[1], ..., r'_n[1]), ..., f(r'_1[\delta(k,t)], ..., r'_n[\delta(k,t)])) \in U_t$$

for some $r'_1, ..., r'_n \in R_t$. In addition, we know that there are coefficients $\alpha_1, \alpha_2, ..., \alpha_n \in \mathbb{Z}$ such that $\sum_{i=1}^{n} \alpha_i = 1$ and $f(x_1, x_2, ..., x_n) = \sum_{i=1}^{n} \alpha_i x_i$ by definition of a balanced operation. Thus for the chosen tuples $r'_i \in R_t$ we can generalize this as $\sum_{i=1}^{n} \alpha_i r'_i \in U_t$. Now using these coefficients we can construct an alternating operation with the same properties. We now replace $\alpha_i r'_i$ by $r'_i + ... + r'_i (\alpha_i$ times) if α_i is positive and by $-x_i - ... - x_i$ ($-\alpha_i$ times) if α_i is negative. Since all α_i sum up to 1, we know that these new coefficients $1, -1$ will sum to 1 as well. By simply rearranging the terms we can create an alternating operation that preserves the equality of the original function f.

This shows that statements 1 and 2 are equivalent. To finalize the proof we show that statements 1 and 3 are equivalent. We do so by proving the contrapositive for both directions of the equivalence.

case ($\Leftarrow$):
Let R be some non-t-balanced relation. This means there is some partial balanced operation $f : \{0,1\}^n \to \{0,1\}$ for which there is an entry $u' \in U_t$ for which there are tuples $r'_1, ..., r'_n \in R_t$ such that $(f(r'_1[1], ..., r'_n[1]), ..., f(r'_1[\delta(k,t)], ..., r'_n[\delta(k,t)])) = u'$. Now by Definition 2.10 we know that there exist integer coefficients $\alpha_1, ..., \alpha_n$ which satisfy $\sum_{i=1}^{n} \alpha_i = 1$, $(x_1, ..., x_n)$ is in the domain of f if and only if $\sum_{i=1}^{n} \alpha_i x_i \in \{0,1\}$, and $f(x_1, ..., x_n) = \sum_{i=1}^{n} \alpha_i x_i$ for all tuples in the domain of f. Now since $f(x_1, ..., x_n) = \sum_{i=1}^{n} = \alpha_i x_i$ for all tuples in the domain of f, we can express u' as $u' = \sum_{i=1}^{n} \alpha_i r'_i$. By Definition 2.16 this is equivalent to $u' \in Span_{\mathbb{Z}}(R_t)$ thus proving this case.

case ($\Rightarrow$):
Let $u' \in U_t$ be the t-extended form of some unsatisfying assignment such that $u' \in Span_{\mathbb{Z}}(R_t)$. By Definition 2.16, we know that there exist integer coefficients $\alpha_1, ..., \alpha_n$ exist such that $u' = \sum_{i=1}^{n} \alpha_i r'_i$ for some $r'_1, ..., r'_n \in R_t$. Using these $\alpha_1, ..., \alpha_n$, we construct the partial operation $f : \{0,1\}^n \to \{0,1\}$ such that $f(x_1, ..., x_n) = \sum_{i=1}^{n} \alpha_i x_i$. In addition, for this f we define that any tuple $(x_1, ..., x_n)$ is in the domain of f if and only of $\sum_{i=1}^{n} \alpha_i x_i \in \{0,1\}$. Now we need to prove that f is a balanced operation. Clearly by construction, requirements 2 and 3 of Definition 2.10 are already satisfied. Thus it remains to be proven that $\sum_{i=1}^{n} \alpha_i = 1$. By choice of $\alpha_1, ..., \alpha_n$, we know that $u' = \sum_{i=1}^{n} \alpha_i r'_i$, and therefor $u'[1] = \sum_{i=1}^{n} \alpha_i r'_i[1]$. By definition of a t-extended tuple, any arbitrary t-extended tuple a would satisfy $a[1] = 1$ as we have stated previously. Thus $u'[1] = 1$ and also $r'_i[1] = 1$ for all $i \in [n]$. Thus we get that the summation $u'[1] = \sum_{i=1}^{n} \alpha_i r'_i[1]$ is equivalent to $1 = \sum_{i=1}^{n} \alpha_i$, and therefore f is a balanced partial operation. Finally it is easy to see that, since $u' = \sum_{i=1}^{n} \alpha_i r'_i = (f(r'_1[1], ..., r'_n[1]), ..., f(r'_1[\delta(k,t)], ..., r'_n[\delta(k,t)]))$ and $r_1, ..., r_n$

are elements of R_t, f does not t-preserve R and thus R is not t-balanced which proves this case.

Now with the equivalence of statements 1 and 3 proven, we can infer by the equivalence of 1 and 2 that 2 and 3 are equivalent as well. Thus statements 1,2 and 3 are equivalent for any Boolean relation $R \subseteq \{0,1\}^k$. □

This gives us a powerful tool to identify t-balanced relations. In order to sparsify an instance $CSP(\Gamma)$ over a t-balanced language Γ, it would be desirable if this relation can be captured by a degree t polynomial as this allows us to construct a kernel of size $O(n^t)$ by Theorem 2.15. To this end we prove Lemma 3.6, which is an extension of method used for constructing linear polynomials for balanced relations by Chen, Jansen and Pieterse [12].

Lemma 3.6. *Let $R \subseteq \{0,1\}^k$ be a k-ary t-balanced Boolean relation and let $U = \{0,1\}^k \setminus R$ be the set of Boolean assignments of arity k not satisfying R. For each $u \in U$ there exists a k-variate polynomial p_u of degree t over $\mathbb{Z}/q\mathbb{Z}$, for some integer $q \in \mathbb{N}$, that captures u with respect to R.*

Proof. Let $R_t = \{r'_1, r'_2, ..., r'_m\}$ denote the t-extended form of $R = \{r_1, r_2, ..., r_m\}$ such that r'_i is the t-extended form of r_i for $i \in [m]$. Now since R is t-balanced, we know that there is no balanced partial operation $f : \{0,1\}^n \to \{0,1\}$ which will result in $(f(r'_1[1], ..., r'_n[1]), ..., f(r'_1[\delta(k,t)], ..., r'_n[\delta(k,t)])) \in U_t$ for any choice of $r'_1, ..., r'_n \in R_t$.

For R_t we construct its matrix representation, which contains all tuples of R_t as row vectors, which will be used to construct the desired polynomial. This is shown in Equation (6).

$$\begin{pmatrix} r'_1[1] & r'_1[2] & r'_1[3] & \cdots & r'_1[\delta(k,t)] \\ r'_2[1] & r'_2[2] & r'_2[3] & \cdots & r'_2[\delta(k,t)] \\ r'_3[1] & r'_3[2] & r'_3[3] & \cdots & r'_3[\delta(k,t)] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r'_m[1] & r'_m[2] & r'_m[3] & \cdots & r'_m[\delta(k,t)] \end{pmatrix} \tag{6}$$

The matrix of R_t is thus an $m \times \delta(k,t)$ matrix, since there is a row for each satisfying assignment in the matrix. Using this matrix we will show that for each $u \in U$ there exists a k-variate polynomial p_u of degree t for which there exists an integer q such that p_u is defined over the ring $\mathbb{Z}/q\mathbb{Z}$. For the remainder of this proof, we will pick an arbitrary choice of $u \in U$ for which a polynomial p_u will be constructed. Since R is t-balanced, we know that $u' \notin Span_{\mathbb{Z}}(R_t)$ for all $u' \in U_t$ by Lemma 3.5. By using Lemmas 2.17 and 2.18 since $u' \notin Span_{\mathbb{Z}}(R_t)$, we can find integers q, $c \not\equiv_q 0$ and $\alpha_1, ..., \alpha_{\delta(k,t)}$ such that the following system is satisfied.

$$\begin{pmatrix} r'_1[1] & r'_1[2] & r'_1[3] & \cdots & r'_1[\delta(k,t)] \\ r'_2[1] & r'_2[2] & r'_2[3] & \cdots & r'_2[\delta(k,t)] \\ r'_3[1] & r'_3[2] & r'_3[3] & \cdots & r'_3[\delta(k,t)] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r'_m[1] & r'_m[2] & r'_m[3] & \cdots & r'_m[\delta(k,t)] \\ u'[1] & u'[2] & u'[3] & \cdots & u'[\delta(k,t)] \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_{\delta(k,t)} \end{pmatrix} \equiv_q \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ c \end{pmatrix} \tag{7}$$

Note that the matrix on the left is an $(m+1) \times \delta(k,t)$ matrix which represents R_t with some $u' \in U_t$ concatenated as the last row. In addition, the vector of values α_i for $i \in [\delta(k,t)]$ is of length $\delta(k,t)$ and the desired outcome is a vector of length $m+1$. For the values $\alpha_1, ..., \alpha_{\delta(k,t)}$ an arbitrary assignment is chosen which satisfies (7) and fixed for the remainder of the proof. We claim that for u there is a k-variate polynomial p_u defined over $\mathbb{Z}/q\mathbb{Z}$ of degree t such that R is captured by p_u with respect to u which is given by Equation (8).

$$p_u(x_1, x_2, ..., x_k) \equiv_q \sum_{i=1}^{\delta(k,t)} \alpha_i \prod_{j \in S_i} x_j \tag{8}$$

Here the sets S_i for all $i \in [\delta(k,t)]$ are defined as $S_1, S_2, ..., S_{\delta(k,t)} \in \binom{[k]}{\leqslant t}$ sorted in lexicographical order. Since this uses the same sets S_i as in Definition 3.2 we see that there is a multivariate term in p_u corresponding to each value in a tuple $r'_i \in R_t$ and similarly for all $u' \in U_t$. In addition, since the size of any $S_i \in \binom{[k]}{\leqslant t}$ is at most t, the construction thus creates a polynomial which is the sum of terms which are at most of degree t. Thus the entire polynomial which is constructed is of degree t.

Now suppose that p_u does not capture u with respect to R_t. This means that either for some $r \in R$, $p(r) \not\equiv_q 0$ or for $u \in U$ it holds that $p_u(u) \equiv_q 0$. The second case would imply that $p_u(u) = \sum_{i=1}^{\delta(k,t)} u'[i] \cdot \alpha_i = u'\vec{\alpha} \equiv_q 0$ where $\vec{\alpha} = (\alpha_1, ..., \alpha_{\delta(k,t)})^T$. However, this contradicts Lemma 2.18 as this ensures that $u'\vec{\alpha} \equiv_q c$ for some $c \not\equiv_q 0$.

Thus it remains to be proven that for all $r \in R$, $p_u(r) \equiv_q 0$. Assume the inverse. This would mean that there is some $r \in R$ for which $p_u(r) \not\equiv_q 0$. Now we know by equation 8 that $p_u(x_1, ..., x_k) = \sum_{i=1}^{\delta(k,t)} \alpha_i \prod_{i \in S_i} x_i$ for $S_1, ..., S_{\delta(k,t)} \in \binom{[k]}{\leqslant t}$. In addition, the values for some entry r', where r' is the t-extended form of some tuple $r \in R$, were also constructed similarly by taking $r'[i] = \prod_{i \in S_i} r[i]$. Now one can rewrite $p_u(x_1, ..., x_k)$ to the inner product of the row vector $\vec{x}$ and to column vector $\vec{\alpha}$ as $p_u(x_1, ..., x_k) = \vec{x}\vec{\alpha}$. Here $\vec{x}$ is the row vector containing multivariate monomials of p_u sorted in lexicographical order in terms of the sets $S_i \in \binom{[k]}{\leqslant t}$ used to define the related monomial. Then by definition, if our assumption is to be correct, this would mean that $p_u(r) = \sum_{i=1}^{\delta(k,t)} r'[i] \cdot \alpha_i = r'\vec{\alpha} \not\equiv_q 0$. However, this contradicts that the values of $\vec{\alpha}$ for a solution to (7). Thus we conclude that this polynomial satisfies $p_u(r) \equiv_q 0$ for all $r \in R$.

From this we can conclude that, if R is t-balanced, for each $u \in U$ we can find a polynomial of degree t that captures u with respect to R. □

Using Lemma 3.6, we want to find a kernel for $CSP(\Gamma)$ with $O(n^t)$ constraints. We thus want to prove Lemma 3.7.

