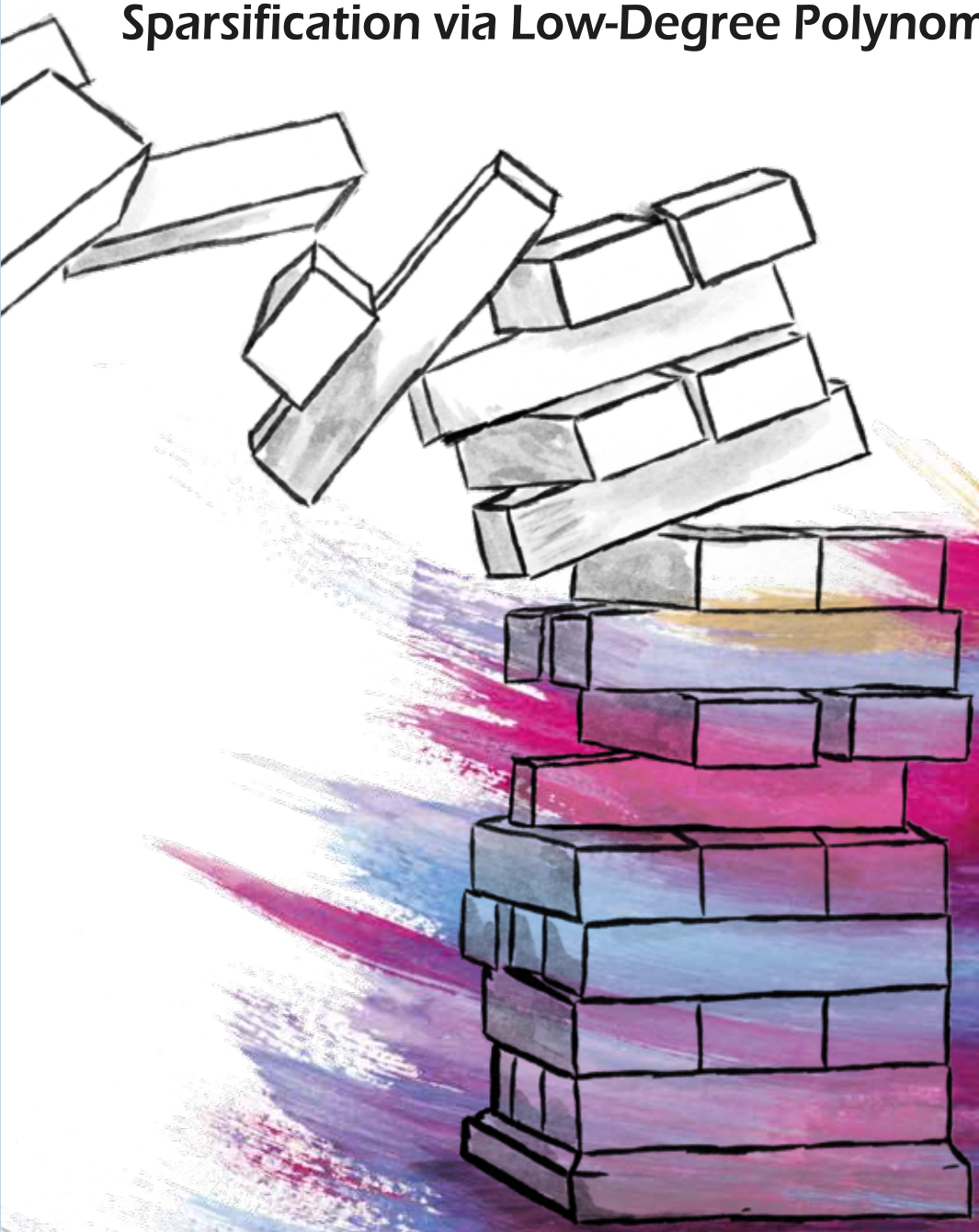
Tight Parameterized Preprocessing Bounds

Sparsification via Low-Degree Polynomials



Astrid Pieterse

Astrid Pieterse

Tight Parameterized Preprocessing Bounds

Sparsification via Low-Degree Polynomials

The work in this thesis has been supported by the Netherlands Organization for Scientific Research NWO under project number 024.002.003 (Networks).

Cover design by Astrid Pieterse.

The cover represents the method of kernelization by carefully removing redundant constraints, as described in Chapter 3 of this thesis.

Printed by Ipskamp Printing, Enschede.

A catalogue record is available from the Eindhoven University of Technology Library.

ISBN: 978-90-386-4821-7.

Tight Parameterized Preprocessing Bounds

Sparsification via Low-Degree Polynomials

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de rector magnificus prof.dr.ir. F.P.T. Baaijens, voor een commissie aangewezen door het College voor Promoties, in het openbaar te verdedigen op woensdag 4 september 2019 om 16:00 uur

door

Astrid Pieterse

geboren te Weert

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

Voorzitter:	prof.dr. J.J. Lukkien
Promotor:	prof.dr. M.T. de Berg
Co-promotor:	dr. B.M.P. Jansen
Leden:	prof.dr. N. Bansal
	prof.dr. S. Kratsch *(Humboldt-Universität zu Berlin)*
	dr.hab. M. Pilipczuk *(University of Warsaw)*
	prof.dr. H.L. Bodlaender *(Universiteit Utrecht)*

Het onderzoek of ontwerp dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

Astrid Pieterse,
July 2019

Acknowledgments

I still remember when Mark introduced me and a couple of fellow students to Bart, who would teach us a course about parameterized complexity. This led to the decision to do my master's thesis with Bart. In the first few weeks of that project, I realized how much fun research can be and after giving it a bit of thought, I accepted a PhD position supervised by him.

As such, I would first of all like to thank Bart for his supervision. Thank you for all the advice on research and writing, and for always being encouraging. Our weekly meetings were both useful and enjoyable, and I think almost all results in this thesis spent some time on one of your whiteboards. When finishing a paper (a little too) shortly before the SODA deadline I learned that you are not only a good supervisor, but also generally great to work with.

Secondly, I would like to thank my promotor Mark de Berg. You always had an open door, for questions or to help with any practical affairs.

Next, I would like to thank all my other committee members, Stefan Kratsch, Nikhil Bansal, Marcin Pilipczuk, and Hans Bodlaender, for reviewing this thesis. Stefan, thank you for inviting me for a productive research visit to Berlin, it was great to spend the week working with you and Eva-Maria. Marcin, thanks for inviting me for a great week of research at a workshop in Poland[1].

I want to thank Hubie Chen for inviting Bart and me to visit him in London, and for visiting us in Eindhoven, each resulting in a chapter of this thesis. I'd like to thank Gerhard Woeginger for supervising an honors project during my master's that resulted in a publication during this PhD. I would also like to thank my co-authors Sudeshna Kolay and Hans Bodlaender, for the project we started at the Lorentz center, the results of which have been accepted to WADS.

Now this whole PhD adventure would have been significantly less fun without the people I shared an office with throughout these 4 years. Thanks to my old office mates Sándor, Aleks[2], Mehran, and Bruce, for the great office

[1] If people remember me muttering that it was in some unreachable location (Karpacz), I can only say that the view of the snowy mountains was worth it.

[2] You're welcome. We'll meet again.

atmosphere. Thanks also to my new office mates Martijn and Henk with whom I shared an office during the final stages of this PhD project.

Furthermore I would like to thank all my colleagues for the friendly atmosphere and daily joint lunches. Thanks to Mehran, Aleks, Sándor, Martijn, Henk, Ali, Max, Thom, Max, Dani, Huib, Arthur, Quirijn, Tim, Willem, Jules, Bram, Jorn, Sudeshna, Kevin, Bart, Herman, Kevin, Irina, Ignaz, Wouter, Marcel, Jesper, Hans, Mark, and Bettina.

Because I was part of the Networks project, I joined many of the Networks Training weeks that were held twice a year. Thank you to all Networks and non-Networks members (I cannot name you all, but surely you know who you are) for making the training weeks enjoyable, for playing games in the evening, and going for walks in the forests. Furthermore, as part of the Networks project I did a short "internship" at CWI with Pieter Kleer and Guido Schaefer. I really enjoyed working with you both.

During this PhD position, I got to travel to various conferences, meetings, and workshops. It took me to places I had always wanted to visit (Vienna, Canada), wonderful places I would otherwise never have thought of (Wrocław, Bergen (NO)) and places I had already been (London). Apart from listening to research presentations, it also gave me the opportunity to do some sightseeing and to meet (new or not so new) people there. Thanks to Tom, Marieke, Sándor, Lars, Eva-Maria, Huib, Jari, Bart, and a bunch of people I am forgetting to mention here, for fun conversations, joint lunches and dinners, and for the sightseeing we did together in the free time left by the conference schedule.

Among the many people I met at conferences during this PhD, one is special to me, and not only because he just-so happened to start working at TU/e on the day that we met. Huib, thanks for all the love and support, you're amazing.

After spending nine years here at TU/e, first as a student, then as a PhD student, it is strange to leave. I'd like to thank the many teachers I had here for sparking my interest in algorithms, contributing to my (very helpful) background in mathematics, or otherwise being great teachers. Furthermore, I would like to thank everyone who contributed to making the TU/e feel like home for all these years. I would like to especially thank those I studied with: Daan, Thom, Willem, Kay, Peter, Laurent, and Niels, for the many successful yet never too serious group projects we did together, and all the silly jokes and fun.

A special thanks also to my friends outside of this university, for distracting me from research (and possibly listening to me complain about it some): Emmy, Inge, Nadia, Jelleke, Rose, and Saskia.

Last but foremost, I would like to thank my family[3], in particular my parents and my brother. You have been the best support I could possibly have hoped for during this PhD and for all the years before that. Dank jullie wel!

[3] and a thanks to Nala for being a (cute, and occasionally hilarious) distraction from everything.

Contents

Chapter 1

Introduction

In this thesis, we will study the effectiveness of preprocessing. To see what kind of preprocessing we will consider, let us start by looking at a puzzle. In Figure 1.1 you see a set of points. The challenge is to cover all these points, by drawing at most eight lines. The lines can be arbitrarily long, but they must be straight. A point is only covered if you draw a line right through its center. Let us attempt to solve the puzzle!

The puzzle in Figure 1.1 has 48 points. With this many points to cover, how do you start solving the puzzle? Observe that if there is a line L you can draw that has at least nine points on it, you must use that line. Otherwise, you would have to draw separate lines through each of the nine points. After all, any line containing at least two out of these nine points, corresponds to line L itself. Given that you cannot afford nine lines, you must indeed choose to draw the line L in order to be able to solve the puzzle. In Figure 1.2, such a line is indicated. Now, you can continue from there, with only 7 lines remaining, and many fewer points. Since we have only seven lines left, if we can find a line with at least eight points on it we must choose that line. The puzzle depicted here is such that, by simply continuing this reasoning, you can in fact solve the entire puzzle. Refer to Figure 1.3, for the solution. This of course need not always be true, the above method is no more than a preprocessing rule: there is no reason to believe that it should always apply, and also no reason to believe that the solution is trivial in puzzles where it does not apply.

Yet we can observe something else. When the above rule no longer applies,

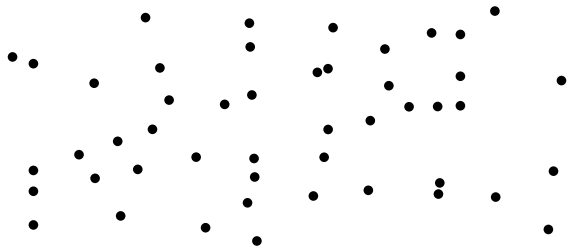


Figure 1.1 Can you cover all points by drawing at most 8 straight lines?

we still have k lines left that we can use, for some $k \leq 8$, and no line exists that covers at least $k+1$ points. Stated differently, every line you can draw will cover at most k of the remaining points, and you can only use k lines in total. But then, for the puzzle to be solvable, we cannot actually have many points. If the puzzle has more than $k \cdot k = k^2$ points remaining, it cannot be solved at all. So even if our preprocessing step does not solve the puzzle, it does guarantee that after preprocessing, the remaining puzzle is "small"!

While the above example is somewhat artificial, it does demonstrate a general solving strategy: before trying to solve a problem, first try to preprocess it by applying (simple) rules that reduce its size or its complexity. This strategy can be applied in a wide range of different scenarios, and preprocessing is indeed widely used in practical applications. For example, many real-world scheduling problems can be modeled as mixed-integer programs, and then solved by a solver such as the CPLEX optimizer. These solvers often start by using a number of rules to reduce the solution space as well as to identify redundant constraints, before solving the problem [2, 5]. This preprocessing has for example been shown to greatly add to CPLEX's ability to solve optimization instances [3, 12]. Disabling CPLEX's presolving step was shown to increase average solving time on the more challenging problems by a factor 11 [3].

Preprocessing techniques were also shown to be useful in the PACE competition in recent years [30, 31]. This is a competition in which teams design and implement (parameterized) algorithms that solve a given problem as quickly as possible on a variety of input instances. Here preprocessing is often applied. For example, the winner of one of the challenges of the 2016 edition [30] used an efficient preprocessing algorithm for the FEEDBACK VERTEX SET problem as a subroutine [54].

As seen from the examples above, a good preprocessing algorithm must be efficient, as we want to speed up the overall computation. Furthermore, a preprocessing algorithm should be guaranteed to be correct, meaning that the answer or solution to a preprocessed input instance can quickly be turned into a solution for the original instance. Finally, we would like to be able to give some

guarantee that the preprocessed instance is always "small", or in some sense easier than the original instance.

In this thesis, we will study the power of such preprocessing algorithms. In particular, we will study preprocessing algorithms that are required to run in polynomial time, such that the preprocessing step can be done efficiently. The problems we will consider in this thesis are NP-hard. As such, we cannot expect to find polynomial-time preprocessing algorithms that always manage to reduce the size of the given instance. After all, such a preprocessing step could then be used repeatedly, until the instance is so small that it is trivial to solve. This would yield a polynomial-time algorithm for an NP-hard problem, which is unlikely to exist. For this reason, we will measure the effectiveness of our preprocessing algorithm by some additional parameter, other than the input size, that (hopefully) measures the complexity of the given input instance.

To study the effectiveness of preprocessing, we therefore consider *parameterized problems*, whose inputs consist of the "regular" input, together with an additional parameter (an integer) that is a measure of the complexity of the given input. A preprocessing algorithm must now guarantee that the size of a preprocessed instance is bounded by some (hopefully small) function of this additional parameter.

1.1 Kernelization

To formally study this type of preprocessing, we will use the notion of kernelization. A *kernelization algorithm* (or *kernel*) is a polynomial-time preprocessing algorithm that takes an instance of a parameterized problem, and outputs an equivalent instance to the same problem whose size is bounded by some function of the considered parameter. This bound on the instance size in terms of the initial parameter value is known as the *size* of the kernel, and can only depend on the value of the initial parameter. In this thesis, we will only consider *decision problems*, which are problems that only have a yes/no-answer. As such, a kernel is only required to preserve this yes/no-answer for the preprocessed instance to be considered equivalent to the original one.

The example of a preprocessing algorithm above where we wanted to cover points with lines, contains all observations needed to provide a kernelization algorithm for the problem. Suppose we have an arbitrary instance asking if it is possible to cover n points with k lines, and let k be the considered parameter. The kernel repeatedly finds a line through at least $k+1$ points, removes all relevant points, and decreases k by one. When this rule is no longer applicable, it checks if there are at most k^2 remaining points. If yes, then this is the preprocessed instance. If not, then the puzzle has no solution, and a trivial *no*-instance can be returned. As such, Point-Line Cover (the problem of covering points with

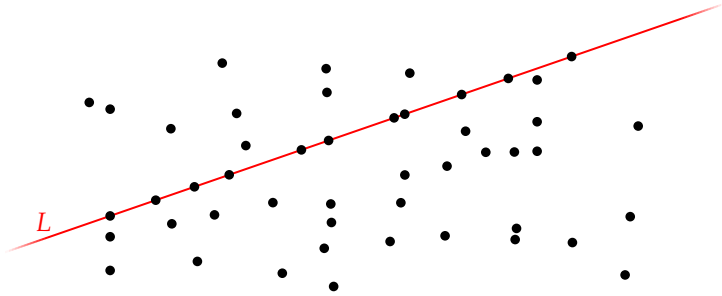


Figure 1.2 Can you cover all points not already covered by L by drawing at most 7 additional straight lines?

lines) has a kernelization algorithm that always outputs an instance with at most $\mathcal{O}(k^2)$ points[4].

One would like to distinguish parameterized problems that have "good" kernels, such as the one for Point-Line Cover given above, from those that do not. In this direction, one may ask which parameterized problems allow for so-called polynomial kernels, which are kernels whose size bound is polynomial in the parameter value. Such kernels are generally considered good to have, as their size grows not-too-fast when the parameter increases. Many (general) tools have been developed to find (polynomial) kernels for a wide variety of problems [15, 39, 67], and there are several books about the area [29, 40]. Furthermore, methods have been developed to be able to rule out the existence of polynomial kernels[5].

The easiest method to rule out polynomial kernels is perhaps to provide a certain type of reduction [18] starting from a parameterized problem that is already known not to have a polynomial kernel, to the problem under consideration. Note that, while NP-hardness reductions may indeed transfer kernelization lower bounds, this is not always the case, as more care needs to be taken to ensure that not only the size of the new instance is polynomial in the original size, but that also the new parameter is appropriately bounded. As usual, such a reduction requires a suitable starting problem.

Another lower bound technique is that of cross-compositions, which was

[4]The *size* of this kernel is not necessarily bounded by $\mathcal{O}(k^2)$, as the size of a kernel is measured by the number of bits needed to represent the resulting instance. This depends on the representation of the input points, which we will not discuss for this example.

[5]All "non-existence" or lower bound results mentioned in this introduction are based on complexity theoretic assumptions, since for example P = NP would imply constant-size kernels for all problems considered in this thesis. Refer to Chapter 2, in particular §2.4.1, for further details.

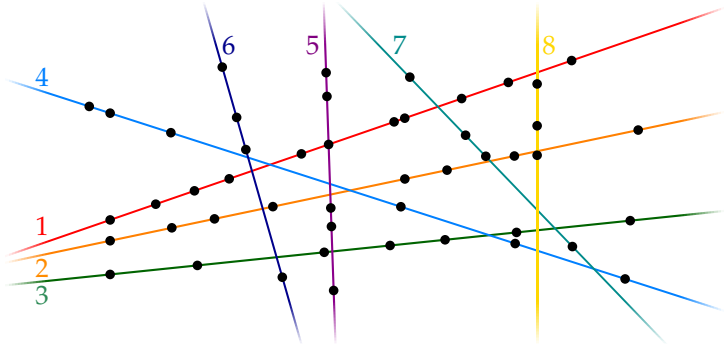


Figure 1.3 One possible way to cover all points given in Figure 1.1 with 8 straight lines. An order in which one could find these lines by repeatedly applying the preprocessing rule is indicated.

introduced by Bodlaender *et al.* [16] and relies on the non-existence of so-called distillation algorithms for problems that are NP-hard, which was proven by Fortnow and Santhanam [41]. Using these methods, it has been shown that a number of problems do not have polynomial kernels [14, 18, 68].

For those parameterized problems that do have a polynomial kernel, we can still ask what the best bound is that can be achieved. For example, we have seen that the Point-Line Cover problem has a kernel with $\mathcal{O}(k^2)$ points, and one may ask whether this can be improved to $\mathcal{O}(k^{1.5})$, or even better. Such questions can sometimes be answered, at least in the negative direction, by using a more refined definition of cross-compositions [16]. This allows us to give very precise bounds on the worst-case size of a best-possible kernel. One of the first results in this direction was obtained by Dell and Van Melkebeek [33].

In this thesis, we will be interested in answering the question: Given a parameterized problem, what is the smallest possible kernel? In particular, we will consider a number of classic graph and logic problems.

1.2 Polynomial-time sparsification

Observe that a graph with n vertices can have $\binom{n}{2} = \Theta(n^2)$ edges, which is quadratic in the number of vertices. One specific method of preprocessing for a graph problem, would be trying to make the input graph less dense, by reducing the number of edges. Of course, this removal of edges should not change the answer to the problem at hand. We will refer to such a procedure as a sparsification.

As an example, consider the graph depicted in Figure 1.4. We can ask

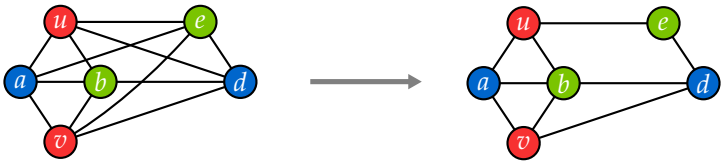


Figure 1.4 Depicted is an instance for 3-COLORING (left). On the right is an equivalent instance, obtained by removing a number of edges that (provably) do not change the colorability of the graph. In fact, one may verify that both instances have the exact same 3-colorings.

ourselves whether this graph can be colored with at most three colors, say red, green, and blue, such that the endpoints of every edge get distinct colors. This problem is also known as the 3-COLORING problem. Since it is NP-hard [46], we do not expect to find a polynomial-time algorithm to solve the problem. We can however try to eliminate certain edges in polynomial time. For example, since vertices a and b are connected by an edge, a proper 3-coloring will assign them different colors. Since vertex u connects to both a and b, it will be colored by the only remaining color. Similarly, vertex v should get a color different from a and b and will thus receive the same color as vertex u.

Now, we can use the knowledge that u and v always receive the same color, to remove a number of edges from the depicted instance: vertex e is connected to both u and v, but we can actually forget about its connection to v, as the remaining connection to u is enough to ensure that e and v receive distinct colors. Similarly, we can eliminate the connection between d and u. We can furthermore observe that vertices a and d always receive the same color, and remove the connection between e and a. This results in the sparsified instance depicted in Figure 1.4 on the right.

The argumentation we used above to sparsify the instance ensures that the solution does not change. In this thesis we will study whether or not we can find such rules that are not only correct in this way, but also powerful enough to give some non-trivial upper bound on the number of edges of the reduced instance.

Sparsification can also be applied to logic problems. Consider as an example the d-CNF-SAT problem, which is one of the most well-known logic problems. The input to d-CNF-SAT is a formula in d-CNF form, thus a formula over n variables consisting of a conjunction of clauses, where each clause is a disjunction of d literals. A literal is simply a variable or its negation. Consider for example the following 3-CNF-SAT instance.

$$\mathcal{F} = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee \neg x_3 \vee x_4).$$

The problem now asks whether there is a Boolean assignment to the variables that satisfies the formula. In the example above, letting $x_1 := x_2 := \mathit{true}$ and

$x_3 := x_4 :=$ *false* would satisfy $\mathcal{F}$.

The goal of sparsification for logic problems is to efficiently reduce the number of clauses (or constraints) in terms of the number of variables. If we look at the formula $\mathcal{F}$ given above, we may observe that removing the last clause does not influence the set of satisfying assignments to $\mathcal{F}$: any assignment satisfying the first two clauses, also satisfies the third. We will therefore say that this last clause is *redundant*. Define $\mathcal{F}'$ to be the formula consisting of only the first two clauses of $\mathcal{F}$. We can see that $\mathcal{F}'$ and $\mathcal{F}$ are equivalent, as follows. Consider any satisfying assignment for $\mathcal{F}'$. Clearly, this assignment satisfies the first clause $(x_1 \vee x_2 \vee \neg x_3)$. Thus, the assignment will set either x_1, x_2 or $\neg x_3$ to *true*. If x_2 is assigned *true*, this assignment also satisfies the last clause of $\mathcal{F}$. Similarly, if $\neg x_3$ is *true*, and thus if x_3 is *false*, the assignment satisfies the last clause. It remains to consider the case where only x_1 is set to *true*. But then, to satisfy the second clause, at least one of x_2 and x_4 is set to *true* (as $\neg x_1$ is *false*) and again the last clause is satisfied. As such, we see that $\mathcal{F}'$ has exactly the same satisfying assignments as the original formula!

An interesting challenge is to give general methods that efficiently find such redundant clauses, and significantly reduce the number of clauses.

For both graph and logic problems, we can observe that we can easily obtain a sparsification that gives some upper bound on the number of edges or clauses, respectively. A simple graph on n vertices can have only $\binom{n}{2} = \mathcal{O}(n^2)$ edges. A d-CNF formula on n variables can have at most $(2n)^d = \mathcal{O}(n^d)$ distinct clauses. The simplest sparsification for d-CNF-SAT would therefore be to simply remove duplicate clauses. We will therefore say that graph problems for which the input is a simple graph have a *trivial* sparsification of size $\mathcal{O}(n^2)$ (do nothing), and a problem like d-CNF-SAT has a trivial sparsification of size $\mathcal{O}(n^d)$ (remove duplicate clauses).

The question that is studied in this thesis is whether a *non-trivial* sparsification exists: a sparsification that achieves better bounds than the ones given above. For example, can we efficiently reduce the number of edges to $\mathcal{O}(n)$ for a given problem defined on general graphs? And can we reduce the number of clauses in a logic problem with clauses of size d to $\mathcal{O}(n^c)$ for some $c < d$ in polynomial time?

Many of the earlier results in this direction were negative [33,56,59], and in this thesis we will add several (graph) problems to the list of problems that do not have a non-trivial sparsification. On the other hand, we will show a number of surprising non-trivial sparsifications for logic problems, demonstrating that non-trivial sparsifications are possible for a remarkable number of such problems. Let us continue by looking at the problems that will be studied in this thesis in more detail.

1.3 Graph coloring

In this thesis, we will study a very general graph problem, for which it was previously not known whether a non-trivial sparsification was possible. A special case of the problem that we will study is the q-Coloring problem. It generalizes the 3-Coloring problem discussed earlier. Given a graph G, it asks whether we can assign every vertex one of q colors, such that each edge has differently colored endpoints.

This is a very well-studied problem [63, 73], with many applications [6, 45, 77]. For example, graph coloring can be used to solve scheduling problems, where jobs need to be assigned to machines. If we assume for simplicity that all machines are identical and one machine cannot handle multiple jobs at the same time, this can be seen as a coloring problem. We can model each job as a vertex, and each machine as one of the colors. We connect two jobs with an edge if they overlap in time. Coloring the resulting graph gives a solution to the original scheduling problem, as it is ensured that each job is assigned a machine (color), and no machine does two tasks at the same time.

In the classic q-Coloring problem, vertices of the same color cannot be connected by an edge, but there are no further restrictions on the coloring. In some cases, you may want to forbid other color combinations as well. For example, one could want to color the graph with five colors, such that every edge has differently colored endpoints and furthermore, a pink vertex can never be connected to an orange one. To capture these kind of constraints, we will look at the H-Coloring problem, which is defined for every fixed graph H. The colorset we now use is given by the vertex set of H, meaning every vertex of H corresponds to a color we may use. An input to the problem consists of an undirected graph G, and the question is to color each vertex of G with one of these colors. If we now look at any edge in G, we require that the coloring of its endpoints is allowed by H, meaning that the corresponding vertices in H are connected by an edge. To see that this is a generalization of the q-Coloring problem, observe that H-Coloring corresponds to q-Coloring when we let H be a clique on q vertices.

In this thesis, we will show a sparsification lower bound for H-Coloring under certain conditions on H. Furthermore, we will obtain a tight kernelization bound for the problem when parameterized by the size of a vertex cover in the input graph. We then extend this result to obtain the same kernel bound for a smaller parameter.

1.4 Constraint satisfaction problems

When introducing sparsification, we used the logic problem d-CNF-SAT as an example. In the d-CNF-SAT problem, we are given a formula that consists of a

number of clauses and the formula is satisfied if at least one literal evaluates to *true* in each clause. It is easy to think of many variations to this problem, by changing the type of clauses we use. For example, the case where the formula is satisfied when each clause contains *exactly* one literal that is set to *true* corresponds to the problem that is known as EXACT SAT.

We can put these problems in a common framework by studying the CONSTRAINT SATISFACTION PROBLEM (CSP) over a certain domain D. Broadly speaking, the input for this problem consists of a set of variables and a number of constraints over these variables. The question is whether it is possible to assign each variable a value from D, such that all constraints are satisfied. In this thesis we will mostly study the case where $D = \{0,1\}$ (or equivalently, $D = \{\mathit{false}, \mathit{true}\}$), known as so-called BOOLEAN CSPs.

In principle, a constraint could be almost anything: it could capture that "exactly one of the variables x,y,z is *true*", or that "$x = \neg y$", or perhaps that "$x = \mathit{true}$ or $y = z$". Clearly, the expressiveness of a CSP depends on the type of constraints one allows. The type of allowed constraints is given by a so-called *constraint language*, often denoted by Γ. We can thus study the problem $\mathrm{CSP}(\Gamma)$, which deals with input instances that only use constraints allowed by Γ. This study of CSPs led to a complete dichotomy for Boolean CSPs: if Γ satisfies certain (verifiable) conditions, $\mathrm{CSP}(\Gamma)$ is polynomial-time solvable, otherwise $\mathrm{CSP}(\Gamma)$ is NP-complete [83]. The hardness of $\mathrm{CSP}(\Gamma)$ over non-Boolean domains remained open for many decades, until recently also in this situation a complete dichotomy was obtained. The dichotomy was proven independently by Bulatov and Zhuk [22, 92].

We can obtain a more fine-grained view of the complexity of constraint satisfaction problems, by studying how the sparsifiability of the problem depends on properties of Γ. The goal is then to identify relevant properties of Γ that determine the best-possible size of a sparsification for $\mathrm{CSP}(\Gamma)$. While we are far from a complete classification, we put a number of existing sparsification results into a common framework in this thesis. Furthermore, we give a widely applicable technique to sparsify CSPs and show that the obtained results are tight in several cases.

1.5 Thesis overview

We will start by giving the necessary preliminaries (including the formal definitions of all concepts mentioned in this introduction) in Chapter 2.

In Chapter 3, we study the sparsifiability of a specific type of constraint satisfaction problem. In particular, we consider CSPs that are given as equations of polynomials over multiple variables over various rings and fields. In this case, we obtain sparsification results where the size of the sparsification depends on the degree of the given polynomial equalities and the properties of the ring

or field over which these polynomials are given. This method of sparsification turns out to generalize a number of existing sparsification results for well-known logic problems.

We furthermore give lower bounds to show that the results obtained by this sparsification technique are best-possible, by providing constraint languages for which we obtain an optimal sparsification.

In Chapter 4 we continue our study of CSPs, by applying the methodology introduced in Chapter 3 to a number of constraint satisfaction problems. We start by showing how to apply the sparsification method from Chapter 3 to a wider range of CSPs. We continue by giving a general property for constraint languages that can be used to obtain sparsification lower bounds.

Using these results, we show that we can fully distinguish between Boolean CSPs that have a non-trivial sparsification and those that do not. We then give a sufficient condition of a constraint language that ensures that the corresponding CSP has a linear sparsification. We use this result to fully classify all symmetric Boolean CSPs that have a linear sparsification. We conclude the chapter by fully classifying the sparsifiability of CSPs over constraint languages where each constraint concerns at most 3 variables.

After studying CSPs, we change our focus to a graph problem, namely the H-Coloring problem, which is a generalization of the well-known q-Coloring problem. In Chapter 5, we show a sparsification lower bound for this problem. Under complexity-theoretic assumptions, we show that for all graphs H satisfying certain conditions, the H-Coloring problems does not have a sparsification with $\mathcal{O}(n^{2-\varepsilon})$ bits, for any $\varepsilon > 0$. Among other things, this implies a sparsification lower bound for the well-known 3-Coloring problem. This chapter combines kernelization lower bound techniques, with techniques from the algebraic analysis of constraint satisfaction problems.

In Chapter 6, we will study the kernelization of the H-Coloring problem, when parameterized by the size of a vertex cover in the input graph. A kernel for q-Coloring parameterized by the size of a vertex cover was previously known, but while this kernel had size $\mathcal{O}(k^q)$, the best lower bound was a factor k smaller. We show how to improve the kernel to have $\mathcal{O}(k^{q-1})$ vertices and edges by cleverly using the sparsification for CSPs introduced in Chapter 3. We further generalize this result to obtain a kernel for H-Coloring parameterized by the size of a twin-cover.

1.6 List of publications

This thesis is based on the following publications. Chapter 3 is based on the following publication.

Hubie Chen, Bart M. P. Jansen, and Astrid Pieterse. Best-case and worst-case sparsifiability of Boolean CSPs. In *Proceedings of the 13th International*

Symposium on Parameterized and Exact Computation (IPEC 2018), volume 115, pages 15:1–15:13, 2019. doi:10.4230/LIPIcs.IPEC.2018.15

Chapter 4 is based on the following publication.

Bart M. P. Jansen and Astrid Pieterse. Optimal sparsification for some binary CSPs using low-degree polynomials. In *Proceedings of the 41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*, volume 58, pages 71:1–71:14, 2016. doi:10.4230/LIPIcs.MFCS.2016.71

Chapter 5 is based on the following manuscript.

Hubie Chen, Bart M. P. Jansen, and Astrid Pieterse. Sparsification lower bounds for H-coloring. Submitted

Chapter 6 is based on the following publication.

Bart M. P. Jansen and Astrid Pieterse. Optimal data reduction for graph coloring using low-degree polynomials. *Algorithmica*, 2019. Online First. doi:10.1007/s00453-019-00578-5

The following publications do not appear in this thesis.

Hans L. Bodlaender, Sudeshna Kolay, and Astrid Pieterse. Parameterized complexity of conflict-free graph coloring. In *Proceedings of the 16th Algorithms and Data Structures Symposium (WADS 2019)*, 2019. To appear. URL: https://arxiv.org/abs/1905.00305

Bart M. P. Jansen and Astrid Pieterse. Polynomial kernels for hitting forbidden minors under structural parameterizations. In *Proceedings of the 26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112, pages 48:1–48:15, 2018. doi:10.4230/LIPIcs.ESA.2018.48

Bart M. P. Jansen and Astrid Pieterse. Sparsification upper and lower bounds for graph problems and not-all-equal SAT. *Algorithmica*, 79(1):3–28, 2017. doi:10.1007/s00453-016-0189-9

Astrid Pieterse and Gerhard J. Woeginger. The subset sum game revisited. In *Proceedings of the 5th International Conference on Algorithmic Decision Theory (ADT 2017)*, volume 10576, pages 228–240, 2017. doi:10.1007/978-3-319-67504-6_16

Rafael G. Cano, Kevin Buchin, Thom Castermans, Astrid Pieterse, Willem Sonke, and Bettina Speckmann. Mosaic drawings and cartograms. *Computer Graphics Forum*, 34(3):361–370, 2015. doi:10.1111/cgf.12648

Chapter 2

Preliminaries

In this section we introduce the background material that will be used throughout this thesis. We start in Section 2.1 by introducing some basic notation. In Section 2.2, we give notation and definitions relating to graphs and graph problems. We continue in Section 2.3 by introducing parameterized complexity, and giving the formal definition of kernelization. In Section 2.4, we describe the methods that will be used to obtain kernelization lower bounds, and the complexity-theoretic assumption under which we are able to obtain these lower bounds. In Sections 2.5 and 2.6, we describe notation and definitions relating to linear algebra, matrices, and polynomials. In Section 2.7, we formally introduce Constraint Satisfaction Problems, and provide some known results. Finally, in Section 2.8 we introduce graph coloring, graph homomorphism, and the relation between graph homomorphism and Constraint Satisfaction Problems.

2.1 General notation

For a positive integer n, we define $[n] := \{1, 2, \ldots, n\}$. Let $\mathbb{Z} := \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$ be the set of all integers. We use $\mathbb{N} := \{0, 1, 2, 3, \ldots\}$ to indicate the non-negative integers. We use $\mathbb{N}_+ := \{1, 2, 3, \ldots\}$ to denote the set of all positive integers.

For a set S we use the notation $\binom{S}{k} := \{S' \subseteq S \mid |S'| = k\}$ to denote the set of all size-k subsets of S. We use the notation $S^k := \{(s_1, \ldots, s_k) \mid s_1, \ldots, s_k \in S\}$

to denote the set of all k-tuples with elements from S. In particular, $[n]^2$ denotes all 2-tuples of elements from $[n]$.

We use the notation $\widetilde{\mathcal{O}}$ to suppress polylogarithmic factors, thus $\widetilde{\mathcal{O}}(f(n)) = \mathcal{O}(f(n) \cdot \log^c n)$ for some constant c.

2.2 Graphs

All graphs considered in this thesis are finite, simple, and undirected, unless explicitly mentioned otherwise. A graph G has vertex set $V(G)$ and edge set $E(G)$. An edge is a nonempty subset of $V(G)$ of size at most two; for non-simple graphs the singleton sets correspond to self-loops (although we may sometimes denote a self-loop on vertex v by edge $\{v,v\}$ for convenience). In a simple graph, all edges have size two. For a simple graph G, let $\bar{G}$ denote the *complement* of G, such that $V(\bar{G}) := V(G)$ and $E(\bar{G}) := \{\{u,v\} \mid u,v \in V(G), \{u,v\} \notin E(G)\}$.

Definition 2.1 (Neighborhood) Let G be a (not necessarily simple) undirected graph. For a vertex $v \in V(G)$, let $N_G(v) := \{u \mid \{u,v\} \in E(G)\}$ denote its *open neighborhood*. These are simply all vertices that are connected to v by an edge. A vertex v is contained in its own open neighborhood, if and only if there is a self-loop at v. We let $N_G[v] := N_G(v) \cup \{v\}$ be its *closed neighborhood*.

Similarly, for a set of vertices $S \subseteq V(G)$, we denote its closed neighborhood by $N_G[S] := \bigcup_{v \in S} N_G[v]$. In a simple graph G, its open neighborhood is given by $N_G(S) := N_G[S] \setminus S$. In general, the open neighborhood of S is given by $N_G(S) := (N_G[S] \setminus S) \cup \{s \mid s \in S, \{s\} \in E(G)\}$. We may omit the subscript G when it is clear from context.

For a vertex $v \in V(G)$ in a graph G, we use the notation $d_G(v)$ to denote its *degree* in G. If G is a simple graph, then $d_G(v) := |N_G(v)|$. If G has self-loops, $d_G(v) := |N_G(v)| + 1$ if $\{v\} \in E(G)$ and $d_G(v) := |N_G(v)|$ otherwise (such that a vertex with only a self-loop has degree 2). We use $\Delta(G) := \max_{v \in V(G)} d_G(v)$ to denote the maximum degree of any vertex in G.

For a graph G and a set $S \subseteq V(G)$, we say S is a *clique* in G if $\{u,v\} \in E(G)$ for all $u,v \in S$ with $u \neq v$. We use $\omega(G)$ to denote the size of the largest clique in G, known as the *clique number* of G. We say a set $S \subseteq V(G)$ is an *independent set* if for all $s,s' \in S$, it holds that $\{s,s'\} \notin E(G)$. Observe that for a simple graph G, the set S is an independent set in G if and only if S is a clique in $\bar{G}$.

Let G be a graph and let $S \subseteq V(G)$. We say S is a *vertex cover* of G if for every edge $e \in E(G)$ at least one of its endpoints is in S, thus $e \cap S \neq \emptyset$. From this definition it is easy to see that S is a vertex cover in a graph G, if and only if $V(G) \setminus S$ is an independent set.

A set $S \subseteq V(G)$ is a *dominating set* in G if for every vertex $v \notin S$, at least one of its neighbors is in S, meaning $N_G[v] \cap S \neq \emptyset$.

A graph H is a *subgraph* of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. We say H is a *spanning subgraph* of G if H is a subgraph of G such that $V(H) = V(G)$. We sometimes also say that a graph H is a (spanning) subgraph of G if H is isomorphic to a subgraph G' of G.

Let S be a subset of the vertices of G. We use $G[S]$ to denote the subgraph of G *induced by* S, thus $G[S]$ is the graph with vertex set S and edge set $\{\{s,s'\} \mid \{s,s'\} \in E(G) \wedge s,s' \in S\}$. For a subset of the vertices S we use the notation $G - S$ do denote the graph $G[V(G) \setminus S]$, which is the graph G after removing all vertices in S and their incident edges.

Definition 2.2 (Paths, walks, cycles) For a (not necessarily simple) graph G, a *walk* of length k in G is a sequence of vertices $(x_0, x_1, \ldots, x_k)$ such that $\{x_{i-1}, x_i\} \in E(G)$ for all $i \in [k]$. A walk is *closed* if $x_0 = x_k$. A *path* is a walk on which all vertices are distinct. A *cycle* is a closed walk $(x_0, x_1, \ldots, x_k)$ where all vertices are distinct except that $x_0 = x_k$[6]. For vertices $u, v \in V(G)$, a path or walk of length k from u to v is a path or walk for which $x_0 = u$ and $x_k = v$.

A path or cycle is *odd* if its length is odd. Note that a self-loop is an odd cycle (of length 1).

Definition 2.3 (Girth) The *(odd) girth* of a (not necessarily simple) graph G is the length of its shortest (odd) cycle, or $+\infty$ if no such cycle exists.

For an integer $\alpha \geq 1$, we use the notation C_α for the cycle on α vertices, such that $V(C_\alpha) := [\alpha]$, and $E(C_\alpha) := \{\{x, x+1\} \mid x \in [\alpha - 1]\} \cup \{\alpha, 1\}$. We let K_α be the complete graph (clique) on α vertices, such that $V(K_\alpha) := [\alpha]$ and $E(K_\alpha) := \{\{x,y\} \mid x,y \in [\alpha], x \neq y\}$.

We say a graph G is *bipartite* if its vertex set can be partitioned into sets S and T, such that S and T are independent sets in G, meaning the only edges in G connect a vertex in S to a vertex in T. Observe that a graph is bipartite, if and only if it is 2-colorable. We use $K_{\alpha,\beta}$ to denote the complete bipartite graph whose partite sets have size α and β, such that $V(K_{\alpha,\beta}) := S \cup T$ with $S := \{s_1, \ldots, s_\alpha\}$ and $T := \{t_1, \ldots, t_\beta\}$. Furthermore, $E(K_{\alpha,\beta}) := \{\{s,t\} \mid s \in S, t \in T\}$.

2.3 Parameterized complexity and kernelization

Let Σ be a fixed finite alphabet. Define Σ^* as the set of all finite strings with elements from Σ (including the empty string). A *parameterized problem* is a language $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$. This means that such a problem has input instances (x,k), where the second component is called the *parameter*. The parameter is commonly referred to by the letter k. In this thesis, we will sometimes deviate from this convention. When the chosen parameter corresponds to the number

[6] By this definition, paths and cycles are necessarily *simple* paths and cycles.

of variables in a logic problem, or the number of vertices in a graph problem, we will regularly denote it by n.

Now that we have defined parameterized problems, we can formalize the definition of kernelization that was presented in the introduction of this thesis.

Definition 2.4 ((Generalized) kernel) Let $\mathcal{Q}, \mathcal{Q}' \subseteq \Sigma^* \times \mathbb{N}$ be parameterized problems and let $h\colon \mathbb{N} \to \mathbb{N}$ be a computable function. A *generalized kernel for $\mathcal{Q}$ into $\mathcal{Q}'$ of size $h(k)$* is an algorithm that, on input $(x,k) \in \Sigma^* \times \mathbb{N}$, takes time polynomial in $|x| + k$ and outputs an instance (x',k') such that:

1. $|x'|$ and k' are bounded by $h(k)$, and
2. $(x',k') \in \mathcal{Q}'$ if and only if $(x,k) \in \mathcal{Q}$.

The algorithm is a *kernel* for $\mathcal{Q}$ if $\mathcal{Q}' = \mathcal{Q}$. It is a *polynomial (generalized) kernel* if $h(k)$ is a polynomial. We use the words *kernel* and *kernelization (algorithm)* interchangeably.

Observe that, for a given parameterized problem, it is entirely possible that no kernelization exists. Intuitively, this happens when the parameter does not capture the complexity of the problem very well. As an extreme example, choosing a parameter that is constant means that a kernel only exists if the problem is solvable in polynomial time. In this thesis we will study parameterized problems that do have kernelization algorithms, and investigate what the smallest kernel is for these problems.

One of the other questions that parameterized complexity deals with is whether a parameterized problem can be solved in time polynomial in $|x|$ (but possibly exponential in the parameter). More formally:

Definition 2.5 (FPT) We say that a parameterized problem $\mathcal{Q}$ is *Fixed-Parameter Tractable (FPT)* if there is a computable function f and an algorithm to determine for any $(x,k) \in \Sigma^* \times \mathbb{N}$ if $(x,k) \in \mathcal{Q}$ in time $f(k) \cdot |x|^c$ for a constant c.

It turns out that the problems that admit FPT algorithms, and those that have kernels, coincide.

▶ **Theorem 2.6** ([13, Theorem 1]) *Let $\mathcal{Q}$ be a parameterized problem. Then $\mathcal{Q}$ is FPT if and only if $\mathcal{Q}$ is decidable and has a kernel.* ◀

By this result, showing that a problem is (not) FPT can be used to show that it does (not) have a kernel.

2.4 Kernelization lower bound framework

Now that we have seen the definition of a kernel, we will look at the methods to prove kernelization lower bounds. We start by describing the necessary complexity-theoretic assumptions.

2.4.1 Complexity-theoretic assumptions

The kernelization lower bounds obtained in this thesis, are proven under complexity-theoretic assumptions. This is necessary, since the problems we consider are all contained in NP. As such, P = NP would imply that all considered problems have a kernel of size $\mathcal{O}(1)$, obtained by solving the problem in polynomial time, and then outputting a constant-size *yes*- or *no*-instance, depending on the answer.

Thereby, we will have to assume P $\neq$ NP to prove meaningful lower bounds. However, kernelization lower bounds are generally only obtained under an even stronger assumption, namely NP $\not\subseteq$ coNP/poly. We assume the reader is familiar with the complexity theory relating to the classes P, NP, and coNP. In the remainder of this section we will give an informal description of the meaning of the assumption that NP is not contained in coNP/poly.

The class coNP/poly is the class of problems that can be co-nondeterministically decided by a Turing machine M, in polynomial time, when given polynomial advice. Machine M co-nondeterministically decides a problem, if for a *yes*-instance, all computation paths lead to an *accept*. When given a no-instance, there must be at least one computation path leading to a *reject*.

This polynomial advice means that M may have an additional, read-only, input tape. An input to this machine is now a "normal" input x, together with an input $f(|x|)$ on the advice tape of length polynomial in the length of the input. Observe that this advice may only depend on the *length* of the given input x.

Alternatively, the class coNP/poly can be seen as those problems for which there is an infinite sequence of algorithms $A_1, A_2, \ldots$, one for every input length n, such that A_n co-nondeterministically decides the inputs of size exactly n in time $\mathcal{O}(n^c)$ for some constant c independent of n, and additionally the description size of each algorithm A_n is bounded by $\mathcal{O}(n^c)$.

Clearly, coNP $\subseteq$ coNP/poly, but it is known that coNP is in fact a strict subset of coNP/poly. The argument for this is not very complicated: We can show that coNP/poly contains undecidable problems, while it is obvious that coNP does not. In fact, even P/poly contains problems that are not decidable. Consider for example the undecidable HALTING PROBLEM. The problem is, given a Turing machine, to decide whether the given Turing machine will halt (within a finite number of steps) when given an empty input string. Suppose we encode the inputs to the halting problem in unary (thus, by only 1's), thereby giving each possible input instance a unique length. This can be done, by fixing an enumeration of all possible Turing machines. Then if an input for the HALTING PROBLEM corresponds to the i'th Turing machine, we can encode i in unary.