Lemma 3.7. *Let Γ be some Boolean constraint language that is t-balanced. Then $CSP(\Gamma)$ parameterized by the number of variables n admits a kernel with $O(n^t)$ constraints which are a subset of the original constraints.*

Proof. Let the instance (I, n) denote an instance $I = (C, V)$, where C is the set of constraints and V the set of variables, of $CSP(\Gamma)$ parameterized by the number of variables $n = |V|$. Let $CSP(\Gamma)$ be defined over a t-balanced language Γ. By definition of a t-balanced language, we know that all $R_i \in \Gamma$ are t-balanced. For each $R_i \in \Gamma$, compute its set of unsatisfying assignments $U_i = \{0,1\}^k \setminus R_i$. For any subset of constraints $C_i \subseteq C$ defined over the relation R_i which is part of some problem $CSP(\Gamma)$, where $R_i(x_1, ..., x_k)$ is a constraint in C_i and $u \in U_i$, we know that polynomials $p_u(x_1, ..., x_k)$ exists by Lemma 3.6 that capture these constraints since R_i is t-balanced. By applying Theorem 2.15 we get a set of constraints $C'_i \subseteq C_i$ which is satisfied if and only if C_i is satisfied. In addition, we know that C'_i is of size $O(n^t)$. The kernelized instance of (I, n) is given by (I', n) where $I' = (C', V)$. Here C' is constructed by $C' = \cup_{i=1}^{|\Gamma|} C'_i$. Since each C'_i is of size $O(n^t)$ and since $|\Gamma|$ is constant, we find that $C' = |\Gamma|O(n^t) = O(n^t)$. Now it is easy to see that I' is also an instance of $CSP(\Gamma)$ equivalent to I from which we conclude that Lemma 3.7 is correct. □

We have thus found a way to sparsify any instance of $CSP(\Gamma)$ defined over a Boolean t-balanced constraint language Γ to an equivalent instance with $O(n^t)$ constraints. Using computer aided calculations, we found that 1-IN-4-SAT and 4-NAE-SAT are 1-balanced and 3-balanced respectively. This implies that these problems have kernels of $\mathcal{O}(n)$ and $\mathcal{O}(n^3)$ constraints respectively. So the presented method agrees with the established kernel bounds of previous work [10].
In addition we explored a new relation, which is given in Equation (9).

$$R = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} \tag{9}$$

This relation represents all possible ways to select an independent vertex set from the graph shown in Figure 1. The columns in Equation (9) represent the vertices sorted in numerical order and the rows represent a specific selection of vertices such that the selection forms an independent set.

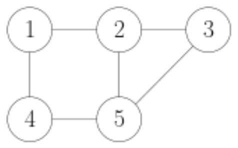


Figure 1: Graph related to the relation in Equation (9)

We found that this relation was 2-balanced, thus implying that a variant of $CSP(\Gamma)$ which is defined over this relation has kernel of $\mathcal{O}(n^2)$ constraints. This result is significant as this relation has not yet been shown to have a nontrivial kernel.

3.2 Kernel for non-uniform CSP over finite domains

While we can construct a kernel of $O(n^t)$ constraints for any $CSP(\Gamma)$ defined over a t-balanced Boolean constraint language, it would be desirable to extend this methodology to constraint languages over any domain D. The main property that we will apply is the notion of a choice representation which transforms a relation R over a finite domain D into a binary representation. This representation can then be used to create polynomials which capture unsatisfying assignments $u \in D^k \setminus R$ with respect to R. The definition of a choice assignment is given in Definition 3.8.

Definition 3.8 (Choice representation)**.** *Let $D = \{d_1, ..., d_h\}$ be some arbitrary domain. Let $x \in D^k$ be some vector of size k defined over D. The choice representation of x, denoted by x_c, is the $k \cdot h$-dimensional vector satisfying for all $i \in [k], j \in [h]$, $x_c[(i-1) \cdot h + j] = 1$ if and only if $x[i] = d_j$.*

This essentially means that for each value of $i \in [k]$, there is only a single value $j \in [h]$ for which it holds that $x_c[(i-1) \cdot h + j]$ is set to 1. This effectively means that $x_c[(i-1) \cdot h + j]$ represents the binary choice of the values which could be assigned to $x[i]$, as $x_c[(i-1) \cdot h + j] = 1$ if and only if $x[i] = d_j$ and any entry $x[i]$ can only be assigned a single value of D. Definition 3.8 can then be applied to a relation $R \subseteq D^k$ to transform a relation over a finite domain into a binary one. We let the set R_c denote the set choice representations of all tuples in the relation R. This can also be thought of as a relation of arity $k \cdot h$ which we will denote by k'. In Equation (10) an example is shown for 3-coloring. The left matrix shows the matrix representation of the original relation for an edge with vertices v_1, v_2 and the right matrix shows the matrix representation of the choice representation of the same relation. Here the row vectors represent the tuples of the given relations. For example, for the tuple $(1, 2)$ of 2-coloring shown in the left matrix, the choice representation is given by the tuple $(1, 0, 0, 0, 1, 0)$ in the right matrix.

$$\begin{array}{cc} v_1 & v_2 \\ \end{array} \quad \begin{array}{cccccc} v[1] & v[2] & v[3] & v[4] & v[5] & v[6] \end{array}$$
$$\begin{pmatrix} 1 & 2 \\ 1 & 3 \\ 2 & 1 \\ 2 & 3 \\ 3 & 1 \\ 3 & 2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \tag{10}$$

This will be used to construct a set of k'-variate polynomials p_u defined over $\mathbb{Z}/q\mathbb{Z}$ for some integer q which captures $u \in U_c$, where $U = D^k \setminus R$ is the set of unsatisfying k-ary assignments with respect to R and U_c the set of all choice representations of the tuples in U. In addition, we need to use the t-extended form of these choice representations. For a choice representation R_c of R we denote the t-extended form by $(R_c)_t$ as we first take the choice representation of R and then t-extend it. For brevity we will denote this by R_{ct}. To construct these polynomials, a new definition for balanced relations must be constructed using the choice representation, which is given by Definitions 3.9, 3.10 and 3.11.

Definition 3.9. *Let $D = \{d_1, ..., d_h\}$ be a domain and let $R \subseteq D^k$ be some k-ary relation defined over the domain D. Let R_c and $(D^k)_c = D^k_c$ be the $k \cdot h = k'$-ary choice representations of R and D^k respectively. Let R_{ct} and $(D^k_c)_t = D^k_{ct}$ be the t-extended form of R_c and D^k_c respectively of arity $\delta(k', t)$. Let $U_c = D^k_c \setminus R_c$ be the k'-ary choice representations of all assignments not in R_c and let $(U_c)_t = U_{ct}$ be the t-extended form of U_c of arity $\delta(k', t)$. A choice representation R_c is choice t-preserved by a partial function $f : \{0,1\}^n \to \{0,1\}$ when for each choice of tuples $r'_1, r'_2, ..., r'_n \in R_{ct}$, if the entries in the tuple $(f(r'_1[1], ..., r'_n[1]), ..., f(r'_1[\delta(k', t)], ..., r'_n[\delta(k', t)]))$ are defined and this tuple is in D^k_{ct}, then this tuple is in R_{ct}.*

Definition 3.10. *Let $D = \{d_1, ..., d_h\}$ be a domain and let $R \subseteq D^k$ be some k-ary relation defined over the domain D. Let R_c and $(D^k)_c = D^k_c$ be the $k \cdot h = k'$-ary choice representations of R and D^k respectively. A choice representation R_c is choice t-balanced if it is choice t-preserved by all balanced operations $f : \{0,1\}^n \to \{0,1\}$.*

Definition 3.11. *Let $R \subseteq D^k$ be a k-ary relation over a domain D and let R_c be the choice representation of R. We say that R is t-balanced if R_c is choice t-balanced. By extension, a constraint language Γ is t-balanced if all relations $R \in \Gamma$ are t-balanced.*

Using these definitions we first prove that there is always an alternating operation that choice t-preserves R_c with R being a t-balanced relation over a domain D.

Lemma 3.12. *Let $R \subseteq D^k$ be a relation of arity k over the domain $D = \{d_1, ..., d_h\}$. Let R_c be the choice representation of R. Let $U = D^k \setminus R$ be the set of assignments not satisfying R and U_c be the choice representation of U. Then the following statements are equivalent:*

1. *R is t-balanced*
2. *for all odd $n \geqslant 1$ the alternating operation of arity n choice t-preserves R_c*
3. *for all $u' \in U_{ct}$, it holds that $u' \notin Span_{\mathbb{Z}}(R_{ct})$*

Proof. Similar to the proof of Lemma 3.5 we will first prove that 1 and 2 are equivalent. First off, it is easy to verify that if R is t-balanced, that the alternating operation would choice t-preserve R_c as the alternating operation is a balanced operation. Now it remains to be proven that when the alternating operation choice t-preserves R_c that R is t-balanced.

Consider a non t-balanced k-ary relation R over a domain D. Since R is not t-balanced, then for the t-extended choice assignment representation R_{ct} of R there exists a balanced partial function $f : \{0,1\}^n \to \{0,1\}$ which does not choice t-preserve R_{ct}. Let $U \subseteq D^k \setminus R$ be the set of all unsatisfying assignments of arity k over domain D and let U_c be the choice representation of U. We again denote the t-extended form of U_c by U_{ct}. This thus means that for some $r'_1, ..., r'_n \in R_{ct}$ it holds that $(f(r'_1[1], ..., r'_n[1]), ..., f(r'_1[\delta(k', t)], ..., r'_n[\delta(k', t)])) \in U_{ct}$, with $k' = k \cdot h$. By extension, there exist integer coefficients $\alpha_1, \alpha_2, ..., \alpha_n$ for which it holds that $\sum_{i=1}^n \alpha_i r'_i = u'$ for some $u' \in U_{ct}$. It is then possible to replace each term $\alpha_i r'_i$ in this summation by $r'_i + ... + r'_i$ (α_i times) if α_i is positive and $-r'_i - ... - r'_i$ ($-\alpha_i$ times) if α_i is negative. By reordering these newly created terms it is possible to create an alternating operation since we know that $\sum_{i=1}^n \alpha_i = 1$ which thus proves this case.

Finally it is easy to confirm that 1 and 3 are equivalent. This will be proven by proving the contrapositive if this equivalence.

($\Leftarrow$):
Suppose R is not t-balanced. This implies that there is a balanced partial operation $f : \{0,1\}^n \to \{0,1\}$ which does not choice t-preserve R_c. This means that there are some tuples $r'_1, ..., r'_n \in R_{ct}$ for which it holds that $(f(r'_1[1], ..., r'_n[1]), ..., f(r'_1[\delta(k', t)], ..., r'_n[\delta(k', t)])) = u'$ and $u' \in U_{ct}$ by Definition 2.10. In addition, since f is balanced partial operation, we know that there exist integers $\alpha_1, ..., \alpha_n \in \mathbb{Z}$ such that $\sum_{i=1}^n \alpha_i = 1$, $(x_1, ..., x_n)$ is in the domain of f if and only if $\sum_{i=1}^n \alpha_i x_i \in \{0,1\}$ and $f(x_1, ..., x_n) = \sum_{i=1}^n \alpha_i x_i$ for all tuples in the domain of f. Thus the tuple u' can be rewritten as $u' = \sum_{i=1}^n \alpha_i r'_i$. By Definition 2.16 we can conclude that $u' \in Span_{\mathbb{Z}}(R_{ct})$.

($\Rightarrow$):
Suppose that $u' \in Span_{\mathbb{Z}}(R_{ct})$. By Definition 2.16 there exist integers $\alpha_1, ..., \alpha_n \in \mathbb{Z}$ such that $u' = \sum_{i=1}^{n} \alpha_i r'_i$ for some $r'_1, ..., r'_n \in R_{ct}$. Using these values for $\alpha_1, ..., \alpha_n$ we construct the partial operation $f : \{0,1\}^n \to \{0,1\}$ such that $f(x_1, ..., x_n) = \sum_{i=1}^{n} \alpha_i x_i$ and a tuple $(x_1, ..., x_n)$ is in the domain of f if and only if $\sum_{i=1}^{n} \alpha_i x_i \in \{0,1\}$. Now we will prove that f is a balanced operation. f already fulfills the second and third property of Definition 2.10. It thus remains to prove that $\sum_{i=1}^{n} \alpha_i = 1$. Note that for any t-extended tuple a satisfies $a[1] = 1$. Thus also $u'[1] = 1$ and $r'_i[1] = 1$ for any choice of $i \in [n]$. Since we know $u' = \sum_{i=1}^{n} \alpha_i r'_i$, then also $u'[1] = \sum_{i=1}^{n} \alpha r'_i[1]$ should hold. Therefore $1 = u'[1] = \sum_{i=1}^{n} \alpha_i r'_i = \sum_{i=1}^{n} \alpha_i$, which leads us to conclude that f is a balanced partial operation. Now it is easy to see that by definition $(f(r'_1[1], ..., r'_n[1]), ..., f(r'_1[\delta(k',t)], ..., r'_n[\delta(k',t)])) = u'$ and that $u' \notin R_{ct}$. This means that R_c is not choice t-balanced and thus R is not t-balanced.

Since we have proven both cases, we can conclude that 1 and 3 are equivalent. Thus by equivalence of 1 and 2, we can conclude that 2 and 3 are equivalent as well. This proves that statements 1, 2 and 3 are equivalent for any relation $R \subseteq D^k$ defined over a finite domain D. □

Using Lemma 3.12 we will show that it is possible to construct polynomials which capture a relation R in terms of its choice assignment representation R_c.

Lemma 3.13. *For a t-balanced relation $R \subseteq D^k$ with the set of unsatisfying assignments $U = D^k \setminus R$ and partial choice assignment representation R_c and U_c respectively, there exists an integer q and a $k \cdot |D|$-variate polynomial p_u over domain $\mathbb{Z}/q\mathbb{Z}$ which captures $u \in U_c$ with respect to R_c.*

Proof. Let R_{ct} be the t-extended form of R_c. Using this form we construct for each $u \in U_c$ a k'-variate p_u, with $k' = k \cdot |D|$, over the ring $\mathbb{Z}/q\mathbb{Z}$ which captures u with respect to R_c for some integer q. We denote the tuples in R_{ct} by $r'_1, ..., r'_n$. We construct the system in Equation 11 using this information.

$$\begin{pmatrix} r'_1[1] & r'_1[2] & r'_1[3] & \cdots & r'_1[\delta(k',t)] \\ r'_2[1] & r'_2[2] & r'_2[3] & \cdots & r'_2[\delta(k',t)] \\ r'_3[1] & r'_3[2] & r'_3[3] & \cdots & r'_3[\delta(k',t)] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r'_n[1] & r'_n[2] & r'_n[3] & \cdots & r'_n[\delta(k',t)] \\ u'[1] & u'[2] & u'[3] & \cdots & u'[\delta(k',t)] \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_{\delta(k',t)} \end{pmatrix} \equiv_q \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ c \end{pmatrix} \tag{11}$$

u' in Equation 11 denotes some tuple from U_{ct}. We know by Lemmas 2.17 and 2.18 and the fact that R_c is choice t-balanced, as this implies that $u' \notin Span_{\mathbb{Z}}(R_{ct})$ by Lemma 3.12, that there exist integers q, c, where $c \not\equiv_q 0$, and $\alpha_1, ..., \alpha_{\delta(k',t)}$ which will ensure that equation 11 is satisfied. For the remainder of the proof we will fix these values of q, c and $\alpha_1, ..., \alpha_{\delta(k',t)}$. It now remains to use this system of equations to construct a polynomial of degree t. Since we have a solution for this system, we know that we can construct the polynomial

$$p_u(x_1, x_2, ..., x_{k'}) \equiv_q \sum_{i=1}^{\delta(k',t)} \alpha_i \prod_{l \in S_i} x_l \tag{12}$$

for all $S_1, S_2, ..., S_{\delta(k',t)} \in \binom{k'}{t\leq}$. Now if we take some $r \in R_c$ and define $\vec{\alpha} = (\alpha_1, ..., \alpha_{\delta(k',t)})^T$ we get $p_u(r) \equiv_q \sum_{i=1}^{\delta(k',t)} \alpha_i \prod_{l \in S_i} r_l \equiv_q \sum_{i=1}^{\delta(k',t)} \alpha_i r'[i] \equiv_q r'\vec{\alpha}$. Now by Equation 11 and Lemmas 2.17

and 2.18 we can confirm that $r'\vec{\alpha} \equiv_q 0$ which is exactly the desired property. It can similarly be reasoned that $p_u(u) \not\equiv_q 0$. Thus p_u captures u with respect to R_c. □

While Lemma 3.13 shows us how to construct a set of polynomials which together capture R_c, it should still be possible to convert the obtained result into a kernel for the original problem over the domain $D = \{d_1, ..., d_h\}$. Since by construction the values of a choice assignment tuple r_c satisfy $r_c[(i-1) \cdot h + j] = 1$ if and only if for the related tuple $r \in R$ it holds that $r[i] = d_j$, we have a trivial way of converting our solution from the choice system to the original system. Now we must still show how to construct a kernel of size $O(n^t)$ for an instance of $CSP(\Gamma)$. First we introduce the notion of a choice equivalent constraint.

Definition 3.14. *Let $R \subseteq D^k$ be a k-ary relation defined over domain $D = \{d_1, ..., d_h\}$. Let R_c denote the choice representation of R. For a set C of constraints $R(x_1, ..., x_k)$ defined over the variables $x_1, ..., x_k \in V$, we define the set of* choice equivalent constraints *as the set B defined over the set of variables $V_c = \{x_{i,1}, ..., x_{i,h} \mid x_i \in V\}$ such that for each $R(x_1, ..., x_k) \in C$, the set B contains the constraint $R_c(x_{1,1}, ...x_{1,h}, ..., x_{k,1}, ..., x_{k,h})$.*

In addition, we define what it means for a Boolean assignment f to be a k-choice assignment.

Definition 3.15. *Let f be a Boolean assignment to the variables $x_1, ..., x_n$. Let k be an integer such that $k \cdot d = n$ with d being some integer. f is called a k-choice assignment if for all $i \in [d]$ it holds that*

$$\sum_{j=k\cdot(i-1)+1}^{k\cdot i} f(x_j) = 1$$

With this definition we show that $CSP(\Gamma)$ defined over a finite domain and t-balanced constraint language Γ has a kernel of $O(n^t)$ constraints in Theorem 3.16.

Theorem 3.16. *For any t-balanced constraint language (not necessarily Boolean) Γ, the problem $CSP(\Gamma)$ parameterized by the number of variables n admits a kernel of with $O(n^t)$ constrains which are a subset of the original constraints.*

Proof. Let (I, n) be the parameterized instance of $CSP(\Gamma)$ with $I = (C, V)$ being an instance of $CSP(\Gamma)$, where C is the set of constraints and V is the set of variables, and $n = |V|$. Let $D = \{d_1, ..., d_h\}$ be the domain over which $CSP(\Gamma)$ is defined. Let $R \subseteq D^k$ be some k-ary relation contained in Γ. For the remainder of the proof we will fix this choice of R. Since R was chose arbitrarily, this will imply that the proposed method will work for any choice of R. Let $C_R \subseteq C$ denote the subset of constraints defined over the relation R. Let $U = D^k \setminus R$ be the set of assignments over D that do not satisfy R. Let R_c, U_c be the choice representations of R, U respectively. Let B_R be the set of choice equivalent constraints of C_R defined over the variables V'. We note that R_c, U_c are relations of arity $k' = k \cdot h$.

By applying Lemma 3.13 we know that there exists a set of polynomials where each $p_u \in P$ captures $u \in U_c$ with respect to R_c. These polynomials are hard coded during execution and we let P denote the set of such polynomials for R_c. Let R^* be the set of Boolean assignments $r^* \in \{0,1\}^{k'}$ which satisfy that for all $p_u \in P$ it holds that $p_u(r^*) = 0$. It is easy to verify that $R_c \subseteq R^*$ since all polynomials in P were designed to satisfy $p_u(r) = 0$ for all $r \in R_c$. In addition, there may be some other $r \in \{0,1\}^{k'}$ which are in R^* but not in R_c since they are not choice assignments. We also note that $U_c \cap R^* = \emptyset$ since all polynomials $p_u \in P$ were designed so that for each $u \in U_c$, $p_u(u) \neq 0$. Let A_R denote the set of constraints defined over the set of variables V' and the relation R^* containing the constraints $R^*(x_{1,1}, ..., x_{k,h})$ such that $R_c(x_{1,1}, ..., x_{k,h}) \in B_R$. Now we need to confirm that a choice assignment f satisfies all constraints in A_R if and only if f satisfies all constraints in B_R.

Claim 3.17. *Any h-choice assignment f satisfies all constraints in A_R if and only if it satisfies all constraints in B_R.*

Proof.
$(\Rightarrow)$:
Let f be a choice assignment such that all $R^*(x_{1,1}, ..., x_{k,m}) \in A_R$ satisfy $(f(x_{1,1}), ..., f(x_{k,h})) \in R^*$. By construction we know that f satisfies all $p_u(f(x_{1,1}), ..., (x_{k,h})) = 0$. Now suppose f does not satisfy some $R_c(x_{1,1}, ..., x_{k,h}) \in B_R$, which implies $(f(x_{1,1}), ..., f(x_{k,h})) \notin R_c$. Let $u = (f(x_{1,1}), ..., f(x_{k,h}))$. Since f is an h-choice assignment, it must hold that $u \in U_c$, where U_c is set of all choice representations not contained in R_c. However, this would mean by construction of all $p_u \in P$, there is some p_u such that $p_u(u) \neq 0$. This contradicts that f satisfies all constrains in A_R. Thus, we conclude that if f is an h-choice assignment which satisfies A_R, then it also satisfies B_R.

$(\Leftarrow)$:
Let f be an h-choice assignment such that all constraints $R_c(x_{1,1}, ..., x_{k,m}) \in B_R$ are satisfied. By construction of the polynomials in P, it must hold that this assignment also satisfies $p(f(x_{1,1}), ..., f(x_{k,h})) = 0$ for all polynomials $p \in P$. Now assume that f does not satisfy some constraint $R^*(x_{1,1}, ..., x_{k,h})$ in A_R, implying that $(f(x_{1,1}), ..., f(x_{k,h})) \notin R^*$. By construction of R^*, this means that there is some polynomial $p \in P$ for which $p(f(x_{1,1}), ..., f(x_{k,h})) \neq 0$. However, this is a contradiction as we have already shown that $p(f(x_{1,1}), ..., f(x_{k,h})) = 0$ for all p as f satisfies all constraints in B_R. As such we conclude that this case holds.

Since we have proven all cases, we conclude that the claim holds. ■

We can now apply Theorem 2.15 to A_R and P to obtain a set of constraints $A'_R \subseteq A_R$. Using this we create a new set of constraints B'_R in which we put all constraints $c \in A'_R$ defined over R_c rather than R^*. First off, note that since A_R was constructed to contain the constraints $R^*(x_{1,1}, ..., x_{k,h})$ if B_R contains the constraint $R_c(x_{1,1}, ..., x_{k,h})$ and since $A'_R \subseteq A_R$ and since B'_R contains the constraints $R_c(x_{1,1}, ..., x_{k,h})$ if A'_R contains the constraint $R^*(x_{1,1}, ..., x_{k,h})$, it follows that $B'_R \subseteq B_R$. The following claim remains to be proven:

Claim 3.18. *Any h-choice assignment f satisfies B_R if and only if it satisfies B'_R.*

Proof. Since $B'_R \subseteq B_R$, it is trivial that if f satisfies all constraints in B_R, then it also satisfies all constraints in B'_R. For the other direction let f be an h-choice assignment satisfying all constraints in B'_R. Since all constraints $R_c(x_{1,1}, ..., x_{k,h})$ in B'_R are added if $R^*(x_{1,1}, ..., x_{x_{k,h}})$ is in A'_R, and since we know by construction of R^* that any choice assignment satisfies $R_c(x_{1,1}, ..., x_{k,h})$ if and only if it satisfies $R^*(x_{1,1}, ..., x_{x_{k,h}})$, we conclude that f must satisfy all constraints in A'_R. By Theorem 2.15 it must hold that f satisfies A_R and by claim 3.17 f must satisfy B_R as well. Thus if a choice assignment f satisfies B'_R, then it also satisfies B_R, proving the claim. ■

We can then translate B'_R to a set $C'_R \subseteq C_R$ by simply transforming the constraints on the choice representation R_c to equivalent constraints on R. It is not hard to see that an assignment $f : V \to D$ satisfies all constraints in C_R if and only if it satisfies all constraints in C'_R. Finally, since Γ is t-balanced, all polynomials constructed for B_R have degree at most t by Lemma 3.13. Thus by Theorem 2.15, $|A'| = O(n^t)$ from which it follows by construction that $|C'| = O(n^t)$.

If we do this for all $R \in \Gamma$ and take $C' = \cup_{R\in\Gamma} C'_R$ to be the set of constraints for the instance $I' = (C', V)$ we can construct a new parameterized problem (I', n). It follows that I and I' are equivalent. Since each C'_R is of size $O(n^t)$ and since $|\Gamma|$ is constant, we get that $|C'| = |\Gamma|O(n^t) = O(n^t)$. We have thus proven that it is possible for a parameterized instance (I, n) of $CSP(\Gamma)$, with $n = |V|$ as the parameter and the language Γ being t-balanced, to find a kernel (I', n) with $O(n^t)$ constraints. □

We have thus shown how to construct a kernel with $O(n^t)$ constraints for a parameterized instance of $CSP(\Gamma)$ over a finite domain, defined over a t-balanced language Γ with the parameter being the number of variables. Using computer aided calculations we were able to use these results to find kernels for some interesting problems. The main problem of interest which we considered was d-Uniform Hypergraph q-Coloring where the question is if for a d-Uniform Hypergraph G there exist a coloring of the vertices of G such that each contains two distinctly colored vertices. Note that in a d-Uniform Hypergraph each edge contains exactly d vertices. During our tests it turned out that it was not possible to obtain a nontrivial kernel for all variants of this problem on which we performed our tests. However, if we ask whether a coloring exists such that each edge contains at least three distinctly colored vertices, we do find there exists a nontrivial kernel. When looking at the problem 4-Uniform Hypergraph 4-Coloring it turned out that when each edge must contain at least three distinctly colored vertices, that the problem admits a kernel of $\mathcal{O}(n^3)$ constraints, which are edges in this case, which is a nontrivial kernel. Even more so, the problem 4-Uniform Hypergraph 3-Coloring when each edge must contain at least three distinctly colored vertices admits a kernel of size $\mathcal{O}(n^2)$ constraints which is even less. This shows that the provided method can be utilized to find nontrivial kernels for $CSP(\Gamma)$ defined over finite domains that are not Boolean.