Even though the problem is now encoded in unary, the HALTING PROBLEM remains undecidable, but a Turing machine with polynomial advice can now trivially solve this version of the HALTING PROBLEM in constant time. The advice string can simply consist of a single bit, that indicates whether the unique

instance of length n is a *yes*-instance.

Note however, that the above should not be taken as evidence that perhaps $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ holds. While it indeed shows that coNP/poly is in some sense significantly larger than coNP, this is mostly the case due to a very specific input encoding. In fact, there are at least two different reasons to believe $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$.

First (and perhaps foremost), if $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ would hold, this has implications for the *polynomial hierarchy*. The polynomial hierarchy is a hierarchy of complexity classes, all contained in PSPACE that starts with P at the lowest level. The level above contains NP (and also coNP). The hierarchy continues with infinitely more levels. It is not only believed that $\mathsf{P} \subsetneq \mathsf{NP}$, but also that every other level is a strict subset of the levels above it. Just like proving $\mathsf{P} \neq \mathsf{NP}$, proving this remains an open question. However, if $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ holds, the entire hierarchy would collapse to the third level [91, Theorem 2], or in other words, $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ implies $\mathsf{PH} = \Sigma_3^p$.

Secondly, the complexity classes NP and coNP are believed to be incomparable due to the way they are defined. For problems in NP it is easy to verify a *yes*-instance, when given a (small) certificate, but potentially difficult to verify a *no*-instance. For problems in coNP however the opposite is true; for problems in coNP verifying *no*-instances is easy. This is one of the reasons that NP and coNP are believed to be distinct. A polynomial amount of advice that is only allowed to depend on the length of the given input, is not expected to change this situation, because over a binary alphabet there are exponentially many inputs of the same size.

2.4.2 Linear-parameter transformations

The most straightforward way to obtain kernelization lower bounds, is by giving an appropriately bounded transformation that starts from a problem for which a kernelization lower bound is known. We start by defining this type of transformation.

Definition 2.7 (Linear-parameter transformation) Let $\mathcal{Q}, \mathcal{Q}' \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. A *linear-parameter transformation* from $\mathcal{Q}$ to $\mathcal{Q}'$ is a polynomial-time algorithm that, given an instance $(x,k) \in \Sigma^* \times \mathbb{N}$ of $\mathcal{Q}$, outputs an instance $(x',k') \in \Sigma^* \times \mathbb{N}$ of $\mathcal{Q}'$ such that the following holds:

1. $(x,k) \in \mathcal{Q}$ if and only if $(x',k') \in \mathcal{Q}'$, and
2. $k' = \mathcal{O}(k)$.

We denote the existence of such a transformation by $\mathcal{Q} \leq_{\text{lpt}} \mathcal{Q}'$.

The next theorem shows how linear-parameter transformations give kernelization lower bounds. In particular, the contraposition of the theorem states that if it is known that $\mathcal{Q}'$ does not admit a generalized kernel of size $\mathcal{O}(k^d)$ for

some d (possibly under additional assumptions), then $\mathcal{Q}$ does not admit such a kernel either (under the same assumptions).

▶ **Theorem 2.8** *Let $\mathcal{Q}$ and $\mathcal{Q}'$ be parameterized problems with $\mathcal{Q} \leq_{lpt} \mathcal{Q}'$. If $\mathcal{Q}'$ admits a generalized kernel of size $\mathcal{O}(k^d)$, then $\mathcal{Q}$ admits a generalized kernel of size $\mathcal{O}(k^d)$.*

Proof. This result has been used several times [16, 18], we give a short proof here for completeness. Suppose $\mathcal{Q}'$ admits a generalized kernel of size $\mathcal{O}(k^d)$, we show how to obtain a generalized kernel of the same size for $\mathcal{Q}$. Let (x,k) be an instance for $\mathcal{Q}$. Use the linear-parameter transformation from $\mathcal{Q}$ to $\mathcal{Q}'$ to obtain an instance (x',k') for $\mathcal{Q}'$ such that $k' = \mathcal{O}(k)$ and $(x,k) \in \mathcal{Q}$ if and only if $(x',k') \in \mathcal{Q}'$. Apply the assumed generalized kernel for $\mathcal{Q}'$ to obtain an instance (x'',k'') for some problem $\mathcal{Q}''$ such that $(x',k') \in \mathcal{Q}'$ if and only if $(x'',k'') \in \mathcal{Q}''$, $|x''| = \mathcal{O}((k')^d)$ and $k'' = \mathcal{O}((k')^d)$.

Clearly hereby $(x'',k'') \in \mathcal{Q}''$ if and only if $(x,k) \in \mathcal{Q}$ and $|x''| = \mathcal{O}((k')^d) = \mathcal{O}(k^d)$ and $k'' = \mathcal{O}(k^d)$. As such, this gives a generalized kernel for $\mathcal{Q}$ of size $\mathcal{O}(k^d)$. ◀

As a starting point for linear-parameter transformations, we will regularly use the following kernelization lower bound for d-CNF-SAT, that was proven by Dell and Van Melkebeek [33]. Let us start by formally defining d-CNF-SAT. We say a formula $\mathcal{F}$ is in d-CNF if it is a conjunction of clauses, where every *clause* is a disjunction of d literals. A *literal* is simply a variable or its negation.

d-CNF-SAT

Input: A d-CNF formula $\mathcal{F}$ on variables $x_1, \ldots, x_n$.
Parameter: The number of variables n.
Question: Does there exist an assignment $\tau\colon \{x_1, \ldots, x_n\} \to \{0,1\}$ satisfying $\mathcal{F}$? Here $\mathcal{F}$ is satisfied if every clause contains at least one literal that evaluates to 1.

▶ **Theorem 2.9** ([33, Theorem 1]) *Let $d \geq 3$ be an integer. Then d-CNF-SAT parameterized by the number of variables n does not have a generalized kernel of size $\mathcal{O}(n^{d-\varepsilon})$ for any $\varepsilon > 0$, unless* NP $\subseteq$ coNP/poly. ◀

Another problem we will use as the starting problem for linear-parameter transformations is the VERTEX COVER problem defined below.

VERTEX COVER

Input: An undirected graph G and an integer k.
Parameter: The number of vertices n.
Question: Does G have a vertex cover of size at most k?

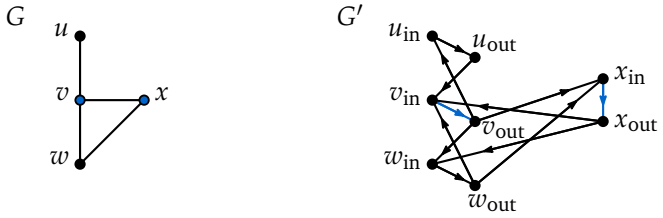


Figure 2.1 Linear-parameter transformation from a VERTEX COVER instance (left) to a FEEDBACK ARC SET instance (right). A minimum vertex cover (respectively, feedback arc set) is indicated in blue.

▶ **Theorem 2.10** ([33, Theorem 2]) VERTEX COVER *parameterized by the number of vertices n does not have a generalized kernel of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless* NP $\subseteq$ coNP/poly. ◀

From the definition of a linear-parameter transformation, it is clear that on some occasions, kernelization lower bounds for a certain NP-hard problem $\mathcal{Q}$ may follow directly from its NP-hardness proof. This is the case if an appropriate lower bound is known for the starting problem for the reduction, and furthermore the parameter only increases linearly. Below, we describe an example where this is indeed the case.

▶ **Example 2.11** Consider the following graph problem.

FEEDBACK ARC SET

Input: A directed graph G on n vertices and an integer k.
Parameter: The number of vertices n.
Question: Is is possible to remove at most k arcs from G, such that the resulting graph is acyclic?

We can show by a linear-parameter transformation from VERTEX COVER, that FEEDBACK ARC SET parameterized by the number of vertices n does not have a generalized kernel of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless NP $\subseteq$ coNP/poly.

The linear-parameter transformation will correspond to the many-one reduction given by Karp [65], that is used to prove the NP-hardness of FEEDBACK ARC SET. Let us start by repeating the construction here for completeness.

Let an instance (G,k) for VERTEX COVER be given, we create an instance (G',k') for FEEDBACK ARC SET as follows. Let $k' := k$. Let $V(G') := \{v_{\text{in}}, v_{\text{out}} \mid v \in V(G)\}$. For every edge $\{u,v\} \in E(G)$, add the arcs $(u_{\text{out}}, v_{\text{in}})$ and $(v_{\text{out}}, u_{\text{in}})$ to G'. For every vertex $v \in V(G)$, add the arc $(v_{\text{in}}, v_{\text{out}})$ to G'. An illustration of this construction can be found in Figure 2.1.

It is straightforward to show that G' has a feedback arc set of size k', if and only if G has a vertex cover of size k, we leave this as an exercise to the reader. Furthermore, we can observe that $|V(G')| = 2|V(G)| = \mathcal{O}(n)$, and thereby we

have given a linear-parameter transformation from VERTEX COVER to FEEDBACK ARC SET parameterized by the number of vertices n. The result follows from Theorems 2.8 and 2.10. ◀

2.4.3 Cross-compositions

Another way to obtain a kernelization lower bound for a parameterized problem $\mathcal{Q}$, is by giving a cross-composition from an NP-hard problem $\mathcal{P}$ to $\mathcal{Q}$. There are two types of cross-compositions, namely AND-cross-compositions and OR-cross-compositions. All kernelization lower bounds in this thesis will be obtained via OR-cross-compositions, and we will often omit the prefix "OR". We will introduce the notion of cross-compositions in this section.

Informally speaking, a cross-composition is a polynomial-time algorithm that is given a t instances for $\mathcal{P}$, for some large number t. It outputs a single instance for $\mathcal{Q}$ such that this instance acts as the logical OR of the given input instances. Furthermore, the parameter value of this instance should be appropriately bounded.

To formally define cross-compositions, we first need some additional definitions.

Definition 2.12 (Polynomial equivalence relation, [16, Def. 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 set $S \subseteq \Sigma^*$ the equivalence relation $\mathcal{R}$ partitions the elements of S into a number of classes that is polynomially bounded in the size of the largest element of S.

As mentioned, the input for a cross-composition consists of t instances of a chosen starting problem. It is allowed to assume that these instances are equivalent under a polynomial equivalence relation of our choice. We can use this to ensure that the inputs given for a cross-composition are somewhat similar. For example, for a graph problem, it can be used to ensure that all inputs have the same number of vertices, by letting input instances with the same number of vertices be equivalent. If the inputs have a vertex set that is partitioned into for example red vertices and blue vertices, one can even use a polynomial equivalence relation to ensure that inputs have the same number of red (and blue) vertices. A polynomial equivalence relation can also in many cases be used to ensure that the inputs all ask for a solution of the same size.

Definition 2.13 (OR-cross-composition, [16, Def. 3.3]) Let $L \subseteq \Sigma^*$ be a language, let $\mathcal{R}$ be a polynomial equivalence relation on Σ^*, let $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem, and let $f\colon \mathbb{N} \to \mathbb{N}$ be a function. An *OR-cross-composition of L into $\mathcal{Q}$* (with respect to $\mathcal{R}$) *of cost $f(t)$* is an algorithm that, given t

instances $x_1, x_2, \ldots, x_t \in \Sigma^*$ of L belonging to the same equivalence class of $\mathcal{R}$, takes time polynomial in $\sum_{i=1}^{t} |x_i|$ and outputs an instance $(y,k) \in \Sigma^* \times \mathbb{N}$ such that:

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

The lower bound result obtained by a cross-composition depends on the cost of the given cross-composition. It is known that cross-compositions of cost $t^{o(1)}$ can be used to rule out the existence of polynomial kernels [16, Corollary 3.9]. In this thesis however we focus on giving tight kernelization lower bounds for problems that do have a polynomial kernel. We will use the following result.

▶ **Theorem 2.14** ([16, Theorem 3.8]) *Let $L \subseteq \Sigma^*$ be a language, let $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem, and let d, ε be positive reals. If L is NP-hard under Karp reductions, has an* OR*-cross-composition into $\mathcal{Q}$ with cost $f(t) = t^{1/d+o(1)}$, where t denotes the number of instances, and $\mathcal{Q}$ has a polynomial (generalized) kernelization with size bound $\mathcal{O}(k^{d-\varepsilon})$, then* NP $\subseteq$ coNP/poly. ◀

For $d \in \mathbb{N}$ we will refer to an OR-cross-composition of cost $f(t) = t^{1/d+o(1)}$ as a *degree-d cross-composition*. By Theorem 2.14, a degree-d cross-composition can be used to rule out generalized kernels of size $\mathcal{O}(k^{d-\varepsilon})$. Note that when studying sparsification, we use the number of vertices or variables in the instance (which is usually denoted by n) as the parameter value.

Let us describe the "standard" approach to giving a degree-2 OR-cross-composition, when dealing with a graph problem $\mathcal{Q}$ parameterized by the number of vertices n. This approach is based on the table-based approach introduced by Dell and Marx [32].

As an input for the cross-composition, we are given t instances of a suitably chosen input problem $\mathcal{P}$. It will turn out that for the construction of a degree-2 cross-composition, it is often convenient to assume that $\sqrt{t}$ is an integer, and it is sometimes even desirable that $\log t$ and $\log \sqrt{t}$ are integers. We can easily ensure this, by duplicating one of the input instances until we have $t' \geq t$ input instances of $\mathcal{P}$, with t' an integer such that $\log \sqrt{t'}$ is integer. Observe that, in particular, for any $c \in \mathbb{N}$ we have that 2^{2c} satisfies this condition, since $\sqrt{2^{2c}} = 2^c$ and $\log \sqrt{2^{2c}} = c$. As such, there exists an appropriate t' with $t' \leq 4t$: Let t'' be the smallest power of two that is larger than t, such that $t'' = 2^d$. Observe that $t'' \leq 2t$. If d is even, then $t' := t''$ satisfies our requirements. If however d is odd, then $t' := 2t'' = 2^{d+1}$ satisfies the requirements. In both cases, $t' \leq 2t'' \leq 4t$.

To select a suitable NP-hard starting problem $\mathcal{P}$, it can be helpful to consider a restricted variant of the goal problem $\mathcal{Q}$. Often, it is convenient to have a starting problem that is defined on a restricted set of graphs, for which the

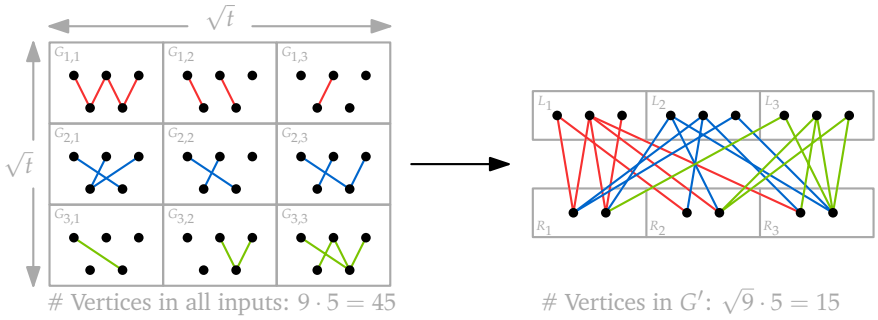


Figure 2.2 Basic structure of a degree-2 cross-composition. Inputs for a problem $\mathcal{P}$ defined on bipartite graphs are depicted on the left. The result of applying the table-like structure for cross-compositions is depicted on the right. The figure is adapted from an earlier publication [81].

vertices can be partitioned into two sets, say L and R, such that $G[L]$ and $G[R]$ always have a known structure. An example would be bipartite graphs, where $G[L]$ and $G[R]$ are always independent sets, but one can also think of cases where $G[L]$ is for example a path, or a union of disjoint triangles, etcetera.

Next, one may define a polynomial equivalence relation that lets inputs be equivalent if the sizes of L and R are the same. So, suppose we are given t instances of $\mathcal{P}$ with $|L| = m$ and $|R| = n$. We now need to encode these instances into one single instance of $\mathcal{Q}$, whose number of vertices can be approximately $\mathcal{O}(\sqrt{t} \cdot \text{poly}(n+m))$. The challenge is now that we do not have the budget to copy every vertex of every input instance. We show how to resolve this, using a table-like structure.

As mentioned, we can assume that $\sqrt{t}$ is integer. As such, we can label the t input graphs as $G_{i,j}$ for $i, j \in [\sqrt{t}]$. By the choice of our equivalence relation and input problem, all $G_{i,j}$ have partite sets $L_{i,j}$ and $R_{i,j}$ and all $G_{i,j}[L_{i,j}]$ are isomorphic. The same holds for $G_{i,j}[R_{i,j}]$. We now show how to create an instance G' for $\mathcal{Q}$. Start by creating $\sqrt{t}$ copies of $G_{1,1}[L_{1,1}]$ and label them L_i for $i \in [\sqrt{t}]$. Similarly, create $\sqrt{t}$ copies of $G_{1,1}[R_{1,1}]$ labeled R_j for $j \in [\sqrt{t}]$. Now, we can represent the edges of all input instances, by connecting vertices in L_i to R_j in such a way, that $G'[L_i \cup R_j]$ is isomorphic to graph $G_{i,j}$. Observe that G' at this point has $\sqrt{t} \cdot (n+m)$ vertices, which is appropriately bounded for a degree-2 cross-composition. Refer to Figure 2.2 for an example.

The created graph G' now represents all input instances. It remains to ensure that G' is a *yes*-instance for $\mathcal{Q}$ if and only if there exist i and j such that $G_{i,j}$ is a *yes*-instance for $\mathcal{P}$. This is generally done by adding additional gadgets, that select one index i and one index j. For example, one could add gadgets such that the solution in all-but-one L_i is "for free", and add similar gadgets for R_j. Of course, the actual design of these gadgets and their precise working is

highly problem-dependent.

As mentioned earlier, apart from OR-cross-compositions, there also exist AND-cross-compositions. These are defined similarly except that the resulting instance should be a logical AND of the given input instances. It follows from a result by Drucker [35] that they result in the same kernelization lower bounds as OR-cross-compositions.

2.5 (Linear) algebra

For an integer q, we let $\mathbb{Z}/q\mathbb{Z}$ denote the integers modulo q. These form a field if q is prime, and a ring otherwise. We will use $x \equiv_q y$ to denote that x and y are congruent modulo q, and $x \not\equiv_q y$ to denote that they are incongruent modulo q.

We denote the greatest common divisor of two integers x and y as $\gcd(x,y)$ and their least common multiple as $\operatorname{lcm}(x,y)$. The two concepts are closely related, since $\operatorname{lcm}(x,y) = |x \cdot y| / \gcd(x,y)$ for x or y nonzero.

▶ **Theorem 2.15** (Bézout's identity) *Let $x,y,z \in \mathbb{Z}$ with $gcd(x,y) = z$. There exist integers a and b such that $ax + by = z$.* ◀

We will use $x \mid y$ to indicate that x divides y (over the integers) and $x \nmid y$ to indicate that it does not.

For definitions and basic properties of rings, fields, vector spaces, and related concepts, we refer the reader to [72]. We will recall the most relevant definitions here, and introduce some notation.

Definition 2.16 (Span) Given a set $S = \{\mathbf{s_1}, \ldots, \mathbf{s_n}\}$ of k-ary vectors over a ring $\mathcal{R}$, we define $\operatorname{span}_{\mathcal{R}}(S)$ as the set of all vectors $\mathbf{y}$ in $\mathcal{R}^k$ for which there exist $\alpha_1, \ldots, \alpha_n \in \mathcal{R}$ such that $\mathbf{y} = \sum_{i \in [n]} \alpha_i \mathbf{s_i}$. We may omit the subscript indicating the relevant ring, if it is clear from the context.

For a positive integer q, we use $\operatorname{span}_q(S)$ as a shorthand for $\operatorname{span}_{\mathbb{Z}/q\mathbb{Z}}(S)$.

For an $m \times n$ matrix A, we commonly use $\mathbf{a_i}$ for $i \in [m]$ to denote the i'th row of A and $a_{i,j}$ to denote the element in row i and column j.

Definition 2.17 (Row space) Let A be an $m \times n$ matrix over a ring $\mathcal{R}$. Then the *row space* of A is defined as the span of the set of rows of A, or equivalently, as $\operatorname{span}_{\mathcal{R}}(\{\mathbf{a_1}, \ldots, \mathbf{a_m}\})$.

Definition 2.18 (Basis) A *basis* of a vector space V is defined as a set $B \subseteq V$ such that $\operatorname{span}(B) = V$ and the vectors in B are linearly independent.

It is well-known that for a vector space, its *dimension* is defined as the size of a basis of this vector space, which is well-defined since any basis of a vector space has the same size. Furthermore, for a matrix over a field, it is known that the dimension of its row space equals the dimension of its column space. Hereby, the dimension of the row space of such a matrix, is bounded by its number

of columns (and vice versa), which is useful information when considering matrices with $m \gg n$.

Observe that when considering matrices over a ring, the situation is more complex. Consider for example the 2×1 matrix A over the integers modulo 6, given by

$$A := \begin{pmatrix} 2 \\ 3 \end{pmatrix}.$$

Now, since A has only one column, you might hope to find a size-1 subset of the rows that spans the entire row span of A. This is however not possible, the vector (1) is in the row space of A, as it is the result of subtracting the first row from the second. It is however not a multiple of either of the two rows. Thus, there is no single row that spans the entire row space of A. This shows an important difference between matrices with elements from a field, and matrices with elements from a ring.

Definition 2.19 (Diagonal matrix) We say an $m \times n$ matrix A is a *diagonal matrix* if all entries $a_{i,j}$ with $i \neq j$ are zero. Thus, all non-zero elements occur on the diagonal. Note that under this definition a matrix can be diagonal even if $m \neq n$.

Definition 2.20 (Unimodular matrix [48, Definition 367]) An $n \times n$ matrix U over $\mathbb{Z}$ is called *unimodular* if its determinant is 1 or -1.

The relevant property of unimodular matrices that we will use is that a unimodular matrix over $\mathbb{Z}$ is invertible over $\mathbb{Z}$. This means that if U is a unimodular matrix over $\mathbb{Z}$, then there exists a matrix U^{-1} over $\mathbb{Z}$ such that $U \cdot U^{-1} = U^{-1} \cdot U = I$ where I is the identity matrix (a diagonal matrix with all-ones on the diagonal). In Chapters 3 and 4 we will use two normal forms of certain types of matrices in our proofs, namely the Smith Normal Form and the Howell Normal Form. We introduce them below.

Definition 2.21 (Smith normal form [48, Theorem 368]) Let A be an $m \times n$ matrix over the integers. There exist unimodular matrices U and V over the integers, such that U is an $m \times m$ matrix and V is an $n \times n$ matrix, and a diagonal $m \times n$ matrix S over $\mathbb{Z}$ such that

$$A = USV,$$

the diagonal entries of S are $d_1, d_2, \ldots, d_r, 0, \ldots, 0$, each d_i a positive integer, and $d_i | d_{i+1}$ for $i \in [r-1]$. We will call U, S, and V the *Smith decomposition* of A. Finally, S is called the *Smith Normal Form* of A.

The Smith normal form can be useful when solving systems of linear equations over the integers. In particular, when one wants to solve $A\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$, one can alternatively solve for $S\mathbf{x}' = U^{-1}\mathbf{b}$. Observe that this system is easier to solve since S is a diagonal matrix. If this gives a solution $\mathbf{x}'$, then $\mathbf{x} := V^{-1}\mathbf{x}'$ is a solution for $A\mathbf{x} = \mathbf{b}$, as is easily verified:

$$A\mathbf{x} = AV^{-1}\mathbf{x}' = USVV^{-1}\mathbf{x}' = U(S\mathbf{x}') = U(U^{-1}\mathbf{b}) = \mathbf{b}.$$

We will see this in more detail in some of the proofs in Chapter 4.

► **Example 2.22** Consider the matrix

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 1 \\ 4 & 0 & 1 \\ 2 & 4 & 4 \end{pmatrix},$$

then its Smith decomposition looks as follows:

$$U = \begin{pmatrix} 1 & 2 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 4 & -3 & -1 & 0 \\ 2 & 2 & 1 & 1 \end{pmatrix}, \quad S = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 16 \\ 0 & 0 & 0 \end{pmatrix}, \quad V = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 8 & 1 \\ 0 & -1 & 0 \end{pmatrix}.$$

One may verify that U and V are indeed invertible with

$$U^{-1} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 5 & -1 & 0 \\ 3 & -11 & 2 & 0 \\ -1 & -1 & 0 & 1 \end{pmatrix}, \text{ and } V^{-1} = \begin{pmatrix} 1 & -1 & -6 \\ 0 & 0 & -1 \\ 0 & 1 & 8 \end{pmatrix}.$$

◄

► **Theorem 2.23** ([64, Theorem 4]) *The Smith decomposition of a matrix can be computed in polynomial time.* ◄

We will furthermore use the Howell form of a matrix, which was first introduced by Howell [51]. The following definition is based on [87].

Definition 2.24 (Howell form) Let $q \in \mathbb{N}$ be an integer and let A be an $m \times n$ matrix over $\mathbb{Z}/q\mathbb{Z}$. Then there exists an invertible $n \times n$ matrix U over $\mathbb{Z}/q\mathbb{Z}$, and an $m \times n$ matrix H over $\mathbb{Z}/q\mathbb{Z}$, such that $A = UH$ and the following statements hold.

- Let r be the number of non-zero rows of H. Then the first r rows of H are non-zero.
- For $i \in [r]$, let h_{i,j_i} be the first nonzero entry in row i. Then $j_1 < j_2 < \ldots < j_r$.
- $h_{i,j_i} | q$ for all $i \in [r]$.
- For $1 \leq k < i \leq r$, we have $0 \leq h_{k,j_i} < h_{i,j_i}$.
- When $\mathbf{x} \in \text{span}_q(\{\mathbf{h_1}, \ldots, \mathbf{h_m}\})$ has zeros as its first $j_i - 1$ components, then $\mathbf{x} \in \text{span}_q(\{\mathbf{h_i}, \ldots, \mathbf{h_m}\})$.

We say that H is in *Howell form*. Furthermore, we may call H the Howell form of matrix A (which is known to be unique [51]).

For a matrix to be in Howell form, the above five statements must all hold. For our proofs however, we are mostly interested in the first two properties, the remaining properties will not be used. The next theorem shows that the Howell form can be efficiently computed.

► **Theorem 2.25** ([87, §3]) *Given a matrix A over $\mathbb{Z}/q\mathbb{Z}$, we can compute matrices U and H in polynomial time, such that U is invertible over $\mathbb{Z}/q\mathbb{Z}$, $A = UH$, and H is in Howell form.* ◄

2.6 Polynomials

We continue by providing the definitions and results that we will use considering polynomials and polynomial equalities.

A *multivariate polynomial* (over a ring $\mathcal{R}$) is a function $p\colon \mathcal{R}^k \to \mathcal{R}$ of the type

$$p(x_1, \ldots, x_k) = \sum_{(d_1,\ldots,d_k)\in\mathbb{N}^k} \alpha_{d_1,\ldots,d_k} \prod_{i\in[k]} (x_i)^{d_i}$$

with coefficients $\alpha_{d_1,\ldots,d_k} \in \mathcal{R}$ that are non-zero for a finite number of choices for $(d_1, \ldots, d_k) \in \mathbb{N}^k$. A term of the type $\prod_{i\in[k]} (x_i)^{d_i}$ is referred to as a *monomial*. The *degree* of a monomial is simply $\sum_{i\in[k]} d_i$. The degree of a polynomial is the maximum of the degrees of all monomials in this polynomial that have a non-zero coefficient.

We say that a monomial is *multilinear* if for all $i \in [k]$, we have $d_i \in \{0, 1\}$. In other words, a monomial is multilinear if it corresponds to a multiplication of distinct variables. A polynomial is multilinear if all its monomials that have a non-zero coefficient are multilinear.

► **Example 2.26** Consider the following polynomial over $\mathbb{Q}$:

$$p(x, y, z) := 5 \cdot x^2 y + \frac{1}{2} \cdot xyz^3 - x + 3y.$$

This is a polynomial of degree 5 (achieved by the second monomial), that is not multilinear, since its first two monomials are not multilinear. ◄

We will often need to bound the number of (multilinear) monomials of a degree-d polynomial. We use the following lemma.

► **Lemma 2.27** *There are at most $n^d + 1$ multilinear monomials of degree at most d over a set of n variables.*

Proof. The number of multilinear monomials over n variables of degree at most d is equal to $\sum_{i=0}^{d} \binom{n}{i}$. We will show that $\sum_{i=1}^{d} \binom{n}{i} \leq n^d$. The left side counts all non-empty subsets of $[n]$ of size at most d. Each of these can be mapped to a distinct d-tuple containing numbers from $[n]$, by repeating an arbitrary element. Since there are n^d possible d-tuples, the claim follows. ◄

Observation 2.28 *Let $\mathcal{F}$ be a field, and let $S \subseteq \mathcal{F}$ be a set of size k. Then there is a degree-k polynomial p in one variable, such that $p(x) = 0$ if and only if $x \in S$; and the polynomial defined by*

$$p(x) := \prod_{s \in S}(x - s)$$

satisfies these conditions.

2.7 Constraint satisfaction problems

We continue by formally introducing constraint satisfaction problems, as were (informally) introduced in Chapter 1.

Definition 2.29 ((Boolean) relation) A *relation* over the set D is a subset of D^k; here, k is a natural number called the *arity* of the relation. A relation of arity k may be called a *k-ary relation*. A *Boolean relation* is a relation over $\{0,1\}$. A 2-ary relation is sometimes referred to as a *binary relation*.

When considering the Boolean domain, we will regularly refer to the 0-element with *false*, and to the 1-element with *true*.

Definition 2.30 (k-OR) For each $k \geq 1$, we use k-OR to denote the relation $\{0,1\}^k \setminus \{(0,\ldots,0)\}$.

By the above definition, 2-OR $= \{(1,1),(1,0),(0,1)\}$.

Definition 2.31 (R^+) For a binary relation R, let R^+ denote the transitive closure of R. For $n \in \mathbb{N}$ let R^n be defined as

$$\{(u,v) \mid \exists x_1,\ldots,x_{n+1} : \forall_{i \in [n]}(x_i, x_{i+1}) \in R \wedge u = x_1 \wedge v = x_{n+1}\}.$$

Observe that for any binary relation R over universe D, we have $R^0 = \{(x,x) \mid x \in D\}$.

Definition 2.32 (Constraint language) A *constraint language over D* is a finite set of relations over D; a *Boolean constraint language* is a constraint language over $\{0,1\}$.

For a constraint language Γ over a domain D, we define CSP(Γ) as follows.

CSP(Γ)

Input: A tuple $(\mathcal{C}, V)$, where $\mathcal{C}$ is a finite set of constraints, V is a finite set of variables, and each constraint is a pair $R(x_1,\ldots,x_k)$ for $R \in \Gamma$ and $x_1,\ldots,x_k \in V$.
Parameter: $n = |V|$.
Question: Does there exist a *satisfying assignment*, that is, an assignment $f\colon V \to D$ such that for each constraint $R(x_1,\ldots,x_k) \in \mathcal{C}$ it holds that $(f(x_1),\ldots,f(x_k)) \in R$?

There are three well-known logic problems, that can be seen as constraint satisfaction problems, that we will regularly encounter. First of all, d-CNF-SAT, as introduced previously. Furthermore, d-EXACT SAT and d-NAE-SAT, which are defined below.

d-NAE-SAT

Input: A d-CNF formula $\mathcal{F}$, where every clause has size at most d.
Parameter: The number of variables n.
Question: Does there exist a satisfying assignment to $\mathcal{F}$, such that each clause of $\mathcal{F}$ contains at least one *false* and at least one *true* literal?

d-EXACT SAT

Input: A d-CNF formula $\mathcal{F}$, where every clause has size at most d.
Parameter: The number of variables n.
Question: Does there exist a satisfying assignment to $\mathcal{F}$, such that each clause of $\mathcal{F}$ contains exactly one *true* literal?

We define CNF-SAT, EXACT SAT, and NAE-SAT as in definitions above, except that for these problems there is no bound on the clause length.

▶ **Example 2.33** We can define 3-EXACT SAT as a constraint satisfaction problem as follows. Consider the relations

$$\begin{aligned} R_0 &:= \{(0,0,1),(0,1,0),(1,0,0)\} \\ R_1 &:= \{(1,0,1),(1,1,0),(0,0,0)\} \\ R_2 &:= \{(1,1,1),(1,0,0),(0,1,0)\} \\ R_3 &:= \{(1,1,0),(1,0,1),(0,1,1)\}. \end{aligned}$$

Let $\Gamma := \{R_0, R_1, R_2, R_3\}$, then 3-EXACT SAT corresponds to CSP(Γ), as follows. The satisfying assignments of clauses with no negated variables are given by R_0. Clauses with one negated variable can be rewritten to a constraint using R_1 by swapping the literals such that the negated variable is in the first position. Similarly, clauses with two or three negated literals can be represented using R_2 and R_3 respectively. For example, the 3-EXACT SAT formula

$$(x \vee y \vee z) \wedge (\neg x \vee \neg z \vee w) \wedge (x \vee v \vee \neg w)$$

gives the CSP(Γ) instance $(\mathcal{C}, V)$ with:

$$\mathcal{C} = \{R_0(x,y,z), R_2(x,z,w), R_1(w,x,v)\}, V = \{v,w,x,y,z\}. \quad \blacktriangleleft$$

We define a *Boolean operation* as a mapping from $\{0,1\}^k$ to $\{0,1\}$, where k is said to be the *arity* of the operation. From here, we define a *partial Boolean operation* as a mapping from a subset of $\{0,1\}^k$ to $\{0,1\}$.

Definition 2.34 (Idempotent, self-dual) We say that a (partial) Boolean operation of arity k is *idempotent* if $f(0,\dots,0) = 0$ and $f(1,\dots,1) = 1$. We say the operation is *self-dual* if for all $a_1,\dots,a_k \in \{0,1\}$, when $f(a_1,\dots,a_k)$ is defined, then $f(\neg a_1,\dots,\neg a_k)$ is defined and $f(a_1,\dots,a_k) = \neg f(\neg a_1,\dots,\neg a_k)$.

We now introduce the notion of polymorphisms of a constraint language. In the study of $\mathrm{CSP}(\Gamma)$, it turned out that studying the polymorphisms of Γ is a useful tool in understanding the complexity of $\mathrm{CSP}(\Gamma)$.

Definition 2.35 (Polymorphism) Let f be a (partial) operation of arity k on a domain D, and let $R \subseteq D^n$ be a relation. We say that f is a *polymorphism* of R when for any tuples $\mathbf{t_1} = (t_{1,1},\dots,t_{1,n}),\dots,\mathbf{t_k} = (t_{k,1},\dots,t_{k,n}) \in R$, if all entries of the tuple $(f(t_{1,1},\dots,t_{k,1}),\dots,f(t_{1,n},\dots,t_{k,n}))$ are defined, then this tuple is in R. We say f is a *polymorphism* of a constraint language Γ if f is a polymorphism of each relation in Γ.

We say that a relation R is *preserved* by a (partial) operation f if f is a polymorphism of R, similarly we say a constraint language Γ is preserved by f if f is a polymorphism of Γ.

Let $f\colon D^k \to D$ be a k-ary (partial) operation and let $\mathbf{t_1},\dots,\mathbf{t_k} \in D^n$ be tuples. We use the following notational convention:

$$f(\mathbf{t_1},\dots,\mathbf{t_k}) := (f(t_{1,1},\dots,t_{k,1}),\dots,f(t_{1,n},\dots,t_{k,n})).$$

▶ **Example 2.36** Consider the 2-ary operation $f\colon \{0,1\}^2 \to \{0,1\}$ corresponding to the binary OR, meaning that f is given by $f(a_1,a_2) := a_1 \vee a_2$. Consider the 3-ary Boolean relation

$$R := \{(0,0,1),(0,1,1),(1,0,1),(1,1,0),(1,1,1)\}.$$

We can for example compute that

$$\begin{aligned} f((0,0,1),(1,1,0)) &= (f(0,1),f(0,1),f(1,0)) = (0\vee 1, 0\vee 1, 1\vee 0) \\ &= (1,1,1). \end{aligned}$$

In the example above, we see that for the tuples $(0,0,1),(1,1,0) \in R$, we get $f((0,0,1),(1,1,0)) = (1,1,1) \in R$. We leave it as an exercise to the reader to show that for any two tuples $\mathbf{t_1},\mathbf{t_2} \in R$ it holds that $f(\mathbf{t_1},\mathbf{t_2}) \in R$. It follows that f is a polymorphism of R; or equivalently that R is preserved by f. ◀

For $b \in \{0,1\}$, let $u_b\colon \{0,1\} \to \{0,1\}$ be the unary operation defined by $u_b(0) = u_b(1) = b$; let $\mathsf{major}\colon \{0,1\}^3 \to \{0,1\}$ be the operation defined by $\mathsf{major}(x,y,z) := (x\wedge y)\vee(x\wedge z)\vee(y\wedge z)$, hence, the value of $\mathsf{major}(x,y,z)$ is the Boolean that occurs at the majority of the positions of x,y,z. Let $\mathsf{minor}\colon \{0,1\}^3 \to \{0,1\}$ be the operation defined by $\mathsf{minor}(x,y,z) := x \oplus y \oplus z$, where $\oplus$ denotes exclusive OR. Observe that hereby $\mathsf{minor}(1,1,1) = 1$ and $\mathsf{minor}(0,0,0) = 0$.

The following is a well-known result, it is a rephrasing of the original theorem that was proven by Schaefer [83]. See [25] for a proof; in particular, refer there to the proof of Theorem 3.21.

▶ **Theorem 2.37** *Let Γ be a Boolean constraint language. The problem* $\mathrm{CSP}(\Gamma)$ *is polynomial-time decidable when Γ is preserved by one of the six following operations:* $u_0, u_1, \wedge, \vee$, major, minor. *Otherwise,* $\mathrm{CSP}(\Gamma)$ *is* NP*-complete.* ◀

As such, we will say a Boolean constraint language Γ is *tractable* when it is preserved by one of the operations $u_0, u_1, \wedge, \vee$, major, minor, and *intractable* otherwise.

▶ **Example 2.38** Recall the constraint language used for 3-Exact sat in Example 2.33. We can show that 3-Exact sat is NP-complete using Theorem 2.37 by verifying that $\Gamma = \{R_0, R_1, R_2, R_3\}$ is not preserved by any of the six operations above. We will show this by showing that R_0 is not preserved by these operations.

Since $u_0(1,0,0) = (0,0,0)$ and $(1,0,0) \in R_0$, while $(0,0,0) \notin R_0$, it follows that R_0 is not preserved by u_0. Similarly, since $u_1(1,0,0) = (1,1,1) \notin R_0$, we obtain that R_0 is not preserved by u_1. Furthermore, while $(0,0,1), (0,1,0) \in R_0$, we have $(0 \vee 0, 0 \vee 1, 1 \vee 0) = (0,1,1) \notin R_0$. Similarly, $(0 \wedge 0, 0 \wedge 1, 1 \wedge 0) = (0,0,0) \notin R_0$. Therefore, R_0 is not preserved by operations $\wedge$ and $\vee$. Finally, one may observe that applying the majority operation to all tuples in R_0 results in the tuple $(0,0,0) \notin R_0$, while taking the minority operation results in the tuple $(1,1,1) \notin R_0$. We conclude that Γ is intractable (observe that we may even conclude that the constraint language $\Gamma' := \{R_0\}$ is intractable). ◀

Definition 2.39 (Γ^*) When Γ is a constraint language over D, we use Γ^* to denote the expansion of Γ where each element of D appears as a relation. That is, we define $\Gamma^* := \Gamma \cup \{\{(d)\} \mid d \in D\}$.

Effectively, the added relations in Γ^* make it possible to enforce via a constraint that a variable must be assigned a fixed value by any satisfying assignment.

Definition 2.40 (pp-definability [25]) A relation $T \subseteq D^k$ is *pp-definable* (short for *primitive positive definable*) from a constraint language Γ if for some $m \geq 0$ there exists a finite conjunction $\mathcal{C}$ consisting of constraints over Γ and equalities over variables $\{v_1, \ldots, v_k, x_1, \ldots, x_m\}$ such that

$$T(v_1, \ldots, v_k) \equiv \exists x_1, \ldots, \exists x_m : \mathcal{C}.$$

That is, for each map $f\colon \{v_1, \ldots, v_k\} \to \{0,1\}$, it holds that f can be extended to a satisfying assignment of $\mathcal{C}$ if and only if $(f(v_1), \ldots, f(v_k)) \in T$.

▶ **Example 2.41** Recall the constraint language $\Gamma = \{R_0, R_1, R_2, R_3\}$ used to define 3-Exact sat in Example 2.33. We can show that Γ pp-defines the 3-or relation given by $T = \{(v_1, v_2, v_3) \in \{0,1\}^3 \mid v_1 \vee v_2 \vee v_3\}$. Consider the

v_1	v_2	v_3	x_1	x_2	x_3	x_4
1	1	1	1	0	0	1
1	1	0	1	0	0	0
1	0	1	0	1	0	1
1	0	0	0	1	0	0

v_1	v_2	v_3	x_1	x_2	x_3	x_4
0	1	1	0	0	0	1
0	1	0	0	0	0	0
0	0	1	0	0	1	0
0	0	0	✗	✗	✗	✗

Table 2.1 All assignments to v_1, v_2, and v_3, together with an extension that satisfies Equation (2.1) if $(v_1, v_2, v_3) \in$ 3-OR.

following pp-definition:

$$\mathcal{C} := R_1(v_1, x_1, x_2) \wedge R_0(v_2, x_2, x_3) \wedge R_1(v_3, x_3, x_4), \tag{2.1}$$

over variable set $V = \{v_1, v_2, v_3, x_1, x_2, x_3, x_4\}$. To show that $\exists x_1, \ldots, x_4 : \mathcal{C}$ is a pp-definition of T, we need to show that an assignment to v_1, v_2, v_3 can be extended to an assignment satisfying all constraints in $\mathcal{C}$ if and only if $(v_1, v_2, v_3) \in$ 3-OR.

Suppose the assignment is such that $(v_1, v_2, v_3) \in$ 3-OR, thus there exists $i \in [3]$ such that $v_i = 1$. One may verify (see Table 2.1) that in this case, the assignment can be extended to a satisfying assignment of the entire formula. Suppose the assignment is such that $(v_1, v_2, v_3) \notin$ 3-OR, thus $v_1 = v_2 = v_3 = 0$. Then the first constraint in $\mathcal{C}$ implies that $x_1 = x_2 = 0$, and the third constraint implies that $x_3 = x_4 = 0$, in any assignment satisfying $\mathcal{C}$. However, this does not satisfy the second constraint, since $(0, 0, 0) \notin R_0$, which is a contradiction. ◀

The next theorem gives an important relation between pp-definability and polymorphisms.

▶ **Theorem 2.42** ([20, 47]) *Let Γ be a constraint language over domain D. A non-empty relation R over domain D is pp-definable over Γ if and only if each polymorphism of Γ is a polymorphism of R.* ◀

The following proposition is a known fact.

▶ **Proposition 2.43** ([25, Exercise 5.3]) *If Γ is an intractable Boolean constraint language, then every Boolean relation is pp-definable from Γ^*.*

Proof. The idea is as follows. It follows from the intractability of Γ and Theorem 5.1 of [25], that all polymorphisms of Γ are "essentially unary". This means that if f is a k-ary polymorphism of Γ, there exists $i \in [k]$ and $g\colon \{0,1\} \to \{0,1\}$ such that $f(x_1, \ldots, x_k) = g(x_i)$ for all $(x_1, \ldots, x_k) \in \{0,1\}^k$. It follows from this that the only polymorphisms of Γ^* are in fact constant. Since any relation has any constant function as a polymorphism, it follows from Theorem 2.42 that any Boolean relation is pp-definable from Γ^*. ◀

We now give a result about the existence of linear-parameter transformations. Recall that Γ^* is defined as the relation Γ together with all constants, by

Definition 2.39.

▶ **Theorem 2.44** (Follows from [23]) *Let Γ be a constraint language over a finite set D such that each unary operation $u\colon D \to D$ that preserves Γ is a bijection. Then, there exists a linear-parameter transformation from $\mathrm{CSP}(\Gamma^*)$ parameterized by the number of variables to $\mathrm{CSP}(\Gamma)$ parameterized by the number of variables.*

Note that in particular, an *intractable* Boolean constraint language can only be preserved (recall Definition 2.35) by unary operations that are bijections, as otherwise it is preserved by u_0 or u_1 and thus the constraint language is tractable by Theorem 2.37. Hence the above theorem implies that for intractable Boolean Γ, there is a linear-parameter transformation from $\mathrm{CSP}(\Gamma^*)$ to $\mathrm{CSP}(\Gamma)$.

Proof of Theorem 2.44. The desired transformation is the final polynomial-time reduction given in the proof of Theorem 4.7 of [23]. This reduction translates an instance of $\mathrm{CSP}(\Gamma^*)$ with n variables to an instance of $\mathrm{CSP}(\Gamma \cup \{=_D\})$ with $n + |D|$ variables; here, $=_D$ denotes the equality relation on domain D. Each constraint of the form $=_D\ (v, v')$ may be removed (while preserving satisfiability) by taking one of the variables v, v', and replacing each instance of that variable with the other. The resulting instance of $\mathrm{CSP}(\Gamma)$ has $\leq n + |D|$ variables. ◀

2.8 Graph coloring problems

A proper *q-coloring* of a graph G is a function $f\colon V(G) \to [q]$ such that for all $\{u, v\} \in E(G)$ it holds that $f(u) \neq f(v)$. Based on this, the q-Coloring problem for a fixed integer q is defined as follows.

q-Coloring

Input: A graph G.
Question: Does G have a proper q-coloring?

A generalization of the q-Coloring problem is the H-Coloring problem. To define this problem, we first need the notion of homomorphisms.

Definition 2.45 (Homomorphism) Let G and H be (not necessarily simple) graphs. A *homomorphism* from G to H is a mapping $f\colon V(G) \to V(H)$ such that $\{f(u), f(v)\} \in E(H)$ for all $\{u, v\} \in E(G)$. We denote the existence of a homomorphism from G to H by $G \to H$. If $G \to H$ we say G is *homomorphic* to H, and if both $G \to H$ and $H \to G$ we say G and H are *homomorphically equivalent*, which we denote by $G \leftrightarrow H$.

For a fixed graph H, we can now define the H-Coloring problem as follows.

H-COLORING

Input: A graph G.
Question: Does there exist a homomorphism from G to H?

When there is a homomorphism f from G to H we will often say G is *H-colorable*, and for a vertex $v \in V(G)$ we may refer to $f(v)$ as the *color* of vertex v, as is consistent with the terminology for q-COLORING. Observe that the q-COLORING problem is equivalent to the problem of K_q-COLORING, where K_q is the clique on q vertices.

When H is a simple graph, H-COLORING can be viewed as equivalent to a constraint satisfaction problem over the domain $V(H)$. In particular, H-COLORING corresponds to $\mathrm{CSP}(H)$ where H is defined as the constraint language containing the single relation F, which is the symmetric binary relation given by $F = \{(u,v) \mid \{u,v\} \in E(H)\}$. A mapping $f\colon V(G) \to V(H)$ is a homomorphism from a graph G to H if and only if it is a satisfying assignment of the $\mathrm{CSP}(\{F\})$ instance on variables $V(G)$ and constraints $\{F(u',v') \mid \{u',v'\} \in E(G)\}$.

▶ **Example 2.46** Let H be the graph defined on three vertices a, b, and c, that form a triangle, such that H is the graph K_3. Then the constraint language $H := \{F\}$ is defined by

$$F := \{(a,b),(b,a),(a,c),(c,a),(b,c),(c,b)\}.$$

If now G is an arbitrary graph, checking whether G is homomorphic to H is equivalent to checking if the $\mathrm{CSP}(H)$-instance $(V,\mathcal{C})$ given by $V := V(G)$ and $\mathcal{C} := \{F(u,v) \mid \{u,v\} \in E(G)\}$ is satisfiable over domain $D = \{a,b,c\}$. In this example, observe that any satisfying assignment simply corresponds to a 3-coloring of G with colors a, b, and c. ◀

Definition 2.47 (H^*) In correspondence with Definition 2.39, we define H^* as the constraint language that contains the relation $F = \{(u,v) \mid \{u,v\} \in E(H)\}$ and, for each $v \in V(H)$, the arity-1 relation $\{(v)\}$. The arity-1 relations in H^* effectively allow the value of a variable to be forced to a constant value, so that $\mathrm{CSP}(H^*)$ corresponds to the PRE-H-COLORING EXTENSION problem.

Definition 2.48 (Core) A graph H is a *core* if there exists no homomorphism from H to an induced proper subgraph of H. We may say that a graph H' is the core of a graph H if H' is an induced subgraph of H such that H' is a core, and $H' \leftrightarrow H$. Observe that for any graph H, its core is unique up to isomorphism.

Chapter 3

Sparsification Using Low-Degree Polynomials

In this chapter we provide techniques to obtain sparsifications for certain Constraint Satisfaction Problems. Here we aim to reduce the number of constraints, without changing the satisfiability of the formula. In particular, we want to obtain guarantees on the number of remaining constraints, in terms of the number of variables. To this end, we will study CSPs where for each constraint, the satisfying assignments can be captured by low-degree polynomials in an appropriate sense. Let us start by giving a simple example of this technique, before studying the problem in full generality, by considering the EXACT SAT problem (as defined in Section 2.7).

Let an instance of EXACT SAT with m clauses over variable set V with $|V| = n$ be given. Any constraint of an instance of EXACT SAT is satisfied by an assignment $\tau\colon V \to \{0,1\}$ if and only if it contains exactly one *true* literal, where we equate *true* with the value 1 and *false* with 0. This is equivalent to saying that a clause $(x_1, \ldots, x_i, \neg x_{i+1}, \ldots, \neg x_d)$ is satisfied by τ if and only if the following equation holds:

$$\sum_{j=1}^{i} \tau(x_j) + \sum_{j=i+1}^{d} (1 - \tau(x_j)) = 1.$$

To find redundant clauses, transform each of the m input clauses into a linear equality, to obtain a system of equalities $A\mathbf{x} = \mathbf{b}$ where A is an $m \times n$ matrix,

$\mathbf{x}$ is the column vector $(x_1, \ldots, x_n)$, and $\mathbf{b}$ is an integer column vector. Using Gaussian elimination, we can efficiently compute a basis B for the row space of the extended matrix $(A|\mathbf{b})$. So, B is a set of equalities such that every equality corresponding to a clause can be written as a linear combination of equalities in B. One can even compute B such that it is a subset of the set of original equalities. Since $(A|\mathbf{b})$ has $n+1$ columns, it follows that it has rank at most $n+1$ and thus B contains at most $n+1$ equalities. To perform the sparsification, remove all clauses from the EXACT SAT instance for which the corresponding equality does not appear in B.

Since we only removed clauses, it is clear that any assignment satisfying the original instance, will satisfy the sparsified instance. The other direction follows from the fact that if $\mathbf{x}$ satisfies $f_1(\mathbf{x}) = b_1$ and $f_2(\mathbf{x}) = b_2$, then it will also satisfy $f_1(\mathbf{x}) + f_2(\mathbf{x}) = b_1 + b_2$. Using that any equality we are interested in can be written as a linear combination of equalities in B gives us that any assignment satisfying the sparsified instance, also satisfies the original input instance. Thus, we have given a sparsification for EXACT SAT with $\mathcal{O}(n)$ clauses.

In terms of kernelization, the measure for kernel size is not the number of clauses of the sparsified instance, but the number of bits required to store it. If clauses are assumed to have size at most d for some constant d, the instance can be stored in $\mathcal{O}(n \log n)$ bits. This can for example be done by encoding each clause individually by storing the d variables it contains. Since there are n variables, we can represent a single variable by $\log n$ bits, resulting in a total size of $\mathcal{O}(d \cdot n \log n)$, which equals $\mathcal{O}(n \log n)$ for d constant.

When the number of literals in a clause can be as large as $\mathcal{O}(n)$, it may however take $\Omega(n)$ bits to encode a single clause. After reducing the number of clauses in EXACT SAT to $n+1$, it may therefore still take $\Theta(n^2)$ bits to encode the instance. Observe that obtaining an encoding of bitsize $\mathcal{O}(n^2)$ instead of $\mathcal{O}(n^2 \log n)$ is in this case possible: we can safely assume that each clause contains every literal at most twice. Since there are $2n$ literals, we can represent each clause by storing for each literal whether it occurs zero, one, or two times in the clause, which can be done in $\mathcal{O}(n)$ bits. This leads to an encoding of size $\mathcal{O}(n^2)$ when we have $n+1$ clauses.

The change from $\mathcal{O}(n \log n)$ bits for bounded clause length to $\mathcal{O}(n^2)$ bits for unbounded clause length turns out to be unavoidable: we prove that EXACT SAT has no kernel of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless NP $\subseteq$ coNP/poly in Corollary 3.23.

The example given above uses that d-EXACT SAT can be expressed using degree-1 polynomials over the rationals. We show that d-NAE-SAT and d-CNF-SAT can be expressed using equalities of polynomial expressions of degree $d-1$ and d. We therefore study the following problem, where F is a field. Refer to Section 2.6 for relevant definitions on polynomials.

d-POLYNOMIAL ROOT CSP over F

Input: A list L of polynomial equalities over variables $V = \{x_1, \ldots, x_n\}$. An equality is of the form $f(x_1, \ldots, x_n) = 0$, where f is a multivariate polynomial over F of degree at most d.
Parameter: The number of variables n.
Question: Does there exist an assignment of the variables $\tau\colon V \to \{0,1\}$ satisfying all equalities in L?

When interpreting truth assignments as elements of a ring or field, we equate the value *true* with the 1 element in the ring (multiplicative identity), and the value *false* with the 0 element (additive identity). Consequently, for a Boolean variable x its negation $\neg x$ corresponds to $(1 - x)$. Note that over a field, we can assume this without loss of generality: for any a and b in F with $a \neq b$, there exists a linear bijection that maps 0 to b and 1 to a, and this bijection does not change the degree of the resulting polynomial.

Overview We show in Section 3.1.1 that using a generalization of the argument presented for EXACT SAT, the number of constraints in an instance of *d*-POLYNOMIAL ROOT CSP can efficiently be reduced to $\mathcal{O}(n^d)$, even when the number of variables that occur in a constraint is not restricted. Assuming that each constraint can be encoded in $\widetilde{\mathcal{O}}(n)$ bits, this allows us to compress instances of *d*-POLYNOMIAL ROOT CSP to bitsize $\widetilde{\mathcal{O}}(n^{d+1})$. This reduction argument in fact works over arbitrary fields F. We then extend our results to obtain similar results over $\mathbb{Z}/m\mathbb{Z}$. When m is not prime, the resulting structure is not a field and this imposes additional technical difficulties.

In Section 3.1.2, we consider Boolean CSPs whose constraints are formed by *disequalities*, rather than equalities, of degree-d polynomials. This leads to the following generic problem:

d-POLYNOMIAL NON-ROOT CSP over F

Input: A list L of polynomial disequalities over variables $V = \{x_1, \ldots, x_n\}$. A disequality is of the form $f(x_1, \ldots, x_n) \neq 0$, where f is a multivariate polynomial over F of degree at most d.
Parameter: The number of variables n.
Question: Does there exist an assignment of the variables $\tau\colon V \to \{0,1\}$ satisfying all disequalities in L?

This problem formulation is related to the *weak representation* of Boolean functions by polynomials: a polynomial f weakly represents a Boolean function g when $f(x_1, \ldots, x_n) \neq 0$ if and only if $g(x_1, \ldots, x_n) \neq 0$ (cf. [7, 11]). This means that effectively, the *d*-POLYNOMIAL NON-ROOT CSP problem asks to find an assignment that satisfies a list of constraints which are weakly represented by degree-d polynomials. For a prime p, we show that the number of constraints

Problem	d-Poly. root CSP		d-Poly. non-root CSP	
	Lower bound[1]	Upper bound[2]	Lower bound[1]	Upper bound[2]
$\mathbb{Q}$	$\Omega(n^{d+1-\varepsilon})$	$\widetilde{\mathcal{O}}(n^{d+1})$	Superpolynomial	
$\mathbb{Z}/p\mathbb{Z}$	$\Omega(n^{d+1-\varepsilon})$	$\widetilde{\mathcal{O}}(n^{d+1})$	$\Omega(n^{d(p-1)-\varepsilon})$	$\widetilde{\mathcal{O}}(n^{d(p-1)+1})$
$\mathbb{Z}/m\mathbb{Z}$	$\Omega(n^{d+1-\varepsilon})$	$\widetilde{\mathcal{O}}(n^{d+1})$	$\Omega(n^{(d/2)^r-\varepsilon})$	?
Theorem	3.26, 3.27	3.1, 3.6, 3.12	3.28, 3.30, 3.29	3.13

[1] The lower bounds hold for any $\varepsilon > 0$, for the problems that are not polynomial-time solvable and under the assumption that NP $\not\subseteq$ coNP/poly.

[2] The upper bounds hold when each n-variate polynomial constraint in the input can be encoded in $\widetilde{\mathcal{O}}(n)$ bits.

Table 3.1 Summary of the kernel upper and lower bounds obtained in this chapter, expressed in the number of bits. The bounds depend on whether the polynomials defining the constraints are over the rationals $\mathbb{Q}$, the integers modulo a prime p, or the integers modulo a composite m. The integer r denotes the number of distinct prime divisors of m. The values of p, m, r, and d are treated as constants in these bounds.

for d-Polynomial non-root CSP over $\mathbb{Z}/p\mathbb{Z}$ can be efficiently reduced to $\mathcal{O}(n^{d(p-1)})$, resulting in an instance of bitsize $\widetilde{\mathcal{O}}(n^{d(p-1)+1})$.

We conclude this chapter (Section 3.2) by showing a number of kernelization lower bounds that (nearly) match the upper bounds obtained in Section 3.1. We show that for d-Polynomial root CSP no kernel of size $\mathcal{O}(n^{d+1-\varepsilon})$ is possible, unless NP $\subseteq$ coNP/poly. For d-Polynomial non-root CSP over $\mathbb{Z}/p\mathbb{Z}$ for a prime p, we obtain a lower bound of $\mathcal{O}(n^{d(p-1)-\varepsilon})$, which leaves a factor-$n$ gap with the upper bound. For general d-Polynomial non-root CSP, we show that it does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly. For d-Polynomial non-root CSP over $\mathbb{Z}/m\mathbb{Z}$ we give a lower bound. We do not have a polynomial upper bound for this problem, the difficulties in obtaining a polynomial kernel are discussed in Section 3.3. An overview of the upper and lower bound results obtained for d-Polynomial root CSP and d-Polynomial non-root CSP can be found in Table 3.1.

3.1 Upper bound techniques

3.1.1 Sparsification for Polynomial root CSP

We start by showing how to reduce the number of constraints in instances of d-Polynomial root CSP, by extending the argument given for Exact SAT in the introduction. Let a field F be *efficient* if the field operations and Gaussian elimination can be done in polynomial time in the size of a reasonable input encoding. By this definition, $\mathbb{Q}$ and $\mathbb{Z}/p\mathbb{Z}$ for a prime p are examples of efficient

fields.

▶ **Theorem 3.1** *There is a polynomial-time algorithm that, given an instance (L, V) of d-POLYNOMIAL ROOT CSP over an efficient field F, outputs an equivalent instance (L', V) with at most $n^d + 1$ constraints such that $L' \subseteq L$.*

Proof. Given a list L of polynomial equalities over variables V for d-POLYNOMIAL ROOT CSP, we use linear algebra to find redundant constraints. Observe that $(x_i)^c = x_i$ for all $0/1$-assignments and $c \geq 1$. As constraints are evaluated over $0/1$-assignments, we may assume without loss of generality that the monomials in each of the polynomials are multilinear: each monomial consists of a coefficient from F multiplied by distinct variables. Note that we also need the constant monomial 1.

Create a matrix A with $|L|$ rows and a column for every multilinear monomial of degree at most d over variables from V. Let position $a_{i,j}$ in A be the coefficient of the monomial corresponding to column j in the polynomial equality corresponding to row i.

Compute a basis B of the row space of matrix A, for example using Gaussian elimination [50], and let L' consist of the equalities in L whose corresponding row appears in the basis. Since $L' \subseteq L$, it follows that if the original instance has a satisfying assignment, the reduced instance has a satisfying assignment as well. The crucial part of the correctness proof is to establish the converse.

▷ **Claim 3.2** *If an assignment $\tau\colon V \to \{0,1\}$ of the variables in V satisfies the equalities in L', then it satisfies all equalities in L.*

Proof. Consider any equality $(f(\mathbf{x}) = 0) \in L \setminus L'$, and assume it corresponds to the i'th matrix row. Let $f_j(\mathbf{x})$ be the polynomial represented in the j'th row of matrix A for $j \in [|L|]$. Without loss of generality, let the basis of A correspond to its first m rows $\mathbf{a_1}, \ldots, \mathbf{a_m}$. We then have $i > m$, and by the definition of basis there exist $\beta_1, \ldots, \beta_m \in F$ such that $\mathbf{a_i} = \sum_{j=1}^{m} \beta_j \mathbf{a_j}$. Let $\mathbf{t}$ be the column vector containing, for each multilinear monomial of degree $\leq d$ in variables $x_1, \ldots, x_n$, the evaluation under τ. For example, for monomial $x_1 x_3$ it contains $\tau(x_1) \cdot \tau(x_3)$. By using the same order of monomials as in the construction of A, we obtain for all $j \in [|L|]$ that $f_j(\tau(x_1), \ldots, \tau(x_n)) = \mathbf{a_j t}$, the inner product of $\mathbf{a_j}$ and $\mathbf{t}$. It follows that $\mathbf{a_j t} = 0$ for all $j \in [m]$, since satisfying L' implies $f_j(\tau(x_1), \ldots, \tau(x_n)) = 0$. Now observe that

$$f_i(\mathbf{x}) = \mathbf{a_i t} = \sum_{j=1}^{m} (\beta_j \mathbf{a_j}) \mathbf{t} = \sum_{j=1}^{m} \beta_j (\mathbf{a_j t}) = \sum_{j=1}^{m} \beta_j \cdot 0 = 0,$$

which proves the claim. ◁

We now prove an upper bound on the number of constraints in the resulting kernel.

▷ **Claim 3.3** *The number of constraints in the resulting kernel is bounded by* $n^d + 1$.

Proof. The size of a basis of any matrix over a field equals its rank, which is bounded by the number of columns. As there is exactly one column for each multilinear monomial of degree at most d, it follows from Lemma 2.27 that there are at most $n^d + 1$ columns and the claim follows. ◁

This concludes the proof of Theorem 3.1. ◀

When each constraint can be encoded in $\widetilde{\mathcal{O}}(n)$ bits, for example when each polynomial can be represented as an arithmetic circuit of size $\mathcal{O}(n)$, Theorem 3.1 gives a kernelization of size $\widetilde{\mathcal{O}}(n^{d+1})$. When constraints can be encoded in $\widetilde{\mathcal{O}}(1)$ bits, which may occur when constraints have constant arity, we obtain kernels of bitsize $\widetilde{\mathcal{O}}(n^d)$.

Example application Let us consider the following problem, that generalizes the d-EXACT SAT and d-NAE-SAT problems. Optionally, a prime p may be chosen.

GENERALIZED d-SAT (MOD p)

Input: A set of clauses $\mathcal{C}$ over variables $V := \{x_1, \ldots, x_n\}$, and for each clause a set $S_i \subset \mathbb{N}$ with $|S_i| \leq d$. Each clause is a set of distinct literals of the form x_i or $\neg x_i$.
Parameter: $|V| = n$.
Question: Does there exist a truth assignment $\tau \colon V \to \{0,1\}$ such that the number of satisfied literals in clause i modulo p lies in S_i for all i?

We can now use the results obtained for d-POLYNOMIAL ROOT CSP, to give a kernelization for the above problem.

▶ **Corollary 3.4** GENERALIZED d-SAT *and* GENERALIZED d-SAT MOD p *both have a kernel with* $n^d + 1$ *clauses, such that the kernelized instance can be encoded in* $\mathcal{O}(n^{d+1} \log n)$ *bits. Furthermore, the clauses of the kernelized instance are a subset of the clauses of the original instance.*

Proof. To reduce the number of clauses using Theorem 3.1, we only have to provide a polynomial of degree at most d to represent each constraint. Consider a clause involving k variables $x_{i_1}, \ldots, x_{i_k}$, with set S_ℓ. Let $t_j = x_{i_j}$ if variable x_{i_j} occurs positively in the clause, and let $t_j = (1 - x_{i_j})$ if the variable occurs negatively. Then the number of satisfied literals in the clause is given by the degree-1 polynomial $f(x_{i_1}, \ldots, x_{i_k}) := \sum_{j=1}^{k} t_j$. Let $F(x)$ be a polynomial with root set $S_\ell \pmod p$ of degree at most $|S_\ell|$, which exists by Observation 2.28. We obtain that $F(f(\mathbf{x})) \equiv_p 0$ if and only if $\mathbf{x}$ satisfies the clause. Note that the degree of $F(f(\mathbf{x}))$ is at most $|S_\ell| \leq d$.

Applying Theorem 3.1 to the resulting instance of d-POLYNOMIAL ROOT CSP identifies a subset of at most $n^d + 1$ constraints which preserve the answer to the SAT problem. Each clause contains at most $2n$ literals, which can be encoded in $\mathcal{O}(\log n)$ bits each. Additionally, for each clause we need to store the set S_ℓ of at most d integers, which have value at most $2n$ in relevant inputs. As d is a constant, the instance can be encoded in $\mathcal{O}(n^{d+1} \log n)$ bits. ◀

Corollary 3.4 yields a new way to get a non-trivial compression for d-NAE-SAT, which is conceptually simpler than the existing approach which requires an unintuitive lemma by Lovász [75]. The new approach gives the same size bound as given earlier [59, Theorem 6].

▶ **Corollary 3.5** *d-NAE-SAT has a kernel with $n^{d-1} + 1$ clauses, such that the kernelized instance can be encoded in $\mathcal{O}(n^{d-1} \log n)$ bits.*

Proof. A clause of size $k \leq d$ is not-all-equal satisfied if and only if the number of satisfied literals lies in $S := \{1, \ldots, k-1\}$. Using Corollary 3.4 we can reduce the number of clauses to $n^{d-1} + 1$. Each clause has $d \in \mathcal{O}(1)$ variables and can thus be encoded in $\mathcal{O}(\log n)$ bits. ◀

We can generalize Theorem 3.1 to also obtain a sparsification for d-POLYNOMIAL ROOT CSP over the integers modulo a non-prime. We give two different approaches for sparsifying such problems. The first approach gives the smallest number of constraints after reduction, but has the disadvantage that the resulting list of constraints is not necessarily a subset of the original list of constraints. The second approach results in a larger (but still bounded) number of constraints, which form a subset of the original constraints. We first give some linear-algebraic background.

Consider an instance (L, V) of d-POLYNOMIAL ROOT CSP over a ring R with n variables and m constraints. We consider the matrix A over R with m rows and $\sum_{i=0}^{d} \binom{n}{d}$ columns, in which the i'th row contains the coefficients of the multilinear monomials in the polynomial for the i'th constraint. The satisfiability of the constraints by a $0/1$-assignment then comes down to the following question: is there a $0/1$-assignment to the variables, such that the vector $\mathbf{x}$ consisting of all multilinear monomial evaluations of the variables $x_1, \ldots, x_n$ satisfies $A\mathbf{x} = \mathbf{0}$ over R? The key insight for the sparsification is that any matrix B for which the row-space over R is equal to that of A, satisfies $A\mathbf{x} = \mathbf{0} \Leftrightarrow B\mathbf{x} = \mathbf{0}$. (Recall that the row-space over R consists of the vectors that can be written as a linear combination of the rows, with coefficients from R.) Hence we can obtain an encoding of an equivalent problem by selecting a matrix B whose row-space equals that of A. When working over a field we can just extract a basis for the row-space to obtain B, which is exactly what happened in Theorem 3.1. When working over the integers modulo m for composite m, the existence of a basis is not guaranteed (for more details, see the discussion after Definition 2.18). For our first approach we therefore use the Howell

normal form of the matrix, which is a canonical matrix form which has the same row-space.

▶ **Theorem 3.6** *There is a polynomial-time algorithm that, given an instance (L, V) of d-*POLYNOMIAL ROOT CSP *over $\mathbb{Z}/m\mathbb{Z}$ for some integer $m \geq 2$, outputs an equivalent instance (L', V) of d-*POLYNOMIAL ROOT CSP *over $\mathbb{Z}/m\mathbb{Z}$ with at most $n^d + 1$ constraints.*

Proof. In a similar way as in Theorem 3.1, we use linear algebra to find redundant constraints. Let a list L of polynomial equalities over variable set V be given. We again assume without loss of generality that the monomials in each of the polynomials are multilinear. Construct a matrix A with $|L|$ rows and a column for every multilinear monomial of degree at most d over variables from V.

We now compute the Howell form H of matrix A (Definition 2.24), such that $A = PH$, where P is invertible over $\mathbb{Z}/m\mathbb{Z}$. This can be done in polynomial time by Theorem 2.25. Let H' be the matrix H with all zero rows removed. Let L' contain the polynomial equations where the left-hand sides are given by the rows of H' and the right-hand sides are all 0. We now prove the correctness of this procedure.

▷ **Claim 3.7** *An assignment $\tau : V \to \{0,1\}$ of the variables in V satisfies the equalities in L', if and only if it satisfies the equalities in L.*

Proof. ($\Rightarrow$) Suppose assignment $\tau : V \to \{0,1\}$ satisfies all equalities in L'. Consider the vector $\mathbf{x}$ with the assignment given to the j'th monomial on position j. Then

$$H'\mathbf{x} = \mathbf{0} \Leftrightarrow H\mathbf{x} = \mathbf{0} \Rightarrow PH\mathbf{x} = P\mathbf{0} \Rightarrow A\mathbf{x} = \mathbf{0},$$

which implies that τ is also a satisfying assignment for L.

($\Leftarrow$) Suppose assignment $\tau : V \to \{0,1\}$ satisfies all equalities in L. Consider the vector $\mathbf{x}$ with the assignment given to the j'th monomial on position j. Then

$$A\mathbf{x} = \mathbf{0} \Leftrightarrow PH\mathbf{x} = \mathbf{0} \Rightarrow P^{-1}PH\mathbf{x} = P^{-1}\mathbf{0} \Rightarrow H\mathbf{x} = \mathbf{0} \Rightarrow H'\mathbf{x} = \mathbf{0},$$

which implies that τ is also a satisfying assignment for L'. ◁

▷ **Claim 3.8** *The number of constraints in the resulting kernel L' is bounded by $n^d + 1$.*

Proof. The number of constraints in L' equals the number of rows in H'. We will use the following properties of a matrix in Howell form (recall Definition 2.24) to give an upper bound on the number of non-zero rows in H.

- Let r be the number of non-zero rows of H. Then the first r rows of H are non-zero.

- For $1 \leq i \leq r$ let the first non-zero entry in row i of H be in column j_i. Then $j_1 < j_2 < \ldots < j_r$.

By these two properties any matrix in Howell form has at most as many non-zero rows as it has columns. Thereby there are at most $n^d + 1$ polynomial equations in L'. ◁

This concludes the proof of Theorem 3.6. ◀

Compared to Theorem 3.1, the sparsification of Theorem 3.6 has the disadvantage that it may output polynomials (representing constraints) that were not part of the input. If the input polynomials had an efficient encoding, for example as an arithmetic circuit, this property may be lost in the transformation. In general, to represent an output polynomial one may have to store all its $\mathcal{O}(n^d)$ coefficients individually. We present an alternative approach that alleviates this issue by ensuring that the set of constraints in the output instance is a subset of the original constraints. However, it comes at the expense of increasing the number of constraints. The following lemma captures the key linear-algebraic insight behind the approach.

▶ **Lemma 3.9** *Let $m \geq 2$ be an integer with r distinct prime divisors. For any $S \subseteq \mathbb{Z}/m\mathbb{Z}$ there exists a subset $S' \subseteq S$ of size at most r such that any element in S can be written as a linear combination over $\mathbb{Z}/m\mathbb{Z}$ of elements in S'. For any fixed m, one can compute S' and expressions of all $a \in S$ as linear combinations of S' in polynomial time.*

Proof. Let $p_1, \ldots, p_r$ be the distinct prime divisors of m, which can be found in constant time for fixed m. For a prime p and positive integer a, define:

$$\mu_p(a) := \max\{k \in \mathbb{N} \mid p^k \text{ divides } a\}.$$
$$\nu_p(a) := \max\{k \in \mathbb{N} \mid p^k \text{ divides both } a \text{ and } m\}.$$

Observe that $\nu_p(a) \leq \mu_p(a)$ for all a. For any a that divides m we have $\nu_p(a) = \mu_p(a)$.

Using these notions we construct the set S' as follows. For each $i \in [r]$ select an element $a \in S$ that minimizes $\nu_{p_i}(a)$ and add this element to S'. Since m is constant this can be done in polynomial time. The resulting set S' has size at most r. We prove it spans S using the following claim.

▷ **Claim 3.10** *Let d be the largest integer that simultaneous divides m and all elements of S'. For any $b \in S \setminus S'$, the integer d divides b. Equivalently:*

$$\gcd(m, S') \mid b.$$

Proof. If $d = 1$ then the claim is trivial. Suppose all prime factors p of d are also prime factors of b with $\mu_p(d) \leq \mu_p(b)$. Then the factorization of b can be

written as the factorization of d multiplied by remaining factors. Hence $d \mid b$, and the claim follows.

Now suppose there is a prime factor p of d with $\mu_p(d) > \mu_p(b)$. Since p is a factor of $d = \gcd(m, S')$, we know p is a factor of m and was therefore considered during the construction of S'. Since d divides m we know that $\mu_p(d) = \nu_p(d)$. Combined with the fact that $\nu_p(b) \leq \mu_p(b)$ it follows that $\nu_p(b) \leq \mu_p(b) < \mu_p(d) = \nu_p(d)$. Since d divides all members of S', it follows that $\nu_p(b) < \nu_p(d) \leq \nu_p(a)$ for all $a \in S'$. But then b should have been added to S' during its construction; a contradiction. ◁

To conclude the proof, we use Claim 3.10 to show that any $b \in S \setminus S'$ can efficiently be written as a linear combination of S' over $\mathbb{Z}/m\mathbb{Z}$. By Bézout's identity (Theorem 2.15), the greatest common divisor of a set of integers can be written as an integer linear combination of the elements in that set. Such a combination can efficiently be found using the extended Euclidean algorithm. Hence there are integer coefficients α_i such that $d = \gcd(m, S') = \alpha_m \cdot m + \sum_{a \in S'} \alpha_a \cdot a$. Let $b' := b / \gcd(m, S')$, which is integral by Claim 3.10. But then

$$b = b' \cdot \gcd(m, S') = (b' \cdot \alpha_m)m + \sum_{a \in S'} (b' \cdot \alpha_a)a,$$

which implies that

$$b \equiv_m \sum_{a \in S'} (b' \cdot \alpha_a)a \equiv_m \sum_{a \in S'} ((b' \cdot \alpha_a) \bmod m)a$$

is a linear combination over $\mathbb{Z}/m\mathbb{Z}$ resulting in b. ◀

The following lemma follows from a procedure similar to Gaussian elimination, using Lemma 3.9 as a subroutine.

▶ **Lemma 3.11** *Let $m \geq 2$ be an integer with r distinct prime divisors. For any matrix A over $\mathbb{Z}/m\mathbb{Z}$ in which $k \geq 1$ columns contain a nonzero element, there is a subset B of $r \cdot k$ rows of A that spans the row-space of A. For any fixed m, such a subset B can be found in polynomial time.*

Proof. Proof by induction on k. Consider the first column c_i of A that contains a nonzero and let S be the elements appearing in that column. Using Lemma 3.9, compute a subset $S' \subseteq S$ of size at most r that spans S, and find the corresponding linear combinations. For each element $a \in S'$ select one row with value a in column c_i and add it to B_1. If c_i is the only column containing a nonzero, then it is easy to see that $B := B_1$ is a valid output for the procedure. Otherwise, since all elements of S are linear combinations of elements of S', by subtracting the relevant linear combinations of rows of B_1 from rows in A we can obtain zeros at all positions in column c_i, without introducing nonzeros in earlier columns. Let A' be the resulting matrix, which therefore has at most $k - 1$

nonzero columns. Apply induction to find a spanning subset B' of the rows of A' of size at most $r \cdot (k-1)$. Let B_2 be the rows of A corresponding to rows B' in A'. Then $B_1 \cup B_2$ consists of at most $r + (k-1)r = rk$ rows of A. It is easy to verify that these rows indeed span the row-space of A. The inductive proof directly translates into a polynomial-time recursive algorithm, using the fact that the procedure of Lemma 3.9 provides the required linear combinations. ◀

Using the lemmas above, we can now give our sparsification procedure for d-POLYNOMIAL ROOT CSP over $\mathbb{Z}/m\mathbb{Z}$ that outputs a subset of the original constraints.

▶ **Theorem 3.12** *There is a polynomial-time algorithm that, given an instance (L, V) of d-POLYNOMIAL ROOT CSP over $\mathbb{Z}/m\mathbb{Z}$ for some fixed integer $m \geq 2$ with r distinct prime divisors, outputs an equivalent instance (L', V) of d-POLYNOMIAL ROOT CSP over $\mathbb{Z}/m\mathbb{Z}$ with at most $r \cdot (n^d + 1)$ constraints such that $L' \subseteq L$.*

Proof. We proceed similarly as in the proof of Theorem 3.6. Consider an input (L, V) of d-POLYNOMIAL ROOT CSP over $\mathbb{Z}/m\mathbb{Z}$ with $n := |V|$ variables. Let A be the matrix with $|L|$ rows and $\sum_{i=0}^{d} \binom{n}{i}$ columns, containing the coefficients of the multilinear monomials that form the constraints for each of the $|L|$ constraint polynomials. A $0/1$-assignment to the variables satisfies all constraints if and only if the vector $\mathbf{x}$ of all monomial evaluations satisfies $A\mathbf{x} = \mathbf{0}$. Use Lemma 3.11 to compute a subset B of at most $r \cdot \sum_{i=0}^{d} \binom{n}{i} \leq r \cdot (n^d + 1)$ rows of A that span the row-space of A. Let L' contain the constraints whose corresponding row appears in B and output the instance (L', V) as the result of the procedure. Using the guarantee of Lemma 3.11 this procedure runs in polynomial time for fixed m. Since $L' \subseteq L$, the instance (L', V) can be satisfied if (L, V) can. For the reverse direction, consider a satisfying assignment for (L', V) and the corresponding vector $\mathbf{x}$ of evaluations of multilinear monomials of degree at most d. Then $B\mathbf{x} = \mathbf{0}$ since the assignment satisfies all constraints in L'. As any row in A can be written as a linear combination of rows in B, it follows that $A\mathbf{x} = \mathbf{0}$, showing that (L, V) is satisfiable and hence that the output instance is equivalent to the input. ◀

3.1.2 Sparsification for Polynomial non-root CSP

We will now change focus to constraint satisfaction problems with constraints given by polynomial disequalities. Therefore, we consider the d-POLYNOMIAL NON-ROOT CSP problem. In Section 3.2.3 we will show that, over the field of rational numbers, the problem cannot be compressed to size polynomial in n, unless NP $\subseteq$ coNP/poly. We therefore consider the field $\mathbb{Z}/p\mathbb{Z}$ of integers modulo a prime p.

▶ **Theorem 3.13** *There is a polynomial-time algorithm that, given an instance (L, V) of d-POLYNOMIAL NON-ROOT CSP over $\mathbb{Z}/p\mathbb{Z}$, outputs an equivalent instance (L', V) with at most $n^{d(p-1)} + 1$ constraints such that $L' \subseteq L$.*

Proof. Suppose we are given a list of polynomial disequalities L over variables V. Observe that a disequality $f(\mathbf{x}) \not\equiv_p 0$ is equivalent to $f(\mathbf{x}) \bmod p \in \{1, \ldots, p-1\}$. Recall that Fermat's little theorem states that $a^p \equiv_p a$ for any integer a and prime p, which implies that $a^{(p-1)} \equiv_p 1$ if and only if $a \neq 0$. Hence the disequality $f(\mathbf{x}) \not\equiv_p 0$ can equivalently be stated as $(f(\mathbf{x}))^{p-1} - 1 \equiv_p 0$. Therefore, L can be written as an instance of $d(p-1)$-POLYNOMIAL ROOT CSP by replacing every polynomial disequality $f(\mathbf{x}) \not\equiv_p 0$ by the equality $(f(\mathbf{x}))^{p-1} - 1 \equiv_p 0$. By Theorem 3.1, the theorem statement follows. ◀

In Section 3.2.3 we will establish a nearly-matching lower-bound counterpart to Theorem 3.13. We do not have upper bounds for d-POLYNOMIAL NON-ROOT CSP modulo a composite number m, the difficulty of obtaining these is discussed in the conclusion of this Chapter (Section 3.3).

3.2 Tightness of upper bound techniques

We now turn our attention to lower bounds and show that the results obtained for d-POLYNOMIAL ROOT CSP and d-POLYNOMIAL NON-ROOT CSP in Sections 3.1.1 and 3.1.2 are (almost) tight.

3.2.1 1-Polynomial root CSP over the rationals

We will start with d-POLYNOMIAL ROOT CSP over $\mathbb{Q}$ and over $\mathbb{Z}/m\mathbb{Z}$. We start by proving that EXACT RED-BLUE DOMINATING SET does not have generalized kernels of bitsize $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless NP $\subseteq$ coNP/poly. The same lower bound for both variants of 1-POLYNOMIAL ROOT CSP will follow by a linear-parameter transformation. We then show how to generalize this result to d-POLYNOMIAL ROOT CSP. As a starting problem for the cross-composition we will use the NP-hard RED-BLUE DOMINATING SET (RBDS) problem [34, 65].

RED-BLUE DOMINATING SET (RBDS)

Input: A bipartite graph $G = (R \cup B, E)$ containing red (R) and blue (B) vertices, and an integer k.
Question: Does there exist a set $D \subseteq R$ with $|D| \leq k$ such that every vertex in B has at least one neighbor in D?

EXACT RED BLUE DOMINATING SET (ERBDS) is defined similarly, except that every vertex in B must have *exactly one* neighbor in D. Furthermore we will not bound the size of such a set, but merely ask for the existence of any ERBDS.

Finally, we define a weakening of the notion of an ERBDS of a graph, called a SEMI-ERBDS. Given a bipartite graph G and set $S \subseteq B$, a set $X \subseteq V(G)$ is a SEMI-ERBDS of G with respect to S if it is a RBDS of G and furthermore, any blue vertex $x \notin S$ has exactly one neighbor in X. Vertices from S may be dominated multiple times.

The following lemma gives a degree-2 cross-composition from RBDS to SEMI-ERBDS, which will be used to prove Theorem 3.22. It is proven separately because the construction will also be used in the proofs of Theorems 3.26 and 3.27, which is also the reason we require part (4) of the lemma statement.

▶ **Lemma 3.14** *There exists a polynomial-time algorithm that, given t instances of* RBDS *with $\sqrt{t} \in \mathbb{N}$, labeled X_{ℓ_1,ℓ_2} with $\ell_1, \ell_2 \in [\sqrt{t}]$, which all ask for a solution of size k and all have m_R red and m_B blue vertices, constructs a bipartite graph G' with vertices partitioned into red (R) and blue (B) vertices, and a subset V of the blue vertices, such that the following holds:*

1. $|R| + |B| \leq \mathcal{O}(\sqrt{t} \cdot (m_R + m_B)^3)$.
2. *If there exist $\ell_1, \ell_2 \in [\sqrt{t}]$ such that X_{ℓ_1,ℓ_2} has a* RBDS *of size k, then G' has an* ERBDS.
3. *If G' has a* SEMI-ERBDS *with respect to V, then there exist $\ell_1, \ell_2 \in [\sqrt{t}]$ such that X_{ℓ_1,ℓ_2} has a* RBDS *of size k.*
4. *There are at most 2 vertices in $B \setminus V$ with degree more than $m_R + k + 2$.*

In particular, the lemma shows how to embed a series of t size-n instances $X_{\ell_1,\ell_2} = (G_{\ell_1,\ell_2}, k)$ for $\ell_1, \ell_2 \in [\sqrt{t}]$ that share the same target value k, into a single graph G' with $\mathcal{O}(\sqrt{t} \cdot \text{poly}(n))$ vertices such that G' has an ERBDS if and only if some input instance has a size-k RBDS. This straightforwardly gives a kernelization lower bound for ERBDS: since the number of output vertices is roughly $\sqrt{t}$, by choosing a suitable polynomial equivalence relation we get a degree-2 cross composition. Now the actual lemma statement is even stronger than the statement "some input has a RBDS $\Leftrightarrow$ G' has an ERBDS", because the $(\Leftarrow)$ implication already holds when G' has a SEMI-ERBDS. The fact that it is only required to be exact on a set of vertices $B \setminus V$ that has almost only small-degree vertices, will be used later. Later constructions "pay extra" for checking exactness of large-degree vertices, and the bound in (4) guarantees this does not happen too often.

Before proving the lemma, let us give the main ideas. The standard approach to give a degree-2 cross composition (see Section 2.4.3) is to have a table-like structure with sets of vertices U_ℓ consisting of m_R vertices and V_ℓ consisting of m_B vertices for all $\ell \in [\sqrt{t}]$. In this way we can add connections between U and V such that $G'[U_{\ell_1} \cup V_{\ell_2}]$ is isomorphic to G_{ℓ_1,ℓ_2}, thereby embedding the adjacency information of all t individual inputs while only needing $\sqrt{t} \cdot (m_R + m_B)$ vertices in the graph. Selector gadgets are then used to ensure that the

part of a (SEMI)-ERBDS in G' in U_{ℓ_1} for some ℓ_1 corresponds to a RBDS of size k in G_{ℓ_1,ℓ_2} for some ℓ_2. In the case of ERBDS however, difficulties arise when we try to use this type of construction. Given a RBDS for some input instance G_{ℓ_1,ℓ_2}, finding an ERBDS in G' can be problematic. The issue is that adding the vertices in U_{ℓ_1} corresponding to a solution in G_{ℓ_1,ℓ_2} to an ERBDS in G', may dominate some of the vertices from V multiple times. This is not easy to avoid, as there is simply no guarantee on how many times a vertex in the set V_ℓ with $\ell \neq \ell_2$ will be dominated by this choice of red vertices.

To resolve this problem, every set U_ℓ and V_ℓ has k copies of each vertex. Connections are made such that the i'th copy of a vertex may only connect to the i'th copy of another vertex, such that $G[U_{\ell_1} \cup V_{\ell_2}]$ contains k disjoint copies of G_{ℓ_1,ℓ_2}. To translate a RBDS in G_{ℓ_1,ℓ_2} to a ERBDS in G', we take at most one vertex from the i'th set of copies in U_{ℓ_1}. Hereby, any vertex in V is dominated at most once. Furthermore, for each vertex in V_{ℓ_2}, at least one of its copies is dominated. We add additional gadgets to ensure that the remaining vertices can also be dominated.

Proof of Lemma 3.14. Let instance X_{ℓ_1,ℓ_2} have graph G_{ℓ_1,ℓ_2}, with red vertices R_{ℓ_1,ℓ_2} and blue vertices B_{ℓ_1,ℓ_2}. For each input graph G_{ℓ_1,ℓ_2} enumerate the red vertices as $r_1,\ldots,r_{m_R}$ and the blue vertices as $b_1,\ldots,b_{m_B}$, arbitrarily. Create a graph G' by the following steps. Figure 3.1 shows a sketch of G'.

1. Create $\sqrt{t}$ sets $U_1,\ldots,U_{\sqrt{t}}$ each consisting of $k \cdot m_R$ red vertices, with $U_\ell := \{u^\ell_{i,j} \mid i \in [k], j \in [m_R]\}$ for each $\ell \in [\sqrt{t}]$. Let U be the union of all sets U_ℓ, for $\ell \in [\sqrt{t}]$.

2. Similarly create $\sqrt{t}$ sets $V_1,\ldots,V_{\sqrt{t}}$, each consisting of $k \cdot m_B$ blue vertices, and define $V_\ell := \{v^\ell_{i,j'} \mid i \in [k], j' \in [m_B]\}$ for all $\ell \in [\sqrt{t}]$. Let V be the union of all sets V_ℓ. Note that a SEMI-ERBDS wrt. V must dominate all blue vertices that are created in the remainder of the construction exactly once.

3. For each $i \in [k]$ add the edge from $u^{\ell_1}_{i,j}$ to $v^{\ell_2}_{i,j'}$ if $\{r_j, b_{j'}\}$ is an edge in instance X_{ℓ_1,ℓ_2} with $\ell_1, \ell_2 \in [\sqrt{t}]$, $j \in [m_R]$, and $j' \in [m_B]$.

By Steps 1 to 3, the subgraph of G' induced by the vertices in $U_{\ell_1} \cup V_{\ell_2}$ consists of k vertex-disjoint copies of G_{ℓ_1,ℓ_2}. The next steps are used to ensure that there are exactly k vertices from U in any SEMI-ERBDS, which must all belong to the same set U_ℓ. These vertices will correspond to a RBDS in one of the input instances.

4. Create blue vertices d^ℓ_i for $\ell \in [\sqrt{t}]$ and $i \in [k]$. Connect vertex d^ℓ_i to all vertices $u^\ell_{i,j}$ with $j \in [m_R]$. Define $D := \{d^\ell_i \mid \ell \in [\sqrt{t}], i \in [k]\}$. These blue vertices ensure that a SEMI-ERBDS wrt. V, which dominates each vertex of D

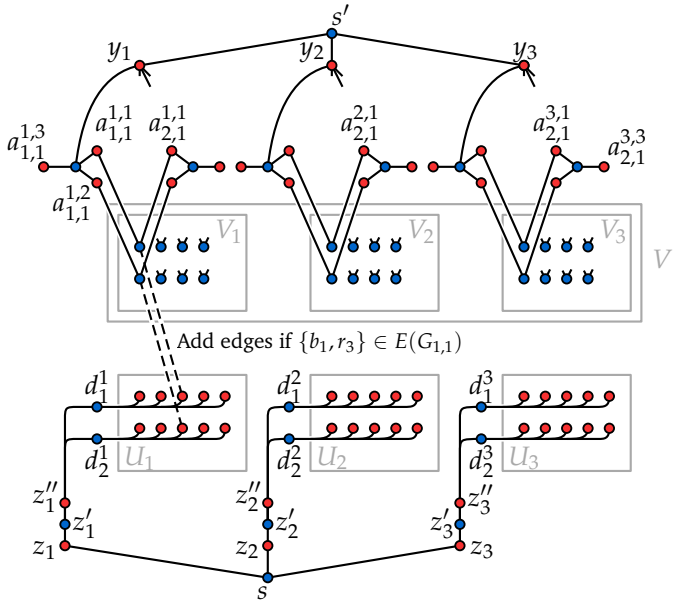


Figure 3.1 The graph G' created in the proof of Lemma 3.14, for $k = 2$, $m_R = 5$, $m_B = 4$, and $t = 9$. Edges between U and V are left out for simplicity. Of the 24 gadgets in C only $c^\ell_{1,1}$ and $c^\ell_{2,1}$ are shown for all $\ell \in [\sqrt{t}]$. Vertices in R are shown in red and vertices in B are shown in dark blue. The set V of vertices that may be dominated multiple times by a SEMI-ERBDS wrt. V is highlighted by a rectangle.

exactly once, cannot contain two vertices $u^\ell_{i,j}$ and $u^\ell_{i,j'}$ belonging to the same row of the same set U_ℓ.

5. Add blue vertex s and add the vertices $Z := \{z_\ell, z'_\ell, z''_\ell \mid \ell \in [\sqrt{t}]\}$. Let z_ℓ and z''_ℓ be red and let z'_ℓ be blue for all $\ell \in [\sqrt{t}]$. Connect z''_ℓ to d^ℓ_i for $i \in [k]$ and $\ell \in [\sqrt{t}]$. Add the edges $\{z_\ell, z'_\ell\}$ and $\{z'_\ell, z''_\ell\}$ for all $\ell \in [\sqrt{t}]$. Connect each vertex z_ℓ to s for $\ell \in [\sqrt{t}]$, thereby ensuring that exactly one vertex z_ℓ is contained in a SEMI-ERBDS wrt. V. Intuitively, the index ℓ_1 for which z_{ℓ_1} belongs to a SEMI-ERBDS controls the first index of the input instance X_{ℓ_1,ℓ_2} to which the solution corresponds.

The next steps ensure that some of the blue vertices in one set V_{ℓ_2} need to be dominated by vertices from U, while all other vertices in V can be dominated "for free". This will control the second index of the input instance X_{ℓ_1,ℓ_2} to which the solution corresponds.

7. Add sets of gadgets C_ℓ for $\ell \in [\sqrt{t}]$. Each set C_ℓ consists of $m_B \cdot k$ selector gadgets $c^\ell_{i,j'}$ for $i \in [k]$, $j' \in [m_B]$. Selector gadget $c^\ell_{i,j'}$ consists of $k+1$ red

vertices labeled $a^{\ell,1}_{i,j'}, \ldots, a^{\ell,k+1}_{i,j'}$ that are all connected to a blue vertex $b^{\ell}_{i,j'}$ that is the only blue vertex inside the gadget. Furthermore, for $j' \in [m_B]$, $\ell \in [\sqrt{t}]$ and $i \in [k]$, in gadget $c^{\ell}_{i,j'}$ the vertex $a^{\ell,x}_{i,j'}$ for $x \in [k]$ is connected to $v^{\ell}_{x,j'}$. We refer to the vertex set of gadget $c^{\ell}_{i,j'}$ by $V(c^{\ell}_{i,j'})$.

By Step 7 of the construction a SEMI-ERBDS uses at most one red vertex from each gadget, which can be used to dominate one vertex from V. Using vertex $a^{\ell,k+1}_{i,j'}$ of a gadget, the blue vertex of that gadget can be dominated without dominating any other blue vertices. Using the k gadgets introduced for $j' \in [m_B], \ell \in [\sqrt{t}]$, we can thus precisely dominate all vertices in $\{v^{\ell}_{i,j'} \mid i \in [k]\}$. Now we will ensure that there is a $\ell_2 \in [\sqrt{t}]$ such that in V_{ℓ_2}, for each $j' \in [m_B]$, one of the vertices $\{v^{\ell_2}_{i,j'} \mid i \in [k]\}$ is not dominated by a gadget and must therefore be dominated by a vertex from U.