3.3 Difficulties of constructing single capturing polynomials

While in practice it has been observed that it is possible to generate a single polynomial which captures each relation of some $CSP(\Gamma)$ [10, 14], directly extending Lemma 2.18 in order to create polynomials which capture multiple unsatisfying assignments at once with respect to some relation R can be shown to be infeasible. More specifically, it is not possible to use Lemma 2.18 and extend it so that it can be applied to multiple row vectors $c_1, c_2, ..., c_m$ which are not in $Span_q(A)$ of some matrix A, where q is the same for all $c_1, c_2, ..., c_m$. This will be shown by constructing an example on which we cannot simply apply the methodology provided by Lemma 2.18 in order to find a solution for the system. First we construct a matrix A and a matrix of row vectors C which are not in $Span_q(A)$. In this example we will use $q = 2$ and thus also for all row vectors in C, $c = 1$ as this is the only value larger than zero in the ring $\mathbb{Z}/2\mathbb{Z}$.

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \quad C = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

It can be verified that for each of the row vectors in C it would be possible to find $q = 2$ by applying Lemma 2.17. In addition, for each row in C we can apply Lemma 2.18 to find both a solution x and constant c for which there is a solution to the system $RA''x = b$ where $b = (0, 0, 0, 1)$, $A' = \left(\frac{A}{c_i}\right)$, which means that A' is the matrix A with row vector c_i appended at the end, and c_i is a row vector of C. If we now try to construct a single polynomial for all $c_i \in C$ applying the same process, it becomes clear that this becomes impossible. First we construct the following system of equations:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} x \equiv_2 \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} \tag{13}$$

It is easy to verify that this system of equations has no solution in the unknown x. This follows from the fact that if we satisfy the fourth and fifth equation, which only happens when the solution x satisfies $x[4] = x[5] = 1$, then the sixth equation cannot be satisfied as this will imply that $a'_6 x \equiv_2 a'_6[4] + a'_6[5] \equiv_2 1 + 1 \equiv_2 0$ and this will thus not capture the sixth assignment. In addition one can verify that the smith normal form decomposition of A in this example has the property $S = A$ since A is a diagonal matrix. Note that via this construction, we can only break the schema provided by Lemma 2.18 because we have at least 3 row vectors which we try to append to A. Finally we note that A is a balanced relation, thus showing that there is no way to directly extend Lemma 2.18.

3.4 Importance of rings when constructing single capturing polynomials

While Section 3.3 shows that it is not always possible to construct a single polynomial that captures some relation R while it is possible to construct polynomials that capture each unsatisfying assignment with respect to R defined over the same ring $\mathbb{Z}/q\mathbb{Z}$, one may wonder whether this means that there does not exist such a polynomial at all. As it turns out, this is not the case. While it is not possible to construct a single polynomial over the ring $\mathbb{Z}/q\mathbb{Z}$ that captures R, it may be possible to construct such a polynomial over a different ring. To illustrate this we will use the relations shown in Equation (14) which show the matrix form of some 3-ary balanced relation R and the unsatisfying assignments $U = \{0, 1\}^3 \setminus R$.

$$R = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \quad U = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{14}$$

It is not hard for one to confirm that for each $u \in U$ it is possible to construct a polynomial p_u over the ring $\mathbb{Z}/2\mathbb{Z}$ which captures u with respect to R. This can actually be done with 2 polynomials, i.e. $p_1(x_1, x_2, x_3) \equiv_2 x_1 + x_2$ and $p_2(x_1, x_2, x_3) \equiv_2 x_3$. However, for this choice of R there is no single degree-1 polynomial defined over the ring $\mathbb{Z}/2\mathbb{Z}$ which captures all $u \in U$ with respect to R. We can confirm this as follows, suppose for a contradiction that p is the polynomial which captures all $u \in U$ with respect to R. Let p be given by $p(x_1, x_2, x_3) \equiv_2 \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3$. We know that in order to ensure that p evaluates to 0 on $(0, 0, 0) \in R$ it must hold that $\alpha_0 = 0$. In addition, to ensure that p evaluates to 0 on $(1, 1, 0) \in R$, p must satisfy $\alpha_1 = \alpha_2$ since $q = 2$. Since we are computing over the ring $\mathbb{Z}/2\mathbb{Z}$, there are thus only 4 valid assignments of these coefficients that are given in Table 1.

α_1	α_2	α_3
0	0	0
1	1	0
0	0	1
1	1	1

Table 1: valid assignments for the coefficients of the polynomial p

Trivially, the first assignment will return $p(x) = 0$ for all possible $x \in \{0, 1\}^3$ and thus also for all values of U, which in turn means that p does not capture any unsatisfying assignment with respect to R. For all subsequent assignments of $\alpha_1, \alpha_2, \alpha_3$ we can find some $u \in U$ which is not captured by p. For the second assignment $(1, 1, 1)$ is not captured, for the third one $(0, 1, 0)$ is not captured and for the last one $(0, 1, 1)$ is not captured. So it turns out that over the ring $\mathbb{Z}/2\mathbb{Z}$ it is not possible to capture all $u \in U$ with our choice of R. This does not mean there does not exist any $q \in \mathbb{Z}$ which allows for the construction of a single polynomial which captures all $u \in U$ with respect to R. To illustrate pick $q = 4$ and let $p(x_1, x_2, x_3) \equiv_4 2x_1 + 2x_2 + x_3$. It is easy to confirm that this polynomial does capture all $u \in U$ with respect to R. Thus while it may not be possible to find a single polynomial to capture a relation using already known values for q, it may still be possible to use a different value for q and still get a single polynomial which captures all $u \in U$ with respect to R.

3.5 Reducing a set of capturing polynomials

Although these examples do illustrate that it is not trivial to construct a single polynomial which captures some relation R, it is possible to create a single polynomial which captures at least 2 unsatisfying assignments with respect to R. More specifically, for two arbitrary unsatisfying assignments $u, u' \in U$, where $U = D^k \setminus R$, it is possible to find a single polynomial which captures both u, u' with respect to R. This is shown in Lemma 3.19

Lemma 3.19. *Let $R \subseteq D^k$ be a k-ary relation over the domain D and let $u, u' \in D^k \setminus R$ be two unsatisfying assignments with respect to R. If there exist two polynomials $p_u, p_{u'}$ defined over their respective rings $\mathbb{Z}/q\mathbb{Z}$, $\mathbb{Z}/q'\mathbb{Z}$ which capture u, u' respectively with respect to R, then it is possible to construct a polynomial p which captures both u, u' with respect to R.*

Proof. Let $p_u, p_{u'}$ be the polynomials which capture the unsatisfying assignments $u, u' \in U$ respectively with respect to R. Let p_u be defined over the ring $\mathbb{Z}/q\mathbb{Z}$ and $p_{u'}$ over the ring $\mathbb{Z}/q'\mathbb{Z}$. If we have $p_u(u') \not\equiv_q 0$ or $p_{u'}(u) \not\equiv_{q'} 0$, it is easy to see that the polynomial which satisfies this property already captures both u, u'. Thus if either of these hold, simply pick $p = p_u$ or $p = p_{u'}$ depending on which case is satisfied.

Suppose neither of these cases hold and we have $p_u(u') \equiv_q 0$ and $p_{u'}(u) \equiv_{q'} 0$. This allows p to be constructed as follows. First we create a new $\mathbb{Z}/q''\mathbb{Z}$. We define $q'' = q \cdot q'$. Then define p to be

$$p(x) = p_u(x) \cdot q' + p_{u'}(x) \cdot q$$

To confirm that this polynomial captures u, u' all that needs to be done is to insert them and check the results. First, $p(u)$ gives the following:

$$\begin{aligned} p(u) &\equiv_{q''} p_u(u) \cdot q' + p_{u'}(u) \cdot q \\ &\equiv_{q''} (q \cdot d + c) \cdot q' + (q' \cdot d') \cdot q \\ &\equiv_{q''} c \cdot q' + d \cdot q'' + d' \cdot q'' \\ &\equiv_{q''} c \cdot q' \end{aligned}$$

Now we need to confirm that $c \cdot q' \not\equiv_{q''} 0$. First, we know that c is the value of $p_u(u)$ over the ring $\mathbb{Z}/q\mathbb{Z}$. Now since we are left with $c \cdot q'$ we note that $0 < c < q$ by definition of p_u. Thus $c \cdot q'$ is no multiple of q'' as $q'' = q \cdot q'$. This thus allows us to conclude that $c \cdot q' \not\equiv_{q''} 0$ and thus p captures u with respect to R. This can similarly be reasoned for u'.

Finally, we need to confirm for all $r \in R$, $p(r) \equiv_{q''} 0$. Let $r \in R$ be some arbitrary satisfying assignment. Then we get the following:

$$\begin{aligned} p(r) &\equiv_{q''} p_u(r) \cdot q' + p_{u'}(r) \cdot q \\ &\equiv_{q''} (q \cdot d) \cdot q' + (q' \cdot d') \cdot q \\ &\equiv_{q''} d \cdot q'' + d' \cdot q'' \\ &\equiv_{q''} 0 \end{aligned}$$

This equality holds since $p_u(r) = d \cdot q$ for some $d \in \mathbb{Z}$ and similarly $p_{u'}(r) = d' \cdot q'$ for some $d \in \mathbb{Z}$ if these polynomials were used over the integers. By this fact we conclude that the above equation is correct.

Now since u, u' are captured with respect to R and since for any arbitrary $r \in R$ it holds that $p(r) \equiv_{q''} 0$, we conclude that p has the desired property. □

So it is always possible to construct a single polynomial which captures two unsatisfying assignments u, u' with respect to a relation R. We also note that this newly generated polynomial p has the same degree as the highest degree of the original polynomials since p was created by addition of the original polynomials. Since we know for a t-balanced relation R that degree t polynomials exist which capture the unsatisfying constraints with respect to R by Lemmas 3.6 and 3.13, it is possible to reduce the amount of degree t polynomials used to capture the unsatisfying assignments of R.

We also note that this approach may not work when trying to combine 3 polynomials. It is possible that two polynomials cancel each other out, which was not possible when combining 2 polynomials. In the proof of Lemma 3.19 it is important that either 1 polynomial of $p_u, p_{u'}$ captures both u, u' or both polynomials do not capture the other unsatisfying assignment, i.e. $p_u(u') \equiv_q 0$ and $p_{u'}(u) \equiv_{q'} 0$. Now with 3 unsatisfying assignments it could be that there is a new case where each polynomial captures 2 unsatisfying assignments except for 1. More concretely, for u, u', u'' and respective polynomials $p_u, p_{u'}, p_{u''}$ it could be that the following equalities are all satisfied:

$$\begin{gathered}p_u(u) \not\equiv_q 0\\ p_u(u') \not\equiv_q 0\\ p_u(u'') \equiv_q 0\\[1em] p_{u'}(u) \equiv_{q'} 0\\ p_{u'}(u') \not\equiv_{q'} 0\\ p_{u'}(u'') \not\equiv_{q'} 0\\[1em] p_{u''}(u) \not\equiv_{q''} 0\\ p_{u''}(u') \equiv_{q''} 0\\ p_{u''}(u'') \not\equiv_{q''} 0\end{gathered}$$

This gives a structure where each polynomial does not capture exactly 1 unsatisfying assignment from u, u', u''. Because of this structure, if we were to simply add these polynomials over a new ring, it could be that for some unsatisfying assignment the results of those polynomials cancel each other and give 0 as a result, thus not capturing said unsatisfying assignment.

For example, let $p(x) = p_u(x) \cdot q' \cdot q'' + p_{u'}(x) \cdot q \cdot q'' + p_{u''}(x) \cdot q \cdot q'$ be defined over the ring $\mathbb{Z}/q'''\mathbb{Z}$ with $q''' = q \cdot q' \cdot q''$. If we were to insert u into this polynomial we would get the following result:

$$\begin{aligned}p(u) &\equiv_{q'''} \quad p_u(x) \cdot q' \cdot q'' + p_{u'}(x) \cdot q \cdot q'' + p_{u'''}(x) \cdot q \cdot q'\\ &\equiv_{q'''} \quad (d + c \cdot q) \cdot q' \cdot q'' + c' \cdot q \cdot q' \cdot q'' + (d'' + c'' \cdot q'') \cdot q \cdot q'\\ &\equiv_{q'''} \quad d \cdot q' \cdot q'' + d'' \cdot q \cdot q'\end{aligned}$$

We cannot guarantee that $d \cdot q' \cdot q'' + d'' \cdot q \cdot q' \not\equiv_{q'''} 0$ as it could be that they sum up to 0 modulo q'''. If this could be guaranteed, however, this approach would be able to construct a single polynomial for any relation R. Otherwise if one could guarantee that this case never occurs in practice, the same result could be achieved. This is no trivial task and requires further investigation.

4 Lower Bound for 3-Uniform Hypergraph 3-Coloring

First off, the problems which will be used to prove our lower bound are defined as follows:

3-Uniform Hypergraph 3-Coloring
Input: A graph G with $V(G)$ the set of vertices and $E(G) \subseteq \binom{V(G)}{3}$ the set of 3-hyperedges.
Question: Is there a 3-coloring $\gamma : V(G) \to [3]$ where for each hyperedge $e \in E(G)$ it holds that there exists two distinct vertices $v, v' \in e$ such that $\gamma(v) \neq \gamma(v')$?