8. Add red vertices $Y := \{y_1, \ldots, y_{\sqrt{t}}\}$. For each $\ell \in [\sqrt{t}]$ connect y_ℓ to the blue vertices of the gadgets $c^{\ell}_{1,j'}$ for all $j' \in [m_B]$, thereby making gadget $c^{\ell}_{i,j'}$ special for $i = 1$. Connect $y_1, \ldots, y_{\sqrt{t}}$ to the new blue vertex s', which ensures that exactly one vertex $y_{\ell_2} \in Y$ belongs to any SEMI-ERBDS wrt. V.

This concludes the construction of graph G', with red vertices

$$R := U \cup Y \cup \{z_\ell, z''_\ell \mid \ell \in [\sqrt{t}]\} \\ \cup \{a^{\ell,x}_{i,j'} \mid \ell \in [\sqrt{t}],\ x \in [k+1],\ i \in [k],\ j' \in [m_B]\},$$

and blue vertices

$$B := V \cup D \cup \{s, s'\} \cup \{b^{\ell}_{i,j'} \mid \ell \in [\sqrt{t}], i \in [k],\ j' \in [m_B]\} \cup \{z'_\ell \mid \ell \in [\sqrt{t}]\}.$$

The following observation follows immediately from the construction above.

Observation 3.15 *Let $x \notin V$ be a blue vertex in G'. The neighborhood of x depends only on m_R, m_B, t, and k; it is independent of the structure of the given input instances.*

Furthermore, we can show that requirement 4 of this lemma is satisfied.

▷ **Claim 3.16** *There are at most 2 vertices in $B \setminus V$ with degree more than $m_R + k + 2$.*

Proof. We list all vertices in $B \setminus V$, together with an upper bound on their degree.

Vertices s and s': It follows from Steps 5 and 8 that these two vertices both have large degree, namely $\sqrt{t}$.

Vertices in D: It follows from Steps 4 and 5 that these vertices have degree $m_R + 1$.

Vertices in $Z \cap B$: It follows from Step 5 that vertex z'_ℓ has degree two for all $\ell \in [\sqrt{t}]$.

Vertices in gadgets: The blue vertex of any gadget has degree at most $k+2$, the incident edges are added in Steps 7 and 8.

Thus there are at most 2 vertices of degree larger than $m_R + k + 2$ in $B \setminus V$. ◁

We now continue by providing a number of relevant properties of the graph G', that will be used to prove that the construction satisfies requirement 3 of the lemma statement.

▷ **Claim 3.17** *For any* SEMI-ERBDS *E of G' wrt. V, there exists an index $\ell_1 \in [\sqrt{t}]$ such that $U_\ell \cap E = \emptyset$ for all $\ell \neq \ell_1 \in [\sqrt{t}]$ and $|E \cap \{u^{\ell_1}_{i,j} \mid j \in [m_R]\}| = 1$ for all $i \in [k]$.*

Proof. By Step 5, blue vertex $s \notin V$ has neighborhood $\{z_\ell \mid \ell \in [\sqrt{t}]\}$. Since the semi-exact RBDS is exact on blue vertices outside V, exactly one neighbor of s is contained in E; let this be z_{ℓ_1}. Thereby, for all $\ell \in [\sqrt{t}]$ with $\ell \neq \ell_1$ we obtain $z_\ell \notin E$. Since blue vertex z'_ℓ has neighborhood exactly $N_{G'}(z'_\ell) = \{z_\ell, z''_\ell\}$, it follows that $z''_\ell \in E$ for all $\ell \neq \ell_1$ with $\ell \in [\sqrt{t}]$.

Let $\ell \in [\sqrt{t}]$ with $\ell \neq \ell_1$, we show that no vertex in U_ℓ is in E. Consider vertex $u^\ell_{i,j}$ with $i \in [k]$, $j \in [m_R]$. Then $u^\ell_{i,j} \in N_{G'}(d^\ell_i)$ for blue vertex d^ℓ_i. Since $z''_\ell \in N_{G'}(d^\ell_i)$ and $z''_\ell \in E$, it follows that $u^\ell_{i,j} \notin E$.

It remains to show that $|E \cap \{u^{\ell_1}_{i,j} \mid j \in [m_r]\}| = 1$ for all $i \in [k]$. Since $N_{G'}(z'_{\ell_1}) = \{z_{\ell_1}, z''_{\ell_1}\}$ and $z_{\ell_1} \in E$, it follows that $z''_{\ell_1} \notin E$. As $d^{\ell_1}_i \in B \setminus V$ and E is an exact RBDS on vertices outside V, it follows that $|E \cap N_{G'}(d^{\ell_1}_i)| = 1$ for all $i \in [k]$. Since $z''_{\ell_1} \notin E$, it thereby follows that $|E \cap \{u^{\ell_1}_{i,j} \mid j \in [m_R]\}| = 1$ for all $i \in [k]$. ◁

▷ **Claim 3.18** *For any* SEMI-ERBDS *E of G' wrt. V, there exists an index $\ell_2 \in [\sqrt{t}]$ such that $E \cap V(c^{\ell_2}_{1,j'}) = \emptyset$ for all $j' \in [m_B]$.*

Proof. By Step 8, blue vertex s' has neighborhood $\{y_\ell \mid \ell \in [\sqrt{t}]\}$. Since $s' \notin V$, exactly one of these vertices is contained in E; let this be y_{ℓ_2}. It is connected to the blue vertex of all gadgets $c^{\ell_2}_{1,j'}$ for $j' \in [m_B]$. Since all red vertices in a gadget $c^{\ell_2}_{1,j'}$ for $j' \in [m_B]$ have the blue neighbor $b^{\ell_2}_{1,j'}$ that is also adjacent to $y_{\ell_2} \in E$, the red vertices in these gadgets are not present in E, as $b^{\ell_2}_{1,j'}$ has exactly one red neighbor in E. ◁

▷ **Claim 3.19** *For any* SEMI-ERBDS E *of* G' *wrt.* V*, there exists an index* $\ell_2 \in [\sqrt{t}]$ *such that for every* $j' \in [m_B]$ *at least one of the vertices in* $\{v^{\ell_2}_{i,j'} \mid i \in [k]\}$ *has a neighbor in* $E \cap U$*.*

Proof. By Claim 3.18 there exists $\ell_2 \in [\sqrt{t}]$ such that $E \cap V(c^{\ell_2}_{1,j'}) = \emptyset$ for all $j' \in [m_B]$. Consider an arbitrary $j' \in [m_B]$. The k vertices in $\{v^{\ell_2}_{i,j'} \mid i \in [k]\}$ are connected to vertices of the k gadgets $c^{\ell_2}_{1,j'}, c^{\ell_2}_{2,j'}, \ldots, c^{\ell_2}_{k,j'}$, and to some vertices in U. From each gadget, at most one red vertex is in E, since the red vertices have a common blue neighbor that is not in V. Any red gadget vertex is connected to only one vertex in V. Since no vertex of gadget $c^{\ell_2}_{1,j'}$ is in E, at most $k-1$ of the vertices in $\{v^{\ell_2}_{i,j'} \mid i \in [k]\}$ have a neighbor in $E \cap C_{\ell_2}$. Consequently, at least one of these vertices has a neighbor in $E \cap U$ for each $j' \in [m_B]$. ◁

We can now prove that G' and V fulfill requirement 3 of the lemma statement.

▷ **Claim 3.20** *If* G' *has a* SEMI-ERBDS *wrt.* V*, then some input* X_{ℓ_1,ℓ_2} *has a* RBDS *of size at most* k*.*

Proof. Assume G' has a SEMI-ERBDS wrt. V, say E. By Claim 3.19, there exists $\ell_2 \in [\sqrt{t}]$, such that for every $j' \in [m_B]$ at least one of the vertices in $\{v^{\ell_2}_{i,j'} \mid i \in [k]\}$ has a neighbor in $E \cap U$. By Claim 3.17, there exists $\ell_1 \in [\sqrt{t}]$ such that for all $\ell \neq \ell_1$ we have $U_\ell \cap E = \emptyset$, so these neighbors lie in U_{ℓ_1}.

We now construct a RBDS E' for instance X_{ℓ_1,ℓ_2}. For each $j \in [m_R]$, add r_j to E' if $E \cap \{u^{\ell_1}_{i,j} \mid i \in [k]\} \neq \emptyset$. By Claim 3.17, it follows that E' has size at most k, as required. It remains to show that every vertex in B_{ℓ_1,ℓ_2} has a neighbor in E'. If some vertex $b_{j'}$ from B_{ℓ_1,ℓ_2} does not have a neighbor in E', then none of the vertices $\{v^{\ell_2}_{i,j'} \mid i \in [k]\}$ have a neighbor in $E \cap U_{\ell_1}$. This contradicts our choice of ℓ_2. Hence E' is an RBDS of size at most k for instance X_{ℓ_1,ℓ_2}. ◁

Furthermore we show that requirement 2 is fulfilled in the following claim.

▷ **Claim 3.21** *If some input instance has a* RBDS *of size at most* k*, then* G' *has an* ERBDS*.*

Proof. Suppose instance X_{ℓ_1,ℓ_2} has a RBDS E' of size k consisting of vertices $r_{i_1}, \ldots, r_{i_k} \subseteq R_{\ell_1,\ell_2}$. We construct an ERBDS E for G'. Start by choosing vertices $u^{\ell_1}_{x,i_x}$ for $x \in [k]$, so for every vertex in E' we pick one vertex in the ERBDS for G'. Add the red vertex z_{ℓ_1} and the vertices z''_ℓ for all $\ell \neq \ell_1$ to E. Furthermore, we let the vertex y_{ℓ_2} be in E.

To exactly dominate the blue vertices in V, we use the gadgets in C as follows. For $\ell \neq \ell_2 \in [\sqrt{t}]$, add red vertex $a^{\ell,x}_{x,j'}$ of gadget $c^{\ell}_{x,j'}$ if vertex $v^{\ell}_{x,j'}$ does

not yet have a neighbor in E, for $j' \in [m_B]$ and $x \in [k]$. Else, add vertex $a^{\ell,k+1}_{x,j'}$ of gadget $c^{\ell}_{x,j'}$ to E, in order to exactly dominate the blue vertex of this gadget.

To exactly dominate the vertices in V_{ℓ_2} we apply a similar procedure, except that gadget $c^{\ell_2}_{1,j'}$ cannot be used since its blue vertex $b^{\ell_2}_{1,j'}$ is already dominated by y_{ℓ_2}. Since E' is a RBDS of instance X_{ℓ_1,ℓ_2}, for each $j' \in [m_B]$ at least one vertex from set $\{v^{\ell_2}_{i,j'} \mid i \in [k]\}$ has a neighbor in $E \cap U$. As such, the $k-1$ remaining gadgets can be used to each dominate one of the $k-1$ remaining vertices in this set, if they do not already have a neighbor in $E \cap U$. If no red vertex of a gadget $c^{\ell_2}_{x,j'}$ is needed to dominate, we choose vertex $a^{\ell_2,k+1}_{x,j'}$ of the gadget in E to dominate the blue vertex in the gadget.

It is straight-forward to verify that this results in an ERBDS for G'. ◁

From Claims 3.20 and 3.21 it follows that graph G' has a SEMI-ERBDS wrt. V if and only if at least one of the input instances has a RBDS of size at most k. The graph G' has $\mathcal{O}(\sqrt{t} \cdot (m_R + m_B)^3)$ vertices and can be constructed in polynomial time. ◀

Using the lemma above, we now prove the kernel lower bound for ERBDS.

▶ **Theorem 3.22** EXACT RED-BLUE DOMINATING SET *parameterized by the number of vertices n does not have a generalized kernel of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless* NP $\subseteq$ coNP/poly.

Proof. We will prove this result by giving a degree-2 cross-composition from RBDS to ERBDS. We start by giving a polynomial equivalence relation $\mathcal{R}$ on inputs of RBDS. Let two instances of RBDS be equivalent under $\mathcal{R}$ if they have the same number of red vertices m_R, the same number of blue vertices m_B, and the same maximum size k of a RBDS. It is easy to check that $\mathcal{R}$ is a polynomial equivalence relation.

Assume we are given t instances of RBDS, labeled X_{ℓ_1,ℓ_2} for $\ell_1, \ell_2 \in [\sqrt{t}]$, from the same equivalence class of $\mathcal{R}$. If the number of instances given is not a square, we duplicate one of the input instances until a square number is reached. Since this changes the number of inputs by at most a factor four, this does not influence the cross-composition. Call the number of red vertices in every instance m_R, the number of blue vertices m_B, and the required size of the dominating set k. By Lemma 3.14, we can in polynomial time construct graph G' such that

- $|V(G)| \leq \sqrt{t} \cdot \text{poly}(m_B + m_R)$ and
- G' has an ERBDS if and only if at least one input instance has a RBDS. This follows from requirements 2 and 3 from Lemma 3.14, and the fact that any ERBDS is also a SEMI-ERBDS.

Thereby we have given a degree-2 cross-composition and the lower bound follows from Theorem 2.14. ◀

Using Theorem 3.22 we provide lower bounds for constraint satisfaction problems. It is easy to give a linear-parameter transformation from ERBDS to both 1-POLYNOMIAL ROOT CSP and EXACT SAT, by introducing a variable for each red vertex and adding a constraint for each blue vertex such that exactly one of its neighbors is chosen in any assignment. Using Theorem 2.8, this results in the following corollary.

▶ **Corollary 3.23** *The problems* EXACT SAT *and* 1-POLYNOMIAL ROOT CSP *over* $\mathbb{Q}$*, parameterized by the number of variables n, do not have a generalized kernel of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless* NP $\subseteq$ coNP/poly. ◀

3.2.2 Polynomial root CSP (modulo an integer)

In order to also establish a lower bound for 1-POLYNOMIAL ROOT CSP over the integers modulo m, we will need the following lemma. It allows us to enforce a linear equality constraint over $\mathbb{Q}$ using constraints over $\mathbb{Z}/m\mathbb{Z}$, through the use of auxiliary $0/1$-dummy variables. Since $\sum_i x_i = 1$ implies $\sum_i x_i \equiv_m 1$, the non-trivial part is to add extra constraints which, together with $\sum_i x_i \equiv_m 1$, also imply $\sum_i x_i = 1$.

▶ **Lemma 3.24** *Let $m \geq 3$ be an integer. Given a linear equality $\sum_{i\in[N]} x_i = 1$ over $\mathbb{Q}$, there exists a system S of linear equalities over $\mathbb{Z}/m\mathbb{Z}$ using the variables $\{x_i \mid i \in [N]\}$ and at most $4N$ additional variables, such that*

1. *Any $0/1$-solution to the system S sets exactly one of the variables $\{x_i \mid i \in [N]\}$ to 1,*
2. *any assignment to $\{x_i \mid i \in [N]\}$ setting exactly one variable x_i to 1 can be extended to a $0/1$-solution of S, and*
3. *S can be constructed in polynomial time.*

Proof. Given the linear equality $\sum_{i\in[N]} x_i = 1$, first of all add the equation

$$\sum_{i\in[N]} x_i \equiv_m 1$$

to S. Any choice of $x_1, \ldots, x_N$ satisfying $\sum_{i\in[N]} x_i = 1$ also satisfies the equality modulo m. Furthermore, any $0/1$-assignment of $x_1, \ldots, x_N$ satisfying the equation $\sum_{i\in[N]} x_i \equiv_m 1$ ensures that *at least* one variable x_i is set to 1.

To ensure that *at most* one of these variables is set to 1, we add additional constraints in the following way. Construct a complete binary tree with $N' := 2^{\lceil \log N \rceil}$ leaves, implying $N \leq N' < 2N$. Identify the first N leaves with variables

$x_1, \ldots, x_N$ and introduce dummy variables for all other vertices. For every non-leaf d in the tree with children d_ℓ and d_r, each corresponding to a unique variable, add the equation

$$d_\ell + d_r \equiv_m d.$$

It is clear that this construction can be done in polynomial time, thus Property 3 holds. To show that Properties 1 and 2 hold, we prove the following claim.

▷ **Claim 3.25** *Let a* $0/1$*-assignment satisfying all equalities in* S *be given. The value assigned to any variable* x *corresponds to the number of leaves in the subtree rooted in* x *that are assigned value* 1.

Proof. We prove this by induction on the height of the tree rooted in x. If x is a leaf, the result is obvious. Suppose the tree has height larger than one and let x_ℓ and x_r be the left and right child of x. By the induction hypothesis, the values of x_ℓ and x_r correspond to the number of leaves in the left (respectively, right) subtree that were assigned 1. Since $x_\ell, x_r \in \{0,1\}$ and $x \equiv_m x_\ell + x_r$ with $m > 2$, the result follows. ◁

Suppose we are given any $0/1$-assignment satisfying all equalities in S. Hence the variable corresponding to the root r of the binary tree has value 0 or 1. By Claim 3.25, it follows that the number of leaves (and thus the number of variables in $\{x_1, \ldots, x_N\}$) that are assigned the value 1 is at most one. As we have seen earlier, at least one variable x_i is set to 1, to fulfill $\sum_{i\in[N]} x_i \equiv_m 1$, and thus $\sum_{i\in[N]} x_i = 1$. Hence Property 1 holds.

Given a $0/1$-assignment to $x_1, \ldots, x_N$ such that $\sum_{i\in[N]} x_i = 1$, it can be extended to a satisfying assignment of S by setting all dummy leaves to 0. For every other dummy vertex, let its value be the number of variables corresponding to leaves in its subtree, that are set to 1. Note that this number is always either 0 or 1 since there is only one leaf whose corresponding variable is set to 1. Therefore Property 2 holds as well. ◀

For $m = 2$, an input to the problem 1-Polynomial root CSP over the integers mod m only consists of linear equations over the two-element field $\{0,1\}$ and is thus polynomial time solvable by Schaefer's dichotomy theorem (Theorem 2.37). For larger moduli, we use Lemma 3.24 to prove the following result.

► **Theorem 3.26** *Let* $m \geq 3$ *be an integer. The problem* 1-Polynomial root CSP *over* $\mathbb{Z}/m\mathbb{Z}$*, parameterized by the number of variables* n*, does not have a generalized kernel of size* $\mathcal{O}(n^{2-\varepsilon})$ *for any* $\varepsilon > 0$*, unless* NP $\subseteq$ coNP/poly.

Proof. We will use the graph constructed in Lemma 3.14, by transforming the constructed instance G' of (semi)-erbds of size $\mathcal{O}(\sqrt{t} \cdot \text{poly}(m_R + m_B))$ to an instance I of 1-Polynomial root CSP over $\mathbb{Z}/m\mathbb{Z}$ with $\mathcal{O}(\sqrt{t} \cdot \text{poly}(m_R + m_B))$ variables. In this way we obtain a degree-2 cross-composition from rbds to 1-Polynomial root CSP over $\mathbb{Z}/m\mathbb{Z}$, proving the lower bound.

Suppose we are given t instances of RBDS, such that $\sqrt{t}$ is integer and such that every instance has m_B blue vertices and m_R red vertices and asks for a RBDS of size $k \leq m_R$. This can be assumed by choosing an appropriate polynomial equivalence relation. Apply Lemma 3.14 to obtain graph G' and $V \subseteq V(G')$. By requirements 2 and 3 of Lemma 3.14, it is sufficient to ensure that G' has a SEMI-ERBDS with respect to V if I is satisfiable, and that I is satisfiable if G has an ERBDS, to obtain the cross-composition.

Recall that a SEMI-ERBDS of G' with respect to V contains at least one neighbor of each blue vertex, and contains *exactly* one neighbor of each blue vertex in $V(G') \setminus V$.

First of all introduce a variable v_r for every red vertex r in G'. For every blue vertex b, we add the following equation to ensure that it has at least one neighbor in the SEMI-ERBDS:

$$\sum_{r \in N_{G'}(b)} v_r \equiv_m 1. \tag{3.1}$$

For every blue vertex $b \notin V$, we add a number of linear equations that ensure b has exactly one neighbor in a SEMI-ERBDS, using at most $4 \cdot |N_{G'}(b)|$ additional variables. This is done by applying Lemma 3.24 to the equation $\sum_{r \in N_{G'}(b)} v_r = 1$.

This completes the construction. If G' has an ERBDS, then I can be satisfied by setting the variables corresponding to the ERBDS to 1 and all other variables corresponding to vertices to 0. The dummy variables can then be chosen in such a way that all equations are satisfied according to Lemma 3.24.

For the opposite direction, suppose I has a satisfying assignment. Define set Y to contain the vertices whose corresponding variable is set to 1. From Equation (3.1) it follows that every blue vertex has at least one neighbor in the set Y. Furthermore every blue vertex not in V has exactly one neighbor in Y by Lemma 3.24. It follows that Y is a SEMI-ERBDS of G'.

It remains to bound the number of used variables. The key idea is that we have only few variables outside of V whose corresponding vertex has a large neighborhood, and for which the number of dummy variables added depends on $\sqrt{t}$. Furthermore there are many variables whose corresponding vertices have small neighborhoods, with size depending only on $m_B + m_R$. Note that the degree of any vertex, and the total number of vertices, is bounded by the order of the graph $\mathcal{O}(\sqrt{t} \cdot (m_R + m_B)^3)$.

For every blue vertex in $V(G) \setminus V$ with a degree larger than $m_R + k + 2$ we add $\mathcal{O}(\sqrt{t}(m_B + m_R)^3)$ dummy variables. By requirement 4 of Lemma 3.14, there are at most 2 such vertices. Furthermore for any vertex with a degree smaller than $m_R + k + 2$ we add $\mathcal{O}(m_R + k)$ dummy vertices. This together results in using $\mathcal{O}(\sqrt{t} \cdot \text{poly}(m_B + m_R))$ variables, which is properly bounded for a degree-2 cross-composition. ◀

We now generalize this result to polynomial equalities of higher degree.

► **Theorem 3.27** *Let $m \geq 2$, and $d \geq 2$ be integers. The problems d-*POLYNOMIAL ROOT CSP *over $\mathbb{Z}/m\mathbb{Z}$ and d-*POLYNOMIAL ROOT CSP *over $\mathbb{Q}$ parameterized by the number of variables n do not have a generalized kernel of size $\mathcal{O}(n^{d+1-\varepsilon})$ for any $\varepsilon > 0$, unless* NP $\subseteq$ coNP/poly.

Proof. We will only provide the proof over $\mathbb{Z}/m\mathbb{Z}$, the result for d-POLYNOMIAL ROOT CSP over $\mathbb{Q}$ can be obtained in the same way (using the same equations without the moduli). Let $m \geq 2, d \geq 2$ be given. The result will be proven by a degree-$(d+1)$ cross-composition from RBDS, using Lemma 3.14. Suppose we are given $t = r^{d+1}$ instances of RBDS, all having m_R red vertices, m_B blue vertices, and the same target size k. By a similar padding argument as before, we may assume r is an integer. Split the inputs into r^{d-1} groups of size r^2 each and apply the algorithm given by Lemma 3.14 to each group. We obtain r^{d-1} instances of (SEMI)-ERBDS with $\mathcal{O}(r \cdot (m_R + m_B)^3)$ vertices each, such that the answer to each composed instance is the logical OR of the answers to the RBDS instances in its group. Label the instances resulting from the group compositions $X_{i_1,\ldots,i_{d-1}}$ with $i_1, \ldots, i_{d-1} \in [r]$. Let instance $X_{i_1,\ldots,i_{d-1}}$ have graph $G_{i_1,\ldots,i_{d-1}}$ and let the set on which the RBDS is not required to be exact be $V_{i_1,\ldots,i_{d-1}}$. All produced graphs have the same number of red and blue vertices; let the number of red vertices in each graph be $N \leq |V(G_{i_1,\ldots,i_{d-1}})| \leq \mathcal{O}(r \cdot (m_R + m_B)^3)$. We create N new variables and identify each red vertex x with one variable v_x. It is essential for the remaining part of this proof that vertices from the produced (SEMI)-ERBDS instances that had the same label are mapped to the same new variable and vice versa. Since the set $V_{i_1,\ldots,i_{d-1}}$ that is produced by Lemma 3.14 does not depend on the structure of the input graphs, only on their size, all produced graphs have the same labeled vertices in the set $V_{i_1,\ldots,i_{d-1}}$. Hence we can treat it as a single set V of vertex labels. Create an instance for d-POLYNOMIAL ROOT CSP as follows.

1. Add sets $Y_1, \ldots, Y_{d-1}$ of r variables each, where $Y_i := \{y_{i,j} \mid j \in [r]\}$. Add the requirement $\sum_{j \in [r]} y_{i,j} \equiv_m 1$ to L' for each $i \in [d-1]$.
2. Consider each graph $G_{i_1,\ldots,i_{d-1}}$ for $i_1, \ldots, i_{d-1} \in [r]$. For each blue vertex b in this instance, add the following equation to L':

$$\left(\sum_{x \in N_{G_{i_1,\ldots,i_{d-1}}}(b)} v_x \right) \cdot \prod_{z \in [d-1]} y_{z,i_z} \equiv_m \prod_{z \in [d-1]} y_{z,i_z}. \tag{3.2}$$

Furthermore, if b is not an element of V, then for every pair of distinct vertices $x, x' \in N(b)$ add the following constraint to L':

$$v_x \cdot v_{x'} \equiv_m 0. \tag{3.3}$$

Note that, by Observation 3.15, the neighborhood of a blue vertex $b \notin V$ does not depend on the graphs to which Lemma 3.14 is applied, but only on the number of red and blue vertices and the target size of the RBDS. As these are identical for all applications of the lemma, it does not matter in which of the graphs we evaluate $N(b)$ when finding relevant pairs x, x'.

The polynomial equalities have degree $\leq d$ as d is at least two. The number of variables, which is the parameter of the CSP, is suitably bounded for a degree-$(d+1)$ cross-composition:

$$N + (d-1) \cdot r \in \mathcal{O}(r \cdot (d + (m_R + m_B)^3)) = \mathcal{O}(t^{1/(d+1)}(m_R + m_B)^3).$$

As the construction can easily be performed in polynomial time, it remains to show that the constraints in L' can be satisfied if and only if one of the input instances of RBDS has a solution of size k. First assume that some input instance of RBDS indeed has a solution of size k. Consider the indices $i_1, \ldots, i_{d-1}$ of the group containing the satisfiable RBDS instance. Then Lemma 3.14 ensures that $G_{i_1,\ldots,i_{d-1}}$ has an ERBDS. Set the variables corresponding to vertices in the ERBDS of $G_{i_1,\ldots,i_{d-1}}$ to 1 and the others to 0. Furthermore, set variables y_{z,i_z} for $z \in [d-1]$ to 1. Set all other variables to 0. Thereby the sum of variables in each set Y_i is 1, as required. Furthermore, each equation defined by (3.2) is satisfied in the following way. If it was defined for $X_{i_1,\ldots,i_{d-1}}$, it is satisfied since the large summation equals one (exactly one neighbor is in the exact dominating set) and the product term is one on both sides. Equations belonging to any other instance are trivially satisfied since the term $\prod_z y_{z,j_z}$ is zero on both sides if there exists $z \in [d-1]$ for which $j_z \neq i_z$. It remains to show that the equations defined by (3.3) are satisfied. This is follows from Observation 3.15 and the fact that an ERBDS contains at most one neighbor of each blue vertex.

For the reverse direction, suppose the constraints in L' are satisfied by some 0/1-assignment to the variables. Then from each set Y_i with $i \in [d-1]$, at least one variable is set to 1. So suppose variables y_{z,i_z} are set to 1 for $z \in [d-1], i_z \in [r]$. We show instance $X_{i_1,\ldots,i_{d-1}}$ has a SEMI-ERBDS wrt. V consisting of the vertices whose corresponding variable is set to 1. Since the product $\prod_{z\in[d-1]} y_{z,i_z}$ is 1 on both sides of the equations defined by (3.2) for $G_{i_1,\ldots,i_{d-1}}$, for each blue vertex b in the graph we have:

$$\sum_{x \in N_{G_{i_1,\ldots,i_{d-1}}}(b)} v_x \equiv_m 1$$

implying all blue vertices have at least one neighbor in the SEMI-ERBDS. Furthermore if $x \notin V$, we know that it has at most one neighbor in the SEMI-ERBDS since the multiplication of any two of its neighbors yields zero by (3.3). Hence $G_{i_1,\ldots,i_{d-1}}$ has a SEMI-ERBDS wrt. V. By Lemma 3.14, this implies the group of RBDS instances from which it was constructed contained a satisfiable instance. Hence there was a *yes*-instance among the inputs of the cross-composition. ◀

Observe that the polynomials constructed in Theorem 3.27 have a simple form: each polynomial is a product of $(d-1)$ Y-variables multiplied by a sum of variables corresponding to red vertices, or simply a multiplication of two variables corresponding to red vertices. Each polynomial can therefore be encoded in $\widetilde{\mathcal{O}}(n)$ bits, where n is the number of variables in the constructed CSP. The sparsification of Theorem 3.1 therefore encodes such instances in $\widetilde{\mathcal{O}}(n^{d+1})$ bits. The lower bound shows that this is optimal up to $n^{o(1)}$ factors.

3.2.3 Polynomial non-root CSP

We start our lower bound discussion for d-POLYNOMIAL NON-ROOT CSP by considering polynomials over the rationals. Using existing kernel lower bounds for CNF-SAT parameterized by the number of variables, we first show that 1-POLYNOMIAL NON-ROOT CSP over $\mathbb{Q}$ does not have a generalized kernel of size bounded by any polynomial in n, unless NP $\subseteq$ coNP/poly.

▶ **Theorem 3.28** *1-POLYNOMIAL NON-ROOT CSP over $\mathbb{Q}$ parameterized by the number of variables n does not have a generalized kernel of polynomial size unless* NP $\subseteq$ coNP/poly.

Proof. We present a linear-parameter transformation from CNF-SAT with unbounded clause length parameterized by the number of variables. Existing results [33, 41] imply that this problem does not have a generalized kernel of polynomial size. The linear-parameter transformation will transfer this lower bound to 1-POLYNOMIAL NON-ROOT CSP over $\mathbb{Q}$.

A clause in conjunctive normal form can directly be translated into a non-root constraint of a degree-1 polynomial over $\mathbb{Q}$. For example, the clause $(x_1 \vee \neg x_3 \vee x_4)$ is satisfied by a 0/1-assignment if and only if $x_1 + (1 - x_3) + x_4 \neq 0$ over $\mathbb{Q}$. More generally, a clause $(x_{i_1} \vee \ldots \vee x_{i_k} \vee \neg x_{i_{k+1}} \vee \ldots \vee \neg x_{i_\ell})$ translates into the constraint $(\sum_{j=1}^{k} x_{i_j}) + (\sum_{j=k+1}^{\ell}(1 - x_{i_j})) \neq 0$. Hence the system of disequalities derived by transforming all clauses in a CNF-formula is satisfiable if and only if the formula is. As the number of variables is preserved by this transformation, the theorem follows. ◀

We now turn our attention to d-POLYNOMIAL NON-ROOT CSP over finite rings and fields. In Theorem 3.13 we provided a kernel for d-POLYNOMIAL NON-ROOT CSP over $\mathbb{Z}/p\mathbb{Z}$ for primes p. It is natural to ask whether similar results can be obtained when working with polynomials modulo an arbitrary integer m. When m is composite, our kernelization fails. We can show that this is not a shortcoming of our proof strategy, but a necessity due to the fact that constraints expressed by degree-d polynomials modulo composite numbers can model more complex constraints than degree-d polynomials modulo a prime. For example, it is known (cf. [7, §2]) that there is a degree-3 polynomial f over the integers

modulo 6 which represents a logical OR of size 27 in the following way:

$$f(x_1, \ldots, x_{27}) \not\equiv_6 0 \Leftrightarrow (x_1 \vee \ldots \vee x_{27}). \tag{3.4}$$

By this expressibility of a size-27 OR by a polynomial of degree 3 over $\mathbb{Z}/6\mathbb{Z}$ using the same variables, one easily constructs a linear-parameter transformation from 27-CNF-SAT to 3-POLYNOMIAL NON-ROOT CSP over $\mathbb{Z}/6\mathbb{Z}$ by mimicking the proof of Theorem 3.28. Since 27-CNF-SAT does not have a kernel of size $\mathcal{O}(n^{27-\varepsilon})$ for any $\varepsilon > 0$ unless NP $\subseteq$ coNP/poly (Theorem 2.9), this linear-parameter transformation rules out kernels of size $\mathcal{O}(n^{27-\varepsilon})$ for 3-POLYNOMIAL NON-ROOT CSP over $\mathbb{Z}/6\mathbb{Z}$ under the same conditions. Plugging in the degree of 3 and modulus 6 into the bound of Theorem 3.13 would give a reduction to $\mathcal{O}(n^{3\cdot(6-1)}) = \mathcal{O}(n^{15})$ constraints and would contradict the lower bound. The example therefore shows that the problem is more complex for composite moduli: the bound for the prime case cannot be matched. In particular, we will see that the exponent in the kernel size may depend super-linearly on the degree d of the CSP. For general non-primes, we give a lower bound using a construction by Bhowmick *et al.* [11] of low-degree polynomials representing OR in the sense of Equation (3.4).

▶ **Theorem 3.29** *Let m be a non-prime with a prime factorization consisting of r distinct primes, such that $m = \prod_{i\in[r]} p_i$. Let d be an even integer. Then d-POLYNOMIAL NON-ROOT CSP over $\mathbb{Z}/m\mathbb{Z}$ parameterized by the number of variables n does not have a generalized kernel of size $\mathcal{O}(n^{(d/2)^r-\varepsilon})$ for any $\varepsilon > 0$, unless* NP $\subseteq$ coNP/poly.

Proof. For any integer $N \geq 1$, Bhowmick *et al.* [11, Appendix A] provide a way to construct a polynomial f of degree $2\lceil N^{1/r}\rceil$ such that for all $x_1, \ldots, x_N \in \{0,1\}$,

$$f(x_1, \ldots, x_N) \not\equiv_m 0 \Leftrightarrow (x_1 \vee \cdots \vee x_N).$$

This implies that for even values of d and $N = (d/2)^r$, we can find a polynomial f of degree d satisfying the above equation. As such, d-POLYNOMIAL NON-ROOT CSP can express a logical OR of size $(d/2)^r$ without introducing auxiliary variables. As in the proof of Theorem 3.28, this gives a linear-parameter transformation from $(d/2)^r$-CNF-SAT to d-POLYNOMIAL NON-ROOT CSP. By Theorem 2.9, the latter problem does not have a generalized kernel of size $\mathcal{O}(n^{(d/2)^r-\varepsilon})$ for any $\varepsilon > 0$, unless NP $\subseteq$ coNP/poly. Hence the same lower bound applies to the CSP. ◀

In case m does not have a prime factorization in which all primes are distinct, it is possible to obtain weaker a lower bound using a result by Barrington *et al.* [8], which proves that there exists a polynomial of degree $\mathcal{O}(\ell N^{1/r})$ that represents a logical OR when taken modulo m. Here ℓ is the largest prime factor of m. For prime moduli, the following result provides a lower bound almost matching the upper bound in Theorem 3.13.

▶ **Theorem 3.30** *Let p be a prime. Then d-*POLYNOMIAL NON-ROOT *CSP over $\mathbb{Z}/p\mathbb{Z}$ parameterized by the number of variables n does not have a generalized kernel of size $\mathcal{O}(n^{d(p-1)-\varepsilon})$ for any $\varepsilon > 0$, unless* NP $\subseteq$ coNP/poly.

Proof. We use a linear-parameter transformation from $d(p-1)$-CNF-SAT. We proceed similarly as in the proof of Theorem 3.29. It is known (cf. [9, Theorem 24]) that for each prime p and integer d, there is a polynomial f of degree d modulo p, such that for any $x_1, \ldots, x_{d(p-1)} \in \{0,1\}$ we have:

$$f(x_1, \ldots, x_{d(p-1)}) \not\equiv_p 0 \Leftrightarrow (x_1 \vee x_2 \vee \cdots \vee x_{d(p-1)}).$$

This allows the linear-parameter transformation to be carried out as in Theorem 3.29. ◀

3.3 Conclusion

We have given upper and lower bounds on the kernelization complexity of Boolean CSPs that can be represented by polynomial (dis)equalities, obtaining (nearly) tight sparsification bounds in several cases. For d-POLYNOMIAL NON-ROOT CSP over the integers modulo a prime, there is a factor n difference between the upper and lower bound. It would be interesting to see whether this can be resolved.

Our main conceptual contribution is to analyze constraints on Boolean variables based on the minimum degree of multivariate polynomials whose roots, or non-roots, capture the satisfying assignments. The ultimate goal of this line of research is to characterize the optimal sparsification size of a Boolean CSP based on easily accessible properties of the constraint language. To reach this goal, several significant hurdles have to be overcome.

One such issue is that we do not have upper bounds for d-POLYNOMIAL NON-ROOT CSP modulo a composite number m. For example, for d-POLYNOMIAL NON-ROOT CSP over the integers modulo 6, we do not know of any way to reduce the number of constraints to polynomial in n. This difficulty is connected to longstanding questions regarding the minimum degree of a multivariate polynomial modulo 6 that represents the OR-function of n variables in the sense of Equation (3.4). In the lower bound construction in Theorem 3.29, we will use that if the OR-function with $g(d)$ inputs can be represented by polynomials of degree d, then d-POLYNOMIAL NON-ROOT CSP cannot be compressed to size $\mathcal{O}(n^{g(d)-\varepsilon})$ unless NP $\subseteq$ coNP/poly. By contraposition, a kernelization with size bound $\widetilde{\mathcal{O}}(n^{h(d)})$ implies a lower bound of $h^{-1}(d)$ on the degree of a polynomial representing an OR of arity $h(d)$, assuming NP $\not\subseteq$ coNP/poly. Kernel upper bounds where $h(d)$ is polynomially bounded in d, would therefore establish lower bounds of the form $\Omega(n^{\alpha})$ on the degree of polynomials representing an n-variable OR modulo 6, for some $\alpha > 0$. However, the current-best degree

lower bound [88] is only $\Omega(\log n)$, which has not been improved in nearly two decades (cf. [11, §1.4]).

A simple example of a CSP whose kernelization complexity is currently unclear has constraints of the form "the number of satisfied literals is one or two, modulo six". The approach of Theorem 3.1 fails, since there is no polynomial modulo six with root set $\{1,2\}$.

On the other hand, we will see in the next chapter that this framework does give a number of insights into the sparsifiability of Boolean CSPs, as it for example allows us to classify which Boolean CSPs have a non-trivial sparsification.

Fully classifying the sparsifiability of all CSPs furthermore requires understanding of CSPs over larger domains. Our upper bound techniques easily extend to CSPs over domains of size $k > 2$. Suppose we have to determine whether there is an assignment $\tau\colon V \to \{0,\ldots,k-1\}$ to a set of variables V, such that $f(\tau(x_1),\ldots,\tau(x_n)) = 0$ for all degree-d polynomial equalities f on a given list L. Using the same technique as in Theorem 3.1, one can efficiently find a subset of $\mathcal{O}(n^d)$ constraints that preserve the answer to the problem. This bound is slightly worse than over the Boolean domain, because we can no longer assume all monomials to be multilinear.

The real challenge in analyzing CSPs over any domain is to find low-degree polynomials that represent constraints of interest. We will see the usefulness of this technique in Chapter 6, in which constraints of the form "variables $x_1,\ldots,x_k$ do not all receive distinct values from $1,\ldots,k$" are represented by polynomials of degree $k-1$ to compress k-coloring problems on graphs.

Finally, we mention that all our results extend to the setting of min-ones and max-ones CSPs, in which one has to find a satisfying assignment that sets at least, or at most, a given number of variables to true. For example, our results easily imply that Exact Hitting Set parameterized by the number of variables n has a sparsification of size $\mathcal{O}(n^2)$, which cannot be improved to $\mathcal{O}(n^{2-\varepsilon})$ unless NP $\subseteq$ coNP/poly.

Chapter 4

Sparsification Bounds for CSPs

In Chapter 3, we have given a way to sparsify CSP instances for which the constraints are given as equalities of low-degree polynomials. In this chapter we will further study the sparsifiability of $\mathrm{CSP}(\Gamma)$ (Sec. 2.7) for finite Boolean constraint languages Γ, applying the main results of the previous chapter.

We start by extending the results obtained in the previous chapter in Section 4.1. We will show for a fixed constraint language Γ, that if Γ can be captured by polynomials in an appropriate sense, the results in the previous chapter allow us to obtain a kernelization whose size will depend on the degree of these polynomials. Clearly, the upper bound results carry over if $\mathrm{CSP}(\Gamma)$ is equivalent to d-Polynomial root CSP for some d, but it turns out that we can still apply our results under somewhat weaker conditions on Γ.

In Section 4.2, we will give a general lower bound technique for $\mathrm{CSP}(\Gamma)$. We give a simple condition that, if satisfied by Γ, leads to a kernelization lower bound for $\mathrm{CSP}(\Gamma)$. The lower bounds are obtained by linear-parameter transformations from Vertex Cover (for quadratic lower bounds) and d-cnf-sat (for other polynomial lower bounds).

Using the results from Sections 4.1 and 4.2, we continue in Section 4.3 with a dichotomy theorem. We fully classify for which Boolean constraint languages $\mathrm{CSP}(\Gamma)$ allows for a non-trivial sparsification, assuming $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$. It turns out that, contrary to the pessimistic picture that arose during the initial investigation of sparsifiability, the phenomenon of non-trivial sparsification is widespread and occurs for almost all Boolean CSPs! When considering constraint

languages whose largest constraint has arity d, a trivial sparsification with $\mathcal{O}(n^d)$ constraints can be obtained for CSP(Γ). Dell and Van Melkebeek [33] showed that d-CNF-SAT does not allow for a non-trivial sparsification. However, it turns out that the d-OR relation is special in this sense: if Γ is a constraint language whose largest constraint has arity-d, then the only reason that CSP(Γ) does not have a non-trivial sparsification, is that it contains a relation that is essentially a d-OR relation. In all other cases, sparsification is possible.

The polynomial-based framework also resulted in some *linear* sparsifications. As seen in the introductory example in Chapter 3, the d-EXACT SAT problem has a sparsification with $\mathcal{O}(n)$ constraints for each constant d. This prompted a detailed investigation into linear sparsifications for CSPs by Lagerkvist and Wahlström [70], who used the toolkit of universal algebra in an attempt to obtain a characterization of the Boolean CSPs with a linear sparsification. Their results give a necessary and sufficient condition on the constraint language of a CSP for having a so-called Maltsev embedding over an infinite domain. They also show that when a CSP has a Maltsev embedding over a *finite* domain, then this can be used to obtain a linear sparsification. Alas, it remains unclear whether Maltsev embeddings over infinite domains can be exploited algorithmically, and a characterization of the linearly-sparsifiable CSPs is currently not known.

In Section 4.4, we give a necessary and sufficient condition for a relation to be captured by degree-1 polynomials. We introduce the notion of balanced operations, and say that a relation is balanced if and only if it is preserved by all balanced operations. We prove that if a Boolean relation R is balanced, then it can efficiently be captured by a degree-1 polynomial and the number of constraints that are applications of this relation can be reduced to $\mathcal{O}(n)$. Hence when *all* relations in a constraint language Γ are balanced—we call such a constraint language *balanced*—then CSP (Γ) has a sparsification with $\mathcal{O}(n)$ constraints. We also show that, on the other hand, if a Boolean relation R is not balanced, then there does not exist a degree-1 polynomial over any ring that captures R in the sense required for application of the polynomial framework. The property of being balanced is (as defined) a universal-algebraic property; these results thus tightly bridge universal algebra and the polynomial framework. In Section 4.7 we compare our universal-algebraic approach to the approach proposed by Lagerkvist and Wahlström [70].

In Section 4.5, we continue our investigation of CSPs that have linear sparsification by considering Boolean symmetric constraint satisfaction problems. A constraint satisfaction problem is symmetric if the satisfiability of a constraint only depends on the number of *true* variables, and not on their positions (see Definition 4.30 for a complete definition). Using the results from Section 4.4, we obtain that CSP(Γ) has a linear sparsification when Γ is balanced. In the case of symmetric CSPs, we can furthermore show that when Γ is intractable and *not* balanced, then there must exist a relation in Γ that cone-defines 2-OR. Using the results from Section 4.2, this implies a quadratic lower bound on the

sparsification size. Consequently, we obtain a characterization of the sparsification complexity of NP-complete Boolean CSPs whose constraint language consists of symmetric relations: there is a linear sparsification if and only if the constraint language is balanced. This yields linear sparsifications in several new scenarios that were not known before.

In Section 4.6, we combine the results on non-trivial sparsification obtained in Section 4.3 with the results on linear sparsification obtained in Section 4.4. This allows us to obtain an exact characterization of the optimal sparsification size for all Boolean CSPs where each relation has arity at most three. For a Boolean constraint language Γ consisting of relations of arity at most three, we characterize the sparsification complexity of Γ as an integer $k \in \{1,2,3\}$ that represents the largest OR that Γ can cone-define. Then we prove that $\mathrm{CSP}(\Gamma)$ has a sparsification of size $\mathcal{O}(n^k)$, but no sparsification of size $\mathcal{O}(n^{k-\varepsilon})$ for any $\varepsilon > 0$, giving matching upper and lower bounds. Hence for all Boolean CSPs with constraints of arity at most three, the polynomial-based framework gives provably *optimal* sparsifications.

4.1 Sparsification for Boolean CSPs captured by polynomials

In this section, we will consider the problem $\mathrm{CSP}(\Gamma)$ for *fixed* Γ. We show that under certain conditions on Γ, the problem allows for a sparsification. We will rely heavily on the results from Section 3.1.1. We will need the following definition.

Definition 4.1 Let R be a k-ary Boolean relation (recall Definition 2.29). We say that a polynomial $p_{\mathbf{u}}$ over a ring $E_{\mathbf{u}}$ *captures* an unsatisfying assignment $\mathbf{u} \in \{0,1\}^k \setminus R$ with respect to R, if the following two conditions hold over $E_{\mathbf{u}}$.

$$p_{\mathbf{u}}(x_1,\ldots,x_k) = 0 \text{ for all } (x_1,\ldots,x_k) \in R, \text{ and} \tag{4.1}$$

$$p_{\mathbf{u}}(u_1,\ldots,u_k) \neq 0. \tag{4.2}$$

We will say a k-ary relation R is *captured by degree-d polynomials* if for all $\mathbf{u} \in \{0,1\}^k \setminus R$ there exists a ring $E_{\mathbf{u}} \in \{\mathbb{Q}\} \cup \{\mathbb{Z}/m_{\mathbf{u}}\mathbb{Z} \mid m_{\mathbf{u}} > 1 \text{ and } m_{\mathbf{u}} \in \mathbb{N}\}$ and polynomial $p_{\mathbf{u}}$ over $E_{\mathbf{u}}$ of degree at most d that captures $\mathbf{u}$ with respect to R.

In the previous chapter, we obtained a sparsification in the case that each constraint was given by a degree-d polynomial over the same field or ring. In this section we will generalize this result by allowing the use of polynomials over different rings, and by allowing multiple polynomials per constraint. We first show the result for constraint languages consisting of a single relation in the next lemma, and then generalize this to arbitrary finite constraint languages in Theorem 4.6

▶ **Lemma 4.2** *Let $R \subseteq \{0,1\}^k$ be a fixed k-ary relation that is captured by degree-d polynomials. There exists a polynomial-time algorithm that, given a set of constraints $\mathcal{C}$ over $\{R\}$ over n variables, outputs $\mathcal{C}' \subseteq \mathcal{C}$ with $|\mathcal{C}'| = \mathcal{O}(n^d)$, such that any Boolean assignment satisfies all constraints in $\mathcal{C}$ if and only if it satisfies all constraints in $\mathcal{C}'$.*

Proof. Let $\mathcal{C}$ be a set of constraints over R and let V be the set of variables used. We will create $|\{0,1\}^k \setminus R|$ instances of d-POLYNOMIAL ROOT CSP with variable set V. For each $\mathbf{u} \in R \setminus \{0,1\}^k$, we create an instance $(L_\mathbf{u}, V)$ of d-POLYNOMIAL ROOT CSP over $E_\mathbf{u}$, as follows. Choose a ring $E_\mathbf{u} \in \{\mathbb{Q}\} \cup \{\mathbb{Z}/m_\mathbf{u}\mathbb{Z} \mid m_\mathbf{u} > 1 \text{ and } m_\mathbf{u} \in \mathbb{N}\}$ and a polynomial $p_\mathbf{u}$ over $E_\mathbf{u}$ such that (4.1) and (4.2) are satisfied for $\mathbf{u}$. For each constraint $(x_1, \ldots, x_k) \in \mathcal{C}$, add the equality $p_\mathbf{u}(x_1, \ldots, x_k) = 0$ to the set $L_\mathbf{u}$; note that these are equations over the ring $E_\mathbf{u}$. Let $L := \bigcup_{\mathbf{u} \notin R} L_\mathbf{u}$ be the union of all created sets of equalities. From this construction, we obtain the following property.

▷ **Claim 4.3** *Any Boolean assignment f that satisfies all equalities in L, satisfies all constraints in $\mathcal{C}$.*

Proof. Let f be a Boolean assignment that satisfies all equalities in L. Suppose f does not satisfy all equalities in $\mathcal{C}$, thus there exists $(x_1, \ldots, x_k) \in \mathcal{C}$, such that $(f(x_1), \ldots, f(x_k)) \notin R$. Let $\mathbf{u} := (f(x_1), \ldots, f(x_k))$. Since $\mathbf{u} \notin R$, the equation $p_\mathbf{u}(x_1, \ldots, x_k) = 0$ was added to $L_\mathbf{u} \subseteq L$. However, it follows from (4.2) that $p_\mathbf{u}(f(x_1), \ldots, f(x_k)) \neq 0$, which contradicts the assumption that f satisfies all equalities in L. ◁

For each instance $(L_\mathbf{u}, V)$ of d-POLYNOMIAL ROOT CSP over $E_\mathbf{u}$ with $E_\mathbf{u} \neq \mathbb{Q}$, apply Theorem 3.6 to obtain an equivalent instance $(L'_\mathbf{u}, V)$ with $L'_\mathbf{u} \subseteq L_\mathbf{u}$ and $|L'_\mathbf{u}| = \mathcal{O}(r_\mathbf{u} n^d)$, where $r_\mathbf{u}$ is the number of distinct prime divisors of $m_\mathbf{u}$. Similarly, for each instance $(L_\mathbf{u}, V)$ of d-POLYNOMIAL ROOT CSP over $E_\mathbf{u}$ with $E_\mathbf{u} = \mathbb{Q}$, apply Theorem 3.1 and obtain an equivalent instance $(L'_\mathbf{u}, V)$ with $L'_\mathbf{u} \subseteq L_\mathbf{u}$ and $|L'_\mathbf{u}| = \mathcal{O}(n^d)$. Let $L' := \bigcup L'_\mathbf{u}$. By this definition, any Boolean assignment satisfies all equalities in L, if and only if it satisfies all equalities in L'. Construct $\mathcal{C}'$ as follows. For any $(x_1, \ldots, x_k) \in \mathcal{C}$, add $(x_1, \ldots, x_k)$ to $\mathcal{C}'$ if there exists $\mathbf{u} \in \{0,1\}^k \setminus R$ such that $p_\mathbf{u}(x_1, \ldots, x_k) = 0 \in L'$. Hereby, $\mathcal{C}' \subseteq \mathcal{C}$. The following two claims show the correctness of this sparsification procedure.

▷ **Claim 4.4** *Any Boolean assignment f satisfies all constraints in $\mathcal{C}'$, if and only if it satisfies all constraints in $\mathcal{C}$.*

Proof. Since $\mathcal{C}' \subseteq \mathcal{C}$, it follows immediately that any Boolean assignment satisfying the constraints in $\mathcal{C}$ also satisfies all constraints in $\mathcal{C}'$. It remains to prove the opposite direction.

Let f be a Boolean assignment satisfying all constraints in $\mathcal{C}'$. We show that f satisfies all equalities in L'. Let $p_\mathbf{u}(x_1, \ldots, x_k) = 0 \in L'$. Thereby,

$(x_1, \ldots, x_k) \in \mathcal{C}'$ and since f is a satisfying assignment, $(f(x_1), \ldots, f(x_k)) \in R$. It follows from property (4.1) that $p_{\mathbf{u}}(f(x_1), \ldots, f(x_k)) = 0$ as desired.

Since f satisfies all equalities in L', it satisfies all equalities in L by the choice of L'. It follows from Claim 4.3 that thereby f satisfies all constraints in $\mathcal{C}$. ◁

▷ **Claim 4.5** $|\mathcal{C}'| = \mathcal{O}(n^d)$.

Proof. By the construction of $\mathcal{C}'$, it follows that $|\mathcal{C}'| \leq |L'|$. Let r be the maximum of all $r_{\mathbf{u}}$ for $\mathbf{u} \in \{0,1\}^k \setminus R$. Since R is fixed, r is a constant. We know $|L'| = \sum_{\mathbf{u} \notin R} |L'_{\mathbf{u}}| \leq \sum_{\mathbf{u} \notin R} \mathcal{O}(r \cdot n^d) \leq 2^k \cdot \mathcal{O}(r \cdot n^d) = \mathcal{O}(n^d)$, as $|R| \leq 2^k$ and k is a constant. ◁

Claims 4.4 and 4.5 complete the proof of Lemma 4.2. ◀

By applying the above lemma repeatedly, we obtain a sparsification for finite Boolean constraint languages consisting of multiple relations.

▶ **Theorem 4.6** *Let Γ be a finite Boolean constraint language such that every $R \in \Gamma$ is captured by degree-d polynomials. Then there exists a polynomial-time algorithm that, given a set of constraints $\mathcal{C}$ over Γ over n variables, outputs $\mathcal{C}' \subseteq \mathcal{C}$ with $|\mathcal{C}'| = \mathcal{O}(n^d)$, such that any Boolean assignment satisfies all constraints in $\mathcal{C}$ if and only if it satisfies all constraints in $\mathcal{C}'$.*

Proof. Suppose we are given an instance of CSP(Γ), with set of constraints $\mathcal{C}$. We show how to define $\mathcal{C}' \subseteq \mathcal{C}$. For a k-ary relation $R \in \Gamma$, let $\mathcal{C}_R$ contain all constraints in $\mathcal{C}$ of the form $R(x_1, \ldots, x_k)$. For all $R \in \Gamma$, apply Theorem 4.2 to obtain $\mathcal{C}'_R \subseteq \mathcal{C}_R$ such that $|\mathcal{C}'_R| = \mathcal{O}(n^d)$ and any Boolean assignment satisfies all constraints in $\mathcal{C}'_R$ if and only if it satisfies the constraints in $\mathcal{C}_R$. Let $\mathcal{C}'$ be the union of all $\mathcal{C}'_R$. It is easy to verify that $|\mathcal{C}'| \leq |\Gamma| \cdot \mathcal{O}(n^d) = \mathcal{O}(n^d)$. ◀

4.2 Lower bounds using cone-definability

In this section we will show that if one of the relations in Γ can be used to define a k-OR in a specific way, then CSP(Γ) does not have a kernel of size $\mathcal{O}(n^{k-\varepsilon})$ unless NP $\subseteq$ coNP/poly. To formalize this, we need the following definition.

Definition 4.7 Let us say that a Boolean relation T of arity m is *cone-definable* from a Boolean relation U of arity n if there exists a tuple $(y_1, \ldots, y_n)$ where:

- for each $j \in [n]$, it holds that y_j is an element of $\{0,1\} \cup \{x_1, \ldots, x_m\} \cup \{\neg x_1, \ldots, \neg x_m\}$;
- for each $i \in [m]$, there exists $j \in [n]$ such that $y_j \in \{x_i, \neg x_i\}$; and,
- for each $f\colon \{x_1, \ldots, x_m\} \to \{0,1\}$, it holds that $(f(x_1), \ldots, f(x_m)) \in T$ if and only if $(\hat{f}(y_1), \ldots, \hat{f}(y_n)) \in U$. Here, $\hat{f}$ denotes the natural extension of f where $\hat{f}(0) = 0$, $\hat{f}(1) = 1$, and $\hat{f}(\neg x_i) = \neg f(x_i)$.

(The prefix *cone* indicates the allowing of **co**nstants and **ne**gation.)

Let us say that two Boolean relations T, U are *cone-interdefinable* if each is cone-definable from the other.

▶ **Example 4.8** Let $R = \{(0,0),(0,1)\}$ and let $S = \{(0,1),(1,1)\}$. We have that R is cone-definable from S via the tuple $(\neg x_2, \neg x_1)$; also, S is cone-definable from R via the same tuple. ◀

The following is a key property of cone-definability; it states that relations that are cone-definable from a constraint language Γ may be simulated by the constraint language, and thus used to prove hardness results for $\mathrm{CSP}(\Gamma)$. Recall that Γ^* is the constraint language Γ together with all constants (as in Definition 2.39).

▶ **Proposition 4.9** *Suppose that Γ is an intractable Boolean constraint language, and that Δ is a constraint language such that each relation in Δ is cone-definable from a relation in Γ. Then, there exists a linear-parameter transformation from $\mathrm{CSP}(\Gamma^* \cup \Delta)$ to $\mathrm{CSP}(\Gamma)$, parameterized by the number of variables.*

Proof. It suffices to give a linear-parameter transformation from $\mathrm{CSP}(\Gamma^* \cup \Delta)$ to $\mathrm{CSP}(\Gamma^*)$, by Theorem 2.44. Let $(\mathcal{C}, V)$ be an instance of $\mathrm{CSP}(\Gamma^* \cup \Delta)$, and let n denote $|V|$. We generate an instance $(\mathcal{C}', V')$ of $\mathrm{CSP}(\Gamma^*)$ as follows.

- For each variable $v \in V$, introduce a primed variable v'. By Proposition 2.43, the relation $\neq$ (that is, the relation $\{(0,1),(1,0)\}$) is pp-definable (Definition 2.40) from Γ^*. Fix such a pp-definition, and let d be the number of variables in the definition. For each $v \in V$, include in $\mathcal{C}'$ all constraints in the pp-definition of $\neq$, but where the variables are renamed so that v and v' are the distinguished variables, and the other variables are fresh.

  The number of variables used so far in $\mathcal{C}'$ is nd.
- For each $b \in \{0,1\}$, introduce a variable z_b and add the constraint $\{(b)\}(z_b)$ to $\mathcal{C}'$. This is equivalent to requiring that $z_0 = 0$ and $z_1 = 1$ in any satisfying assignment. Observe that for this step, it is required that we can use relations from Γ^*, instead of just Γ.
- For each constraint $T(v_1, \ldots, v_k)$ in $\mathcal{C}$ such that $T \in \Gamma^*$, add the constraint to $\mathcal{C}'$.
- For each constraint $T(v_1, \ldots, v_k)$ in $\mathcal{C}$ such that $T \in \Delta \setminus \Gamma^*$, we use the assumption that T is cone-definable from a relation in Γ to add a constraint to $\mathcal{C}'$ that has the same effect as $T(v_1, \ldots, v_k)$. In particular, assume that T is cone-definable from $U \in \Gamma$ via the tuple $(y_1, \ldots, y_\ell)$, and that U has arity ℓ. Add to $\mathcal{C}'$ the constraint $U(w_1, \ldots, w_\ell)$, where, for each $i \in [\ell]$, the entry w_i is defined as follows:

$$w_i = \begin{cases} v_j \text{ if } y_i = x_j, \\ v'_j \text{ if } y_i = \neg x_j, \\ z_0 \text{ if } y_i = 0, \text{ and} \\ z_1 \text{ if } y_i = 1. \end{cases}$$

See Example 4.10 for an example of this whole construction. The set V' of variables used in $\mathcal{C}'$ is the union of $V \cup \{v' \mid v \in V\} \cup \{z_0, z_1\}$ with the other variables used in the copies of the pp-definition of $\neq$. We have $|V'| = nd + 2$. It is straightforward to verify that an assignment $f\colon V \to \{0,1\}$ satisfies $\mathcal{C}$ if and only if there exists an assignment $f'\colon V' \to \{0,1\}$ of f that satisfies $\mathcal{C}'$. ◀

► **Example 4.10** We give an example of the linear-parameter transformation used in the proof of Proposition 4.9. Let us consider $R = \{(0,0,1),(0,1,0),(1,0,0),(0,1,1),(1,0,1),(1,1,0)\}$ and let $\Gamma = \{R\}$. Thus, a three-tuple is in R unless all three variables are equal.

Define $R' = \{(0,0),(1,0),(1,1)\}$ and $\Delta := \{R'\}$. Observe that Γ is an intractable constraint language, as can be shown using Theorem 2.37. We can cone-define R' from R by the following tuple:

$$(x_1, \neg x_2, 0).$$

Verify that indeed $(x_1, x_2) \in R'$ if and only if $(x_1, \neg x_2, 0) \in R$. As such, the preconditions of the proposition are satisfied, we now show the linear-parameter transformation from $\mathrm{CSP}(\Gamma^* \cup \Delta)$ to $\mathrm{CSP}(\Gamma^*)$, based on an example. Consider the input

$$V = \{v,w,x,y,z\},\ \mathcal{C} = \{R'(x,y), R'(v,x), R(w,v,z), \{(0)\}(w)\}$$

for $\mathrm{CSP}(\Gamma^* \cup \Delta)$. We create an instance $(V', \mathcal{C}')$ of $\mathrm{CSP}(\Gamma^*)$. Initialize

$$V' := \{v, v', w, w', x, x', y, y', z, z', z_0, z_1\}.$$

In the first step, we want to add the inequality requirements on the variables and their primed counterparts. For this, we need a pp-definition of $\{(0,1),(1,0)\}$ from Γ^*. For example, we may use the following definition.

$$\exists b, c : R(x_1, x_2, b) \wedge \{(0)\}(b) \wedge R(x_1, x_2, c) \wedge \{(1)\}(c).$$

As such, we add variables $\{b_v, b_w, b_x, b_y, b_z, c_v, c_w, c_x, c_y, c_z\}$ to V' and initialize $\mathcal{C}'$ as

$$\begin{aligned}
&\{R(v,v',b_v), \{(0)\}(b_v), R(w,w',b_w), \{(0)\}(b_w),\\
&\ R(x,x',b_x), \{(0)\}(b_x), R(y,y',b_y), \{(0)\}(b_y),\\
&\ R(z,z',b_z), \{(0)\}(b_z)\} \cup\\
&\{R(v,v',c_v), \{(1)\}(c_v), R(w,w',c_w), \{(1)\}(c_w),\\
&\ R(x,x',c_x), \{(1)\}(c_x), R(y,y',c_y), \{(1)\}(c_y),\\
&\ R(z,z',c_z), \{(1)\}(c_z)\}.
\end{aligned}$$

In this particular case, we could have re-used the variables for b and c, as they are constant, but this is not generally true. In the next step, we add the constraints in

$$\{\{(0)\}(z_0), \{(1)\}(z_1)\}.$$

Then, we add all constraints that are using relations from Γ^*, thus adding the constraints in

$$\{R(w,v,z),\{(0)\}(w)\}$$

to $\mathcal{C}'$. Finally, we use the cone-definability result for Δ, and add the constraints in

$$\{R(x,y',z_0),R(v,x',z_0)\},$$

concluding the construction of $\mathcal{C}'$. ◀

The following theorem combines existing ideas for kernelization lower bounds due to several authors [33,35,70] with the formalism of cone-definition.

▶ **Theorem 4.11** *Let Γ be an intractable Boolean constraint language, and let $k \geq 1$. If there exists $R \in \Gamma$ such that R cone-defines k-OR, then $\mathsf{CSP}(\Gamma)$ does not have a generalized kernel of size $\mathcal{O}(n^{k-\varepsilon})$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.*

Proof. We do a case distinction on k.

($k = 1$) Suppose that there exists $\varepsilon > 0$ such that $\mathsf{CSP}(\Gamma)$ has a (generalized) kernel of size $\mathcal{O}(n^{1-\varepsilon})$ into a parameterized decision problem L. Let $\widetilde{L} := \{x\#1^\ell \mid (x,\ell) \in L\}$ denote the classical (non-parameterized) problem corresponding to L, in which the parameter is written in unary and appended to the end of each input string. Here 1 is an arbitrary character in the encoding alphabet, and # is a separator character that is freshly added to the alphabet.

Using this hypothetical generalized kernel, one could obtain a polynomial-time algorithm that takes as input a series of instances $(\mathcal{C}_1,V_1),\ldots,(\mathcal{C}_t,V_t)$ of $\mathsf{CSP}(\Gamma)$, and outputs in polynomial time an instance $\widetilde{x}$ of $\widetilde{L}$ such that:

- $\widetilde{x} \in \widetilde{L}$ if and only if $(\mathcal{C}_i,V_i)$ is a *yes*-instance of $\mathsf{CSP}(\Gamma)$ for *all* $i \in [t]$, and
- $\widetilde{x}$ has bitsize $\mathcal{O}(N^{1-\varepsilon})$, where $N := \sum_{i=1}^{t} |V_i|$.

To obtain such an AND-compression algorithm from a hypothetical generalized kernel of $\mathsf{CSP}(\Gamma)$ into L, it suffices to do the following:

1. On input a series of instances $(\mathcal{C}_1,V_1),\ldots,(\mathcal{C}_t,V_t)$ of $\mathsf{CSP}(\Gamma)$, form a new instance $(\mathcal{C}^* := \bigcup_{i=1}^{t} \mathcal{C}_i, V^* := \bigcup_{i=1}^{t} V_i)$ of $\mathsf{CSP}(\Gamma)$. Hence we take the disjoint union of the sets of variables and the sets of constraints, and it follows that the new instance has answer *yes* if and only if all the inputs $(\mathcal{C}_i,V_i)$ have answer *yes*.
2. Run the hypothetical generalized kernel on $(\mathcal{C}^*,V^*)$, which has $|V^*| = N$ variables and is therefore reduced to an equivalent instance (x^*,k^*) of L with size and parameter bounded by $\mathcal{O}(N^{1-\varepsilon})$. Let $\widetilde{x}$ be the classical instance corresponding to (x^*,k^*).

If we apply this AND-compression scheme to a sequence of $t_1(m) := m^\alpha$ instances of m bits each (which therefore have at most m variables each), the resulting output has $\mathcal{O}(|V^*|^{1-\varepsilon}) = \mathcal{O}((m \cdot m^\alpha)^{1-\varepsilon}) = \mathcal{O}(m^{(1+\alpha)(1-\varepsilon)})$ bits. By picking α large enough that it satisfies $(1+\alpha)(1-\varepsilon) \leq \alpha$, we therefore compress a sequence of $t_1(m)$ instances of bitsize m into one instance expressing the logical AND, of size at most $t_2(m) \leq \mathcal{O}(m^{(1+\alpha)(1-\varepsilon)}) \leq C \cdot t_1(m)$ for some suitable constant C. Drucker [35, Theorem 5.4] has shown that an error-free deterministic AND-compression algorithm with these parameters for an NP-complete problem into a fixed decision problem L, implies NP $\subseteq$ coNP/poly. Hence the lower bound for $k = 1$ follows since CSP(Γ) is NP-complete.

$(k \geq 2)$ For $k \geq 2$, we prove the lower bound using a linear-parameter transformation (recall Definition 2.7). Let Δ be the set of k-ary relations given by $\Delta := \{\{0,1\}^k \setminus \{u\} \mid u \in \{0,1\}^k\}$. In particular, note that Δ contains the k-OR relation. Since R cone-defines k-OR, it is easy to see that by variable negations, R cone-defines all relations in Δ. Thereby, it follows from Proposition 4.9 that there is a linear-parameter transformation from CSP$(\Gamma^* \cup \Delta)$ to CSP(Γ). Thus, to prove the lower bound for CSP(Γ), it suffices to prove the desired lower bound for CSP$(\Gamma^* \cup \Delta)$. We do a further case distinction on k.

***Case* $(k = 2)$** If $k = 2$, we do a linear-parameter transformation from VERTEX COVER to CSP$(\Gamma^* \cup \Delta)$. Since it is known that VERTEX COVER parameterized by the number of vertices n has no generalized kernel of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless NP $\subseteq$ coNP/poly (Theorem 2.10), the result will follow.

Suppose we are given a graph $G = (V, E)$ on n vertices and integer $k \leq n$, forming an instance of the VERTEX COVER problem. The question is whether there is a set S of k vertices, such that each edge has at least one endpoint in S. We create an equivalent instance $(\mathcal{C}, V')$ of CSP$(\Gamma^* \cup \Delta)$ as follows. We introduce a new variable x_v for each $v \in V$. For each edge $\{u, v\} \in E$, we add the constraint 2-OR(x_u, x_v) to $\mathcal{C}$.

At this point, any vertex cover in G corresponds to a satisfying assignment, and vice versa. It remains to ensure that the size of the vertex cover is bounded by k. Let $H_{n,k}$ be the n-ary relation given by

$$H_{n,k} = \{(x_1, \ldots, x_n) \mid x_i \in \{0,1\} \text{ for all } i \in [n] \text{ and } \sum_{i \in [n]} x_i = k\}.$$

By Proposition 2.43, we obtain that Γ^* pp-defines all Boolean relations. It follows from [70, Lemma 17] that Γ^* pp-defines $H_{n,k}$ using $\mathcal{O}(n+k)$ constraints and $\mathcal{O}(n+k)$ existentially quantified variables. We add the constraints from this pp-definition to $\mathcal{C}$, and add the existentially quantified variables to V'. This concludes the construction of $\mathcal{C}$. It is easy to see that $\mathcal{C}$ has a satisfying assignment if and only if G has a vertex cover of size k. Furthermore, we used $\mathcal{O}(n+k) \in \mathcal{O}(n)$ variables and thereby this is a linear-parameter transformation from VERTEX COVER to CSP$(\Gamma^* \cup \Delta)$.

Case $(k \geq 3)$ In this case there is a trivial linear-parameter transformation from $\mathrm{CSP}(\Delta)$ to $\mathrm{CSP}(\Gamma^* \cup \Delta)$. It is easy to verify that $\mathrm{CSP}(\Delta)$ is equivalent to k-CNF-SAT. The result now follows from the fact that for $k \geq 3$, k-CNF-SAT has no kernel of size $\mathcal{O}(n^{k-\varepsilon})$ for any $\varepsilon > 0$, unless NP $\subseteq$ coNP/poly (Theorem 2.9). ◀

4.3 Classification of CSPs with worst-case sparsification

It is well known that k-CNF-SAT allows no non-trivial sparsification, for each $k \geq 3$ (Theorem 2.9). This means that we cannot efficiently reduce the number of clauses in such a formula to $\mathcal{O}(n^{k-\varepsilon})$. The k-OR relation is special, in the sense that there is exactly one k-tuple that is not contained in the relation. We show in this section that when considering k-ary relations for which there is more than one k-tuple not contained in the relation, a non-trivial sparsification is always possible. In particular, the number of constraints of any input can efficiently be reduced to $\mathcal{O}(n^{k-1})$.

We start by showing that if R is a relation for which at least two k-tuples are not contained in R, then R is captured by degree-$(k-1)$ polynomials. Recall from Section 4.1, that d-NAE-SAT was captured by polynomials of degree $d-1$. In the next lemma, we will reuse some of the ideas that construction. The reader may verify that d-NAE-SAT can indeed be represented by a constraint language for which every relation contains $2^d - 2$ tuples.

Since relations with $|R| = 2^k$ have a sparsification of size $\mathcal{O}(1)$, as constraints over such relations are satisfied by any assignment, this will allow us to obtain the desired sparsification for k-ary relations with $|\{0,1\}^k \setminus R| \neq 1$ using Theorem 4.6.

▶ **Lemma 4.12** *Let R be a k-ary Boolean relation with $|R| < 2^k - 1$. The relation R is captured by degree-$(k-1)$ polynomials.*

Proof. We will prove this by showing that for every $\mathbf{u} \in \{0,1\}^k \setminus R$, there exists a k-ary polynomial $p_{\mathbf{u}}$ over $\mathbb{Q}$ of degree at most $k-1$ satisfying requirements (4.1) and (4.2) from Definition 4.1, such that the result follows.

We will prove the existence of such a polynomial by induction on k. For $k = 1$, the lemma statement implies that $R = \emptyset$. Thereby, for any $\mathbf{u} \notin R$, we simply choose $p_{\mathbf{u}}(x_1) := 1$. This polynomial satisfies the requirements, and has degree 0.

Let $k > 1$ and let $\mathbf{u} = (u_1, \ldots, u_k) \in \{0,1\}^k \setminus R$. Since $|R| < 2^k - 1$, we can choose $\mathbf{w} = (w_1, \ldots, w_k)$ such that $\mathbf{w} \in \{0,1\}^k \setminus R$ and $\mathbf{w} \neq \mathbf{u}$. Choose such $\mathbf{w}$ arbitrarily, we now do a case distinction.

(There exists no $i \in [k]$ for which $u_i = w_i$) This implies $u_i = \neg w_i$ for all i. One may note that for $\mathbf{u} = (0, \ldots, 0)$ and $\mathbf{w} = (1, \ldots, 1)$ this situation corre-

sponds to MONOTONE k-NAE-SAT. We show that there exists a polynomial $p_{\mathbf{u}}$ such that $p_{\mathbf{u}}(u_1,\ldots,u_k) \neq 0$, and $p_{\mathbf{u}}(x_1,\ldots,x_k) = 0$ for all $(x_1,\ldots,x_k) \in R$. Hereby $p_{\mathbf{u}}$ satisfies conditions (4.1) and (4.2) for $\mathbf{u}$. For $i \in [k]$, define $r_i(x) := (1-x)$ if $u_i = 1$ and $r_i(x) := x$ if $u_i = 0$. It follows immediately from this definition that $r_i(u_i) = 0$ and $r_i(w_i) = 1$ for all $i \in [k]$. Define

$$p_{\mathbf{u}}(x_1,\ldots,x_k) := \prod_{i=1}^{k-1}\left(i - \sum_{j=1}^{k} r_j(x_j)\right).$$

By this definition, $p_{\mathbf{u}}$ has degree $k-1$. It remains to verify that $p_{\mathbf{u}}$ has the desired properties. First of all, since $\sum_{j=1}^{k} r_j(u_j) = 0$ by definition, it follows that

$$p_{\mathbf{u}}(u_1,\ldots,u_k) = \prod_{i=1}^{k-1} i \neq 0,$$

as desired. Since $r_i(w_i) = 1$ for all i, we obtain $p_{\mathbf{u}}(w_1,\ldots,w_k) = \prod_{i=1}^{k-1}(i-k) \neq 0$, which is allowed since $\mathbf{w} \notin R$. It is easy to verify that in all other cases, $\sum_{j=1}^{k} r_j(x_j) \in \{1,2,\ldots,k-1\}$ and thereby one of the terms of the product is zero, implying that $p_{\mathbf{u}}(x_1,\ldots,x_k) = 0$.

(There exists $i \in [k]$, such that $u_i = w_i$) Let $\mathbf{u}'$ and $\mathbf{w}'$ be defined as the results of removing coordinate i from $\mathbf{u}$ and $\mathbf{w}$ respectively. Note that $\mathbf{u}' \neq \mathbf{w}'$. Define

$$R' := \{(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_k) \mid (x_1,\ldots,x_{i-1},u_i,x_{i+1},\ldots,x_k) \in R\}.$$

By this definition, $\mathbf{u}', \mathbf{w}' \notin R'$ and thereby R' is a $(k-1)$-ary relation with $|R'| < 2^{k-1} - 1$. By the induction hypothesis, there exists a polynomial $p_{\mathbf{u}'}$ of degree at most $k-2$, such that $p_{\mathbf{u}'}(u'_1,\ldots,u'_{k-1}) \neq 0$ and $p_{\mathbf{u}'}(x'_1,\ldots,x'_{k-1}) = 0$ for all $\mathbf{x}' \in R'$. Now define

$$p_{\mathbf{u}}(x_1,\ldots,x_k) := (1-x_i-u_i)\cdot p_{\mathbf{u}'}(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_k).$$

We show that $p_{\mathbf{u}}$ has the desired properties. By definition, $p_{\mathbf{u}}$ has the degree of $p_{\mathbf{u}'}$ plus one. Since $p_{\mathbf{u}'}$ has degree $k-2$ by the induction hypothesis, it follows that $p_{\mathbf{u}}$ has degree $k-1$. Let $(x_1,\ldots,x_k) \in R$. We do a case distinction on the value taken by x_i.

- $x_i \neq u_i$. In this case, $(1-x_i-u_i) = 0$, and thereby $p_{\mathbf{u}}(x_1,\ldots,x_k) = 0$, thus satisfying condition (4.1).
- $x_i = u_i$. Since $x = (x_1,\ldots,x_{i-1},u_i,x_{i+1},\ldots,x_k) \in R$, it follows that $(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_k) \in R'$. By definition of $p_{\mathbf{u}'}$, it follows that

  $$p_{\mathbf{u}'}(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_k) = 0$$

  and thus $p_{\mathbf{u}}(x_1,\ldots,x_k) = 0$, showing (4.1).

It remains to show that $p_{\mathbf{u}}(u_1,\ldots,u_k) \neq 0$. This follows from $(1-u_i-u_i) \in \{-1,1\}$, and $p_{\mathbf{u}'}(u_1,\ldots,u_{i-1},u_{i+1},\ldots,u_k) \neq 0$, showing that (4.2) holds.

Since we have shown for all $\mathbf{u} \in \{0,1\}^k \setminus R$ that there exists a polynomial $p_{\mathbf{u}}$ over $\mathbb{Q}$ of degree at most $k-1$ satisfying (4.1) and (4.2), R is captured by degree-$(k-1)$ polynomials. ◀

The next lemma formalizes the idea that any k-ary relation with $|\{0,1\}^k \setminus R| = 1$ is equivalent to k-OR, up to negation of variables. The dichotomy result will follow from Lemma 4.12, together with the next lemma and Theorem 4.11.

▶ **Lemma 4.13** *If R is a Boolean k-ary relation with $|R| = 2^k - 1$, then R cone-defines k-OR.*

Proof. Let $\mathbf{u} = (u_1,\ldots,u_k)$ be the unique k-tuple not contained in R. Define the tuple $(y_1,\ldots,y_k)$ as follows. Let $y_i := x_i$ if $u_i = 0$, and let $y_i := \neg x_i$ otherwise. Clearly, this satisfies the first two conditions of cone-definability. It remains to prove the last condition. Let $f\colon \{x_1,\ldots,x_k\} \to \{0,1\}$. Suppose $(f(x_1),\ldots,f(x_k)) \in k$-OR. We show $(\hat{f}(y_1),\ldots,\hat{f}(y_k)) \in R$. It follows from $(f(x_1),\ldots,f(x_k)) \in k$-OR, that there exists at least one $i \in [k]$ such that $f(x_i) \neq 0$. Thereby, $\hat{f}(y_i) \neq u_i$ and thus $(\hat{f}(y_1),\ldots,\hat{f}(y_k)) \neq \mathbf{u}$, implying $(\hat{f}(y_1),\ldots,\hat{f}(y_k)) \in R$.

Suppose $(f(x_1),\ldots,f(x_k)) \notin k$-OR, implying $f(x_i) = 0$ for all $i \in [k]$. But this implies $\hat{f}(y_i) = u_i$ for all $i \in [k]$ and thus $(\hat{f}(y_1),\ldots,\hat{f}(y_k)) = \mathbf{u} \notin R$. ◀

Using Lemmas 4.12 and 4.13, the next theorem will give a complete classification of the constraint languages that admit a non-trivial sparsification.

▶ **Theorem 4.14** *Let Γ be a finite intractable Boolean constraint language. Let k be the maximum arity of any relation $R \in \Gamma$. The following dichotomy holds.*

- *If for all $R \in \Gamma$ it holds that $|R| \neq 2^k - 1$, then $\mathrm{CSP}(\Gamma)$ has a kernel with $\mathcal{O}(n^{k-1})$ constraints that can be stored in $\mathcal{O}(n^{k-1}\log n)$ bits.*
- *If there exists $R \in \Gamma$ with $|R| = 2^k - 1$, then $\mathrm{CSP}(\Gamma)$ has no generalized kernel of bitsize $\mathcal{O}(n^{k-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.*

Proof. Suppose that for all $R \in \Gamma$, it holds that $|R| \neq 2^k - 1$. We give the following kernelization procedure. Suppose we are given an instance of $\mathrm{CSP}(\Gamma)$, with set of constraints $\mathcal{C}$. We show how to define $\mathcal{C}' \subseteq \mathcal{C}$. For each constraint $R(x_1,\ldots,x_\ell) \in \mathcal{C}$ where R is a relation of arity $\ell < k$, add one such constraint to $\mathcal{C}'$ (thus removing duplicate constraints). Note that this adds at most $\mathcal{O}(n^\ell)$ constraints for each ℓ-ary relation $R \in \Gamma$.

Let $\Gamma_k \subseteq \Gamma$ contain all relations of arity k for which $|R| < 2^k - 1$. It follows from Lemma 4.12 that every relation in Γ_k is captured by degree-$(k-1)$ polynomials. Let $\mathcal{C}_k \subseteq \mathcal{C}$ contain all constraints using a relation from Γ_k. Use Theorem 4.6 to obtain $\mathcal{C}'_k \subseteq \mathcal{C}_k$ such that any assignment satisfying all

constraints in $\mathcal{C}'_k$ satisfies all constraints in $\mathcal{C}_k$, and $|\mathcal{C}_k| = \mathcal{O}(n^{k-1})$. Add the constraints in $\mathcal{C}'_k$ to $\mathcal{C}'$. This concludes the construction of $\mathcal{C}'$. Note that the procedure removes constraints of the form $R(x_1, \ldots, x_k)$ with $|R| = 2^k$, as these are always satisfied. It is easy to verify that $|\mathcal{C}'| \leq |\Gamma| \cdot \mathcal{O}(n^{k-1}) = \mathcal{O}(n^{k-1})$. Since each constraint can be stored in $\mathcal{O}(\log n)$ bits, this gives a kernel of bitsize $\mathcal{O}(n^{k-1} \log n)$.

Suppose that there exists $R \in \Gamma$ with $|R| = 2^k - 1$. It follows from Lemma 4.13 that R cone-defines k-OR. Since Γ is intractable, it now follows from Theorem 4.11 that CSP(Γ) has no generalized kernel of size $\mathcal{O}(n^{k-\varepsilon})$, unless NP $\subseteq$ coNP/poly. ◀

4.4 Towards a classification of CSPs with linear sparsification

In this section we will introduce an algebraic property of constraint languages, namely whether a constraint language is *balanced* (introduced below). We will show that for Γ satisfying the property, CSP(Γ) has a kernel with $\mathcal{O}(n)$ constraints. This sparsification will be obtained using the method of representing constraints by low-degree polynomials. The following definitions capture the key notions for this universal-algebraic approach to sparsification via low-degree polynomials. Relevant definitions relating to (partial) Boolean operations and CSPs can be found in Section 2.7.

Definition 4.15 A partial Boolean operation $f\colon \{0,1\}^k \to \{0,1\}$ is *balanced* if there exist integer values $\alpha_1, \ldots, \alpha_k$, called the *coefficients* of f, such that

- $\sum_{i\in[k]} \alpha_i = 1$,
- $(x_1, \ldots, x_k)$ is in the domain of f if and only if $\sum_{i\in[k]} \alpha_i x_i \in \{0,1\}$, and
- $f(x_1, \ldots, x_k) = \sum_{i\in[k]} \alpha_i x_i$ for all tuples in its domain.

Definition 4.16 We say that a Boolean relation is *balanced* if it is preserved by all balanced operations, and that a Boolean constraint language is *balanced* if each relation therein is balanced.

Definition 4.17 Define an *alternating operation* to be a balanced operation $a_k\colon \{0,1\}^k \to \{0,1\}$ such that k is odd and the coefficients alternate between $+1$ and -1, so that $\alpha_1 = +1$, $\alpha_2 = -1$, $\alpha_3 = +1$, …, $\alpha_k = +1$. For example, by this definition we have

$$a_3(x_1, x_2, x_3) = x_1 - x_2 + x_3$$

for all (x_1, x_2, x_3) in the domain of a_3, and the domain of a_3 is $\{(0,0,0), (0,0,1), (0,1,1), (1,0,0), (1,1,0), (1,1,1)\}$.

While alternating operations form a restricted type of balanced operations, the following proposition shows that being preserved by all balanced operations is equivalent to being preserved by all alternating operations.

▶ **Proposition 4.18** *A Boolean relation R is balanced if and only if for all odd $k \geq 1$, the relation R is preserved by the alternating operation of arity k.*

Proof. It suffices to show that if a relation T is not balanced, then there exists an alternating operation that does not preserve T. Let f be a k-ary balanced operation that does not preserve T. Then there exist tuples $\mathbf{t_1}, \ldots, \mathbf{t_k}$ in T such that $\alpha_1\mathbf{t_1} + \cdots + \alpha_k\mathbf{t_k}$ is not in T, where the sum of the α_i is equal to 1 (and where we may assume that no α_i is equal to 0). For each positive α_i, replace $\alpha_i\mathbf{t_i}$ in the sum with $\mathbf{t_i} + \cdots + \mathbf{t_i}$ (α_i times); likewise, for each negative α_i, replace $\alpha_i\mathbf{t_i}$ in the sum with $-\mathbf{t_i} - \cdots - \mathbf{t_i}$ ($-\alpha_i$ times). Each tuple then has coefficient $+1$ or -1 in the sum; since the sum of coefficients is $+1$, by permuting the sum's terms, the coefficients can be made to alternate between $+1$ and -1. ◀

As an example, consider the 2-OR relation given by $\{(1,1),(1,0),(0,1)\}$ and observe that it is *not* balanced, since $(1,0) - (1,1) + (0,1) = (0,0)$ and $(0,0) \notin$ 2-OR. The relation $R := \{(0,0,1),(0,1,0),(1,0,0)\}$ (corresponding to MONOTONE EXACT 3-SAT), is an example of a relation that is balanced.

We will use the following straightforwardly verified fact tacitly, throughout (recall Definition 2.34).

Observation 4.19 *Each balanced operation is idempotent and self-dual.*

The main result of this section is that when Γ is balanced, then CSP(Γ) has a kernel with linearly many constraints. To prove this result, we will use two additional technical lemmas. When S is a matrix, we use $\mathbf{s_i}$ to refer to the i'th row of S. The proof of the following lemma was contributed by Emil Jeřábek.

▶ **Lemma 4.20** *Let S be an $m \times n$ integer matrix. Let $\mathbf{u} \in \mathbb{Z}^n$ be a row vector. If $\mathbf{u} \notin span_{\mathbb{Z}}(\{\mathbf{s_1}, \ldots, \mathbf{s_m}\})$, then there exists a prime power q such that $\mathbf{u} \notin span_q(\{\mathbf{s_1}, \ldots, \mathbf{s_m}\})$. Furthermore, there is a polynomial-time algorithm that computes a (possibly composite) integer q' for which $\mathbf{u} \notin span_{q'}(\{\mathbf{s_1}, \ldots, \mathbf{s_m}\})$.*

Proof. Suppose $\mathbf{u} \notin \text{span}_{\mathbb{Z}}(\{\mathbf{s_1}, \ldots, \mathbf{s_m}\})$, thus $\mathbf{u}$ cannot be written as a linear combination of the rows of S over $\mathbb{Z}$; equivalently, the system $\mathbf{y}S = \mathbf{u}$ has no solutions for $\mathbf{y}$ over $\mathbb{Z}$. We will show that there exists a prime power q, such that $\mathbf{y}S \equiv_q \mathbf{u}$ has no solutions over $\mathbb{Z}/q\mathbb{Z}$ and thus $\mathbf{u} \notin \text{span}_q(\{\mathbf{s_1}, \ldots, \mathbf{s_m}\})$.

There exist an $m \times m$ matrix M and an $n \times n$ matrix N over $\mathbb{Z}$, such that M and N are invertible over $\mathbb{Z}$ and furthermore $S' := MSN$ is in Smith Normal Form (recall Definition 2.21). In particular, this implies that S' is a diagonal matrix. Furthermore, M, S, and N can be computed in polynomial time by Theorem 2.23. Define $\mathbf{u}' := \mathbf{u}N$.

▷ **Claim 4.21** *If $\mathbf{y}'S' = \mathbf{u}'$ is solvable for $\mathbf{y}'$ over $\mathbb{Z}$, then $\mathbf{y}S = \mathbf{u}$ is solvable for $\mathbf{y}$ over $\mathbb{Z}$.*

Proof. Consider $\mathbf{y}'$ such that $\mathbf{y}'S' = \mathbf{u}'$. One can verify that $\mathbf{y} := \mathbf{y}'M$ solves $\mathbf{y}S = \mathbf{u}$, as

$$\mathbf{y}S = \mathbf{y}'MS = \mathbf{y}'MSNN^{-1} = \mathbf{y}'S'N^{-1} = \mathbf{u}'N^{-1} = \mathbf{u}NN^{-1} = \mathbf{u}. \qquad \triangleleft$$

▷ **Claim 4.22** *Let $q \in \mathbb{N}$. If $\mathbf{y}S \equiv_q \mathbf{u}$ is solvable for $\mathbf{y}$, then $\mathbf{y}'S' \equiv_q \mathbf{u}'$ is solvable for $\mathbf{y}'$.*

Proof. Let $\mathbf{y}$ be such that $\mathbf{y}S \equiv_q u$. Define $\mathbf{y}' := \mathbf{y}M^{-1}$. We verify that $\mathbf{y}'S' \equiv_q \mathbf{u}'$ as follows.

$$\mathbf{y}'S' = \mathbf{y}'MSN = \mathbf{y}M^{-1}MSN = \mathbf{y}SN \equiv_q \mathbf{u}N = \mathbf{u}'. \qquad \triangleleft$$

Using these two claims, our proof proceeds as follows. From our starting assumption $\mathbf{u} \notin \mathrm{span}_{\mathbb{Z}}(\{\mathbf{s_1}, \ldots, \mathbf{s_m}\})$, it follows by Claim 4.21 that $\mathbf{y}'S' = \mathbf{u}'$ has no solution $\mathbf{y}'$ over $\mathbb{Z}$. Below, we prove that this implies there exists a prime power q such that $\mathbf{y}'S' \equiv_q \mathbf{u}'$ is unsolvable. Furthermore we show how to find a (possibly composite) integer q in polynomial time such that $\mathbf{y}'S' \equiv_q \mathbf{u}'$ is unsolvable. By Claim 4.22 this will imply that $\mathbf{y}S \equiv_q \mathbf{u}$ is unsolvable and complete the proof.

We inferred that $\mathbf{y}'S' = \mathbf{u}'$ has no solutions over $\mathbb{Z}$. Since all non-zero elements of S' are on the diagonal, this implies that either there exists $i \in [n]$, such that u'_i is not divisible by $s'_{i,i}$, or $s'_{i,i}$ is zero while $u'_i \neq 0$. We finish the proof by a case distinction.

- Suppose there exists $i \in [n]$ such that $s'_{i,i} = 0$, while $u'_i \neq 0$. Choose a prime power q such that $q \nmid u'_i$. Observe that such q can be obtained in polynomial-time, for example by taking the smallest power of two that is larger than u'_i. It is easy to see that thereby, $u'_i \not\equiv_q 0$. Since $s'_{i,i} \equiv_q 0$ holds trivially in this case, the system $\mathbf{y}'S' \equiv_q \mathbf{u}'$ has no solution.
- Otherwise, there exists $i \in [n]$ such that $s'_{i,i} \nmid u'_i$. Choose a prime power q such that $q \nmid u'_i$ and $q \mid s'_{i,i}$. Such a prime power can be chosen by letting $q := p^\ell$ for a prime p that occurs $\ell \geq 1$ times in the prime factorization of $s'_{i,i}$, but less often in the prime factorization of u'_i. Thereby, $u'_i \not\equiv_q 0$, while $s'_{i,i} \equiv_q 0$. It again follows that the system $\mathbf{y}'S' \equiv_q \mathbf{u}'$ has no solutions.

  It is not clear how to obtain q as described above in polynomial time, as it requires computing the prime decompositions of possibly large integers. However, observe that letting $q := s'_{i,i}$ means $u'_i \not\equiv_q 0$ while $s'_{i,i} \equiv_q 0$, implying that for this choice of q the system $\mathbf{y}'S' \equiv_q \mathbf{u}'$ again has no solutions. Since this q is trivial to obtain in polynomial time, that concludes the proof. ◀

Given that $\mathbf{u} \notin \mathrm{span}_{\mathbb{Z}}(\{\mathbf{s_1}, \ldots, \mathbf{s_m}\})$, we can thus use Lemma 4.20 to obtain an integer q such that $\mathbf{u} \notin \mathrm{span}_q(\{\mathbf{s_1}, \ldots, \mathbf{s_m}\})$. The next lemma shows that in this case the system $S'\mathbf{x} \equiv_q \mathbf{b}$, where $\mathbf{b} = (0, \ldots, 0, c)$ and S' is the matrix consisting of rows $\{\mathbf{s_1}, \ldots, \mathbf{s_m}, \mathbf{u}\}$, has a solution $\mathbf{x}$ for some $c \not\equiv_q 0$.

▶ **Lemma 4.23** *Let $q > 1$ be an integer. Let A be an $m \times n$ matrix over $\mathbb{Z}/q\mathbb{Z}$. Suppose $\mathbf{a_m} \notin span_q(\{\mathbf{a_1}, \ldots, \mathbf{a_{m-1}}\})$. Then there exists a constant $c \not\equiv_q 0$ for which the system $A\mathbf{x} \equiv_q \mathbf{b}$ has a solution, where $\mathbf{b} := (0, \ldots, 0, c)^T$ is the vector with c on the last position and zeros in all other positions. Furthermore, $\mathbf{x}$ and c can be computed in polynomial time.*

Proof. Let A' be the $(m-1) \times n$ matrix consisting of the first $m-1$ rows of A. Find the Smith decomposition (Definition 2.21) of A' over $\mathbb{Z}$, thus there exist an $(m-1) \times (m-1)$ matrix M' and an $n \times n$ matrix N, such that $S' := M'A'N$ is in Smith Normal Form and M' and N are invertible over $\mathbb{Z}$. The only property of the Smith Normal Form we will rely on is that S' is a diagonal matrix.

We show that similar properties hold over $\mathbb{Z}/q\mathbb{Z}$. Let $(M')^{-1}$, N^{-1} be the inverses of M' and N over $\mathbb{Z}$. It is easy to verify that $NN^{-1} = I \equiv_q I$ and $M'(M')^{-1} = I \equiv_q I$, such that M' and N are still invertible over $\mathbb{Z}/q\mathbb{Z}$. Furthermore, $S' \pmod q$ remains a diagonal matrix.