A more intuitive way to interpret this problem is to think of a standard graph, where each edge consists of 3 vertices instead of 2 and such an edge is properly colored if and only if it contains at least 2 distinctly colored vertices. Next the other problem which we will use to construct a lower bound is defined as follows:

2-3-Coloring with triangle split decomposition
Input: A graph G with a partition of its vertex set into $V \cup W$ such that $G[V]$ is an edgeless graph and $G[W]$ is a disjoint union of triangles.
Question: Is there a 3-coloring $\gamma : V(G) \to \{1, 2, 3\}$ of G, such that $\gamma(v) \in [2]$ for all $v \in V$?

For 2-3-Coloring with Triangle Split Decomposition the following Lemma was proven:

Lemma 4.1 ([18], Lemma 3). 2-3-Coloring with Triangle Split Decomposition *is NP-complete.*

Using these definitions, t input instances of 2-3-Coloring with Triangle Split Decomposition will be used to construct an instance of 3-Uniform Hypergraph 3-Coloring. To do so, a couple of gadgets must be introduced. The first of which will be referred to as the *restricting-gadget*. Before we introduce this gadget and prove its correctness, we first introduce the notion of a 3-uniform hypergraph clique.

Definition 4.2. *A k-hypergraph d-clique of a k-uniform hypergraph G is a set of vertices $C \subseteq V(G)$ of size d such that for any subset of k vertices of C there is a hyperedge in $E(G)$.*

For brevity a k-hypergraph d-clique will sometimes be refered to as a d-clique from which it will be clear from the context what the value for k is. Using this definition we will show that for 3-Uniform Hypergraph Coloring that a 3-hypergraph 6-clique can only be properly 3-colored if all 3 colors are used exactly twice.

Lemma 4.3. *A 3-hypergraph 6-clique C can be colored using 3 colors if and only if each color is used exactly twice.*

Proof. Assume the inverse and let $\gamma : V(C) \to [3]$ be its related coloring. It is easy to confirm that some color must have been used at least three times, since if not every color was used exactly twice to color C or some vertices were left uncolored, which is not allowed. Let the vertices $v_1, v_2, v_3 \in C$ satisfy $\gamma(v_1) = \gamma(v_2) = \gamma(v_3)$. However, since in a 3-hypergraph 6-clique it holds that any set of 3 vertices, there is also an edge $\{v_1, v_2, v_3\}$. However, this edge is then improperly colored as all vertices share the same color which is a contradiction. From this contradiction the claim follows. □

For simplicity, we will say that a vertex v is connected to a d-clique C if $\{v\} \cup C$ forms a $d+1$-clique. Similarly, if we have two sets of vertices C, D and we say that we connect C and D then edges are added such that $C \cup D$ forms a $(|C| + |D|)$-clique. The restricting gadget is given by the 3-uniform hypergraph in Figure 2. Here the groups of 5 vertices represent 5-cliques and the edges from 1 vertex to a clique means that we connect that vertex to that clique. For example, $\{c_{1,3}\} \cup C_2$ represent 6-clique. It now remains to be shown that this structure actually behaves as our restricting gadget.

Lemma 4.4. *For the 3-uniform hypergraph G shown in Figure 2 the following holds: for any proper 3-coloring $\gamma : V(G) \to [3]$ of G each clique C_i satisfies that $c_{i,5}$ has a unique color within C_i and $\gamma(c_{i,5}) \neq \gamma(c_{j,5})$ for all $i \neq j$.*

Proof. First we observe that each 5-clique C_i has 4 vertices which form 6-cliques with the other 5-cliques. Let for a clique C_i the vertices $c_{i,k}, c_{i,k'}$, where $k, k' \in [4]$ and $k \neq k'$, be the vertices

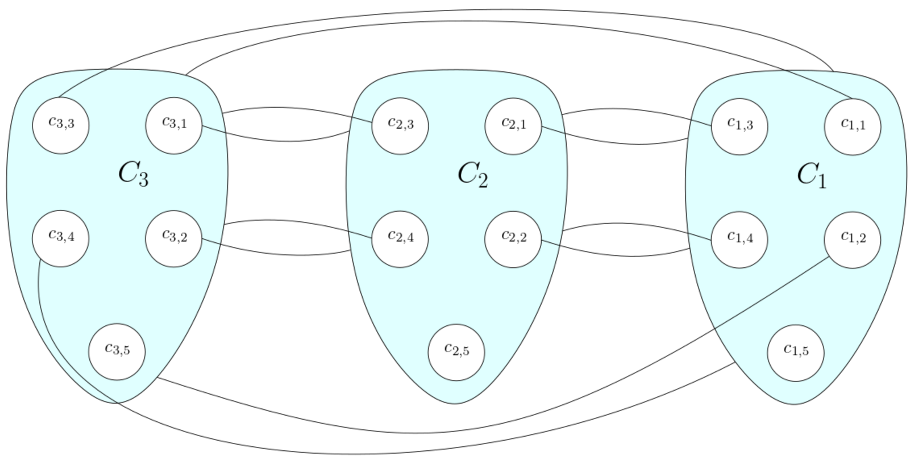


Figure 2: illustration of a degree-1 restricting gadget for 3-UNIFORM HYPERGRAPH COLORING

which are connected to the clique C_j, with $j \in [3]$ and $j \neq i$. Since these vertices form a 6-clique with C_j, Lemma 4.3 states that these cliques must use each color exactly twice. This allows us to infer that $c_{i,k}, c_{i,k'}$ must have the same color, since if we assume $\gamma(c_{i,k}) \neq \gamma(c_{j,k'})$ with γ being some proper 3-coloring, then if either $\{c_{i,k}\} \cup C_j$ or $\{c_{i,k'}\} \cup C_j$ is properly colored, the other must have some hyperedge that is improperly colored. It is easy to confirm that if, without loss of generality, $\{c_{i,k}\} \cup C_j$ would be properly colored, then the colors $[3] \setminus \gamma(c_{i,k})$ are used twice in C_j. Since $c_{i,k'}$ is colored using one of these colors and since $\{c_{i,k'}\} \cup C_j$ is a 6-clique, there must be an edge that is colored using only 1 color, thus contradicting that γ is a proper 3-coloring.

Now we note that the color assigned to $c_{i,k}, c_{i,k'}$ in any proper coloring only occurs once in C_j. For each clique C_i we denote this color which only gets assigned to one vertex in C_i by c_i. It is easy to see by the previous argument, since all $c_{i,5}$ do not form 6-cliques, that these vertices must be assigned c_i, since all other vertices in C_i form pairs which are assigned the same color. Now it remains to be checked that for any two vertices $c_{i,5}, c_{j,5}$ for $i \neq j$ it holds that $\gamma(c_{i,5}) \neq \gamma(c_{j,5})$ for some proper coloring γ. Assume that these vertices are given the same color under γ. We know that in C_j there are two vertices $c_{j,k}, c_{j,k'}$ which are given the color c_i which was established previously. In addition, $\gamma(c_{i,5}) = c_i$ by definition. However, since we assumed $\gamma(c_{i,5}) = \gamma(c_{j,5}) = c_i$, this means that there are 3 vertices in C_j which have been assigned the same color. Since C_j is a 5-clique, there is an edge which is not properly colored, contradicting the assumption that γ is a proper 3-coloring. Thus we conclude that for any pair of vertices $c_{i,5}, c_{j,5}$, where $i \neq j$, any proper 3-coloring γ satisfies $\gamma(c_{i,5}) \neq \gamma(c_{j,5})$ thus proving the Lemma. □

From now on we will refer to the graph shown in Figure 2 as a *degree-1 restricting gadget*. Degree-1 comes from the fact that whenever an arbitrary vertex v would be connected to any 5-clique C_i which is part of the gadget, then v only has one degree of freedom in terms of its coloring, namely only c_i. From here on forth we will assume that without loss of generality any 5-clique C_i in a degree-1 restricting gadget is colored only once by color i.

Using the degree-1 restricting gadget we can construct a degree-2 restricting gadget as follows.

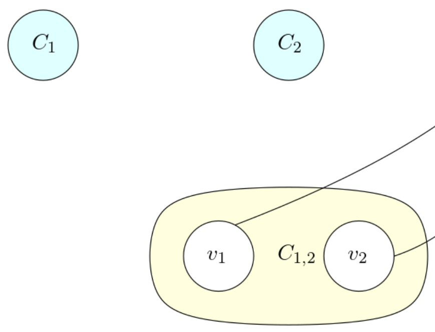


Figure 3: example of a degree-2 restricting gadget $C_{1,2}$ for 3-UNIFORM HYPERGRAPH COLORING

Let $1, 2$ be the colors which we want a certain vertex to be restricted to. First create a set $C_{1,2}$ with vertices v_1, v_2. Connect these vertices individually to the clique C_3 of the degree-1 restricting gadget. Now we know that v_1, v_2 are colored using the color 3. If we now take some arbitrary vertex v which is not part of any of the gadgets and connect it to $C_{1,2}$ the only color which cannot be assigned to v is 3 since the edge $\{v_1, v_2, v\}$ would be improperly colored. Thus v has 2 degrees of coloring freedom, namely 1,2. An example of a degree-2 restricting gadget can be seen in Figure 3.

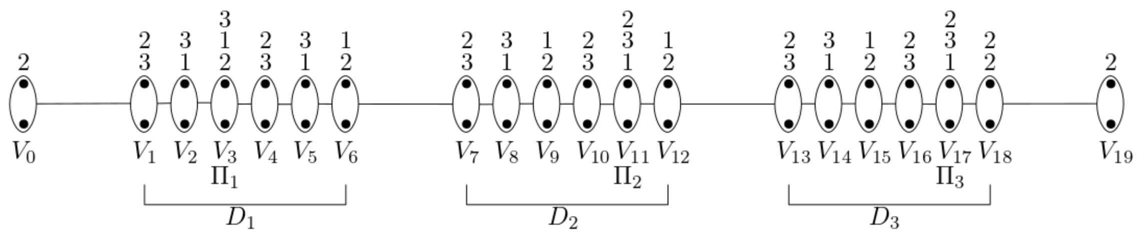
Now in addition we will need another gadget which will allow us to construct the logical-or of our t instances of 2-3-COLORING WITH TRIANGLE SPLIT DECOMPOSITION. This gadget will use a similar construction to that of the blocking-gadget($\mathbf{c}$) as used by Jansen and Pieterse which was first presented by Jaffke and Jansen [9, 19]. Todo so we first introduce the problem 3-UNIFORM HYPERGRAPH 3-LIST-COLORING.

3-UNIFORM HYPERGRAPH 3-LIST-COLORING
Input: A graph G with $V(G)$ being a set of vertices, $E(G) \subseteq \binom{V(G)}{3}$ of 3-hyperedges and for each vertex $v \in V(G)$ a set $l(v) \subseteq [3]$.
Question: Is there a 3-coloring $\gamma : V(G) \to [3]$ where for each vertex $v \in V(G)$ it holds that $\gamma(v) \in l(v)$ and for each hyperedge $e \in E(G)$ there exists two distinct vertices $v, v' \in e$ such that $\gamma(v) \neq \gamma(v')$?

Note that if for all $v \in V$ $l(v) = [3]$, then 3-UNIFORM HYPERGRAPH 3-LIST-COLORING is equivalent to 3-UNIFORM HYPERGRAPH 3-COLORING. We will first show how to construct such a gadget for 3-UNIFORM HYPERGRAPH 3-LIST-COLORING and then extend this gadget to 3-UNIFORM HYPERGRAPH 3-COLORING.