Define M to be the following $m \times m$ matrix

$$M := \left(\begin{array}{c|c} M' & \mathbf{0} \\ \hline \mathbf{0} & 1 \end{array}\right),$$

then M has an inverse over $\mathbb{Z}/q\mathbb{Z}$ that is given by the following matrix

$$M^{-1} \equiv_q \left(\begin{array}{c|c} (M')^{-1} & \mathbf{0} \\ \hline \mathbf{0} & 1 \end{array}\right).$$

Define $S := MAN$ and verify that

$$S := MAN \equiv_q \left(\begin{array}{c} S' \\ \hline \mathbf{a_m}N \end{array}\right), \tag{4.3}$$

meaning that the first $m-1$ rows of S are equal to the first $m-1$ rows of S', and the last row of S is given by the row vector $\mathbf{a_m}N$.

The following two claims will be used to show that proving the lemma statement for matrix S, will give the desired result for A.

▷ **Claim 4.24** *Let $\mathbf{b} := (0, \ldots, 0, c)$ for some constant c. The system $S\mathbf{x}' \equiv_q \mathbf{b}$ has a solution, if and only if the system $A\mathbf{x} \equiv_q \mathbf{b}$ has a solution.*

Proof. Let $\mathbf{x}$ be a solution for $A\mathbf{x} \equiv_q \mathbf{b}$. Define $\mathbf{x}' := N^{-1}\mathbf{x}$. Then $MAN\mathbf{x}' \equiv_q MA\mathbf{x} \equiv_q M\mathbf{b}$. Observe that by the definitions of M and $\mathbf{b}$, $M\mathbf{b} \equiv_q \mathbf{b}$, which concludes this direction of the proof.

For the other direction, let $\mathbf{x}'$ be a solution for $MAN\mathbf{x}' \equiv_q \mathbf{b}$. Define $\mathbf{x} := N\mathbf{x}'$. Then $M^{-1}MAN\mathbf{x}' \equiv_q M^{-1}\mathbf{b}$ and thus $AN\mathbf{x}' \equiv_q M^{-1}\mathbf{b}$ and thereby $A\mathbf{x} \equiv_q M^{-1}\mathbf{b}$. By the definition of M^{-1} and $\mathbf{b}$, we again have $M^{-1}\mathbf{b} \equiv_q \mathbf{b}$. ◁

▷ **Claim 4.25** *Let S be defined as above. Then $\mathbf{s_m} \in span_q(\{\mathbf{s_1}, \ldots, \mathbf{s_{m-1}}\})$ if and only if $\mathbf{a_m} \in span_q(\{\mathbf{a_1}, \ldots, \mathbf{a_{m-1}}\})$.*

Proof. Suppose $\mathbf{s_m} \in \text{span}_q(\{\mathbf{s_1}, \ldots, \mathbf{s_{m-1}}\})$. This implies that there exist $\alpha_1, \ldots, \alpha_{m-1}$ such that $\sum_{i\in[m-1]} \alpha_i \mathbf{s_i} \equiv_q \mathbf{s_m} \equiv_q \mathbf{a_m} N$. Thus, $\sum_{i\in[m-1]} \alpha_i \mathbf{s'_i} \equiv_q \mathbf{a_m} N$, and for $\alpha = (\alpha_1, \ldots, \alpha_{m-1})$ we therefore have $\alpha S' \equiv_q \mathbf{a_m} N$, implying $(\alpha M')A'N \equiv_q \mathbf{a_m} N$. Since N is invertible, it follows that

$$(\alpha M')A' \equiv_q \mathbf{a_m}$$

and thus $\mathbf{a_m} \in \text{span}_q(\{\mathbf{a_1}, \ldots, \mathbf{a_{m-1}}\})$.

For the other direction, suppose $\mathbf{a_m} \in \text{span}_q(\{\mathbf{a_1}, \ldots, \mathbf{a_{m-1}}\})$. Thus, there exists $\alpha \equiv_q (\alpha_1, \ldots, \alpha_{m-1})$ such that $\alpha A' \equiv_q \mathbf{a_m}$. Let $\alpha' := \alpha(M')^{-1}$. Then

$$\alpha' S' \equiv_q \alpha' M' A' N \equiv_q \alpha A' N \equiv_q \mathbf{a_m} N \equiv_q \mathbf{s_m},$$

and it follows from the definition of S in Equation (4.3) that thereby $\mathbf{s_m} \in \text{span}_q(\{\mathbf{s_1}, \ldots, \mathbf{s_{m-1}}\})$. ◁

It follows from Claims 4.24 and 4.25, that it suffices to show that $S\mathbf{x} = (0, \ldots, 0, c)^{\mathrm{T}}$ has a solution for some $c \not\equiv_q 0$ if $\mathbf{s_m} \notin \text{span}_q(\{\mathbf{s_1}, \ldots, \mathbf{s_{m-1}}\})$. Observe that since S' (the first $m-1$ rows of S) is a diagonal matrix, there must exist $i \in [m-1]$ for which there is no α_i satisfying $s_{i,i} \cdot \alpha_i \equiv_q s_{m,i}$. Otherwise, it is easy to see that $\sum_{i\in[m-1]} \alpha_i \mathbf{s_i} \equiv_q \mathbf{s_m}$, contradicting that $\mathbf{s_m} \notin \text{span}_q(\{\mathbf{s_1}, \ldots, \mathbf{s_{m-1}}\})$. We now do a case distinction.

Suppose there exists $i \in [m-1]$ such that $s_{i,i} \equiv_q 0$, while $s_{m,i} \not\equiv_q 0$. Let $\mathbf{x} = (0, \ldots, 0, 1, 0, \ldots, 0)$ be the vector with 1 in the i'th position and zeros in all other positions. It is easy to verify that $S\mathbf{x} \equiv_q (0, \ldots, 0, s_{m,i})^{\mathrm{T}}$ and thereby the system $S\mathbf{x} \equiv_q \mathbf{b}$ has a solution for $c = s_{m,i}$.

Otherwise, choose i such that $s_{i,i} \not\equiv_q 0$ and there exists no integer α_i satisfying $s_{i,i} \cdot \alpha_i \equiv_q s_{m,i}$. We consider the following two cases.

- Suppose $\gcd(s_{i,i}, q) \mid s_{m,i}$ over the integers. Let $d \in \mathbb{Z}$ such that $s_{m,i} = d \cdot \gcd(s_{i,i}, q)$ over the integers. It follows from Bézout's identity (Theorem 2.15) that there exist integers a and b such that $\gcd(s_{i,i}, q) = a \cdot s_{i,i} + b \cdot q$. Thereby,

  $$s_{m,i} \equiv_q d \cdot (a \cdot s_{i,i} + b \cdot q) \equiv_q d \cdot a \cdot s_{i,i}$$

  which is a contradiction with the assumption that no integer α_i exists such that $s_{i,i} \cdot \alpha_i \equiv_q s_{m,i}$.

- Suppose $\gcd(s_{i,i}, q) \nmid s_{m,i}$ over the integers, let $y := q/\gcd(s_{i,i}, q)$. Define $x := (0, \ldots, 0, y, 0, \ldots, 0)^{\mathrm{T}}$ as the vector with y in position i. Then

  $$Sx \equiv_q (0, \ldots, 0, y \cdot s_{i,i}, 0, \ldots, 0, y \cdot s_{m,i})^{\mathrm{T}}.$$

4

Then $y \cdot s_{i,i} = s_{i,i} \cdot q / \gcd(s_{i,i}, q) = \pm\mathrm{lcm}(q, s_{i,i}) \equiv_q 0$, recall that lcm stands for least common multiple. It remains to show that $y \cdot s_{m,i} \not\equiv_q 0$. Suppose for contradiction that $y \cdot s_{m,i} \equiv_q 0$, such that over the integers $q \cdot s_{m,i} / \gcd(s_{i,i}, q) = d \cdot q$ for some $d \in \mathbb{Z}$. But then $s_{m,i} / \gcd(s_{i,i}, q) = d \in \mathbb{Z}$ contradicting that $\gcd(s_{i,i}, q) \nmid s_{m,i}$. It follows that $y \cdot s_{m,i} \not\equiv_q 0$. Thus, for $\mathbf{x}$ defined as above and $c := y \cdot s_{m,i} \not\equiv_q 0$ we obtain that $S\mathbf{x} = (0, \ldots, 0, c)^{\mathrm{T}}$, concluding the proof.

Computing the Smith decomposition of a matrix can be done in polynomial time (Theorem 2.23) and computing the greatest common divisor of two integers can be done in time polynomial in their binary encoding. Hereby, it is straightforward to turn the above proof in a polynomial-time algorithm that computes both $\mathbf{x}$ and c. ◀

In the context of capturing a Boolean relation R by degree-1 polynomials, the constructive proof of Lemma 4.23 effectively shows the following: given a ring $\mathbb{Z}/q\mathbb{Z}$ over which a degree-1 polynomial exists that captures a certain tuple $\mathbf{u} \notin R$, one can constructively find the coefficients $\mathbf{x}$ of a polynomial that captures $\mathbf{u}$ by following the steps in the proof. We will formalize this idea in Theorem 4.28, but first we prove the main sparsification result.

▶ **Theorem 4.26** *Let Γ be a balanced Boolean constraint language. Then $\mathrm{CSP}(\Gamma)$ has a kernel with $\mathcal{O}(n)$ constraints that are a subset of the original constraints. The kernel can be stored using $\mathcal{O}(n \log n)$ bits.*

Proof. We start by showing in the following claim that for all relations $R \in \Gamma$ we can find linear polynomials over an appropriate ring that capture the tuples that are not in R. We will then use this representation to obtain our kernel.

▷ **Claim 4.27** *For all relations R in a balanced Boolean constraint language Γ, for all $\mathbf{u} \notin R$, there exists a linear polynomial $p_{\mathbf{u}}$ over a ring $E_{\mathbf{u}} \in \{\mathbb{Z}/q_{\mathbf{u}}\mathbb{Z} \mid q_{\mathbf{u}}$ is a prime power$\}$ that captures $\mathbf{u}$ with respect to R.*

Proof. Suppose for a contradiction that there exists $R \in \Gamma$ and $\mathbf{u} \notin R$, such that no prime power q and polynomial p over $\mathbb{Z}/q\mathbb{Z}$ exist that capture $\mathbf{u}$ with respect to R. Let $R = \{\mathbf{r_1}, \ldots, \mathbf{r_\ell}\}$. By the non-existence of p and q, the system

$$\begin{pmatrix} 1 & r_{1,1} & r_{1,2} & \ldots & r_{1,k} \\ 1 & r_{2,1} & r_{2,2} & \ldots & r_{2,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & r_{\ell,1} & r_{\ell,2} & \ldots & r_{\ell,k} \\ 1 & u_1 & u_2 & \ldots & u_k \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \end{pmatrix} \equiv_q \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ c \end{pmatrix}$$

has no solution for any prime power q and $c \not\equiv_q 0$. Otherwise, it is easy to verify that q is the desired prime power and $p(x_1, \ldots, x_k) := \alpha_0 + \sum_{i=1}^{k} \alpha_i x_i$ is the desired polynomial.

The fact that no solution exists, implies that $(1, u_1, \ldots, u_k)$ is in the span of the remaining rows of the matrix, by Lemma 4.23. But this implies that for any prime power q, there exist coefficients $\beta_1, \ldots, \beta_\ell$ over $\mathbb{Z}/q\mathbb{Z}$ such that $\mathbf{u} \equiv_q \sum \beta_i \mathbf{r_i}$. Furthermore, since the first column of the matrix is the all-ones column, we obtain that $\sum \beta_i \equiv_q 1$. By Lemma 4.20, it follows that there exist integer coefficients $\gamma_1, \ldots, \gamma_\ell$ such that $\sum \gamma_i = 1$ and furthermore $\mathbf{u} = \sum \gamma_i \mathbf{r_i}$. But it immediately follows that $R \in \Gamma$ is not preserved by the balanced operation given by $f(x_1, \ldots, x_\ell) := \sum \gamma_i x_i$, which contradicts the assumption that Γ is balanced. ◁

The theorem statement now follows directly from Theorem 4.6. ◀

The next theorem shows that given an arbitrary constraint language, it is possible to decide in polynomial time whether it is balanced. Furthermore, for balanced constraint languages there is a polynomial-time algorithm to obtain the capturing polynomials. Here we assume that the relation R is encoded explicitly as a list of tuples contained in the relation, together with a list of tuples not contained in the relation, making the total encoding size of a Boolean k-ary relation at least 2^k.

▶ **Theorem 4.28** *There is a polynomial-time algorithm that, given a k-ary Boolean relation R, decides whether R is balanced and for balanced R outputs a polynomial $p_{\mathbf{u}}$ and integer $q_{\mathbf{u}}$ for all $\mathbf{u} \in \{0,1\}^k \setminus R$ such that $p_{\mathbf{u}}$ captures $\mathbf{u}$ over $\mathbb{Z}/q_{\mathbf{u}}\mathbb{Z}$.*

Proof. Let $R = \{\mathbf{r_1}, \ldots, \mathbf{r_m}\}$. Define $\mathbf{s_i}$ as $(1, r_{i,1}, \ldots, r_{i,k})$ for $i \in [m]$. Relation R is balanced if and only if there exists no $\mathbf{u} \in \{0,1\}^k \setminus R$ such that $\mathrm{ext}(\mathbf{u}) \in \mathrm{span}_{\mathbb{Z}}(\{\mathbf{s_1}, \ldots, \mathbf{s_m}\})$, where $\mathrm{ext}(\mathbf{u}) = (1, u_1, \ldots, u_k)$. Thereby, R is balanced if and only if for all $\mathbf{u} \in \{0,1\}^k \setminus R$, the following system has no solutions over $\mathbb{Z}$.

$$(\alpha_1, \ldots, \alpha_m) \begin{pmatrix} 1 & r_{1,1} & r_{1,2} & \ldots & r_{1,k} \\ 1 & r_{2,1} & r_{2,2} & \ldots & r_{2,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & r_{m,1} & r_{m,2} & \ldots & r_{m,k} \end{pmatrix} = (1, u_1, \ldots, u_k)$$

The solvability of these systems over the integers can be tested in time that is polynomial in the size of an explicit encoding of R that indicates for each tuple whether or not it is contained in R. This computation can be done, for example, by first computing the Smith Normal Form of the matrix.

Suppose R is balanced. Let $\mathbf{u} \in \{0,1\}^k \setminus R$, we show how to find $p_{\mathbf{u}}$ and $q_{\mathbf{u}}$. Let $\mathbf{s_i}$ be defined as above. Then $\mathrm{ext}(\mathbf{u}) \notin \mathrm{span}_{\mathbb{Z}}(\{\mathbf{s_1}, \ldots, \mathbf{s_m}\})$ since R is balanced. Apply Lemma 4.20 to obtain $q_{\mathbf{u}} \in \mathbb{N}$ in polynomial time such that $\mathrm{ext}(\mathbf{u}) \notin \mathrm{span}_{q_{\mathbf{u}}}(\{\mathbf{s_1}, \ldots, \mathbf{s_m}\})$. It follows from Lemma 4.23 that the following

system has a solution

$$\begin{pmatrix} 1 & r_{1,1} & r_{1,2} & \dots & r_{1,k} \\ 1 & r_{2,1} & r_{2,2} & \dots & r_{2,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & r_{m,1} & r_{m,2} & \dots & r_{m,k} \\ 1 & u_1 & u_2 & \dots & u_k \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \end{pmatrix} \equiv_{q_u} \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ c \end{pmatrix}$$

for $(\alpha_0, \alpha_1, \dots, \alpha_k)^{\mathrm{T}}$ and some $c \not\equiv_{q_u} 0$ and that we can find it in polynomial time. Let $p_{\mathbf{u}}(x_1, \dots, x_k) := \alpha_0 + \sum_{i=1}^{k} \alpha_i x_i$. It is straightforward to verify that $p_{\mathbf{u}}$ captures $\mathbf{u}$ with respect to R over $\mathbb{Z}/q_{\mathbf{u}}\mathbb{Z}$. ◀

The kernelization result of Theorem 4.26 above is obtained by using the fact that when Γ is balanced, the constraints in $\mathrm{CSP}(\Gamma)$ can be replaced by linear polynomials. We show in the next theorem that this approach fails when Γ is not balanced.

▶ **Theorem 4.29** *Let R be a k-ary Boolean relation that is not balanced. Then there exists $\mathbf{u} \in \{0,1\}^k \setminus R$ for which there exists no linear polynomial $p_{\mathbf{u}}$ over any ring E that captures $\mathbf{u}$ with respect to R.*

Proof. Suppose R is not balanced. By Proposition 4.18, this implies R is violated by an alternating operation. Let f be an alternating operation that does not preserve R, such that $f(y_1, \dots, y_m) := \sum_{i=1}^{m} (-1)^{i+1} y_i$ for some odd m, and for some (not necessarily distinct) $\mathbf{r_1}, \dots, \mathbf{r_m} \in R$ we have $f(\mathbf{r_1}, \dots, \mathbf{r_m}) = \mathbf{u}$ with $\mathbf{u} \notin R$ (recall the definition of $f(\mathbf{r_1}, \dots, \mathbf{r_m})$ from Definition 2.35).

Suppose for a contradiction that there exists a linear polynomial $p_{\mathbf{u}}$ that captures $\mathbf{u}$ over a ring $E_{\mathbf{u}}$. Let $\mathbf{r_i} = (r_{i,1}, \dots, r_{i,k})$ for $i \in [m]$. Since $f(\mathbf{r_1}, \dots, \mathbf{r_m}) = \mathbf{u}$, we have the following equality over $\mathbb{Z}$:

$$u_i = r_{1,i} - r_{2,i} \dots + r_{m,i}. \tag{4.4}$$

Since $r_{j,i} \in \{0,1\}$ for all $i \in [k]$ and $j \in [m]$, Equation (4.4) holds over any ring, so in particular over $E_{\mathbf{u}}$.

Let $p_{\mathbf{u}}(x_1, \dots, x_k)$ be given by $p_{\mathbf{u}}(x_1, \dots, x_k) := \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \dots + \beta_k \cdot x_k$ for ring elements $\beta_0, \dots, \beta_k$ from $E_{\mathbf{u}}$. By Definition 4.1, $p_{\mathbf{u}}(r_{i,1}, \dots, r_{i,k}) = 0$ for all $i \in [m]$. Thereby the following equalities hold over $E_{\mathbf{u}}$:

$$\begin{aligned} p_{\mathbf{u}}(u_1, \dots, u_k) &= \beta_0 + \sum_{i=1}^{k} \beta_i \cdot u_i \\ &= \beta_0 + \sum_{i=1}^{k} \beta_i \cdot (r_{1,i} - r_{2,i} \dots + r_{m,i}) \\ &= \beta_0 + \sum_{i=1}^{k} \beta_i \cdot r_{1,i} - \beta_i \cdot r_{2,i} \dots + \beta_i \cdot r_{m,i} \end{aligned}$$

$$= (\beta_0 + \sum_{i=1}^{k} \beta_i \cdot r_{1,i}) - (\beta_0 + \sum_{i=1}^{k} \beta_i \cdot r_{2,i}) \ldots + (\beta_0 + \sum_{i=1}^{k} \beta_i \cdot r_{m,i}) \quad (4.5)$$
$$= p_{\mathbf{u}}(r_{1,1}, \ldots, r_{1,k}) - p_{\mathbf{u}}(r_{2,1}, \ldots, r_{2,k}) \ldots + p_{\mathbf{u}}(r_{m,1}, \ldots, r_{m,k})$$
$$= 0,$$

where the fourth equality follows from the fact that in line (4.5) all but one of the terms β_0 cancel, since the summation alternates between addition and subtraction. This contradicts the fact that $p_{\mathbf{u}}(u_1, \ldots, u_k) \neq 0$. Thereby, there exists no linear polynomial that captures $\mathbf{u}$ with respect to R. ◀

4.5 Symmetric CSPs

In this section, we characterize the symmetric Boolean constraint languages Γ for which CSP(Γ) has a linear sparsification. For a Boolean tuple $\mathbf{x} = (x_1, \ldots, x_k)$, let weight($\mathbf{x}$) denote the number of ones in the tuple, such that weight($\mathbf{x}$) := $\sum_i x_i$. We can now define symmetric relations.

Definition 4.30 We say a k-ary Boolean relation R is *symmetric*, if there exists $S \subseteq \{0, 1, \ldots, k\}$ such that a tuple $\mathbf{x} = (x_1, \ldots, x_k)$ is in R if and only if weight($\mathbf{x}$) $\in S$. We call S the set of *satisfying weights* for R. We will say that a constraint language Γ is *symmetric*, if it only contains symmetric relations.

At the end of this section we will prove that for symmetric Boolean constraint languages, CSP(Γ) has no sparsification with $\mathcal{O}(n^{2-\varepsilon})$ constraints if Γ is not balanced, assuming NP $\not\subseteq$ coNP/poly. Together with the results from the previous section, this gives a dichotomy result for the sparsification of CSP(Γ) for symmetric constraint languages.

The following lemma shows that under certain conditions on the satisfying weights, it follows that the relation cone-defines 2-OR, which we will use to obtain the kernelization lower bound.

▶ **Lemma 4.31** *Let R be a k-ary symmetric Boolean relation with satisfying weights $S \subseteq \{0, 1, \ldots, k\}$. Let $U := \{0, 1, \ldots, k\} \setminus S$. If there exist $a, b, c \in S$ and $d \in U$ such that $a - b + c = d$, then R cone-defines* 2-OR.

Proof. We first show the result when $b \leq a$, $b \leq c$, and $b \leq d$. In this case, we use the following tuple to express $x_1 \vee x_2$.

$$(\underbrace{\neg x_1, \ldots, \neg x_1}_{(a-b) \text{ copies}}, \underbrace{\neg x_2, \ldots, \neg x_2}_{(c-b) \text{ copies}}, \underbrace{1, \ldots, 1}_{b \text{ copies}}, \underbrace{0, \ldots, 0}_{(k-d) \text{ copies}}).$$

For $a = b$ or $c = b$, the tuple above does not give a valid cone definition. Observe however that in this case, $c = d$ or $a = d$ respectively, which cannot happen as $a, c \in S$ while $d \notin S$.

Let $f\colon \{x_1, x_2\} \to \{0,1\}$, then

$$(\neg f(x_1), \ldots, \neg f(x_1), \neg f(x_2), \ldots, \neg f(x_2), 1, \ldots, 1, 0, \ldots, 0)$$

has weight $(a-b)(1-f(x_1)) + (c-b)(1-f(x_2)) + b$. It is easy to verify that for $f(x_1) = f(x_2) = 0$, this implies the tuple has weight $a + c - b = d \notin S$ and thus the tuple is not in R. Otherwise, the weight is either a, b, or c. In these cases the tuple is contained in R, as the weight is contained in S.

Note that the above case applies when b is the smallest of all four integers. We now consider the remaining cases. Suppose $a \leq b$, $a \leq c$, and $a \leq d$ (the case where c is smallest is symmetric by swapping a and c). In this case, use the tuple

$$(\underbrace{\neg x_1, \ldots, \neg x_1}_{(d-a) \text{ copies}}, \underbrace{x_2, \ldots, x_2}_{(b-a) \text{ copies}}, \underbrace{1, \ldots, 1}_{a \text{ copies}}, \underbrace{0, \ldots, 0}_{(k-c) \text{ copies}}).$$

Consider an assignment f satisfying $x_1 \vee x_2$, verify that the weight of the above tuple under this assignment lies in $\{a, b, c\}$, and thus the tuple is contained in R. Assigning 0 to both x_1 and x_2 gives weight d, such that the tuple is not in R.

Otherwise, we have $d \leq a$, $d \leq b$, and $d \leq c$ and use the tuple

$$(\underbrace{x_1, \ldots, x_1}_{(a-d) \text{ copies}}, \underbrace{x_2, \ldots, x_2}_{(c-d) \text{ copies}}, \underbrace{1, \ldots, 1}_{d \text{ copies}}, \underbrace{0, \ldots, 0}_{(k-b) \text{ copies}}).$$

It is again easy to verify that any assignment to x_1 and x_2 satisfies this tuple if and only if it satisfies $(x_1 \vee x_2)$, using the fact that $a - d + c = b \in S$. ◀

We now give the main lemma that is needed to prove Theorem 4.34. It shows that if a relation is symmetric and not balanced, it must cone-define 2-OR, by using the result from the lemma above.

▶ **Lemma 4.32** *Let R be a symmetric Boolean relation of arity k. If R is not balanced, then R cone-defines* 2-OR.

Proof. Let f be a balanced operation that does not preserve R. Since f has integer coefficients, it follows that there exist (not necessarily distinct) $\mathbf{r_1}, \ldots, \mathbf{r_m} \in R$, such that $\mathbf{r_1} - \mathbf{r_2} + \mathbf{r_3} - \mathbf{r_4} \cdots + \mathbf{r_m} = \mathbf{u}$ for some $\mathbf{u} \in \{0,1\}^k \setminus R$ and odd $m \geq 3$. Thereby, $\text{weight}(\mathbf{r_1}) - \text{weight}(\mathbf{r_2}) + \text{weight}(\mathbf{r_3}) - \text{weight}(\mathbf{r_4}) \cdots + \text{weight}(\mathbf{r_m}) = \text{weight}(\mathbf{u})$. Let S be the set of satisfying weights for R and let $U := \{0, \ldots, k\} \setminus S$. Define $s_i := \text{weight}(\mathbf{r_i})$ for $i \in [m]$, and $t = \text{weight}(\mathbf{u})$, such that $s_1 - s_2 + s_3 - s_4 \ldots + s_m = t$, and furthermore $s_i \in S$ for all i, and $t \in U$. We show that there exist $a, b, c \in S$ and $d \in U$ such that $a - b + c = d$, such that the result follows from Lemma 4.31. We do this by induction on the length of the alternating sum.

If $m = 3$, we have that $s_1 - s_2 + s_3 = t$ and define $a := s_1$, $b := s_2$, $c := s_3$, and $d := t$.

If $m > 3$, we will use the following claim.

▷ **Claim 4.33** *Let $s_1, \ldots, s_m \in S$ and $t \in U$ such that $s_1 - s_2 + s_3 - s_4 \cdots + s_m = t$. There exist distinct $i, j, \ell \in [m]$ with i, j odd and ℓ even, such that $s_i - s_\ell + s_j \in \{0, \ldots, k\}$.*

Proof. If there exist distinct $i, j, \ell \in [m]$ with i, j odd and ℓ even, such that $s_i \geq s_\ell \geq s_j$, then these i, j, ℓ satisfy the claim statement. Suppose these do not exist, we consider two options.

- Suppose $s_i \geq s_\ell$ for all $i, \ell \in [m]$ with i odd and ℓ even. It is easy to see that thereby, for any i, j, ℓ with i, j odd and ℓ even it holds that $s_i - s_\ell + s_j \geq 0$. Furthermore, $s_i - s_\ell + s_j \leq s_1 - s_2 + s_3 - s_4 \cdots + s_m = t$ and $t \leq k$ since $t \in U$. Thus, any distinct $i, j, \ell \in [m]$ with i, j odd and ℓ even satisfy the statement.
- Otherwise, $s_i \leq s_\ell$ for all $i, \ell \in [m]$ with i odd and ℓ even. It follows that for any i, j, ℓ with i, j odd and ℓ even $s_i - s_\ell + s_j \leq k$, as $s_i - s_\ell \leq 0$ and $s_j \leq k$. Furthermore, $s_i - s_\ell + s_j \geq s_1 - s_2 + s_3 - s_4 \cdots + s_m = t$ and $t \geq 0$ by definition. Thus, any distinct $i, j, \ell \in [m]$ with i, j odd and ℓ even satisfy the statement. ◁

Use Claim 4.33 to find i, j, ℓ such that $s_i - s_\ell + s_j \in \{0, \ldots, k\}$. We consider two options. If $s_i - s_\ell + s_j \in U$, then define $d := s_i - s_\ell + s_j$, $a := s_i$, $b := s_\ell$, and $c := s_j$ and we are done. The other option is that $s_i - s_\ell + s_j = s \in S$. Replacing $s_i - s_\ell + s_j$ by s in $s_1 - s_2 + s_3 - s_4 \cdots + s_m$ gives a shorter alternating sum with result t. We obtain a, b, c, and d by the induction hypothesis.

Thereby, we have obtained $a, b, c \in S$, $d \in U$ such that $a - b + c = d$. It now follows from Lemma 4.31 that R cone-defines 2-OR. ◀

Using the lemma above, we can now prove the main result of this section.

▶ **Theorem 4.34** *Let Γ be a finite Boolean symmetric intractable constraint language.*

- *If Γ is balanced, then $\mathrm{CSP}(\Gamma)$ parameterized by the number of variables n has a kernel with $\mathcal{O}(n)$ constraints that can be stored in $\mathcal{O}(n \log n)$ bits.*
- *If Γ is not balanced, then $\mathrm{CSP}(\Gamma)$ parameterized by the number of variables n does not have a generalized kernel of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.*

Proof. If Γ is balanced, it follows from Theorem 4.26 that $\mathrm{CSP}(\Gamma)$ has a kernel with $\mathcal{O}(n)$ constraints that can be stored in $\mathcal{O}(n \log n)$ bits. Note that the assumption that Γ is symmetric is not needed in this case.

If the symmetric constraint language Γ is not balanced, then Γ contains a symmetric relation R that is not balanced. It follows from Lemma 4.32 that R cone-defines the 2-OR relation. Thereby, we obtain from Theorem 4.11 that $\mathrm{CSP}(\Gamma)$ has no generalized kernel of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. ◀

4.6 Full classification of arity-3 CSPs

In this section, we will give a full classification of the sparsifiability for constraint languages that consist only of low-arity relations. We assume all considered constraint languages to be Boolean. We start by providing some additional properties of cone-definability.

Observation 4.35 *If a Boolean relation T of arity m is cone-definable from a Boolean relation U of arity n, then $m \leq n$.*

Observation 4.36 *Suppose a Boolean relation T is cone-definable from a Boolean relation U, and that g is a partial operation that is idempotent and self-dual. If g preserves U, then g preserves T.*

The above observation follows quite straightforwardly from the definitions. For contradiction, suppose g does not preserve T. The proof strategy is to find a witness to this, and using the cone-definition of T from U to find a witness that g also does not preserve U.

The next observation is that cone-definability is transitive.

Observation 4.37 *Suppose that T_1, T_2, T_3 are Boolean relations such that T_2 is cone-definable from T_1, and T_3 is cone-definable from T_2. Then T_3 is cone-definable from T_1.*

The following two propositions are consequences of the above observations. We will tacitly use them in the sequel. Morally, they show that the properties of relations that we are interested in are invariant under cone-interdefinability. The first proposition is immediate from Observation 4.35.

▶ **Proposition 4.38** *If Boolean relations T, U are cone-interdefinable, then they have the same arity.* ◀

The next proposition is follows directly from Observation 4.36.

▶ **Proposition 4.39** *Suppose that T and U are Boolean relations that are cone-interdefinable, and that g is a partial operation that is idempotent and self-dual. Then, g preserves T if and only if g preserves U.* ◀

The next results will show that when the constraint language consists of only low-arity relations, if the constraint language is not balanced, it can cone-define the 2-OR relation.

Observation 4.40 *Each Boolean relation of arity 1 is balanced.*

▶ **Theorem 4.41** *A Boolean relation of arity 2 is balanced if and only if it is not cone-interdefinable with the 2-OR relation.*

Proof. Let $R \subseteq \{0,1\}^2$ be a relation. We prove the two directions separately.

($\Rightarrow$) Proof by contraposition. Suppose that R is cone-interdefinable with 2-OR. Then in particular, R cone-defines the 2-OR relation. Let (y_1, y_2) be a tuple witnessing cone-definability as in Definition 4.7. Since 2-OR is symmetric in its two arguments, we may assume without loss of generality that y_i is either x_i

or $\neg x_i$ for $i \in [2]$. Define $g\colon \{0,1\}^2 \to \{0,1\}^2$ by letting $g(b_1,b_2) := (\hat{b}_1,\hat{b}_2)$ where $\hat{b}_i = b_i$ if $y_i = x_i$ and $\hat{b}_i = 1 - b_i$ if $y_i = \neg x_i$ for $i \in [2]$. By definition of cone-definability we then have $g(1,0), g(0,1), g(1,1) \in R$ while $g(0,0) \notin R$. But $g(1,0) - g(1,1) + g(0,1) = g(0,0)$, showing that R is not preserved by all alternating operations and therefore is not balanced.

($\Leftarrow$) We again use contraposition. Suppose R is not balanced; we will prove R is cone-interdefinable with 2-OR. Let $f\colon \{0,1\}^k \to \{0,1\}$ be a balanced partial Boolean operation *of minimum arity* that does not preserve R. Let $\alpha_1,\ldots,\alpha_k \in \mathbb{Z}$ be the coefficients of f, as in Definition 4.15. Let $\mathbf{s_1},\ldots,\mathbf{s_k} \in R$ such that $f(\mathbf{s_1},\ldots,\mathbf{s_k}) = \mathbf{u} \in \{0,1\}^2 \setminus R$ witnesses that f does not preserve R. By Definition 4.15 we have $\mathbf{u} = \sum_{i=1}^{k} \alpha_i \mathbf{s_i}$ and $\sum_{i=1}^{k} \alpha_i = 1$. This shows that if $\alpha_i = 0$ for some coordinate i, then that position does not influence the value of f, implying the existence of a smaller-arity balanced relation that does not preserve T. Hence our choice of f as a minimum-arity operation ensures that α_i is nonzero for all $i \in [k]$.

▷ **Claim 4.42** *The tuples $\mathbf{s_1},\ldots,\mathbf{s_k}$ are all distinct.*

Proof. Suppose that $\mathbf{s_i} = \mathbf{s_j}$ for some distinct $i,j \in [k]$, and assume without loss of generality that $i = k-1$ and $j = k$. But then the balanced operation f' of arity $k-1$ defined by the coefficients $(\alpha_1,\ldots,\alpha_{k-2},\alpha_{k-1}+\alpha_k)$ does not preserve T since $f'(\mathbf{s_1},\ldots,\mathbf{s_{k-1}}) = f(\mathbf{s_1},\ldots,\mathbf{s_k}) = \mathbf{u} \notin R$, contradicting that f is a minimum-arity balanced operation that does not preserve R. ◁

▷ **Claim 4.43** *The arity k of operation f is 3.*

Proof. Since $\mathbf{s_1},\ldots,\mathbf{s_k} \in R \subseteq \{0,1\}^2$ are all distinct, while $\mathbf{u} \in \{0,1\}^2 \setminus R$, we have $k \leq 3$. We cannot have $k = 1$ since that would imply $f(\mathbf{s_1}) = \mathbf{s_1} \in R$, contradicting that $f(\mathbf{s_1},\ldots,\mathbf{s_k}) = f(\mathbf{s_1}) = \mathbf{u} \notin R$. It remains to show that $k \neq 2$. So assume for a contradiction that $k = 2$. Since $\mathbf{s_1}$ and $\mathbf{s_2}$ are distinct, there is a position $\ell \in [2]$ such that $s_{1,\ell} \neq s_{2,\ell}$. Assume without loss of generality that $s_{1,\ell} = 1$ while $s_{2,\ell} = 0$. Since $f(\mathbf{s_1},\ldots,\mathbf{s_k}) = f(\mathbf{s_1},\mathbf{s_2}) = \alpha_1\mathbf{s_1} + \alpha_2\mathbf{s_2} = \mathbf{u} \in \{0,1\}^2 \setminus R$, we find $u_\ell = \alpha_1 s_{1,\ell} + \alpha_2 s_{2,\ell} = \alpha_1 \cdot 1 + \alpha_2 \cdot 0 \in \{0,1\}$. Since α_1 is a nonzero integer, we must have $\alpha_1 = 1$. But since $\alpha_1 + \alpha_2 = 1$ by definition of a balanced operation, this implies $\alpha_2 = 0$, contradicting that f is a minimum-arity balanced operation that does not preserve R. ◁

The previous two claims show that there are at least three distinct tuples in $R \subseteq \{0,1\}^2$. Since $\mathbf{u} \in \{0,1\}^2 \setminus R$ it follows that $|R| = 3$. Hence R and 2-OR are both Boolean relations of arity two that each have three tuples. To cone-define one from the other, one may easily verify that it suffices to use the tuple (y_1,y_2), where $y_i = x_i$ if $u_i = 0$ and $y_i = \neg x_i$ otherwise. ◀

In order to show the classification of the arity-3 relations with a linear sparsification in Theorem 4.46, we first present some additional lemmas and

definitions. Let $U \subseteq \{0,1\}^n$ be a relation. We say that $\mathbf{w} \in \{0,1\}^n$ is a *witness* for U if $\mathbf{w} \notin U$, and there exists a balanced operation $f\colon \{0,1\}^k \to \{0,1\}$ and tuples $\mathbf{t_1}, \ldots, \mathbf{t_k} \in U$ such that $\mathbf{w} = f(\mathbf{t_1}, \ldots, \mathbf{t_k})$. Observe that U is not balanced if and only if there exists a witness for U.

▶ **Lemma 4.44** *Suppose that $U \subseteq \{0,1\}^n$ is a Boolean relation, and that there exist an integer c and a natural number $m > 1$ such that, for each $\mathbf{u} \in U$, it holds that*

$$\mathit{weight}(\mathbf{u}) \equiv_m c.$$

Then, if $\mathbf{w}$ is a witness for U, it holds that $\mathit{weight}(\mathbf{w}) \equiv_m c$.

Proof. Since $\mathbf{w}$ is a witness for U, there exist tuples $\mathbf{t_1} = (t_{1,1}, \ldots, t_{1,n}), \ldots, \mathbf{t_k} = (t_{k,1}, \ldots, t_{k,n})$ and a balanced operation $f\colon \{0,1\}^k \to \{0,1\}$ such that

$$f(\mathbf{t_1}, \ldots, \mathbf{t_k}) = \mathbf{w}.$$

Let $\alpha_1, \ldots, \alpha_k$ be the coefficients of f. From $f(\mathbf{t_1}, \ldots, \mathbf{t_k}) = \mathbf{w}$, we obtain that $\alpha_1 \mathrm{weight}(\mathbf{t_1}) + \cdots + \alpha_k \mathrm{weight}(\mathbf{t_k}) = \mathrm{weight}(\mathbf{w})$. Since $\sum_{i \in [k]} \alpha_i = 1$ by definition of a balanced operation, we have

$$\alpha_1 \mathrm{weight}(\mathbf{t_1}) + \cdots + \alpha_k \mathrm{weight}(\mathbf{t_k}) \equiv_m \alpha_1 c + \cdots + \alpha_k c = \left(\sum_{i \in [k]} \alpha_i \right) c = c,$$

and the result follows. ◀

We will view Boolean tuples of arity n as maps $f\colon [n] \to \{0,1\}$, via the natural correspondence where such a map f represents the tuple $(f(1), \ldots, f(n))$. We freely interchange between these two representations of tuples.

For $S \subseteq \mathbb{N}$, we say that $f\colon S \to \{0,1\}$ is a *no-good* of $U \subseteq \{0,1\}^n$ when:

- $S \subseteq [n]$;
- each extension $g\colon [n] \to \{0,1\}$ of f is not an element of U; and
- there exists an extension $h\colon [n] \to \{0,1\}$ of f that is a witness for U.

We say that $f\colon S \to \{0,1\}$ is a *min-no-good* if f is a no-good, but no proper restriction of f is a no-good. Observe that the following are equivalent, for a relation: the relation is not balanced; it has a witness; it has a no-good; it has a min-no-good.

When $U \subseteq \{0,1\}^n$ is a relation and $S \subseteq [n]$, let $s_1 < \cdots < s_m$ denote the elements of S; then, we use $U \restriction S$ to denote the relation $\{(h(s_1), \ldots, h(s_m)) \mid h \in U\}$.

▶ **Proposition 4.45** *Let $U \subseteq \{0,1\}^n$ be a relation, let $S \subseteq [n]$, and suppose that $f\colon S \to \{0,1\}$ is a min-no-good of U. Then f is a min-no-good of $U \restriction S$.*

Proof. Observe that f is not in $U \restriction S$; since f has an extension that is a witness for U, it follows that f is a witness for $U \restriction S$. Thus, f is a no-good of $U \restriction S$. In order to obtain that f is a min-no-good of $U \restriction S$, it suffices to establish that, for any restriction $f^- : S^- \to \{0,1\}$ of f, it holds that f^- is a no-good of U if and only if f^- is a no-good of $U \restriction S$. This follows from what we have established concerning f and the following fact: all extensions $h : S \to \{0,1\}$ of f^- are not in $U \restriction S$ if and only if all extensions $h' : [n] \to \{0,1\}$ of f^- are not in U. ◀

Using these tools we are finally in position to prove Theorem 4.46.

► **Theorem 4.46** *Suppose that $U \subseteq \{0,1\}^3$ is an arity-3 Boolean relation that is not balanced. Then, the* 2-OR *relation is cone-definable from U.*

Proof. Let $f : S \to \{0,1\}$ be a min-no-good of U.

It cannot hold that $|S| = 0$, since then U would be empty and hence preserved by all balanced operations. It also cannot hold that $|S| = 1$, since then f would be a min-no-good of $U \restriction S$ (by Proposition 4.45), which is not possible since $U \restriction S$ would have arity 1 and hence would be preserved by all balanced operations (by Observation 4.40).

For the remaining cases, by replacing U with a relation that is interdefinable with it, we may assume that $f : S \to \{0,1\}$ maps each $s \in S$ to 0.

Suppose that $|S| = 2$, and assume for the sake of notation that $S = \{1,2\}$ (this can be obtained by replacing U with a relation that is interdefinable with it). By Proposition 4.45, f is a min-no-good of $U \restriction S$. By Theorem 4.41, we obtain that $U \restriction S$ contains all tuples other than f, that is, we have $\{(0,1),(1,0),(1,1)\} = U \restriction S$. It follows that there exists a *realization*, where we define a *realization* to be a tuple $(a_1,a_2,a_3) \in \{0,1\}^3$ such that $(0,1,a_1),(1,0,a_2),(1,1,a_3) \in U$. Let us refer to $(0,0,1)$ and $(1,1,0)$ as *bad* tuples, and to all other arity 3 tuples as *good* tuples.

▷ **Claim 4.47** *If there is a realization that is a good tuple, then the* 2-OR *relation is cone-definable from U.*

Proof. We show cone-definability via a tuple of the form (x_1,x_2,y) where $y \in \{0,1,x_1,x_2,\neg x_1,\neg x_2\}$. The right setting for y can be derived from the realization that forms a good tuple.

- choose $y = 0$ for $(0,0,0)$;
- $y = 1$ for $(1,1,1)$;
- $y = x_1$ for $(0,1,1)$;
- $y = x_2$ for $(1,0,1)$;
- $y = \neg x_1$ for $(1,0,0)$; and,
- $y = \neg x_2$ for $(0,1,0)$.

It is easy to verify that this choice of y gives the desired cone-definition. ◁

▷ **Claim 4.48** *There is a realization that is a good tuple.*

Proof. Proof by contradiction. If there exists no realization that is a good tuple, every realization is a bad tuple; moreover, there is a unique realization, for if there were more than one, there would exist a realization that was a good tuple. We may assume (up to interdefinability of U) that the unique realization is $(1,1,0)$. Then, U is the relation $\{(0,1,1),(1,0,1),(1,1,0)\}$ containing exactly the weight 2 tuples; applying Lemma 4.44 to U with $a = 2$ and $m = 3$, we obtain that for any witness $\mathbf{w}$ for U, it holds that $\text{weight}(\mathbf{w}) \equiv_3 2$. This implies that f has no extension $\mathbf{w}'$ that is a witness, since any such extension must have $\text{weight}(\mathbf{w}')$ equal to 0 or 1 as f maps both $s \in S$ to 0; we have thus contradicted that f is a no-good of U. ◁

Together, the two claims complete the case that $|S| = 2$.

Suppose that $|S| = 3$. Since f is a min-no-good mapping all $s \in S$ to 0, it follows that $(0,0,0)$ is a witness. Thereby, it follows that each of the weight 1 tuples $(1,0,0),(0,1,0),(0,0,1)$ is contained in U. We claim that U contains a weight 2 tuple; if not, then U would contain only weight 1 and weight 3 tuples, and by invoking Lemma 4.44 with $a = 1$ and $m = 2$, we would obtain that $\text{weight}((0,0,0)) \equiv_2 1$, a contradiction. Assume for the sake of notation that U contains the weight 2 tuple $(0,1,1)$. Then U cone-defines the 2-OR relation via the tuple $(0,x_1,x_2)$, since $(0,0,0) \notin R$ and $(0,1,0),(0,0,1),(0,1,1) \in R$. ◀

Combining the results in this section with the results in previous sections, allows us to give a full classification of the sparsifiability of constraint languages that only contain relations of arity at most three. Observe that any k-ary relation R such that $R \neq \emptyset$ and $\{0,1\}^k \setminus R \neq \emptyset$ cone-defines the 1-OR relation. Since we assume that Γ is intractable in the next theorem, it follows that k is always defined and $k \in \{1,2,3\}$.

► **Theorem 4.49** *Let Γ be an intractable Boolean constraint language such that each relation therein has arity ≤ 3. Let $k \in \mathbb{N}$ be the largest value for which k-OR can be cone-defined from a relation in Γ. Then $\text{CSP}(\Gamma)$ has a kernel with $\mathcal{O}(n^k)$ constraints that can be encoded in $\mathcal{O}(n^k \log k)$ bits, but for any $\varepsilon > 0$ there is no kernel of size $\mathcal{O}(n^{k-\varepsilon})$, unless $\text{NP} \subseteq \text{coNP/poly}$.*

Proof. To show that there is a kernel with $\mathcal{O}(n^k)$ constraints, we do a case distinction on k.

$(k = 1)$ If $k = 1$, there is no relation in Γ that cone-defines the 2-OR relation. It follows from Observation 4.40 and Theorems 4.41 and 4.46 that thereby, Γ is balanced. It now follows from Theorem 4.26 that $\text{CSP}(\Gamma)$ has a kernel with $\mathcal{O}(n)$ constraints that can be stored in $\mathcal{O}(n \log n)$ bits.

($k = 2$) If $k = 2$, there is no relation $R \in \Gamma$ with $|R| = 2^3 - 1 = 7$, as otherwise by Lemma 4.13 such a relation R would cone-define 3-OR which is a contradiction. Thereby, it follows from Theorem 4.14 that CSP(Γ) has a sparsification with $\mathcal{O}(n^{3-1}) = \mathcal{O}(n^2)$ constraints that can be encoded in $\mathcal{O}(n^2 \log n)$ bits.

($k = 3$) Given an instance $(\mathcal{C}, V)$, it is easy to obtain a kernel of with $\mathcal{O}(n^3)$ constraints by simply removing duplicate constraints. This kernel can be stored in $\mathcal{O}(n^3)$ bits, by storing for each relation $R \in \Gamma$ and for each tuple $(x_1, x_2, x_3) \in V^3$ whether $R(x_1, x_2, x_3) \in \mathcal{C}$. Since $|\Gamma|$ is constant and there are $\mathcal{O}(n^3)$ such tuples, this results in using $\mathcal{O}(n^3)$ bits.

It remains to prove the lower bound. By definition, there exists $R \in \Gamma$ such that R cone-defines the k-OR relation. Thereby, the result follows immediately from Theorem 4.11. Thus, CSP(Γ) has no kernel of size $\mathcal{O}(n^{k-\varepsilon})$ for any $\varepsilon > 0$, unless NP $\subseteq$ coNP/poly. ◀

4.7 Capturing polynomials versus compression via Maltsev embeddings

In this section we compare our polynomial-based framework for linear sparsification from Section 4.4 to the framework of Lagerkvist and Wahlström [70] based on Maltsev embeddings.