Lemma 4.5. *There is a polynomial-time algorithm that, given* $\mathbf{c} = (c_1, ..., c_m) \in [3]^m$*, outputs a* 3-UNIFORM HYPERGRAPH 3-LIST-COLORING *instance* $G = (V, E)$ *where* V *is a set of vertices of size* $O(m)$ *containing distinguished vertices* $(\pi_{1,1}, \pi_{1,2}, ..., \pi_{m,1}, \pi_{m,2})$*, such that the following holds. For each* $\mathbf{d} = (d_1, ..., d_m) \in [3]^m$ *there is a proper 3-uniform-hypergraph 3-list-coloring* γ *of* G *in which* $\gamma(\pi_{i,j}) \neq d_i$ *for all* $i \in [m], j \in [2]$*, if and only if* $(c_1, ..., c_m) \neq (d_1, ..., d_m)$ *and for each pair of distinguished vertices* $\pi_{i,1}, \pi_{i,2}$ *it holds that* $\gamma(\pi_{i,1}) = \gamma(\pi_{i,2})$*.*

Proof. First we create the consecutive vertices $v_{0,1}, v_{0,2}, ..., v_{6m+1,1}, v_{6m+1,2}$ to add to our graph. We will denote the pairs $v_{i,1}, v_{i,2}$ by the set V_i for all $i \in [6m + 1] \cup \{0\}$. In addition, we want that



2
2 3
3 1
3 1 2
2 3
3 1
1 2
V_0 V_1 V_2 V_3 V_4 V_5 V_6
Π_1
D_1
2 3
3 1
1 2
2 3
2 3 1
1 2
V_7 V_8 V_9 V_{10} V_{11} V_{12}
Π_2
D_2
2 3
3 1
1 2
2 3
2 3 1
2 2
V_{13} V_{14} V_{15} V_{16} V_{17} V_{18}
Π_3
D_3
2
V_{19}

Figure 4: Example showcasing the blocking gadget for $\mathbf{c} = (3, 2, 2)$

for any $i \in [6m]$ that all vertices in V_i have the same color. To enforce this we create a 5-clique C_i for each $i \in [6m + 1] \cup \{0\}$ and connect all $v \in V_i$ to C_i. This in turn creates 2 6-cliques which enforces all $v \in V_i$ to have the same colors as a consequence of Lemma 4.3. We then restrict all $v \in V_0 \cup V_{6m+1}$ to only the color 2. We will refer to vertices in V_0 as the sources and the vertices in V_{6m+1} as the sinks. Let $l(v_{i,j})$ denote the set of colors which vertex $v_{i,j}$ is allowed to be assigned. More general, since all vertices in some set V_i are required to get the same color, we define $l(V_i)$ to denote the set of colors which all vertices $v \in V_i$ are allowed to take. Now all sets V_i for $i \in [6m]$ get assigned the following lists $l(V_i) = [3] \setminus \{((i-1) \mod 3) + 1\}$. Thus each vertex that is not a sink is allowed exactly 2 colors. We then add hyperedges such that V_{i-1}, V_i are connected, which we recall means that $V_{i-1} \cup V_i$ forms a 4-clique, for all $i \in [6m+1]$. Observe that this construction as it is does not allow for a proper list-3-uniform-hypergraph-3-coloring. This can be seen since the vertices $v \in V_0$ are forced to be colored using 2 and they form a 4-clique with vertices $v' \in V_1$, which means that all v' are forced to be colored with color 3 since we only allowed colors $[3] \setminus \{((i-1) \mod 3) + 1\} = \{2, 3\}$. In general we find that each vertex $v_{i,j}$ is colored with $((i + 1) \mod 3) + 1$ by this construction. This then results in $v_{6m,j}$ being colored with $(6m + 1 \mod 3) + 1 = 2$ and also $v_{m+1,j}$ to be colored by 2 for all $j \in [2]$. This thus results in a clique of size 4 where all vertices share the same color which means that all edges in this clique are improperly colored. This fact will be used to construct the gadget with the exact requirement stated in the Lemma.

We now identify m groups D_i for $k \in [m]$ where D_k is defined by $D_k = \bigcup_{i=6(k-1)+1}^{6k} V_i$. We then call all vertices $v_{i,j} \in D_k$ such that $i \notin \{6k, 6(k-1)+1\}$ the interior vertices of D_k. We will use these interior vertices to then identify our distinguished vertices $\pi_{1,1}, \pi_{1,2}, ..., \pi_{m,1}, \pi_{m,2}$. We will denote each pair $\{\pi_{i',1}, \pi_{i',2}\}$ by the set $\Pi_{i'}$. All $\pi_{i',j}$ are picked as follows, for each $c_{i'}$ in $(c_1, ..., c_m)$ we pick a set V_i such that $V_i \subset D_{i'}$ such that $c_{i'} \notin l(V_i)$. We then add $c_{i'}$ to the list $l(V_i)$ and this set will thus function as the set $\Pi_{i'}$. An example can be seen in Figure 4.

We will now prove that this construction has the desired property. First, assume that for some $\mathbf{c} = (c_1, ..., c_m)$ we pick $\mathbf{d} = (d_1, ..., d_m)$ such that $\mathbf{c} = \mathbf{d}$. In this case, we observe that since $\gamma(\pi_{i',j}) \neq d_{i'} = c_{i'}$ we cannot use the new colors added to the vertex $v_{i,j}$. This thus means that we can only use the colors which were already on $l(v_{i,j})$ which means that the construction cannot be properly colored as was established earlier.

Thus, we consider the case where $\mathbf{c} \neq \mathbf{d}$. For all sets $\Pi_{i'}$ for which $c_{i'} \neq d_{i'}$ we assign to its vertices $c_{i'}$. We then start from the source and start coloring in the direction of the sink. This is continued until the first pair of distinguished vertices $V_i = \Pi_{i'}$ is reached, where $\gamma(v) = c_{i'}$ for all $v \in V_i$. We then note that this does not impose a problem as the vertices in V_{i-1} must have been colored with $(((i-1)+1) \mod 3)+1$ and the vertices in V_i are colored using $c_{i'}$ which by construction is equal to $((i-1) \mod 3)+1$ and thus they are colored differently. We can also color the vertices in V_{i+1} with $((i+1) \mod 3)+1$ as this color is still part of $v_{i+1,j}$ and different from the color assigned to the vertices in V_i. Now we color the remaining vertices as follows:

1. If i' was the last index of the last $\Pi_{i'}$ such that $c_{i'} \neq d_{i'}$ we color all vertices in $V_{i''}$ for $i'' \in [6m] \setminus [i+2]$ with $(i'' \mod 3)+1$ as this allows the sink vertices to be properly colored.
2. Otherwise we color V_{i+2} with the color $((i+3) \mod 3)+1 = (i \mod 3)+1$ which is part of $l(V_{i+2})$ and does not create conflict since by construction the vertices $v_{i+2,j}$ are not interior vertices. We then propagate this coloring until we reach the next pair of distinguished vertices.

By repeating this scheme the graph will eventually be properly list-colored. □

Thus we see that for List-3-Uniform Hypergraph 3-Coloring we can create such a gadget. Now lifting this gadget to 3-Uniform Hypergraph 3-Coloring is done by constructing degree-1 and 2 restricting gadgets and connecting them such that the proper 3-list coloring is maintained. We also note that a simpler and more straightforward way of interpreting Lemma 4.5 is that there must be some pair Π_i which gets the color c_i assigned from $\mathbf{c} = (c_1, ..., c_m)$. With this we have all the tools at our disposal to prove the kernel lower-bound for 3-Uniform Hypergraph 3-Coloring. To do so we will first prove a kernel lowerbound on 3-Uniform Hypergraph 3-List-Coloring by providing a degree-3 cross-composition and then show how to extend such a construction to 3-Uniform Hypergraph 3-Coloring.

Theorem 4.6. 3-Uniform Hypergraph 3-List-Coloring *parameterized by the number of vertices n does not have a generalized kernel of size $O(n^{3-\varepsilon})$ for any $\varepsilon > 0$, unless $NP \subseteq coNP/poly$.*

Proof. In order to prove this statement we will give a degree-3 cross-composition from 2-3-Coloring with Triangle Split Decomposition to 3-Uniform Hypergraph 3-List-Coloring. To do so we must first establish some polynomial equivalence relation $\mathcal{R}$ between our instances of 2-3-Coloring with triangle split decomposition. An instance (G, V, W) of 2-3-Coloring with Triangle Split Decomposition is defined such that G denotes the graph, V is the set of independent vertices and W the set of vertex disjoint triangles. We refer to the vertex set of the graph as $V(G)$ and the edge set as $E(G)$. We note that in this case $V(G) = V \cup W$. We call two such instances equivalent if their respective sets V and W have the same size. It is easy to see that this is a polynomial equivalence relation.

Let us be given t instances of 2-3-Coloring with Triangle Split Decomposition. We want ensure that we always have a cubic number of instances to work with. This can always be ensured by taking one of our inputs and duplicating it until we get a cubic number of 2-3-Coloring with Triangle Split Decomposition instances. This at most increases the amount of instances by a factor 8 and it does not change the value of the OR. So now suppose we are given t instances 2-3-Coloring with triangle split decomposition such that $t' = \sqrt[3]{t}$ is an integer. We denote our given instances by $X_{i,j,k} = (G_{i,j,k}, V_{i,j,k}, W_{i,j,k})$ for $i, j, k \in [t']$. We also assume that these instances are equivalent under $\mathcal{R}$ and say that $|V_{i,j,k}| = m$ and that $W_{i,j,k}$ contains n vertex disjoint triangles for all instances $X_{i,j,k}$. The vertices in $V_{i,j,k}$ are labeled $v_1, ..., v_m$ and the vertices in $W_{i,j,k}$ are labeled $w_1, ..., w_{3n}$ such that any group of 3 vertices $w_{3\ell-2}, w_{3\ell-1}, w_{3\ell}$ for $\ell \in [n]$ forms a triangle. Using this we now proceed to the construction of the graph G' which is an instance of 3-Uniform Hypergraph 3-List-Coloring.

1. Initialize the graph G' to contain t' sets that contain $m \cdot 3n$ vertices called S_i for $i \in [t']$. We label each vertex $s^i_{\alpha,\ell}$ with $\alpha \in [m], \ell \in [3n], i \in [t']$. Here a set of vertices $s^i_{\alpha,\ell}$ which share the same i, α represents a vertex of the sets of independent vertices V.

2. For each $j \in [t']$, add to G' the vertex set $T_j = \{t^j_1, ..., t^j_{3n}\}$ of size $3n$. These vertices are labeled such that $t^j_{3\ell-2}, t^j_{3\ell-1}, t^j_{3\ell}$ for $\ell \in [n]$ correspond to a triangle $w_{3\ell-2}, w_{3\ell-1}, w_{3\ell}$ in the original graph of some input instance $X_{i,j,k}$. They are not connected so that when they are not part of the 2-3-colorable input instance they can safely be colored using the color 3.

3. For each $k \in [t']$, add to G' the vertex sets $U_k = \{u^k_1, u^k_2, u^k_3\}$ of size 3. These vertices do not correspond to any vertices in the original input instances and are used to ensure the 2-3-colorability of the original input instance.

4. Add edges $\{s^i_{\alpha,\ell}, t^j_\ell, u^k_\beta\}$ for $i, j, k \in [t'], \alpha \in [m], \ell \in [3n], \beta \in [3]$ if there is an edge (v_α, w_ℓ) in $X_{i,j,k}$. This way if one were to take the vertex set $V(G'[S_i \cup T_j])$ for any $i, j \in [t']$ and construct a normal graph G, such that all $s^i_{\alpha,1}, ..., s^i_{\alpha,3n}$ would be merged into a single vertex v_α for all $\alpha \in [m]$, all vertices t^j_ℓ are relabeled to w_ℓ for all $\ell \in [3n]$, add edges between all vertices of the triplet $w_{3\ell'-2}, w_{3\ell'-1}, w_{3\ell'}$ for $\ell' \in [n]$ and introduce edges $\{v_\alpha, w_\ell\}$ if the hyperedges $\{s^i_{\alpha,\ell}, t^j_\ell, v^k_\beta\}$ are in G' for some $k \in [t']$ all $\beta \in [3]$, the derived graph would be equivalent to the graph of the instance $X_{i,j,k}$.

5. Add vertex sets $A = \{a_1, ..., a_{t'}\}, B = \{b_1, ..., b_{t'}\}$ and $D = \{d_1, ..., d_{t'}\}$. We will use these vertex sets to select $i, j, k \in [t']$ such that $X_{i,j,k}$ is 2-3-colorable.

6. For all vertices in $s^i_{\alpha,\ell} \in S_i$ for all $i \in [t']$ we set $l(s^i_{\alpha,\ell}) = [2]$. We do the same for all vertices in the sets A, B and D.

7. For all vertices in $t^j_\ell \in T_j$ assign $l(t^j_\ell) = [3]$ for all $j \in [t']$. Do the same for all vertices in the sets U_k for all $k \in [t']$.

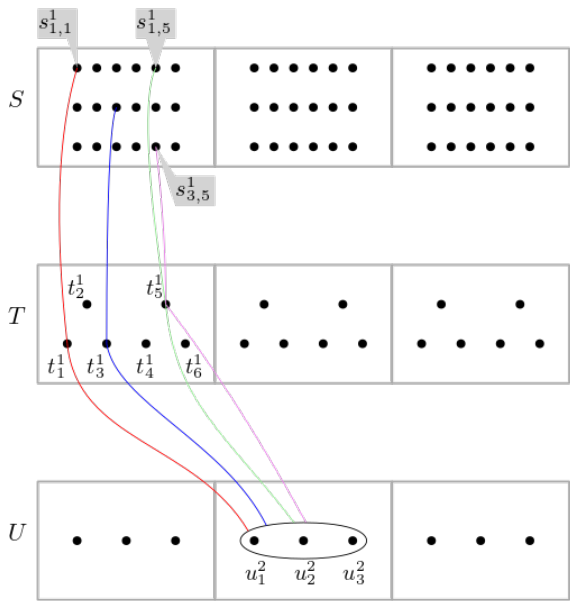
8. For the vertex set A, we add a blocking-gadget with the vector $\mathbf{c} = (c_1, ..., c_{t'})$ where $c_i = 2$ for all $i \in [t']$. We then add edges $(a_i, \pi_{i,1}, \pi_{i,2})$ for all $i \in [t']$. This ensures that at least one vertex in A is colored using 1 as otherwise the blocking-gadget can not be properly colored. We repeat this process for the vertex sets B and D.

9. For all $i \in [t'], \alpha \in [m], \ell \in [3n-1]$ we add blocking gadgets with $\mathbf{c} = (c_1, c_2, 1)$ where $c_1, c_2 \in [2]$ and $c_1 \neq c_2$ and add edges $(s^i_{\alpha,\ell}, \pi_{1,1}, \pi_{1,2}), (s^i_{\alpha,\ell+1}, \pi_{2,1}, \pi_{2,2})$ and $(a_i, \pi_{3,1}, \pi_{3,2})$. This ensures that if a_i is colored 1, then all vertices $s^i_{\alpha,1}, ..., s^i_{\alpha,3n}$ are colored using the same color.