4.7.1 Maltsev embeddings and definitions

To facilitate the discussion, we introduce some additional concepts. A ternary operation $f\colon D^3 \to D$ over a domain D is a *Maltsev operation* if it satisfies the identities $f(x, x, y) = y$ and $f(x, y, y) = x$ for all $x, y \in D$.

Definition 4.50 ([70, Definition 7]) A constraint language Γ over a domain D admits an *embedding* over the constraint language Γ' over $D' \supseteq D$ if there exists a bijection $h\colon \Gamma \to \Gamma'$ such that for every $R \in \Gamma$, if the arity of R is k then the arity of $h(R)$ is k and $h(R) \cap D^k = R$.

If Γ' is preserved by a Maltsev operation, then Γ is said to *admit a Maltsev embedding*. Lagerkvist and Wahlström proved [70, Theorems 10–11] that if Γ is a constraint language (over a possibly non-Boolean domain) that admits a Maltsev embedding over Γ' with a finite domain, then CSP(Γ) has a kernel with $\mathcal{O}(n)$ constraints. Hence constraint languages admitting Maltsev embeddings over finite domains admit linear sparsifications, just as balanced constraint languages.

In a quest to understand which CSPs can be sparsified through this route, they investigated which constraint languages admit Maltsev embeddings using universal algebra. For this purpose, they defined a *universal partial Maltsev*

operation as a partial Boolean operation f such that each Boolean constraint language Γ that admits a Maltsev embedding is preserved by f.

Definition 4.51 For a k-ary operation $f\colon D^k \to D$ on a domain $D \supseteq \{0,1\}$, the partial Boolean operation $f_{|\mathbb{B}}$ is the restriction of f to Boolean arguments that result in a Boolean value. Hence $\text{domain}(f_{|\mathbb{B}}) = \{\mathbf{x} \in \{0,1\}^k \mid f(\mathbf{x}) \in \{0,1\}\}$, and for each k-tuple $\mathbf{x}$ in the domain of $f_{|\mathbb{B}}$ we have $f_{|\mathbb{B}}(\mathbf{x}) = f(\mathbf{x})$.

Definition 4.52 ([70, Definition 14]) The infinite domain D_∞ is recursively defined as follows. It contains the elements 0 and 1 along with each triple (x,y,z) where $x,y,z \in D_\infty$ with $x \neq y$ and $y \neq z$. The Maltsev operation $u\colon D_\infty^3 \to D_\infty$ is defined by setting $u(x,x,y) = y$, setting $u(x,y,y) = x$, and setting $u(x,y,z) = (x,y,z)$ otherwise.

We define $[\{u\}]$ as the set of all operations that can be defined as a term over the algebra $(D_\infty, \{u\})$. Hence an arity-k operation $f \in [\{u\}]$ can be defined either as a projection, so that $f(x_1,\ldots,x_k) = x_i$ for some $i \in [k]$, or can be recursively defined from operations $f_1, f_2, f_3 \in [\{u\}]$ via $f(x_1,\ldots,x_k) = u(f_1(x_1,\ldots,x_k), f_2(x_1,\ldots,x_k), f_3(x_1,\ldots,x_k))$. We will use this recursive decomposition of operations in $[\{u\}]$ later in our proofs. The universal partial Maltsev operations can be characterized precisely [70, Theorems 13–15, p.165] as the operations $f_{|\mathbb{B}}$ for $f \in [\{u\}]$.

Any Boolean constraint language that is preserved by all universal partial Maltsev operations, has [71, Theorem 28] a Maltsev embedding over D_∞. However, since only *finite-domain* Maltsev embeddings lead to linear sparsifications, this infinite-domain embedding does not directly have algorithmic applications.

In the remainder of this section, we explore relations between balanced Boolean constraint languages (which can be sparsified using capturing polynomials) and Boolean constraint languages admitting a Maltsev embedding.

4.7.2 Balanced constraint languages versus Maltsev embeddings

The next theorem shows that being balanced is at least as strong of a requirement as being preserved by all universal partial Maltsev operations. In particular, any balanced relation is preserved by all universal partial Maltsev operations. This implies that if the Maltsev approach does not apply to obtain a linear sparsification for a Boolean CSP, then the polynomial-based framework does not apply either.

▶ **Theorem 4.53** *Let Γ be a Boolean constraint language. If there exists a universal partial Maltsev operation f such that Γ is not preserved by f, then Γ is not balanced.*

Proof. We introduce a function to associate an integer value to every element of D_∞, as follows. Let $\text{val}\colon D_\infty \to \mathbb{N}$ be given by $\text{val}(0) := 0$, $\text{val}(1) := 1$ and

$\mathrm{val}((d_1,d_2,d_3)) := \mathrm{val}(d_1) - \mathrm{val}(d_2) + \mathrm{val}(d_3)$ for $d_1,d_2,d_3 \in D_\infty$. We start by proving the following claim.

▷ **Claim 4.54** *Let $f \in [\{u\}]$ have arity m. There is a sequence $\alpha_1,\dots,\alpha_m \in \mathbb{Z}$ with $\sum_{i\in[m]} \alpha_i = 1$, such that for all $x_1,\dots,x_m \in \{0,1\}$:*

$$\sum_{i\in[m]} \alpha_i x_i = val(f(x_1,\dots,x_m)).$$

Proof. We prove this by induction on the structure of f as given by a term over u.

(Base case) If $f(x_1,\dots,x_m) := x_j$ for $j \in [m]$, we define $\alpha_j := 1$ and $\alpha_i := 0$ for all $i \neq j$. By this definition, $\sum_{i\in[m]} \alpha_i = 1$ and $\mathrm{val}(f(x_1,\dots,x_m)) = x_j = \sum_{i\in[m]} \alpha_i x_i$.

(Step) Let $f(x_1,\dots,x_m) = u(f_1(x_1,\dots,x_m), f_2(x_1,\dots,x_m), f_3(x_1,\dots,x_m))$. For $b \in [3]$, choose coefficients $\alpha_{b,1},\dots,\alpha_{b,m} \in \mathbb{Z}$ with $\sum_{i\in[m]} \alpha_{b,i} = 1$ such that

$$\sum_{i\in[m]} \alpha_{b,i} x_i = \mathrm{val}(f_b(x_1,\dots,x_m))$$

for all $x_1,\dots,x_m \in \{0,1\}$. These coefficients exist by the induction hypothesis. For $i \in [m]$, define $\alpha_i := \alpha_{1,i} - \alpha_{2,i} + \alpha_{3,i}$. By this definition,

$$\sum_{i\in[m]} \alpha_i = \sum_{i\in[m]} \alpha_{1,i} - \sum_{i\in[m]} \alpha_{2,i} + \sum_{i\in[m]} \alpha_{3,i} = 1 - 1 + 1 = 1,$$

as desired. Let $x_1,\dots,x_m \in \{0,1\}$ be given. To show that $\sum_{i\in[m]} \alpha_i x_i = f(x_1,\dots,x_m)$ we distinguish three cases.

Suppose $f_1(x_1,\dots,x_m) = f_2(x_1,\dots,x_m)$. Then $f(x_1,\dots,x_m) = f_3(x_1,\dots,x_m)$ by the definition of u, and thus

$$\begin{aligned}\sum_{i\in[m]} \alpha_i x_i &= \sum_{i\in[m]} \alpha_{1,i} x_i - \sum_{i\in[m]} \alpha_{2,i} x_i + \sum_{i\in[m]} \alpha_{3,i} x_i \\ &= \mathrm{val}(f_1(x_1,\dots,x_m)) - \mathrm{val}(f_2(x_1,\dots,x_m)) + \mathrm{val}(f_3(x_1,\dots,x_m)) \\ &= \mathrm{val}(f_3(x_1,\dots,x_m)) = \mathrm{val}(f(x_1,\dots,x_m)).\end{aligned}$$

If $f_3(x_1,\dots,x_m) = f_2(x_1,\dots,x_m)$, then a symmetric argument to the case above shows that indeed $\sum_{i\in[m]} \alpha_i x_i = f(x_1,\dots,x_m)$.

Otherwise, we know that $f_3(x_1,\dots,x_m) \neq f_2(x_1,\dots,x_m)$ and $f_1(x_1,\dots,x_m) \neq f_2(x_1,\dots,x_m)$. It follows that

$$f(x_1,\dots,x_m) = (f_1(x_1,\dots,x_m), f_2(x_1,\dots,x_m), f_3(x_1,\dots,x_m))$$

and we obtain

$$\begin{aligned}\sum_{i\in[m]} \alpha_i x_i &= \sum_{i\in[m]} \alpha_{1,i} x_i - \sum_{i\in[m]} \alpha_{2,i} x_i + \sum_{i\in[m]} \alpha_{3,i} x_i \\ &= \mathrm{val}(f_1(x_1,\dots,x_m)) - \mathrm{val}(f_2(x_1,\dots,x_m)) + \mathrm{val}(f_3(x_1,\dots,x_m)) \\ &= \mathrm{val}(\,(f_1(x_1,\dots,x_m), f_2(x_1,\dots,x_m), f_3(x_1,\dots,x_m))\,) \\ &= \mathrm{val}(f(x_1,\dots,x_m)),\end{aligned}$$

concluding the proof of this claim. ◁

Using the claim we prove Theorem 4.53. Let $h\colon \{0,1\}^m \to \{0,1\}$ be an m-ary universal partial Maltsev operation that does not preserve Γ. As described in Section 4.7.1, there exists $f \in [\{u\}]$ such that $h = f_{|\mathbb{B}}$. Let $R \in \Gamma$ be a k-ary relation such that R is not preserved by h. We show that there exists a balanced operation g_f that does not preserve R. By Claim 4.54 there exist coefficients $\alpha_1,\dots,\alpha_m \in \mathbb{Z}$ such that

$$\sum_{i\in[m]} \alpha_i x_i = \mathrm{val}(f(x_1,\dots,x_m))$$

for all $x_1,\dots,x_m \in \{0,1\}$. Let g_f be the balanced operation with coefficients α_i. Since $f_{|\mathbb{B}}$ does not preserve R, there are $s_1,\dots,s_m \in R$ such that $f_{|\mathbb{B}}(s_1,\dots,s_m)$ is well-defined and $f_{|\mathbb{B}}(s_1,\dots,s_m) \notin R$. Let $s_i = (s_{i,1},\dots,s_{i,k})$. Then since $f_{|\mathbb{B}}(s_1,\dots,s_m)$ is well-defined, it evaluates to a 0/1-tuple and

$$\mathrm{val}(f(s_{1,j},\dots,s_{m,j})) = f(s_{1,j},\dots,s_{m,j}),$$

for all $j \in [k]$. Since $\sum_{i\in[m]} \alpha_i x_i = \mathrm{val}(f(x_1,\dots,x_m))$ for all $x_1,\dots,x_m \in \{0,1\}$, it follows that g_f violates R. So g_f is a balanced operation that does not preserve Γ, concluding the proof. ◀

4.7.3 Balanced constraint languages versus finite-domain Maltsev embeddings

Theorem 4.53 implies [71, Theorem 28] that any balanced constraint language has a Maltsev embedding over the infinite domain D_∞. We show in the next theorem that in fact, every balanced constraint language allows a Maltsev embedding over a *finite* domain. Thus, there are in fact two ways to obtain a kernel with $\mathcal{O}(n)$ constraints for balanced constraint languages, one given by Theorem 4.26 and one via Maltsev embeddings [70, Theorems 10–11].

▶ **Theorem 4.55** *If Γ is a balanced Boolean constraint language, then Γ admits a Maltsev embedding over a finite domain.*

Proof. Since Γ is balanced, it follows from Claim 4.27 that for every $R \in \Gamma$ and $\mathbf{u} \notin R$ there exists a linear polynomial $p_{\mathbf{u},R}$ and prime power $q_{\mathbf{u},R}$ such that $p_{\mathbf{u},R}$ captures $\mathbf{u}$ with respect to R over $\mathbb{Z}/q_{\mathbf{u},R}\mathbb{Z}$. Enumerate the obtained polynomials arbitrarily as $p_1, \ldots, p_m$ and their corresponding moduli as $q_1, \ldots, q_m$. Let D be the Cartesian product of the rings $\mathbb{Z}/q_{\mathbf{u},R}\mathbb{Z}$ we thus obtained, such that $D = \{0, \ldots, q_1 - 1\} \times \{0, \ldots, q_2 - 1\} \times \ldots \times \{0, \ldots, q_m - 1\}$. We equate the value $(0, \ldots, 0) \in D$ with the Boolean 0 and $(1, \ldots, 1) \in D$ with the Boolean value 1. Let $\varphi \colon D^3 \to D$ be the Maltsev operation given by $\varphi(x, y, z) = w$ with $w_i = x_i - y_i + z_i \pmod{q_i}$ for all $i \in [m]$, for $x, y, z \in D$.

We now define a constraint language Γ' over D such that Γ' is preserved by φ. For any k-ary relation $R \in \Gamma$, we show how to define a corresponding k-ary relation $R' \in \Gamma'$. Initialize R' as R. Then, recursively add any $\mathbf{x} \in D^k$ to R' for which there exist $\mathbf{t_1}, \mathbf{t_2}, \mathbf{t_3} \in R'$ such that $\varphi(\mathbf{t_1}, \mathbf{t_2}, \mathbf{t_3}) = \mathbf{x}$. This completes the definition of Γ'. It is easy to verify that Γ' is preserved by φ. To complete the Maltsev embedding, it remains to show that for all k-ary $R \in \Gamma$ we have $R' \cap \{0,1\}^k = R$.

Suppose for contradiction that there exists $R \in \Gamma$ such that $R' \cap \{0,1\}^k \neq R$, which implies that there exists $\mathbf{x} \in \{0,1\}^k$ such that $\mathbf{x} \in R'$ but $\mathbf{x} \notin R$. From the construction of R' it follows by an easy induction that there exist $\mathbf{t_1}, \ldots, \mathbf{t_\ell} \in R$ with $\mathbf{t_a} = (t_{a,1}, \ldots, t_{a,k})$, such that for all entries $i \in [k]$ of an assignment and all coordinates $j \in [m]$ of a domain element, we have

$$x_i \equiv_{q_j} t_{1,i} - t_{2,i} + t_{3,i} - t_{4,i} \ldots + t_{\ell,i}.$$

Consider the index $j \in [m]$ for which $p_j = p_{\mathbf{x},R}$. Since $p_{\mathbf{x},R}$ is a linear polynomial, it can be written as $p_{\mathbf{x},R}(y_1, \ldots, y_k) = \alpha_0 + \sum_{i=1}^{k} \alpha_i y_i$ for coefficients $\alpha_0, \ldots, \alpha_k \in \mathbb{Z}/q_{\mathbf{x},R}\mathbb{Z}$. Interpreting this polynomial over the integers, we have that $p_j(t_{a,1}, \ldots, t_{a,k}) \equiv_{q_j} 0$ for all $a \in [\ell]$, since p_j captures x with respect to R and $t_a \in R$. Then

$$\begin{aligned}
p_j(x_1, \ldots, x_k) &\equiv_{q_j} \alpha_0 + \sum_{i=1}^{k} \alpha_i x_i \equiv_{q_j} \alpha_0 + \sum_{i=1}^{k} \alpha_i (t_{1,i} - t_{2,i} \ldots + t_{\ell,i}) \\
&\equiv_{q_j} \alpha_0 + \sum_{i=1}^{k} \sum_{a=1}^{\ell} (-1)^{a-1} \alpha_i \cdot t_{a,i} \\
&\equiv_{q_j} \alpha_0 + \sum_{a=1}^{\ell} (-1)^{a-1} \sum_{i=1}^{k} \alpha_i \cdot t_{a,i} \\
&\equiv_{q_j} \sum_{a=1}^{\ell} (-1)^{a-1} \left(\alpha_0 + \sum_{i=1}^{k} \alpha_i \cdot t_{a,i} \right) \\
&\equiv_{q_j} \sum_{a=1}^{\ell} (-1)^{a-1} p_j(t_{a,1}, \ldots, t_{a,k}) \equiv_{q_j} 0.
\end{aligned}$$

Thereby, $p_{\mathbf{x},R}(x_1,\ldots,x_k) \equiv_{q_{\mathbf{x},R}} 0$, contradicting that $p_{\mathbf{x},R}$ captures $\mathbf{x}$ with respect to R. This concludes the proof. ◀

Theorem 4.55 shows that balanced constraint languages have (finite-domain) Maltsev embeddings. In the next theorem, we prove a partial converse: constraint languages that are not balanced, do not admit Maltsev embeddings of a particular type over a (possibly infinite) domain. The particular type we consider consists of embeddings for which the constraint language Γ' into which we embed has a group structure on its domain, and the Maltsev operation that preserves Γ' is the coset generating operation of the group. Let us give the relevant definitions.

Definition 4.56 Let $(D,\cdot)$ be a group. The *coset generating operation* of the group is the Maltsev operation $c\colon D^3 \to D$ defined by $c(x,y,z) = x \cdot y^{-1} \cdot z$.

▶ **Theorem 4.57** *Let Γ be a Boolean constraint language that is not balanced, and let $(D,\cdot)$ be a group with coset generating operation c. Then there is no constraint language Γ' over domain $D \supseteq \{0,1\}$ that is preserved by c and for which Γ admits an embedding over Γ', not even if the identity element of $(D,\cdot)$ is allowed to differ from $\{0,1\}$.*

Proof. Suppose for contradiction that there exists a group $(D,\cdot)$ with coset generating operation c and a constraint language Γ' such that Γ' is preserved by c and Γ admits an embedding over Γ'. Let $R \in \Gamma$ be a relation that is not balanced, suppose R has arity k. Since R is not balanced, take $\mathbf{r_1},\ldots,\mathbf{r_m} \in R$ with $\mathbf{r_1} - \mathbf{r_2} \ldots + \mathbf{r_m} = \mathbf{u} \notin R$. Let $\widehat{R} \in \Gamma'$ be the image of R under the considered Maltsev embedding. We start by proving the following claim.

▷ **Claim 4.58** *For all $i \in [k]$, the following equation holds over the group $(D,\cdot)$:*

$$r_{1,i} \cdot r_{2,i}^{-1} \cdot \ldots \cdot r_{m,i} = u_i.$$

Proof. We show by induction that for all $x_1,\ldots,x_m \in \{0,1\}$ with $x_1 - x_2 \ldots + x_m = y$ for $y \in \{0,1\}$, it holds that $x_1 \cdot x_2^{-1} \cdot \ldots \cdot x_m = y$ over D.

(Base case) If $m = 1$, we trivially obtain that $x_1 = y$.

(Step) Suppose $m > 1$. We start by showing that there exists $j \in [m-1]$ such that $x_j = x_{j+1}$. Suppose not, then we are in one of the two cases below.

- $x_1 = x_3 = \ldots = x_m = 0$ and $x_2 = x_4 = \ldots = x_{m-1} = 1$, or
- $x_1 = x_3 = \ldots = x_m = 1$ and $x_2 = x_4 = \ldots = x_{m-1} = 0$.

In both cases it is easily verified that for this choice of variables, $x_1 - x_2 \ldots + x_m \notin \{0,1\}$, which is a contradiction. Therefore, there exists $j \in [m-1]$ such that $x_j = x_{j+1}$. Suppose for ease of notation that j is even, the case when j is odd follows symmetrically. It is easy to verify that $y = x_1 - x_2 \ldots + x_{j-1} - x_j + x_{j+1} - x_{j+2} \ldots + x_m = x_1 - x_2 \ldots + x_{j-1} - x_{j+2} \ldots + x_m$, and thus it follows

from the induction hypothesis that $x_1 \cdot x_2^{-1} \cdot \ldots \cdot x_{j-1} \cdot x_{j+2}^{-1} \cdot \ldots \cdot x_m = y$. Thereby

$$\begin{aligned} &x_1 \cdot x_2^{-1} \cdot \ldots \cdot x_{j-1} \cdot x_j^{-1} \cdot x_{j+1} \cdot x_{j+2}^{-1} \cdot \ldots \cdot x_m = \\ &x_1 \cdot x_2^{-1} \cdot \ldots \cdot x_{j-1} \cdot (x_j^{-1} \cdot x_j) \cdot x_{j+2}^{-1} \cdot \ldots \cdot x_m = \\ &x_1 \cdot x_2^{-1} \cdot \ldots \cdot x_{j-1} \cdot x_{j+2}^{-1} \cdot \ldots \cdot x_{m-2} = y. \end{aligned}$$

Since $\mathbf{r_1}, \ldots, \mathbf{r_m}$ were chosen such that $\mathbf{r_1} - \mathbf{r_2} \ldots + \mathbf{r_m} = \mathbf{u} \in \{0,1\}^k \setminus R$, the statement of the claim follows. ◁

Recall that $\widehat{R}$ is preserved by c. We have the following claim.

▷ **Claim 4.59** *Let $\widehat{R}$ be a k-ary relation and $m \geq 3$ be odd. If $\widehat{R}$ is preserved by c, then it is preserved by the m-ary operation $f_m \colon D^m \to D$ given by $f_m(x_1, \ldots, x_m) := x_1 \cdot x_2^{-1} \cdot \ldots \cdot x_m$.*

Proof. We show this by a simple induction. If $m = 3$, the statement is true since in this case f and c are equivalent. Let $m > 3$ and let $\mathbf{t_1}, \ldots, \mathbf{t_m} \in \widehat{R}$ be given, we show $f(\mathbf{t_1}, \ldots, \mathbf{t_m}) \in \widehat{R}$. Let $\mathbf{t}' := c(\mathbf{t_1}, \mathbf{t_2}, \mathbf{t_3})$, observe that $\mathbf{t}' \in \widehat{R}$ since $\widehat{R}$ is preserved by c. Choose $i \in [k]$ and observe that

$$\begin{aligned} f_m(t_{1,i}, \ldots, t_{m,i}) &= t_{1,i} \cdot t_{2,i}^{-1} \cdot t_{3,i} \cdot t_{4,i}^{-1} \cdot \ldots \cdot t_{m,i} \\ &= c(t_{1,i}, t_{2,i}, t_{3,i}) \cdot t_{4,i}^{-1} \cdot \ldots \cdot t_{m,i} \\ &= t' \cdot t_{4,i}^{-1} \cdot \ldots \cdot t_{m,i}. \end{aligned}$$

Thereby, $f_m(\mathbf{t_1}, \ldots, \mathbf{t_m}) = f_{m-2}(\mathbf{t}', \mathbf{t_4}, \ldots, \mathbf{t_m})$. It follows from the induction hypothesis that f_{m-2} is preserved by $\widehat{R}$ and thus $f_{m-2}(\mathbf{t}', \mathbf{t_4}, \ldots, \mathbf{t_m}) \in \widehat{R}$. ◁

It follows from the fact that c preserves $\widehat{R}$ and Claim 4.59, that f_m preserves $\widehat{R}$. However, by Claim 4.58, it follows that $f_m(\mathbf{r_1}, \ldots, \mathbf{r_m}) = \mathbf{u}$ and thus $\mathbf{u} \in \widehat{R}$, contradicting that we have given a valid Maltsev embedding of Γ, since $\mathbf{u} \in \{0,1\}^k$, but $\mathbf{u} \notin R$. ◀

We conclude the subsection with a discussion of the implications of Theorems 4.55 and 4.57. On the one hand, Theorem 4.55 shows that balanced constraint languages admit Maltsev embeddings over finite domains. On the other hand, Theorem 4.57 shows that unbalanced constraint languages do not allow Maltsev embeddings over the coset generating operation of a group, not even an infinite group. As a corollary to these results, we therefore obtain an infinite-domain to finite-domain transformation, for embeddings via coset generating operations.

▶ **Corollary 4.60** *If Γ is a Boolean constraint language that has a Maltsev embedding over Γ', where the domain D of Γ' is a (possibly infinite) group and Γ' is preserved by the coset generating operation of the group, then Γ has a Maltsev embedding over a finite domain.* ◀

4.7.4 Preservation by balanced or universal partial Maltsev operations

In this section we compare preservation by balanced operations to preservation by universal partial Maltsev operations. Theorem 4.53 implies that every balanced constraint language is preserved by all universal partial Maltsev operations. At this point, it is unknown whether the converse also holds. If the Boolean constraint language Γ is preserved by all universal partial Maltsev operations, then is it also balanced?

We have not managed to resolve this question, but we present some insights in this direction. Recall that a_i is the alternating (partial) operation of arity i, for odd $i \geq 1$. It is easy to verify that a_3 is equivalent to $u_{|\mathbb{B}}$. We show the following result about a_5.

▶ **Theorem 4.61** *Let Γ be a Boolean constraint language. If Γ is not preserved by a_5, then there is a universal partial Maltsev operation that does not preserve Γ.*

Proof. We start by considering the term $f \in [\{u\}]$ defined as follows:

$$f(x_1,\dots,x_5) := u(x_1,u(x_2,x_3,u(x_1,x_2,x_3)),u(x_5,x_4,u(x_3,x_2,x_1))).$$

The key of the proof is that the Boolean restriction $f_{|\mathbb{B}}$ of f does not preserve Γ. We start by showing how f relates to the alternating operation of arity 5.

▷ **Claim 4.62** *For all $x_1,\dots,x_5 \in \{0,1\}$ such that $a_5(x_1,\dots,x_5) \in \{0,1\}$, it holds that*

$$f(x_1,\dots,x_5) = a_5(x_1,\dots,x_5).$$

Proof. Suppose $a_5(x_1,\dots,x_5) \in \{0,1\}$, we do a case distinction.

$(u(x_5,x_4,u(x_3,x_2,x_1)) \in \{0,1\})$ Observe that in particular, this implies that $u(x_3,x_2,x_1) \in \{0,1\}$, implying that $x_3 = x_2$ or $x_1 = x_2$. If $x_3 = x_2$, we obtain that $u(x_2,x_3,u(x_1,x_2,x_3)) = u(x_3,x_3,u(x_1,x_3,x_3)) = x_1$, implying that $f(x_1,\dots,x_5) = u(x_5,x_4,u(x_3,x_2,x_1))$ and since $u(a,b,c) = a - b + c$ if $u(a,b,c) \in \{0,1\}$, the result follows. Similarly, if $x_1 = x_2$, then $u(x_2,x_3,u(x_1,x_2,x_3)) = u(x_1,x_3,u(x_1,x_1,x_3)) = x_1$. Just like in the previous case, this implies $f(x_1,\dots,x_5) = u(x_5,x_4,u(x_3,x_2,x_1))$ and the result follows.

$(u(x_5,x_4,u(x_3,x_2,x_1)) \notin \{0,1\})$ We again consider two options, based on whether $u(x_3,x_2,x_1)$ is in $\{0,1\}$ or not. Observe that if $u(x_3,x_2,x_1) \in \{0,1\}$, the reason that $u(x_5,x_4,u(x_3,x_2,x_1)) \notin \{0,1\}$ is that $x_5 = u(x_3,x_2,x_1)$ and $x_5 \neq x_4$. But then by definition, $x_5 = x_3 - x_2 + x_1$ and thus if $x_5 = 1$, then $x_4 =$

0 and we obtain $x_5 - x_4 + x_3 - x_2 + x_1 = 2 \notin \{0,1\}$, which is a contradiction. Similarly, if $x_5 = 0$ we obtain that $x_5 - x_4 + x_3 - x_2 + x_1 = -1 \notin \{0,1\}$, which is again a contradiction. We thereby conclude that $u(x_3, x_2, x_1) \notin \{0,1\}$.

By definition, this implies $x_3 = x_1 \neq x_2$. It follows that $x_4 \neq x_5$ to ensure that $a_5(x_1, \ldots, x_5) = x_5 - x_4 + x_3 - x_2 + x_1 \in \{0,1\}$. Furthermore, observe that $x_4 = x_3$ for this same reason. Thus, $x_1 = x_3 = x_4 \neq x_2 = x_5$. If follows that $a_5(x_1, \ldots, x_5) = x_1$. Furthermore, substituting this into the formula shows that $u(x_5, x_4, u(x_3, x_2, x_1)) = u(x_2, x_1, u(x_1, x_2, x_1)) = (x_2, x_1, (x_1, x_2, x_1)) = u(x_2, x_3, u(x_1, x_2, x_3))$, implying that $f(x_1, \ldots, x_5) = x_1$, as desired. ◁

Now let Γ be a constraint language that is not preserved by a_5. It follows from Claim 4.62 that for any $(x_1, \ldots, x_5) \in \text{domain}(a_5)$, it holds that $a_5(x_1, \ldots, x_5) = f(x_1, \ldots, x_5)$ and $(x_1, \ldots, x_5) \in \text{domain}(f_{|\mathbb{B}})$. It follows that Γ is not preserved by $f_{|\mathbb{B}}$, which is a universal partial Maltsev operation [70, Theorem 15]. ◀

Theorem 4.61, together with the observation that a_3 is equivalent to $u_{|\mathbb{B}}$, have the following consequence. If a constraint language Γ is unbalanced because some alternating operation of arity at most five does not preserve it, then Γ does not admit a Maltsev embedding, not even over an infinite domain. We leave it for future work to determine whether there exist Boolean constraint languages that admit finite-domain Maltsev embeddings, but are not balanced. Are there constraint languages for which the Maltsev framework yields linear kernelizations, but the polynomial-based framework does not?

4.7.5 Preservation by partial Maltsev operations versus cone-definability

There is a close relation between cone-definability of 2-OR and preserving the Boolean operation φ_1, where φ_1 is defined by $\varphi_1(x,x,y) = y$, $\varphi_1(x,y,y) = x$, and $\text{domain}(\varphi_1) = \{(0,0,0), (0,0,1), (0,1,1), (1,0,0), (1,1,0), (1,1,1)\}$.

▶ **Proposition 4.63** *Let Γ be a Boolean constraint language. Then Γ is not preserved by φ_1 if and only if there exists $R \in \Gamma$ such that R cone-defines* 2-OR.

Proof. ($\Rightarrow$) Suppose $R \in \Gamma$ is not preserved by φ_1. We show that R cone-defines 2-OR. Let m be the arity of R and let $\mathbf{t_1}, \mathbf{t_2}, \mathbf{t_3} \in R$ and $\mathbf{u} \in \{0,1\}^m \setminus R$ such that $\varphi_1(\mathbf{t_1}, \mathbf{t_2}, \mathbf{t_3}) = \mathbf{u}$.

We cone-define the 2-OR relation on variables x_1 and x_2 via the tuple $(y_1, \ldots, y_m)$ as follows. Consider $i \in [m]$, if $t_{1,i} = t_{2,i} = t_{3,i} = u_i$ let y_1 have the constant value $u_i \in \{0,1\}$. If $(t_{1,i}, t_{2,i}, t_{3,i}) = (1,1,0)$ let $y_i := x_1$, if $(t_{1,i}, t_{2,i}, t_{3,i}) = (0,0,1)$ let $y_i := \neg x_1$. Similarly, if $(t_{1,i}, t_{2,i}, t_{3,i}) = (0,1,1)$ let $y_i := x_2$, if $(t_{1,i}, t_{2,i}, t_{3,i}) = (1,0,0)$ let $y_i := \neg x_2$. Note that this covers all cases for the value of $(t_{1,i}, t_{2,i}, t_{3,i})$.

Let $f\colon \{x_1, x_2\} \to \{0,1\}$. We show that $f(x_1) \vee f(x_2)$ if and only if $(\hat{f}(y_1), \ldots, \hat{f}(y_m)) \in R$, with $\hat{f}$ as in Definition 4.7. We do a case distinction. If

$f(x_1) = 1$ and $f(x_2) = 1$, then one may verify that $(\hat{f}(y_1), \ldots, \hat{f}(y_m)) = \mathbf{t_2} \in R$. If $f(x_1) = 0$ and $f(x_2) = 1$, then $(\hat{f}(y_1), \ldots, \hat{f}(y_m)) = \mathbf{t_3}$. If $f(x_2) = 0$ and $f(x_1) = 1$, then $(\hat{f}(y_1), \ldots, \hat{f}(y_m)) = \mathbf{t_1} \in R$. Finally, if $f(x_1) = f(x_2) = 0$, then $(\hat{f}(y_1), \ldots, \hat{f}(y_m)) = \mathbf{u} \notin R$, as desired.

To complete the cone-definition, it remains to show that there are $i, j \in [m]$ such that $y_i \in \{x_1, \neg x_1\}$ and $y_j \in \{x_2, \neg x_2\}$. Now observe that $\mathbf{t_1} \neq \mathbf{t_2} \neq \mathbf{t_3}$: if $\mathbf{t_1} = \mathbf{t_2}$ then by definition of φ_1 we have $\varphi_1(\mathbf{t_1}, \mathbf{t_2}, \mathbf{t_3}) = \mathbf{t_3} \in R$, and similarly if $\mathbf{t_2} = \mathbf{t_3}$ then $\varphi_1(\mathbf{t_1}, \mathbf{t_2}, \mathbf{t_3}) = \mathbf{t_1} \in R$. Since $\varphi_1(\mathbf{t_1}, \mathbf{t_2}, \mathbf{t_3}) \notin R$, this cannot be. If $\mathbf{t_1} = \mathbf{t_3} \neq \mathbf{t_2}$, then consider a coordinate $k \in [m]$ for which $t_{1,k} = t_{3,k} \neq t_{2,k}$; but then φ_1 is not defined on this coordinate. Consequently, all three tuples $\mathbf{t_1}, \mathbf{t_2}, \mathbf{t_3}$ are distinct. Now, since the argumentation above shows that $(\hat{f}(y_1), \ldots, \hat{f}(y_m))$ can evaluate to each of $\mathbf{t_1}, \mathbf{t_2}, \mathbf{t_3}$, depending on the values assigned to x_1 and x_2, it follows that $\hat{f}$ depends on both x_1 and x_2 which implies the existence of the desired positions $i, j \in [m]$.

($\Leftarrow$) Suppose there exists $R \in \Gamma$ of some arity m that cone-defines 2-OR. Let $(y_1, \ldots, y_m)$ be the tuple witnessing this, with $y_i \in \{0, 1\} \cup \{x_1, \neg x_1, x_2, \neg x_2\}$. Define $f\colon \{x_1, x_2\} \to \{0, 1\}$ as $f(x_1) = 0$ and $f(x_2) = 1$. Let $\mathbf{t_1} := (\hat{f}(y_1), \ldots, \hat{f}(y_m)) \in R$ where $\hat{f}$ is the natural extension of f. Similarly, for $f'(x_1) = f'(x_2) = 1$ let $\mathbf{t_2} := (\hat{f'}(y_1), \ldots, \hat{f'}(y_m))$ and for $f''(x_1) = 1$, $f''(x_2) = 0$ let $\mathbf{t_3} := (\hat{f''}(y_1), \ldots, \hat{f''}(y_m))$. Finally, let $\mathbf{u}$ be the tuple witnessing that for $f'''(x_1) = f'''(x_2) = 0$, $(\hat{f'''}(y_1), \ldots, \hat{f'''}(y_m)) = \mathbf{u} \notin R$. We will show that $\varphi_1(\mathbf{t_1}, \mathbf{t_2}, \mathbf{t_3}) = \mathbf{u}$, thus showing that φ_1 does not preserve R.

We show that $u_i = t_{1,i} - t_{2,i} + t_{3,i}$ for each $i \in [m]$. Suppose $y_i = 0$, then by the definition above $t_{1,i} = t_{2,i} = t_{3,i} = u_i = 0$ as $\hat{f}(y_i) = 0$ for any $f\colon \{x_1, x_2\} \to \{0, 1\}$. Thus, $u_i = t_{1,i} - t_{2,i} + t_{3,i}$ in this case. Similarly, if $y_i = 1$ we obtain $u_i = 1 = t_{1,i} - t_{2,i} + t_{3,i}$.

Suppose $y_i = x_1$. Then by the definition above, $t_{1,i} = 0$ and $t_{2,i} = t_{3,i} = 1$. Furthermore, $u_i = 0 = 1 - 1 + 0$ as desired. For $y_i = \neg x_1$, we have $t_{1,i} = 1$ and $t_{2,i} = t_{3,i} = 0$ and $u_i = 1$, as desired. It is straightforward to verify that also when $y_i = x_2$ and $y_i = \neg x_2$ we obtain $u_i = t_{1,i} - t_{2,i} + t_{3,i}$, concluding the proof. ◀

In their work, [71, Theorem 32] Lagerkvist and Wahlström show the following. For every integer $d \geq 3$ and for every finite set P of partial polymorphisms for which the set of Boolean relations preserved by P can pp-define all Boolean relations, there is a polynomial-parameter transformation [16, 18] from d-CNF-SAT instances with n variables, to equivalent instances of CSP(Γ) on $\mathcal{O}(n^c)$ variables. Here $c \in \mathbb{N}$ depends only on P and Γ is a finite Boolean constraint language whose relations are preserved by all operations in P. Assuming NP $\not\subseteq$ coNP/poly, Theorem 2.9 states that the problem d-CNF-SAT has no kernel of bitsize $\mathcal{O}(n^{d-\varepsilon})$ for any $\varepsilon > 0$. Via the cited transformation, this implies CSP(Γ) has no sparsification of bitsize $\mathcal{O}(n^{d/c-\varepsilon})$. Hence knowing that the constraint language is preserved by any finite set of partial polymorphisms

does not guarantee any polynomial compressibility. Note that any constraint language that can cone-define k-OR for some $k \geq 2$ can also cone-define 2-OR, while Proposition 4.63 shows that being able to cone-define 2-OR is equivalent to being violated by the partial operation φ_1. A constraint language preserved by the single partial polymorphism φ_1 therefore does not cone-define k-OR for any $k \geq 2$. Using the transformation and incompressibility results mentioned above, we find (assuming NP $\not\subseteq$ coNP/poly) that for any $d' \in \mathbb{R}$ there is a finite Boolean constraint language $\Gamma_{d'}$ that does not cone-define 2-OR or larger, but for which CSP($\Gamma_{d'}$) does not have a sparsification of bitsize $\mathcal{O}(n^{d'})$. A general characterization of optimal sparsification size by the arity of the largest cone-definable OR is therefore impossible.

4.8 Conclusion

The ultimate goal of this line of research is to fully classify the sparsifiability of CSP(Γ), depending on Γ. In particular, we would like to classify those Γ for which $\mathcal{O}(n)$ sparsifiability is possible. In this chapter, we have shown that Γ being balanced is a sufficient condition to obtain a linear sparsification; it is tempting to conjecture that this condition is also necessary. The simplest example of a Boolean constraint language for which we currently do not understand whether or not it has a linear sparsification, consists of a single Boolean relation R^* of arity nine. Relation R^* has five satisfying assignments $\mathbf{s_1}, \ldots, \mathbf{s_5} \in \{0,1\}^9$:

$$R^* = \left\{ \begin{array}{l} \mathbf{s_1} = (1,0,0,1,1,1,0,0,1), \\ \mathbf{s_2} = (0,0,0,0,1,0,0,1,1), \\ \mathbf{s_3} = (0,1,0,1,1,0,1,1,0), \\ \mathbf{s_4} = (0,0,0,1,0,1,1,0,0), \\ \mathbf{s_5} = (0,0,1,0,0,1,1,1,1) \end{array} \right\}$$

Relation R^* is not balanced, since

$$\mathbf{s_1} - \mathbf{s_2} + \mathbf{s_3} - \mathbf{s_4} + \mathbf{s_5} = (1,1,1,1,1,1,1,1,1) \notin R^*.$$

By Theorem 4.61, it follows that R^* is violated by some universal partial Maltsev operation. Hence neither our polynomial framework for compression, nor the Maltsev-based approach yields a linear sparsification for CSP($\{R^*\}$). On the other hand, no superlinear lower bound is currently known. Resolving the sparsification complexity of CSP($\{R^*\}$) is the first obstacle in a general classification of linearly-compressible Boolean CSPs.

Note that the matrix consisting of the five satisfying assignments of R^* has a very succinct description: there is one column for each vector $\mathbf{x} \in \{0,1\}^5$ for which $x_1 - x_2 + x_3 - x_4 + x_5 = 1$, except for the all-ones column. The presence of these columns ensures that φ_1 preserves R^*, for the simple reason that $\varphi_1(\mathbf{s_i}, \mathbf{s_j}, \mathbf{s_k})$ for $i,j,k \in [5]$ is only defined when $i = j$ or $j = k$, in which

case the output tuple equals one of the input tuples. In other cases, there is an index ℓ for which $s_{i,\ell} - s_{j,\ell} + s_{k,\ell} \in \{-1, 2\}$, making the output undefined. Hence, using Proposition 4.63, relation R^* does not cone-define 2-OR.

Observe that CSP($\{R^*\}$) is NP-complete by Schaefer's dichotomy theorem (Theorem 2.37): R^* does not have the constantly-1 operation as a polymorphism, nor the constantly-0 operation; the tuples $\mathbf{s_1}, \mathbf{s_2}$ show that R^* is not preserved by the binary AND operation, nor by the binary OR operation; and the tuples $\mathbf{s_1}, \mathbf{s_2}, \mathbf{s_3}$ show that R^* is not preserved by the ternary majority or minority operations.

Resolving the sparsification complexity of CSP($\{R^*\}$), and subsequently obtaining a complete characterization of the Boolean CSPs that admit a linear compression, form the main open problems that remain. Other directions include the investigation of constraint languages of larger arity and the characterization of the CSPs that admit sparsifications with a quadratic, or even larger polynomial number, of constraints.

Acknowledgements We would like to thank Emil Jeřábek for the proof of Lemma 4.20. We thank Andrei Bulatov for insightful discussions, and thank Magnus Wahlström for sharing his ideas and an initial version of the proof of Theorem 4.61.

Chapter 5

Sparsification Lower Bounds for H-Coloring

In this chapter, we will provide a sparsification lower bound for the H-Coloring problem, showing that for a large class of graphs H, it does not have a non-trivial sparsification unless NP $\subseteq$ coNP/poly. This adds to the growing list of problems for which the existence of non-trivial sparsification algorithms has been ruled out under the assumption that NP $\not\subseteq$ coNP/poly, which includes Vertex Cover [33], Dominating Set [59], Feedback Arc Set [59], and Treewidth [56]. To the best of our knowledge, to date there is no non-trivial sparsification algorithm for any NP-hard problem that is defined on general graphs. This chapter adds another problem to this growing list of graph problems not admitting a non-trivial sparsification.

A second motivation for studying H-Coloring, is its relation to constraint satisfaction problems, as described in detail in Section 2.8. In short, any NP-hard H-Coloring problem translates into a CSP with a non-Boolean domain in which constraints have arity two. There have been a number of non-trivial advances in the sparsification for CSPs in over the Boolean domain, as can be seen from Chapters 3 and 4 in this thesis, and from recent work by Lagerkvist and Wahlström [70]. A natural step in that line of research is to consider CSPs over larger domain sizes, of which H-Coloring problems are an example.

Finally, the lower bound for H-Coloring provided in this chapter generalizes earlier results on sparsification of q-Coloring problems [60].

Related work It is well-known that H-Coloring is NP-hard if H is non-bipartite and contains no self-loops [49], and polynomial-time solvable otherwise. The classical complexity of H-Coloring has also been investigated when restricted to planar [76], minor-closed [36], and bounded-degree [43, 84] input graphs G.

Overview We start by showing that when H corresponds to C_α for some odd value for α, the H-Coloring problem has no non-trivial sparsification unless NP $\subseteq$ coNP/poly, in Section 5.1. This will be shown by giving a degree-2 cross-composition from Clique to C_α-Coloring. Recall that C_α is a the cycle on α vertices.

In Section 5.2 we provide some additional preliminaries. In the remainder of this chapter (Section 5.3), we transfer the lower bound thus shown for C_α-Coloring to other choices for the graph H, using linear-parameter transformations. In particular, we will show the lower bound holds in case H contains a triangle, and in case H has odd-girth $\alpha > 3$ and no edge in H is contained in two different C_α-subgraphs.

To obtain a linear-parameter transformation, we cannot easily re-use the techniques that were used in the NP-completeness proofs for H-Coloring when H is nonbipartite. In particular, the constructions used by Hell and Nešetřil introduce gadgets for every edge in the graph, which is not allowed in a linear-parameter transformation. For the same reasons, alternative proofs for the H-Coloring dichotomy cannot be used in our setting. Instead, we show that if H satisfies the relevant preconditions, then there is a subset B of the vertices of H, such that $H[B]$ is homomorphically equivalent to an odd cycle and furthermore, there exists a reduction from C_α-Coloring to H-Coloring that attaches a constant-size gadget to every vertex, forcing it to be colored with a color in B. Hence an H-coloring satisfying the constraints enforced by these gadgets only uses the vertices of a cycle in H to color G, and is therefore a C_α-coloring

5.1 Sparsification lower bound for C_α-Coloring

In this section, we will prove a sparsification lower bound for C_α-Coloring. We show that C_α-Coloring on n vertices has no (generalized) kernel of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$ and odd $\alpha \geq 3$, unless NP $\subseteq$ coNP/poly. We will do this by giving a degree-2 cross-composition from Clique to C_α-Coloring, as in Definition 2.13. An input to the decision problem Clique consists of a graph G and integer k, and asks whether G has a clique on k vertices.

For ease of presentation, we will first prove the lower bound for C_α-List Coloring instead of C_α-Coloring. The input to C_α-List Coloring is a graph G together with a function $\mathcal{L}$ that assigns each vertex v of G a list $\mathcal{L}(v) \subseteq V(C_\alpha)$.

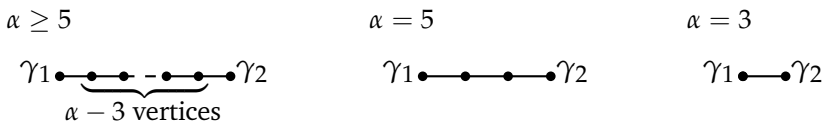


Figure 5.1 The gadget ineq-gadget for Lemma 5.1, depicted for $\alpha \geq 5$ (left), $\alpha = 5$ (middle) and $\alpha = 3$ (right).

The question is whether there is a homomorphism from G to C_α that maps each vertex v to a color on its list $\mathcal{L}(v)$. We will show later that this construction will also imply the sparsification bound for C_α-Coloring.

Before presenting the cross-composition, we need the following additional lemmas.

▶ **Lemma 5.1** *For all odd $\alpha \geq 3$, there exists a C_α-Coloring instance with $\mathcal{O}(\alpha)$ vertices called ineq-gadget that has distinguished vertices γ_1, γ_2, such that any mapping $f\colon \{\gamma_1, \gamma_2\} \to V(C_\alpha)$ can be extended to a C_α-coloring of the ineq-gadget if and only if $f(\gamma_1) \neq f(\gamma_2)$.*

Proof. A gadget with the desired properties was also developed by Maurer *et al.* [79], we will use a simplified version of their gadget.

Define the ineq-gadget as a path on $\alpha - 1$ vertices. Let the first vertex of the path be γ_1 and let the last vertex be γ_2. The gadget is depicted in Figure 5.1. Observe that for $\alpha = 3$, the gadget simply consists of the vertices γ_1 and γ_2, connected by an edge.

Since the gadget is a path of length $\alpha - 2$, the colors used on this path by any C_α-coloring form a walk in C_α of length $\alpha - 2$. Suppose a mapping $f\colon \{\gamma_1, \gamma_2\} \to V(C_\alpha)$ can be extended to a coloring f' of the whole gadget. If now, $f(\gamma_1) = f(\gamma_2)$, the existence of f' means that there is a closed walk of length $\alpha - 2$ in C_α, implying that there is an odd cycle in C_α of length strictly smaller than α, which is a contradiction. Thus, $f(\gamma_1) \neq f(\gamma_2)$.