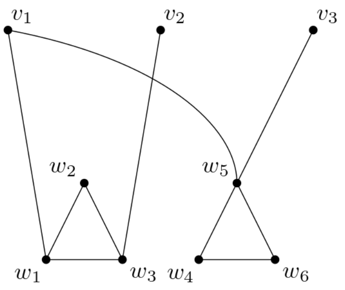
10. For all $j \in [t'], \ell \in [n]$ we add blocking-gadgets with $\mathbf{c} = (c_1, c_2, c_3, 1)$ where $c_1, c_2, c_3 \in [3]$ are not all pairwise distinct. We then add edges $(t^j_{\ell-2}, \pi_{1,1}, \pi_{1,2}), (t^j_{\ell-1}, \pi_{2,1}, \pi_{2,2}), (t^j_\ell, \pi_{3,1}, \pi_{3,2})$ and $(b_j, \pi_{4,1}, \pi_{4,2})$ where $b_j \in B$. This ensures that whenever b_j is colored using color 1, then all sets of 3 vertices $t^j_{\ell-2}, t^j_{\ell-1}, t^j_\ell$ have different colors.

11. For all $k \in [t']$, we add blocking-gadgets with $\mathbf{c} = \{c_1, c_2, c_3, 1\}$ where $c_1, c_2, c_3 \in [3]$ are not all pairwise distinct. We then add edges $(u_1^k, \pi_{1,1}, \pi_{1,2}), (u_2^k, \pi_{2,1}, \pi_{2,2}), (u_3^k, \pi_{3,1}, \pi_{3,2})$ and $(d_k, \pi_{4,1}, \pi_{4,2})$ where d_k is a vertex from D. This ensures that whenever d_k is colored with 1, the blocking-gadgets can only be properly colored whenever u_1^k, u_2^k, u_3^k all have different colors.

This concludes the construction of G'. An example can be seen in Figure 5. Here if an edge from S to T and an edge from T to U share the same color and node in T, then this means there is a hyper edge between the vertex in S, the vertex in T and all vertices in U. For example, this means in Figure 5 that the edges $\{s_{1,5}^1, t_5^1, t_1^2\}, \{s_{1,5}^1, t_5^1, t_2^2\}, \{s_{1,5}^1, t_5^1, t_3^2\}$ are part of G' as they all share the green edge. For simplicity the vertex sets A, B, D and the blocking-gadgets have been omitted.



(a) Constructed graph G'



(b) $X_{1,1,2}$

Figure 5: Construction of G' with $t' = 3, m = 3, n = 2$. Edges between S, T and U are shown for the instance $X_{1,1,2}$. All gadgets and sets A, B, D are left out.

Now we will prove that G' can only be 3-colored if and only if there is some instance $X_{i,j,k}$ which can be properly 2-3-colored. First we will prove some properties of our graph.

Claim 4.7. *For each proper 3-uniform hypergraph 3-coloring $\gamma : V(G') \to [3]$ of the graph G' there exists an $i \in [t']$ such that for all $\alpha \in [m]$ and distinct $\ell, \ell' \in [3n]$ it holds that $\gamma(s_{\alpha,\ell}^i) = \gamma(s_{\alpha,\ell'}^i)$*

Proof. First assume the inverse. Now we pick this i to be such that $\gamma(a_i) = 1$, as we know by step 8 that such a vertex must exists otherwise G' cannot be properly colored. Let $s_{\alpha,\ell}^i$ and $s_{\alpha,\ell'}^i$

be the two vertices such that $\gamma(s^i_{\alpha,\ell}) \neq \gamma(s^i_{\alpha,\ell'})$ for $\alpha \in [m]$ and distinct $\ell, \ell' \in [3n]$. This implies that there is also some $\ell \in [3n-1]$ such that $\gamma(s^i_{\alpha,\ell}) \neq \gamma(s^i_{\alpha,\ell+1})$. We also know that there exist blocking-gadgets with $\mathbf{c} = (c_1, c_2, 1)$ for all distinct colors $c_1, c_2 \in [2]$ which are connected to $a_i, s^i_{\alpha,\ell}, s^i_{\alpha,\ell+1}$ by step 9. Now since a_i is colored using 1, we know that $\pi_{3,1}, \pi_{3,2}$ are not colored using 1, as they are forced to have the same color by construction and assigning them the color 1 would improperly color the edge $(a_i, \pi_{3,1}, \pi_{3,2})$. Now since the blocking gadgets both have $c_1 \neq c_2$ and since there are edges $(s^i_{\alpha,\ell}, \pi_{1,1}, \pi_{1,2}), (s^i_{\alpha,\ell+1}, \pi_{2,1}, \pi_{2,2})$ we know that the only way to color them properly is for $s^i_{\alpha,\ell}$ and $s^i_{\alpha,\ell+1}$ to be assigned the same color, which is a contradiction to our assumption that γ is a proper coloring. ■

We will say that a triple of vertices v_1, v_2, v_3 is colorful under a coloring γ whenever $\gamma(v_1) \neq \gamma(v_2) \neq \gamma(v_3)$, which means they all get different colors.

Claim 4.8. *Let γ be a proper 3-hypergraph coloring of G'. Then there exists some $j \in [t']$ such that for all $\ell \in [n]$ the triple $t^j_{\ell-2}, t^j_{\ell-1}, t^j_\ell$ is colorful under γ.*

Proof. First off, we know that there exists some j such that $\gamma(b_j) = 1$ by step 8. We pick this to be our j of choice. We then assume that for this choice of j there is some triplet $t^j_{\ell-2}, t^j_{\ell-1}, t^j_\ell$ which is not colorful. Now by step 10 we know that there are blocking-gadgets with $\mathbf{c} = (c_1, c_2, c_3, 1)$ with $c_1, c_2, c_3 \in [3]$ not being all pairwise distinct. Now since $\gamma(b_j) = 1$, we know that for all possible pairs of the vertices $t^j_{\ell-2}, t^j_{\ell-1}, t^j_\ell$ it must hold that they are colored distinctly, otherwise there is some gadget introduced in step 10 that can not be properly colored and by extension G'. However, this is a contradiction to the assumption that γ is a proper 3-list-coloring for G'. ■

It is easy to confirm that there exists some $k \in [t']$ such that the triple u^k_1, u^k_2, u^k_3 is colored colorfully by steps 8 and 11 using a similar reasoning to Claim 4.8. Now with this we will prove Claim 4.9.

Claim 4.9. *The hypergraph G' is 3-list-colorable if and only if there is some input instance X_{i^*,j^*,k^*} which is properly 2-3-colorable.*

Proof. We prove the claim by splitting the proof in two cases, one for the forward and one for the backwards direction of this claim.

$(\Rightarrow)$:
Let γ be a proper coloring for G'. We will now show how to use this coloring to construct a new coloring γ' for some instance X_{i^*,j^*,k^*}. First, by construction and Claims 4.7 and 4.8 that there are some $i^*, j^*, k^* \in [t']$ such that $\gamma(a_{i^*}) = \gamma(b_{j^*}) = \gamma(d_{k^*}) = 1$. We will choose these indices for i^*, j^*, k^* to be the indices of the instance X_{i^*,j^*,k^*}, which will be colored using γ'. First for i^*, we know that for all $\alpha \in [m]$ and distinct $\ell, \ell' \in [3n]$ that $\gamma(s^{i^*}_{\alpha,\ell}) = \gamma(s^{i^*}_{\alpha,\ell'})$. We then color all vertices $v_\alpha \in V$ of X_{i^*,j^*,k^*} with $\gamma'(v_\alpha) = \gamma(s^{i^*}_{\alpha,1})$ for all $\alpha \in [m]$.

Similarly we know that for j^* it holds for all $\ell \in [n]$ that any triple $t^{j^*}_{\ell-2}, t^{j^*}_{\ell-1}, t^{j^*}_\ell$ is colorful. Now we assign to all $w_\ell \in W$ for $\ell \in [3n]$ colors as follows $\gamma'(w_\ell) = \gamma(t^{j^*}_\ell)$. We note that this means that all triangles in W are properly colored as all such triangles must by definition of coloring γ

be colored colorfully. It now remains to be checked that all edges between V, W of X_{i^*,j^*,k^*} are properly colored.

We now recall the fact that since $\gamma(d_k) = 1$, that the vertices $u_1^{k^*}, u_2^{k^*}, u_3^{k^*}$ are colored colorfully. In addition, during step 4 of the construction of G' we added edges $(s_{\alpha,\ell}^{i*}, t_\ell^{j^*}, u_\beta^{k^*})$ for $\alpha \in [m], \ell \in [3n], \beta \in [3]$ if there was an edge between (v_α, w_ℓ) in X_{i^*,j^*,k^*}. Now since all $u_1^{k^*}, u_2^{k^*}, u_3^{k^*}$ are colorful under γ and each of these vertices have an edge with the pair $s_{\alpha,\ell}^{i*}, t_\ell^{j^*}$ in G', we can conclude that $\gamma(s_{\alpha,\ell}^{i*}) \neq \gamma(t_\ell^{j^*})$ as otherwise one of those edges would be improperly colored. Thus by extension we know that all $\gamma'(v_\alpha) \neq \gamma'(w_\ell)$ for all $v_\alpha \in V, w_\ell \in W$ such that (v_α, w_ℓ) is an edge in X_{i^*,j^*,k^*}. Finally, by step 6 we know that $\gamma(s_{\alpha,\ell}^{i*}) \in [2]$ and thus also $\gamma'(v_\alpha) \in [2]$. From this we can conclude that all vertices in V are properly 2-colored. In addition, we conclude that all edges between V, W are properly colored and all triangles within W are properly colored. Thus γ' is a proper 2-3-coloring for X_{i^*,j^*,k^*}.

$(\Leftarrow)$:
Let γ be a proper 2-3-coloring for X_{i^*,j^*,k^*}. We will use this coloring to construct a coloring γ' for G'. First off we pick the vertices $a_{i^*}, b_{j^*}, d_{k^*}$ and assign $\gamma'(a_{i^*}) = \gamma'(b_{j^*}) = \gamma'(d_{k^*}) = 1$. We then assign to all $u_\beta^{k^*}$ for $\beta \in [3]$ the coloring $\gamma'(u_\beta^{k^*}) = \beta$. We then color the vertices $s_{\alpha,\ell}^{i^*}$ for $\alpha \in [m], \ell \in [3n]$ with $\gamma'(s_{\alpha,\ell}^{i^*}) = \gamma(v_\alpha)$ where $v_\alpha \in V_{i^*,j^*,k^*}$ and we color $\gamma'(t_\ell^{j^*}) = \gamma(w_\ell)$ where $w_\ell \in W_{i^*,j^*,k^*}$.

We then assign to the vertices in A, B and D that have not yet been colored the color 2. Finally, color all blocking-gadgets in accordance to this coloring. We note that this does allow the blocking gadgets introduced in step 8 to be properly colored as a_{i^*}, b_{j^*} and d_{k^*} were already colored with 1. We then add for all $\beta \in [3]$ and $k \in [t']$ where $i \neq i^*$ the coloring $\gamma'(u_\beta^k) = 3$, for all $\ell \in [3n], j \in [t']$ where $j \neq j^*$ the coloring $\gamma'(t_\ell^j) = 3$ and finally for all $i \in [t'], \alpha \in [m], \ell \in [3n]$ where $i \neq i^*$ we assign to all $s_{\alpha,\ell}^i$ a color from $[2] \setminus \{\gamma'(t_\ell^{j^*})\}$. We will now proceed to show that this coloring is a proper coloring of G'.

It is easy to verify that all edges $(s_{\alpha,\ell}^{i^*}, t_\ell^{j^*}, u_\beta^{k^*})$ for all $\alpha \in [m], \ell \in [3n], \beta \in [3]$ which were added in step 4 are properly 3-colored since by definition of γ, $s_{\alpha,\ell}^{i^*}$ and $t_\ell^{j^*}$ must have gotten different colors since they represent adjacent vertices in X_{i^*,j^*,k^*}. Since γ is a 2-3-coloring we also know for all $s_{\alpha,\ell}^{i^*} \in S_{i^*}$ satisfy $\gamma'(s_{\alpha,\ell}^{i^*}) \in l(s_{\alpha,\ell}^{i^*})$ since all $v \in V_{i^*,j^*,k^*}$ are colored using $1, 2$. For all vertices in A, B, D it also holds that they are properly list colored as they are all colored using either 1 or 2. We also note that for i^*, j^*, k^* the blocking-gadgets added in steps 9-11 are colored properly since all $u_\beta^{k^*}$ for $\beta \in [3]$ were given distinct colors, all triplets $t_{\ell-2}^{j*}, t_{\ell-1}^{j*}, t_\ell^{j*}$ for $\ell \in [n]$ have distinct colors and all vertices $s_{\alpha,\ell}^{i^*}, s_{\alpha,\ell'}^{i^*}$ for $\alpha \in [m]$ and distinct $\ell, \ell' \in [3n]$ satisfy $\gamma'(s_{\alpha,\ell}^{i^*}) = \gamma'(s_{\alpha,\ell'}^{i^*})$.

From this we conclude that coloring all a_i, b_j, d_k for $i, j, k \in [t']$ and $i \neq i^*, j \neq j^*, k \neq k^*$ being colored using the color 2 is fine as each $a_{i^*}, b_{j^*}, c_{k^*}$ has already been assigned color 1. This thus means that all S_i, T_j and U_k are not restricted by the gadgets in steps 9-11 and thus we can color those vertices with any color with respect to the blocking-gadgets. We note that for any choice of $i \neq i^*, j \neq j^*, k \neq k^*$ it holds that the edges added in step 4 are properly colored as all vertices in S_i, T_j, U_k will be colored by $\gamma'(s_{\alpha,\ell}^i) \in [2]$ and $\gamma'(t_\ell^j) = \gamma'(u_\beta^k) = 3$ for $\alpha \in [m], \ell \in [3n], \beta \in [3]$. It thus remains to be proven that any hyperedges in G' that are partially related to i^*, j^*, k^*, which means that there is at least one and at most two indices i, j, k for which $i = i^*, j = j^*, k = k^*$, are properly colored.

Consider that we have a some k where $k \neq k^*$. Here we know that all u_1^k, u_2^k, u_3^k are colored using 3. All edges involving these vertices are trivially properly 3-colored as for any edge, the vertex $s_{\alpha,\ell}^{i'}$ for $i' \in [t'], \alpha \in [m], \ell \in [3n]$ is colored using some color from [2]. This thus means that the edge always has 2 different colors and thus it is properly colored. By this same argument any edge involving vertices from T_j where $j \neq j^*$ are always properly colored, as all these vertices are colored using the color 3 as well.

Thus it remains to be proven that for any $i \neq i^*$ it holds that any edge involving the vertices in S_i are properly colored. We can trivially see that if we have some edge involving some vertex from S_i and T_j for $j \neq j^*$ then the edge is trivially colored properly as all vertices in T_j are colored using the color 3. Thus we consider the edges involving S_i and T_{j^*}. Now by construction, a vertex $s_{\alpha,\ell}^i$ was colored using $[2] \setminus \{\gamma'(t_\ell^{j^*})\}$ for $\alpha \in [m], \ell \in [3n]$. Now we know that this set is never empty as by construction step 4, there is only a single vertex from $t_\ell^{j^*}$ with which $s_{\alpha,\ell}^i$ shares edges. In addition, since this color is chosen such that $\gamma'(s_{\alpha,\ell}^i) \neq \gamma'(t_\ell^{j^*})$, we conclude that these edges are also properly colored.

Since all vertices are colored and since we concluded that all edges are properly colored, we conclude that G' is properly 3-colored and thus the case holds.

Since both cases have been proven the claim follows. ■

It thus remains to bound the cost of the amount of vertices in G'. First in steps 1 through 3 we create $t' \cdot m \cdot 3n$, $t' \cdot 3n$ and $3t'$ vertices respectively. In step 5 we add $3t'$ vertices. In step 8 we add per blocking-gadget vertices in the order $O(t')$. In step 9 we add blocking-gadgets which have constant size for which we add $t' \cdot m \cdot (3n - 1)$ so a total of order $O(t' \cdot m \cdot n)$ vertices. In step 10 we again add blocking-gadgets of constant order of which we add $t' \cdot n$ so a total of vertices of order $O(t' \cdot n)$ were added. Finally in step 11 a total of t' blocking-gadgets of constant order were added, which together add vertices of order $O(t')$. By now summing these amounts of vertices we get a total of $O(t' \cdot m \cdot n) = O(\sqrt[3]{t} \cdot (\max_{i,j,k} |X_{i,j,k}|)^{O(1)})$. This together with Claim 4.9, Lemma 4.1 and Theorem 2.8 allows us to conclude that 3-UNIFORM HYPERGRAPH 3-LIST-COLORING has no kernel of size $O(n^{3-\varepsilon})$ for any $\varepsilon > 0$ unless $NP \subseteq coNP/poly$. □

To now extend this proof to one for 3-UNIFORM HYPERGRAPH 3-COLORING all that is needed is to add degree-1 and 2 restricting gadgets and then add edges so that the lists for all vertices in G' are preserved. Since all restricting gadgets of any degree only require a constant amount of vertices to be added and since these are the only vertices added to G', we can conclude that there is no kernel of size $O(n^{3-\varepsilon})$ for 3-UNIFORM HYPERGRAPH 3-COLORING where $\varepsilon > 0$ unless $NP \subseteq coNP/poly$. Since a kernel of size $O(n^3)$ for 3-UNIFORM HYPERGRAPH 3-COLORING is trivial, the extension on Theorem 4.6 thus proves that there is no non-trivial kernel for 3-UNIFORM HYPERGRAPH 3-COLORING. This result thus leads us to conclude that for 3-UNIFORM HYPERGRAPH 3-COLORING this trivial upper bound is tight.

5 Conclusion

In conclusion, we have provided a framework which allows for kernels to be created for non-uniform CSP. Using the structure of a t-balanced language Γ over which a non-uniform CSP is defined, it is possible to create a kernel of $O(n^t)$ constraints for $CSP(\Gamma)$. This approach expands on already constructed framework as this approach allows for the construction of non-linear kernels and such kernels for $CSP(\Gamma)$ when it is defined over any finite domain, whereas most known methods tend to be restricted to $CSP(\Gamma)$ over a Boolean domain [10, 12]. The results of this thesis were obtained by first rewriting constraints as polynomials of degree at most t that capture the unsatisfying assignments with respect to the satisfying assignments of said constraints. By computing a basis for these polynomials it is then possible to construct a kernel of size $\mathcal{O}(n^t)$. It is interesting to note that it is possible to check whether a relation $R \in \Gamma$ is t-balanced in polynomial time by using the same methodology provided by Chen, Jansen and Pieterse [12]. This was utilized to confirm that the relation shown in Equation (9) is 2-balanced and it thus a set of constraints which utilizes this relation admits a kernel of $\mathcal{O}(n^2)$ constraints. This is a significant result as this relation has a trivial kernel of $\mathcal{O}(n^5)$ constrains. When it comes to larger domains the presented approach allowed for the construction for a nontrivial kernel of hypergraph coloring. More specifically 4-Uniform Hypergraph 4-Coloring with the requirement that each hyperedge contains at least 3 distinctly colored vertices showed to have a kernel of $\mathcal{O}(n^3)$ constraints. Moreover, 4-Uniform Hypergraph 3-coloring with the requirement that each hyperedge contains at least 3 distinctly colored vertices has an even smaller kernel of $\mathcal{O}(n^2)$ constraints. Again these results are significant as for both of these problems a trivial kernel consists of $\mathcal{O}(n^4)$ constraints. This shows that the presented approach can be used to find nontrivial kernels for many instances of non-uniform CSP. In addition we have explored and shown what it would take for $CSP(\Gamma)$ to have a single polynomial that captures a constraint defined over a relation R. First of we showed that it would not be possible to extend the presented methodology in order to find such a single polynomial, although this does not imply that they do not exist. This was followed by showing that it is possible to capture at least 2 unsatisfying constraints of some relation R by combining two already known polynomials for some $CSP(\Gamma)$ into single polynomial. It was shown that this is possible for any pair of polynomials but for anymore than 2 polynomials this is no simple feat.

Finally we have shown a kernel lower bound for 3-Uniform Hypergraph 3-Coloring which showed that no kernel of size $O(n^{3-\varepsilon})$ exists for any $\varepsilon > 0$. As Hypergraph Coloring is a form of $CSP(\Gamma)$ over a finite domain, this result yields insight in the complexity of such problems. Since for 3-Uniform Hypergraph 3-Coloring a trivial kernel contains $\mathcal{O}(n^3)$ constraints, this also shows that 3-Uniform Hypergraph 3-Coloring does not admit nontrivial kernel. Thus the method presented in Section 3.2 is also unable to find a nontrivial kernel for 3-Uniform Hypergraph 3-Coloring. However, as we have shown earlier, different variations of hypergraph coloring do admit nontrivial kernels so there is still a lot to be explored with regards to hypergraph coloring.

In addition, there are still other questions of interest which remain unanswered. First off, one can wonder whether a single polynomial exists per relation $R \in \Gamma$ which captures said relation. This is an important question as it will simplify the polynomial representation of $CSP(\Gamma)$. The question whether $CSP(\Gamma)$ can sparsified for non-finite domains also remains open. Answering this question will allow more problems to be sparsified which can greatly impact the efficiency of solving these computational problems. Finally it is not clear whether the bound $O(n^t)$ for some t-balanced language Γ is tight. Although when put to practice, the provided methodology does agree with already known lower bounds, such as for d-NAE-SAT we find that $t = d - 1$, it is not yet clear if this holds for all languages over which $CSP(\Gamma)$ can be defined.

References

[1] M. R. Krom, “The decision problem for a class of first-order formulas in which all disjunctions are binary,” *Mathematical Logic Quarterly*, vol. 13, no. 1-2, pp. 15–20, 1967.

[2] B. Aspvall, M. F. Plass, and R. E. Tarjan, “A linear-time algorithm for testing the truth of certain quantified boolean formulas,” *Information Processing Letters*, vol. 8, no. 3, pp. 121–123, 1979.

[3] S. Even, A. Itai, and A. Shamir, “On the complexity of timetable and multicommodity flow problems,” *SIAM Journal on Computing*, vol. 5, no. 4, pp. 691–703, 1976.

[4] R. M. Karp, *Reducibility among Combinatorial Problems*, pp. 85–103. Boston, MA: Springer US, 1972.

[5] A. A. Bulatov, “A dichotomy theorem for constraint satisfaction problems on a 3-element set,” *J. ACM*, vol. 53, no. 1, pp. 66–120, 2006.

[6] A. A. Bulatov, “A dichotomy theorem for nonuniform CSPs,” in *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017* (C. Umans, ed.), pp. 319–330, IEEE Computer Society, 2017.

[7] D. Zhuk, “A proof of the CSP dichotomy conjecture,” *J. ACM*, vol. 67, no. 5, pp. 30:1–30:78, 2020.

[8] H. Dell and D. Van Melkebeek, “Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses,” *J. ACM*, vol. 61, July 2014.

[9] B. M. P. Jansen and A. Pieterse, “Optimal data reduction for graph coloring using low-degree polynomials,” *Algorithmica*, vol. 81, no. 10, pp. 3865–3889, 2019.

[10] B. M. P. Jansen and A. Pieterse, “Optimal sparsification for some binary csps using low-degree polynomials,” *ACM Trans. Comput. Theory*, vol. 11, no. 4, pp. 28:1–28:26, 2019.

[11] V. Lagerkvist and M. Wahlström, “Kernelization of constraint satisfaction problems: A study through universal algebra,” in *Principles and Practice of Constraint Programming - 23rd International Conference, CP 2017, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings* (J. C. Beck, ed.), vol. 10416 of *Lecture Notes in Computer Science*, pp. 157–171, Springer, 2017.

[12] H. Chen, B. M. P. Jansen, and A. Pieterse, “Best-case and worst-case sparsifiability of boolean CSPs,” *Algorithmica*, vol. 82, no. 8, pp. 2200–2242, 2020.

[13] M. Datar, T. Feder, A. Gionis, R. Motwani, and R. Panigrahy, “A combinatorial algorithm for MAX CSP,” *Inf. Process. Lett.*, vol. 85, no. 6, pp. 307–315, 2003.

[14] H. Chen, B. M. P. Jansen, K. Okrasa, A. Pieterse, and P. Rzążewski, “Sparsification Lower Bounds for List H-Coloring,” in *31st International Symposium on Algorithms and Computation (ISAAC 2020)* (Y. Cao, S.-W. Cheng, and M. Li, eds.), vol. 181 of *Leibniz International Proceedings in Informatics (LIPIcs)*, (Dagstuhl, Germany), pp. 58:1–58:17, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020.

[15] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms.* Springer, 2015.

[16] R. Majdoddin, "Uniform CSP parameterized by solution size is in W[1]," in *Computer Science - Theory and Applications - 14th International Computer Science Symposium in Russia, CSR 2019, Novosibirsk, Russia, July 1-5, 2019, Proceedings* (R. van Bevern and G. Kucherov, eds.), vol. 11532 of *Lecture Notes in Computer Science*, pp. 275–285, Springer, 2019.

[17] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch, "Kernelization lower bounds by cross-composition," *SIAM J. Discret. Math.*, vol. 28, no. 1, pp. 277–305, 2014.

[18] B. M. P. Jansen and A. Pieterse, "Sparsification upper and lower bounds for graph problems and not-all-equal SAT," *Algorithmica*, vol. 79, no. 1, pp. 3–28, 2017.

[19] L. Jaffke and B. M. P. Jansen, "Fine-grained parameterized complexity analysis of graph coloring problems," in *Algorithms and Complexity* (D. Fotakis, A. Pagourtzis, and V. T. Paschos, eds.), (Cham), pp. 345–356, Springer International Publishing, 2017.