In the other direction, given a C_α-coloring $f\colon \{\gamma_1, \gamma_2\}$ for which $f(\gamma_1) \neq f(\gamma_2)$ we can extend it to a C_α-coloring of the entire gadget by choosing a length-$(\alpha - 2)$ walk in C_α from $f(\gamma_1)$ to $f(\gamma_2)$. Color the vertices of the path using the vertices from this walk, in the same order. ◀

In our construction below, we will use the phrase *connect u to v with an ineq-gadget* to mean the following: create a new ineq-gadget and identify u with γ_1 and v with γ_2. The ineq-gadget can be used to make sure two vertices of the graph are colored with two distinct colors.

We will now define so-called blocking-gadgets$_\alpha$, that allow us to forbid a certain coloring on a set of vertices in a C_α-List Coloring instance. The gadget will be based on a gadget for classic graph coloring created by Jaffke and Jansen [55]. The following lemma is a rephrased version of Lemma 14 in [55].

▶ **Lemma 5.2** *There exists a polynomial-time algorithm that, given a tuple* $\mathbf{c} = (c_1, \ldots, c_m) \in [q]^m$*, outputs a* q-List Coloring *instance called blocking-gadget*$(\mathbf{c})$ *that is a path of length* $\mathcal{O}(m)$ *and contains distinguished vertices* $(\pi_1, \ldots, \pi_m)$*. A mapping* $f\colon \{\pi_i \mid i \in [m]\} \to [q]$ *can be extended to a* q*-list coloring of blocking-gadget*$(\mathbf{c})$ *if and only if* $f(\pi_i) = c_i$ *for some* $i \in [m]$*.* ◀

Using this gadget for q-List Coloring, we show how to construct a gadget for C_α-List Coloring in the following lemma.

▶ **Lemma 5.3** *There is a polynomial-time algorithm that, given* $\mathbf{c} = (c_1, \ldots, c_m) \in [V(C_\alpha)]^m$*, outputs a* C_α-List Coloring *instance with* $\mathcal{O}(\alpha \cdot m)$ *vertices called blocking-gadget*$_\alpha(\mathbf{c})$ *that contains distinguished vertices* $(\pi_1, \ldots, \pi_m)$*.*

A mapping $f\colon \{\pi_i \mid i \in [m]\} \to [V(C_\alpha)]$ *can be extended to a* C_α*-coloring of blocking-gadget*$_\alpha(\mathbf{c})$ *if and only if* $f(\pi_i) = c_i$ *for some* $i \in [m]$*.*

Proof. Enumerate the vertices in C_α such that $V(C_\alpha) = \{1, \ldots, \alpha\}$ arbitrarily and rename the colors in $\mathbf{c}$ accordingly. To obtain blocking-gadget$_\alpha(\mathbf{c})$ start from blocking-gadget$(\mathbf{c})$, with the same lists $\mathcal{L}(v)$ for each vertex.

Replace every edge in blocking-gadget$(\mathbf{c})$ by an ineq-gadget as follows. If $\{u, v\}$ is an edge in blocking-gadget$(\mathbf{c})$, remove it and add a new ineq-gadget with distinguished vertices γ_1 and γ_2. Identify vertex γ_1 with u and γ_2 with v. This concludes the construction of blocking-gadget$_\alpha(\mathbf{c})$.

Since blocking-gadget$(\mathbf{c})$ is a path, it only has $\mathcal{O}(m)$ edges and vertices. Thereby, blocking-gadget$_\alpha(\mathbf{c})$ has at most $\mathcal{O}(\alpha m)$ vertices.

One can now verify that by the properties of the ineq-gadget given in Lemma 5.1 and the properties of a blocking-gadget given in Lemma 5.2, it follows that a coloring f of $\pi_1, \ldots, \pi_m$ can only be extended to a C_α-coloring of blocking-gadget$_\alpha(\mathbf{c})$ if there exists $i \in [m]$ such that $f(\pi_i) = c_i$. ◀

We will use the above lemma when we want to forbid one particular coloring $c_1, \ldots, c_m \in V(C_\alpha)$ from appearing on a particular sequence of vertices $v_1, \ldots, v_m$ of the graph under construction, without adding further restrictions. This can now be achieved by adding a blocking-gadget$_\alpha((c_1, \ldots, c_m))$. Then for each $i \in [m]$, connect π_i to v_i with an ineq-gadget. This ensures that for any proper coloring f, it holds that $f(v_i) \neq f(\pi_i)$ for all $i \in [m]$. Thereby, if $f(v_i) = c_i$ for all i (which we want to forbid), then $f(\pi_i) \neq c_i$ for all i and thus this coloring cannot be extended to properly color all vertices of blocking-gadget$_\alpha(\mathbf{c})$. If however $f(v_i) \neq c_i$ for some i, vertex π_i can be colored with c_i and this coloring can be extended to color blocking-gadget$_\alpha(\mathbf{c})$, as desired.

Using the gadgets introduced above, we can now prove the sparsification lower bound for C_α-Coloring.

▶ **Theorem 5.4** *Let* $\alpha \geq 3$ *be an odd integer.* C_α-Coloring *parameterized by the number of vertices* n *admits no generalized kernel of size* $\mathcal{O}(n^{2-\varepsilon})$ *for any* $\varepsilon > 0$*, unless* NP $\subseteq$ coNP/poly.

Proof. We will start by giving a degree-2 cross-composition from CLIQUE to C_α-LIST COLORING. We define a polynomial equivalence relation $\mathcal{R}$ (Def. 2.12) on instances of CLIQUE. Let any two instances that ask for a clique that is larger than their respective number of vertices be equivalent; these are always no-instances. Let two instances of CLIQUE be equivalent under $\mathcal{R}$, when the input graphs have same number of vertices and the problems ask for a clique of the same size. It is easy to verify that $\mathcal{R}$ is indeed a polynomial equivalence relation.

By duplicating one of the inputs multiple times if needed, we can assume the number of inputs to the cross-composition is a square. Therefore, assume we are given t instances of CLIQUE, such that $t' := \sqrt{t}$ is integer and such that each instance has n vertices and asks for a clique of size k. Enumerate the given instances as $X_{i,j}$ for $i,j \in [t']$ and let $G_{i,j}$ denote the corresponding graph. Name the vertices in each instance arbitrarily as $x_1, \ldots, x_n$. We now create an instance G' of the C_α-LIST COLORING problem. To do this, we use the colorset $\{L, L', R, R', C\} \subseteq V(C_\alpha)$. For $\alpha \geq 5$ the colors are chosen such that (L, L', C, R', R) forms a (simple, but not necessarily induced) path in C_α. If $\alpha = 3$, simply pick three distinct colors L, R, and C and let $L' := R$ and $R' := L$. Observe that the pairs (L, C), (C, C), and (C, R) have a common neighbor in $\{L', R'\}$, but that (L, R) does not (this pair has C as a common neighbor but that is irrelevant to the construction). Refer to Figure 5.2 for a sketch of G'. We construct a C_α-LIST COLORING instance consisting of a graph G' and list $\mathcal{L}(v) \subseteq V(C_\alpha)$ for each $v \in V(G')$.

1. For each $j \in [t']$, $\ell \in [n]$, and $m \in [k]$ create vertices $p^j_{\ell,m}$ and $q^j_{\ell,m}$. Let $\mathcal{L}(q^j_{\ell,m}) := \{L, C\}$ and $\mathcal{L}(p^j_{\ell,m}) := \{R, C\}$. Let Q_j contain all created vertices $p^j_{\ell,m}$ and $q^j_{\ell,m}$. Let $Q := \bigcup_{j \in [t']} Q_j$.

2. Let $\mathbf{c} := (L, C)$. For each $j \in [t']$, $\ell \in [n]$, and $m \in [k]$, create a new blocking-gadget$_\alpha(\mathbf{c})$ and let π_1, π_2 be its distinguished vertices. Connect π_1 to $q^j_{\ell,m}$ with an ineq-gadget and connect π_2 to $p^j_{\ell,m}$ via an ineq-gadget. This ensures that any coloring that assigns L to $q^j_{\ell,m}$, must assign color R to $p^j_{\ell,m}$.

3. For each $f \in \binom{[k]}{2}$, each $e = (e_1, e_2) \in [n]^2$, and each $i \in [t']$, create vertices $r^i_{e,f}$ and $s^i_{e,f}$. Let $S_i := \{r^i_{(e_1,e_2),f}, s^i_{(e_1,e_2),f} \mid f \in \binom{[k]}{2}, (e_1, e_2) \in [n]^2\}$. Note that S_i contains $\binom{k}{2}$ vertices for each possible edge e that could exist in an n-vertex graph (including self-loops). Let $S := \bigcup_{i \in [t']} S_i$. For every vertex in S, let its list be $\{L', R'\}$.

The goal of the construction is to ensure that the C_α-LIST COLORING instance $(G', \mathcal{L})$ acts as the logical OR of the CLIQUE instances $X_{i,j}$, so that G' has a C_α-coloring respecting the lists if and only if some input graph $G_{i,j}$ has a clique

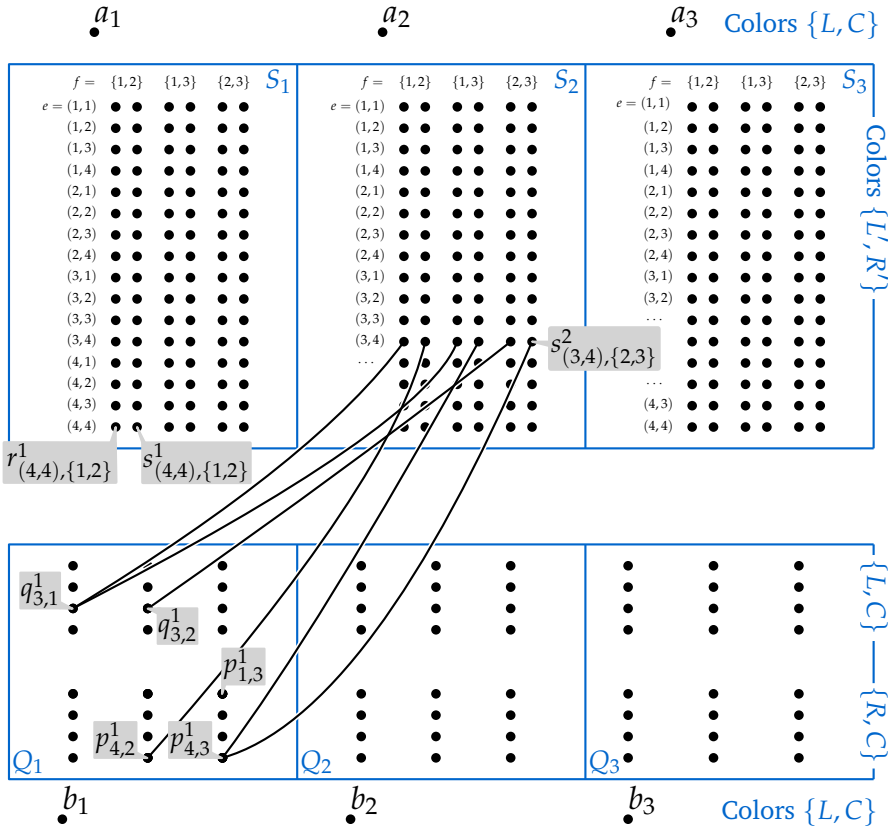


Figure 5.2 A sketch of G' for $t = 9$, $k = 3$, and $n = 4$. The only edges drawn are the edges created in Step 4, for $e = (3, 4)$, $i = 2$, and $j = 1$, in case the edge $\{x_3, x_4\} \notin E(G_{2,1})$.

of size k. The part of G' constructed so far allows colorings of G' to encode the vertex set of a k-clique through its behavior on a set Q_j. To find a proper list coloring of G' entails highlighting vertices from one of the sets Q_j that correspond to a clique in instance $X_{i,j}$ for some $i \in [t']$. The highlighting property will be enforced by ensuring at least one vertex in each set $\{q^j_{\ell,m} \mid \ell \in [n]\}$ receives color L. The index of the vertex that is colored L encodes a vertex in the clique to which the coloring corresponds. The vertices in S_i are used to verify that the selected vertices form a clique in $G_{i,j}$. The next steps add additional vertices and edges, in order to achieve these properties.

4. For each $i, j \in [t']$, consider instance $X_{i,j}$. For all $f \in \binom{[k]}{2}$ and $e = (e_1, e_2) \in [n]^2$, connect $q^j_{e_1,f_1}$ to $r^i_{e,f}$, and connect $p^j_{e_2,f_2}$ to $s^i_{e,f}$, whenever $\{x_{e_1}, x_{e_2}\} \notin$

$E(G_{i,j})$. Here $f_1 < f_2$ are such that $f = \{f_1, f_2\}$.

The above step uses vertices $r^i_{e,f}$ and $s^i_{e,f}$ to ensure that when $\{x_{e_1}, x_{e_2}\}$ is not an edge in some instance, we cannot select both vertices in a clique for this instance.

5. Add vertices a_i with list $\{L, C\}$ for all $i \in [t']$, and let $A := \{a_i \mid i \in [t']\}$.
6. Similarly, add vertices b_j with list $\{L, C\}$ for all $j \in [t']$ and let $B := \{b_j \mid j \in [t']\}$.
7. Let $\mathbf{c} := (L, \ldots, L)$ be the tuple consisting of t' copies of color L. Add a blocking-gadget$_\alpha(\mathbf{c})$ and for all $i \in [t']$ connect the distinguished vertex π_i of blocking-gadget$_\alpha(\mathbf{c})$ to a_i. Add another blocking-gadget$_\alpha(\mathbf{c})$ and for all $j \in [t']$ connect the distinguished vertex π_j of this new gadget to b_j.

The steps above ensure that there exist vertices $a_i \in A$ and $b_j \in B$ that receive color C. This will indicate that $X_{i,j}$ is selected. Now we put further constraints on the coloring of Q_i and S_j when they correspond to a selected instance.

8. Add blocking-gadgets$_\alpha$ and ineq-gadgets to ensure that $r^i_{e,f}$ and $s^i_{e,f}$ receive the same color whenever a_i has color C, as follows. For all $i \in [t']$, $f \in \binom{[k]}{2}$, and $e \in [n]^2$, for all $(c_1, c_2, c_3) \in \{L, L', C, R', R\}^3$ such that $c_1 = C$ and $c_2 \neq c_3$, add a new blocking-gadget$_\alpha(\mathbf{c})$. Connect a_i to π_1 with an ineq-gadget, connect $r^i_{e,f}$ to π_2 with an ineq-gadget and connect $s^i_{e,f}$ to π_3 with an ineq-gadget.
9. Let $\mathbf{c} := (C, \ldots, C)$ be the tuple consisting of $n+1$ copies of color C. For each $j \in [t']$, and $m \in [k]$, add a new blocking-gadget$_\alpha(\mathbf{c})$. Connect π_{n+1} to b_j with an ineq-gadget and for each $\ell \in [n]$ connect $q^j_{\ell,m}$ to π_ℓ with an ineq-gadget.

This concludes the construction of G'. We will start by proving a number of claims about the construction, starting with the "selection property" of the sets A and B.

▷ **Claim 5.5** *Let c be a valid C_α-list coloring of G'. Then there exist $i, j \in [t']$ such that $c(a_i) = c(b_j) = C$.*

Proof. Since c is a proper coloring, it properly colors the blocking-gadgets$_\alpha$ created for sets A (respectively, B) in Step 7 of the construction. By Lemma 5.3, there exists a vertex π_i in the first gadget and a vertex π_j in the second gadget such that $c(\pi_i) = c(\pi_j) = L$. As a result of the ineq-gadgets added in this step, $c(\pi_i) \neq c(a_i)$ and $c(\pi_j) \neq c(b_j)$. Since $\mathcal{L}(a_i) = \mathcal{L}(b_j) = \{L, C\}$, it follows that $c(b_j) = c(a_i) = C$. ◁

We will now show that in any proper C_α-List Coloring, there exists a j such that k vertices in Q_j will be selected, meaning they receive color L.

▷ **Claim 5.6** *Let c be a valid C_α-list coloring of G'. There exists $j \in [t']$ such that for all $m \in [k]$, the set $\{q^j_{\ell,m} \mid \ell \in [n]\}$ contains at least one vertex q with $c(q) = L$.*

Proof. By Claim 5.5, there exists $j \in [t']$ such that $c(b_j) = C$. We show that this choice for j has the desired property. Suppose for a contradiction that for some $m \in [k]$, for all $q \in \{q^j_{\ell,m} \mid \ell \in [n]\}$ it holds that $c(q) = C$. Let $\mathbf{c} := (C, \ldots, C)$ and consider the blocking-gadget$_\alpha(\mathbf{c})$ added in Step 9 for which for all $\ell \in [n]$ vertex π_ℓ was connected to $q^j_{\ell,m}$ with an ineq-gadget and b_j was connected to π_{n+1} with an ineq-gadget. It follows from Lemma 5.1 that for this gadget $c(\pi_i) \neq C$ for all i. Using Lemma 5.3, this contradicts that c is a proper coloring of G'.

Thereby, each set $\{q^j_{\ell,m} \mid \ell \in [n]\}$ contains at least one vertex q with $c(q) \neq C$. Since $\mathcal{L}(q) = \{L, C\}$, it follows that $c(q) = L$. ◁

We continue by proving a claim about the coloring of the set S_i, when i is such that a_i received color C.

▷ **Claim 5.7** *Let c be a valid C_α-list coloring of G'. There exists $i \in [t']$ such that for all $e \in [n]^2$ and $f \in \binom{[k]}{2}$ it holds that $c(r^i_{e,f}) = c(s^i_{e,f})$.*

Proof. Choose $i \in [t']$ such that $c(a_i) = C$, such a value for i exists by Claim 5.5. We show that this choice for i has the desired property.

Suppose for contradiction that there exist $e \in [n]^2$, $f \in \binom{[k]}{2}$ such that $c(r^i_{e,f}) \neq c(s^i_{e,f})$. Let $\mathbf{c} := (C, c(r^i_{e,f}), c(s^i_{e,f}))$. Verify that in Step 8, a blocking-gadget$_\alpha(\mathbf{c})$ was added and its distinguished vertices π_1, π_2, and π_3 were connected with ineq-gadgets to a_i, $r^i_{e,f}$, and $s^i_{e,f}$, respectively. By Lemma 5.1, it follows that $c(\pi_1) \neq C$, $c(\pi_2) \neq c(r^i_{e,f})$ and $c(\pi_3) \neq c(s^i_{e,f})$. But by Lemma 5.3 this contradicts that c properly colors this blocking-gadget$_\alpha$, which concludes the proof. ◁

▷ **Claim 5.8** *Let c be a valid C_α-list coloring of G'. Let $j \in [t']$, $m \in [k]$, and $\ell \in [n]$. Then $c(q^j_{\ell,m}) = L$ implies that $c(p^j_{\ell,m}) = R$.*

Proof. Suppose for contradiction that $c(q^j_{\ell,m}) = L$ but $c(p^j_{\ell,m}) \neq R$. Since $\mathcal{L}(p^j_{\ell,m}) = \{R, C\}$ this implies that $c(p^j_{\ell,m}) = C$. Let $\mathbf{c} := (L, C)$. In Step 2, a blocking-gadget$_\alpha(\mathbf{c})$ was added and its distinguished vertex π_1 was connected to $q^j_{\ell,m}$ with an ineq-gadget, and similarly π_2 was connected to $p^j_{\ell,m}$ with an ineq-gadget. By Lemma 5.1 it follows that $c(\pi_1) \neq c(q^j_{\ell,m}) = L$ and $c(\pi_2) \neq$

$c(p^j_{\ell,m}) = C$. But by Lemma 5.3 it follows that this coloring cannot be extended to color the blocking-gadget$_\alpha$, which contradicts that c is a proper coloring of G'. ◁

When G' is C_α-list-colorable, we need to prove that one of the input instances has a clique of size k. The following claim will be used to show that we never select two vertices into this clique if there is no edge between them.

▷ **Claim 5.9** *Let c be a valid C_α-list coloring of G', such that $c(r^i_{e,f}) = c(s^i_{e,f})$ for some $e = (e_1, e_2) \in [n]^2$, $f = \{f_1, f_2\} \in \binom{[k]}{2}$ with $f_1 < f_2$, and $i \in [t']$. If $\{x_{e_1}, x_{e_2}\} \notin E(G_{i,j})$ for some $j \in [t']$, then $c(q^j_{e_1,f_1}) \neq L$ or $c(q^j_{e_2,f_2}) \neq L$.*

Proof. Let c be a valid coloring. Suppose for contradiction that there exist $i, j \in [t']$, $e = (e_1, e_2) \in [n]^2$, and $f = \{f_1, f_2\} \in \binom{[k]}{2}$ such that $c(r^i_{e,f}) = c(s^i_{e,f})$ and $\{x_{e_1}, x_{e_2}\} \notin E(G_{i,j})$, and suppose $c(q^j_{e_1,f_1}) = c(q^j_{e_2,f_2}) = L$. It follows from Claim 5.8 that $c(p^j_{e_2,f_2}) = R$.

Since $\{x_{e_1}, x_{e_2}\} \notin E(G_{i,j})$, it holds that $\{r^i_{e,f}, q^j_{e_1,f_1}\}, \{s^i_{e,f}, p^j_{e_2,f_2}\} \in E(G')$ by Step 4 of the construction. Since $c(r^i_{e,f}) = c(s^i_{e,f})$ and $\mathcal{L}(r^i_{e,f}) = \mathcal{L}(s^i_{e,f}) = \{L', R'\}$ there should exist a single color in $\{L', R'\}$ to properly color both vertices. This is however impossible, since $r^i_{e,f}$ is connected to a vertex with color L, while $s^i_{e,f}$ is already connected to a vertex of color R. Coloring $c(r^i_{e,f}) = c(s^i_{e,f}) = R'$ would not properly color the edge $\{r^i_{e,f}, q^j_{e_1,f_1}\}$ since R' and L are not neighbors in C_α, while setting $c(r^i_{e,f}) = c(s^i_{e,f}) = L'$ would not properly color edge $\{s^i_{e,f}, p^j_{e_2,f_2}\}$ since L' and R are not neighbors in C_α (note that in the case of $\alpha = 3$, indeed $L' = R$ and $R' = L$). This contradicts that c is a proper C_α-coloring of G'. ◁

The structural properties derived so far, lead to the proof of the following claim. Combined with Claim 5.11, it will show that the composed instance $(G', \mathcal{L})$ acts as the logical OR of the CLIQUE inputs.

▷ **Claim 5.10** *If some input graph G_{i^*,j^*} has a clique of size k, then G' is C_α-list colorable.*

Proof. Take such i^* and j^*, and let $x_1, \ldots, x_n$ be the vertices of G_{i^*,j^*}. Pick $I = (i_1, \ldots, i_k) \subseteq [n]$ such that $\{x_i \mid i \in I\}$ is a clique of size k in G_{i^*,j^*}, thereby all indices in I are distinct. We show how to define a proper C_α-list coloring c of G'.

Let $c(a_{i^*}) := c(b_{j^*}) := C$. For all $i \in [t']$ with $i \neq i^*$, let $c(a_i) := L$ and for all $j \in [t']$ with $j \neq j^*$ let $c(b_j) := L$.

For all $m \in [k]$, let $c(q^{j^*}_{i_m,m}) := L$ and let $c(p^{j^*}_{i_m,m}) := R$. Color all remaining vertices in Q with color C.

For all $e = (e_1, e_2) \in [n]^2$ and $f = \{f_1, f_2\} \in \binom{[k]}{2}$ with $f_1 < f_2$, let $c(r^{i^*}_{e,f}) := c(s^{i^*}_{e,f}) := L'$ when vertex $r^{i^*}_{e,f}$ is connected to vertex $q^{j^*}_{e_1,f_1}$ and $c(q^{j^*}_{e_1,f_1}) = L$ (note that the color of this vertex is already defined). Otherwise, let $c(r^{i^*}_{e,f}) := c(s^{i^*}_{e,f}) := R'$. For all $i \in [t']$ such that $i \neq i^*$, let $c(r^{i}_{e,f}) := L'$ and define $c(s^{i}_{e,f}) := R'$.

Before showing how to extend c to properly color the blocking-gadgets$_\alpha$ and ineq-gadgets, we first show that the coloring defined so far is proper. It is easy to verify that the defined coloring respects the lists of the vertices. The only edges to consider are those between a vertex in S and a vertex in Q. There are two types of edges in the graph.

- Edges of the form $\{r^{i}_{e,f}, q^{j}_{\ell,m}\}$ for $e = (e_1, e_2)$ and $f = \{f_1, f_2\}$ with $f_1 < f_2$. For this edge to exist, it must hold that $\ell = e_1$ and $m = f_1$. If $j \neq j^*$ this edge is properly colored since $c(q^{j}_{\ell,m}) = C$ and $c(r^{i}_{e,f}) \in \{L', R'\}$. If $i \neq i^*$, this edge is properly colored since $c(q^{j}_{\ell,m}) \in \{L, C\}$ and $c(r^{i}_{e,f}) = L'$. So suppose $i = i^*$ and $j = j^*$. By the definition of the coloring, if $c(q^{j}_{\ell,m}) = L$ and this edge exists, it follows that $c(r^{i}_{e,f})$ was defined as L' and the edge is properly colored. Otherwise, $c(q^{j}_{\ell,m}) = C$ (or the edge does not exist), and since $c(r^{i}_{e,f}) \in \{R', L'\}$ the edge is again properly colored.

- Edges of the form $\{s^{i}_{e,f}, p^{j}_{\ell,m}\}$ for $e = (e_1, e_2)$ and $f = \{f_1, f_2\}$ with $f_1 < f_2$. For this edge to exist, it must hold that $\ell = e_2$ and $m = f_2$. Suppose $j \neq j^*$, then since $c(s^{i}_{e,f}) \in \{L', R'\}$ and $c(p^{j}_{e_2,f_2}) = C$ this edge is properly colored. Similarly, if $i \neq i^*$ then $c(s^{i}_{e,f}) = R'$. Since $c(p^{j}_{e_2,f_2}) \in \{R, C\}$ the edge is properly colored.

  So suppose $i = i^*$ and $j = j^*$. If $c(p^{j^*}_{e_2,f_2}) = C$ or $c(s^{i^*}_{e,f}) = R'$ it is easy to verify that this coloring is always proper. So suppose $c(p^{j^*}_{e_2,f_2}) = R$ and $c(s^{i^*}_{e,f}) = L'$. The choice of $c(s^{i^*}_{e,f}) = L'$ implies that $r^{i^*}_{e,f}$ is connected to $q^{j^*}_{e_1,f_1}$ and that $c(q^{j^*}_{e_1,f_1}) = L$. The fact that $c(p^{j^*}_{e_2,f_2}) = R$ implies that we defined $c(q^{j^*}_{e_2,f_2}) = L$. Note that by definition, $f_1 \neq f_2$. But $c(q^{j^*}_{e_2,f_2}) = c(q^{j^*}_{e_1,f_1}) = L$ implies that $e_1, e_2 \in I$ and $e_1 \neq e_2$ since $f_1 \neq f_2$. However, the existence of the edges $\{r^{i^*}_{e,f}, q^{j^*}_{e_1,f_1}\}$ and $\{s^{i^*}_{e,f}, p^{j^*}_{e_2,f_2}\}$ in G' implies that $\{x_{e_1}, x_{e_2}\} \notin E(G_{i^*,j^*})$, which contradicts that $\{x_i \mid i \in I\}$ is a clique in G_{i^*,j^*}. Thus we conclude that the

edge $\{s^i_{e,f}, p^j_{\ell,m}\}$ is properly colored.

It remains to extend the given coloring to properly color all gadgets. We will use that whenever vertices $\pi_1, \ldots, \pi_m$ of some blocking-gadget$_\alpha(\mathbf{c})$ are connected only to vertices $v_1, \ldots, v_m$ with ineq-gadgets, it follows that the coloring of $v_1, \ldots, v_m$ can be extended to color blocking-gadget$_\alpha(\mathbf{c})$ and the ineq-gadgets connecting them whenever there exists $i \in [m]$ such that $c(v_i) \neq c_i$. This follows immediately from Lemmas 5.1 and 5.3. We will show how to color the gadgets created in each step of the construction.

Consider a blocking-gadget$_\alpha(\mathbf{c})$ added in Step 2, note that in this case $\mathbf{c} = (c_1, c_2) = (L, C)$. For every $j \in [t']$, $\ell \in [n]$, and $m \in [k]$ we defined that either $c(q^j_{\ell,m}) = C \neq c_1$ or $c(q^j_{\ell,m}) = L$ and $c(p^j_{\ell,m}) = R \neq c_2$. In both cases, it is easy to see that this coloring can be extended to the blocking-gadget$_\alpha$ and all added ineq-gadgets.

It is easy to see that c can be extended to color the two blocking-gadgets$_\alpha$ and the ineq-gadgets added in Step 7, since we defined $c(a_{i^*}) = C \neq L$ and $c(b_{j^*}) = C \neq L$.

Consider a blocking-gadget$_\alpha((c_1, c_2, c_3))$ added in Step 8, let it be added for some $i \in [t']$, $f \in \binom{[k]}{2}$ and $e \in [n]^2$. Note that $c_1 = C$ and $c_2 \neq c_3$. If $i \neq i^*$, we defined $a_i = L \neq c_1$ and we can extend the coloring to this gadget and the connecting ineq-gadgets. If $i = i^*$, we defined $c(r^{i^*}_{e,f}) = c(s^{i^*}_{e,f})$ and this coloring can again be extended to all gadgets.

Consider a blocking-gadget$_\alpha(\mathbf{c})$ added in Step 9. Note that $\mathbf{c} = (C, \ldots, C)$. If $j \neq j^*$ we defined $c(b_j) = L \neq C$ and the coloring can be extended to the gadgets. If $j = j^*$, let this gadget be added for some $m \in [k]$. Then we defined $c(q^{j^*}_{i_m,m}) = L \neq C$ and again c can be extended to the gadgets.

Thereby, we have shown that c can be extended to a valid C_α-list coloring of G', and thus G' is C_α-list colorable. ◁

▷ **Claim 5.11** *If G' has a proper C_α-list coloring, then there exist $i^*, j^* \in [t']$ such that G_{i^*,j^*} has a clique of size k.*

Proof. Let c be a proper C_α-list coloring of G'. Pick j^* such that for all $m \in [k]$, the set $\{q^{j^*}_{\ell,m} \mid \ell \in [n]\}$ contains at least one vertex of color L, using Claim 5.6. Choose i^* such that for all $e \in [n]^2$, $f \in \binom{[k]}{2}$ it holds that $c(r^{i^*}_{e,f}) = c(s^{i^*}_{e,f})$, by Claim 5.7.

For each $m \in [k]$, choose exactly one index i_m such that $c(q^{j^*}_{i_m,m}) = L$. Note that such an i_m exists by the choice of j^*. We define a clique Z in G_{i^*,j^*} as follows. Remember $V(G_{i^*,j^*}) = x_1, \ldots, x_n$.

$$Z := \{x_{i_m} \mid m \in [k]\}.$$

We start by verifying that $|Z| \geq k$. It is sufficient to show that all i_m are distinct, which we prove by showing that there are no $\ell \in [n]$ and $m, m' \in [k]$ such that $m \neq m'$ and $c(q^{j^*}_{\ell,m}) = c(q^{j^*}_{\ell,m'}) = L$. This follows immediately from the choice of i^* and Claim 5.9, since $\{x_\ell, x_\ell\} \notin E(G_{i^*,j^*})$ because the input graphs have no self-loops. It follows that $|Z| = k$. It remains to verify that Z is a clique in G_{i^*,j^*}.

Suppose for contradiction that there exist distinct vertices $x_\ell, x_{\ell'} \in Z$ such that $\{x_\ell, x_{\ell'}\} \notin E(G_{i^*,j^*})$. Since $x_\ell, x_{\ell'} \in Z$, it follows that there exist $m, m' \in [k]$ with $m \neq m'$ such that $c(q^{j^*}_{\ell,m}) = c(q^{j^*}_{\ell',m'}) = L$.

However, it follows from the choice of i^* and Claim 5.9 that $c(q^{j^*}_{\ell,m}) \neq L$ or $c(q^{j^*}_{\ell',m'}) \neq L$. This is a contradiction with the definition of Z. It follows that Z is a clique of size k in G_{i^*,j^*} and thus X_{i^*,j^*} is a yes-instance. ◁

It is easy to see that the construction of G' can be done in polynomial time. It follows from Claims 5.10 and 5.11 that G' is C_α-list colorable if and only if there exist $i, j \in [t']$ such that $X_{i,j}$ is a yes-instance. It remains to bound the size of G'.

▷ **Claim 5.12** *The number of vertices of G' is bounded by $\mathcal{O}(\sqrt{t} \cdot n^2k^2\alpha)$.*

Proof. We bound the number of vertices of G'. The step of the construction in which the vertices were added to G' is indicated.

$$|V(G')| \leq \underbrace{nk\sqrt{t} \cdot 2}_{\text{Step 1}} + \underbrace{nk\sqrt{t} \cdot \mathcal{O}(\alpha)}_{\text{Step 2}} + \underbrace{n^2k^2\sqrt{t} \cdot 2}_{\text{Step 3}} + \underbrace{2 \cdot \sqrt{t}}_{\text{Step 5,6}} + \underbrace{2 \cdot \mathcal{O}(\alpha\sqrt{t})}_{\text{Step 7}}$$
$$+ \underbrace{n^2k^2\sqrt{t} \cdot \mathcal{O}(\alpha)}_{\text{Step 8}} + \underbrace{k\sqrt{t} \cdot \mathcal{O}(n\alpha)}_{\text{Step 9}} = \mathcal{O}(\sqrt{t} \cdot n^2k^2\alpha).$$ ◁

It follows that we have given a degree-2 cross-composition from Clique to C_α-List Coloring and the sparsification lower bound for C_α-List Coloring follows from Theorem 2.14. To show the bound for C_α-Coloring, we modify G' to a C_α-Coloring instance, such that we obtain a degree-2 cross-composition from Clique to C_α-Coloring.

Start from C_α-List Coloring instance G'. We show how to construct an equivalent C_α-Coloring instance G''. Add a cycle on α vertices to G'. We now do a case distinction on the value of α.

If $\alpha > 3$, choose distinguished vertices $\{L, L', C, R', R\}$ from the newly added cycle such that (L, L', C, R', R) is a (not necessarily induced) path in this cycle. For any $v \in V(G')$, for any $c \notin \mathcal{L}(v)$, connect c (the vertex from the newly added cycle) to v with an ineq-gadget.

If $\alpha = 3$, let the three vertices of the newly added cycle be $\{L, C, R\}$. For any $v \in V(G')$, if color $C \notin \mathcal{L}(v)$ connect v to C. If both L' and R are not contained in $\mathcal{L}(v)$, connect v to vertex R. Finally, if R' and L are not contained

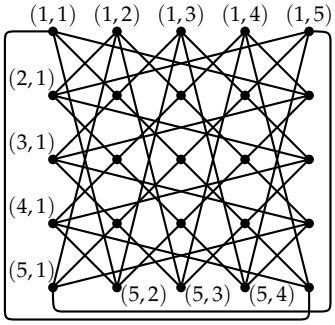


Figure 5.3 This figure depicts C_5^2 (see Definition 5.13), where C_5 is the five-cycle with vertex set $\{1,2,3,4,5\}$.

in $\mathcal{L}(v)$, connect v to L. As such, for C_3-coloring, the we have that colors L' and R coincide, and that colors R' and L coincide.

It is easy to see that G'' is C_α-colorable if and only if G' is C_α-list colorable. Furthermore, $|V(G'')| \leq \mathcal{O}(\alpha) \cdot |V(G')| + \alpha = \mathcal{O}(\sqrt{t} \cdot n^2 k^2 \alpha^2)$.

We have given a degree-2 cross-composition from Clique to C_α-Coloring. It now follows from Theorem 2.14 that for odd $\alpha \geq 3$, C_α-Coloring parameterized by the number of vertices n does not have a generalized kernel of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless NP $\subseteq$ coNP/poly. ◀

5

5.2 Additional preliminaries

In the remainder of this chapter, we will heavily rely on the information given in Section 2.8. In this section, we provide some additional preliminaries. We will regularly use the graph C_α, which is defined as the cycle on α vertices. When doing so, we will label the vertices of C_α with the numbers 1 up to α, such that $V(C_\alpha) := \{1,\ldots,\alpha\}$ and $E(C_\alpha) := \{\{\alpha,1\}\} \cup \{\{i,i+1\} \mid i \in [\alpha-1]\}$.

We continue by two additional definitions for graphs.

Definition 5.13 (G^k) If G is a graph and $k \in \mathbb{N}_+$, then G^k is the graph on vertex set $V(G)^k$ where vertices $(x_1,\ldots,x_k)$ and $(y_1,\ldots,y_k)$ are adjacent if $\{x_i,y_i\} \in E(G)$ for all $i \in [k]$. Refer to Figure 5.3 for an example.

For $k \geq 1$, graph G^k is G-colorable by the mapping $f\colon (x_1,\ldots,x_k) \mapsto x_1$. Hence $G^k \leftrightarrow G$ (recall Definition 2.45).

For an equivalence relation R on a set D, we denote by $[u]_R$ the equivalence class of $u \in D$. We omit the subscript if it is clear from the context.

Definition 5.14 ($G/_R$) Let G be a graph (potentially with self-loops) and let R be an equivalence relation on the vertices of G. Then $G/_R$ is the graph on vertex set $\{[v]_R \mid v \in V(G)\}$ and edge set $\{\{[u],[v]\} \mid \exists u' \in [u], \exists v' \in [v]\colon \{u',v'\} \in$

$E(G)\}$.

Observe that even when G is assumed to be simple, $G/_R$ may contain self-loops.

We give a special name for pp-definable relations (Def. 2.40) of arity 1, which will be of particular interest.

Definition 5.15 Let Γ be a constraint language over domain D. A *definable subdomain* of Γ is a subset S of D that is pp-definable over Γ. It is *strict* if $S \subsetneq D$.

The following corollary is immediate from Theorem 2.42.

▶ **Corollary 5.16** *Let Γ be a constraint language over domain D, and let B be a non-empty subset of D. If B is not a definable subdomain of Γ, then there is a polymorphism of Γ that is not a polymorphism of B; that is, there exists a polymorphism $f\colon D^n \to D$ of Γ such that $f(b_1,\dots,b_n) \notin B$ for some $b_1,\dots,b_n \in B$.* ◀

When presenting pp-definitions of relations, for notational convenience we often take the equivalent viewpoint via positive-primitive formulas involving the relations of Γ and the equality relation (cf. [21, Definition 1]). Hence $R \subseteq D^k$ is pp-definable over Γ if and only if there is a Boolean formula $\psi(x_1,\dots,x_k)$ with free variables $x_1,\dots,x_k$, built using existential quantifiers, conjunctions, the binary equality relation, and applications of relations of Γ, such that $(x_1,\dots,x_k) \in R$ if and only if $\psi(x_1,\dots,x_k)$ holds.

The following two propositions are known and relatively straightforward to verify; we provide brief explanations for the sake of completeness. Recall that G^* is defined as the constraint language that contains the relation $F = \{(u,v) \mid \{u,v\} \in E(G)\}$ and, for each $v \in V(G)$, the arity-1 relation $\{(v)\}$ (see also Definition 2.47).

▶ **Proposition 5.17** *Let G be a graph, $B \subseteq V(G)$ a definable subdomain of G^*, and $G' := G[B]$. If $B' \subseteq B$ is a definable subdomain of $(G')^*$, then B' is a definable subdomain of G^*.* ◀

Proposition 5.17 can be argued in the following way: consider a pp-formula $\psi'_{B'}(x)$ defining B' over $(G')^*$. Let V be the set of all variables occurring in $\psi'_{B'}(x)$ (including x). We can define B' over G^* by the formula

$$\psi_{B'}(x) := \psi'_{B'}(x) \wedge \bigwedge_{v \in V} \psi_B(v),$$

where $\psi_B(v)$ is the formula defining B over G^*. It is easy to verify that $\psi_{B'}(x)$ is a pp-formula defining B' over G^*.

▶ **Proposition 5.18** *Let G be a graph, let R be an equivalence relation on $V(G)$ that is pp-definable in G^*, and let $G' := G/_R$. If $B' \subseteq V(G')$ is a definable subdomain of $(G')^*$, then the pre-image $B'' := \{v \in V(G) \mid [v] \in B'\}$ is a definable subdomain of G^*.* ◀

The key ingredient for the proof of Proposition 5.18 is that for $u, v \in V(G)$ one can pp-define the condition $\{[u],[v]\} \in E(G')$ over G^* via $\exists u', \exists v' \colon uRu' \wedge vRv' \wedge E_G(u', v')$. Similarly, for $u, v \in V(G)$ one can pp-define $[u] = [v]$ by $\exists u', \exists v' : u'Ru \wedge v'Rv \wedge u' = v'$. As such, given a pp-definition of B' over G', one may obtain a pp-definition of B'' over G by applying the above substitutions to the formula.

5.3 Sparsification lower bounds for H-Coloring

In this section we will prove that several classes of H-Coloring problems do not have a non-trivial sparsification unless NP $\subseteq$ coNP/poly.

We will start this section by showing that when H is a core (recall Definition 2.48) and has a definable subdomain B such that $H[B]$ is homomorphically equivalent to C_α, we can transfer the lower bounds for C_α-Coloring that we have proven in the previous section, to H-Coloring.

Note that for a graph H that is a core, the only unary operations on $V(H)$ that preserve the edge relation of H are bijections. Otherwise, such an operation would provide a homomorphism from H to a proper induced subgraph of H, contradicting that H is a core. Hence Theorem 2.44 implies the following.

▶ **Lemma 5.19** *Let H be a graph that is a core. Then there is a linear-parameter transformation from* CSP(H^*) *parameterized by the number of variables, to* CSP(H) *(corresponding to H-Coloring) parameterized by the number of variables.* ◀

The transformation of Lemma 5.19 has a simple graph-theoretic interpretation: an instance of CSP(H^*), which asks whether a given graph G can be H-colored while respecting fixed colors pre-assigned to a subset of the vertices of G, is equivalent to the instance of H-Coloring that is obtained from the disjoint union G' of G and H by identifying all vertices precolored with $x \in V(H)$ with the copy of x in G'. Hence the number of variables (vertices) increases by at most an additive term $|V(H)|$.

When Γ is a constraint language over domain D and $B \subseteq D$, we use $\Gamma|_B$ to denote the constraint language $\{R|_B \mid R \in \Gamma\}$, where, for a relation R of arity k, we define $R|_B$ as $R \cap B^k$.

▶ **Theorem 5.20** *Let Γ be a constraint language over domain D, and let B be a definable subdomain of Γ. Then there is a linear-parameter transformation from* CSP$(\Gamma|_B)$ *to* CSP(Γ).

Proof. Fix a set of constraints $\mathcal{I}$ over Γ, using variable set $V = \{v, x_1, \ldots, x_m\}$, such that $\exists x_1, \ldots, x_m : \mathcal{I}$ witnesses the pp-definability (Def. 2.40) of B over Γ. In other words, $\mathcal{I}$ is chosen such that for all $d \in D$, it holds that $d \in B$ if and only if the mapping sending v to d can be extended to a satisfying assignment of $\mathcal{I}$.

Given an instance $(\mathcal{C}, \{v_1, \ldots, v_n\})$ of $\mathrm{CSP}(\Gamma|_B)$, the transformation produces the instance described as follows. For each $i \in [n]$, let $\mathcal{I}(v_i, x_i^1, \ldots, x_i^m)$ denote the set of constraints in $\mathcal{I}$ but with the variables renamed accordingly. Then, the produced instance has variable set $\bigcup_{i\in[n]}\{v_i, x_i^1, \ldots, x_i^m\}$, and its constraints are the constraints in $\mathcal{C}$ along with all of the constraints in the sets $\mathcal{I}(v_i, x_i^1, \ldots, x_i^m)$, over $i \in [n]$. Observe that the new instance's variable set has size $n(m+1)$. The transformation is correct, as a satisfying assignment for the original instance of $\mathrm{CSP}(\Gamma|_B)$ can be extended to a satisfying assignment for the produced instance due to choice of the instance $(\mathcal{I}(v, x^1, \ldots, x^m), V)$ of $\mathrm{CSP}(\Gamma)$; in the other direction, a satisfying assignment for the produced instance must map each variable v_i to a value in B, due to the same choice, and hence restricting such an assignment to $\{v_1, \ldots, v_n\}$ yields a satisfying assignment for the original instance of $\mathrm{CSP}(\Gamma|_B)$. ◀

Using the results obtained above, we can now show how the lower bound for C_α-Coloring transfers to H-Coloring, under the right conditions on H.

▶ **Lemma 5.21** *Let H be a graph that is a core such that H^* has a definable subdomain $B \subseteq V(H)$ satisfying $H[B] \leftrightarrow C_\alpha$ for some odd $\alpha \geq 3$. Then H-Coloring parameterized by the number of vertices n does not admit a generalized kernel of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.*

Proof. By the lower bound for C_α-Coloring of Theorem 5.4, together with the known fact that linear-parameter transformations transfer generalized kernelization lower bounds (Theorem 2.8), it suffices to give a linear-parameter transformation from C_α-Coloring to H-Coloring, both parameterized by the number of vertices. We build it by applying several sub-transformations in sequence, using the relation to CSPs.

Recall (Definition 2.39) that H^* is the constraint language over domain $V(H)$ consisting of the edge relation of H and unary singleton relations forcing a value to a constant. Hence $(H^*)|_B$ (cf. Theorem 5.20) is the constraint language over domain B consisting of the edge relation of $H[B]$ and constants for $b \in B$, so it is equivalent to $(H[B])^*$. Since $H[B] \leftrightarrow C_\alpha$, there is a trivial linear-parameter transformation from C_α-Coloring to $\mathrm{CSP}((H[B])^*)$. Then Theorem 5.20 gives a linear-parameter transformation from $\mathrm{CSP}((H[B])^*)$ to $\mathrm{CSP}(H^*)$. Since H is a core, by Lemma 5.19 there is a linear-parameter transformation from $\mathrm{CSP}(H^*)$ parameterized by the number of variables, to $\mathrm{CSP}(H)$ parameterized by the number of variables (i.e., H-Coloring). Since linear-parameter transformations compose, this yields a linear-parameter transformation from C_α-Coloring parameterized by the number of variables to H-Coloring, which concludes the proof. ◀

Using Lemma 5.21, the proof of the sparsification lower bound we give for H-Coloring (Theorem 5.44) essentially reduces to showing that H has a definable subdomain that induces a subgraph homomorphically equivalent

to an odd cycle. In Section 5.3.1, we show how this can be done for graphs in which no edge is contained in two distinct minimum-length odd cycles. In Section 5.3.2, we obtain such definable subdomains for all graphs that contain a triangle. This leads to a proof of the main result of this chapter (Theorem 5.44) in Section 5.3.3.

5.3.1 Defining a subdomain homomorphically equivalent to an odd cycle

We say a nonbipartite graph G with odd girth α has the *no-overlap property* if there is no edge in G that is contained in two distinct C_α subgraphs of G. Here we consider two subgraphs distinct if their edge-sets are distinct. The main result of this subsection will be that if a graph G satisfies the no-overlap property, then G^* has a definable subdomain B such that $G[B]$ is homomorphically equivalent to C_α. This will allow us to prove the sparsification lower bound for H-Coloring when H satisfies the no-overlap property, at the end of this chapter.

The result is obtained by a careful analysis of the structure of graph powers of odd cycles, resulting in a lemma about projectivity. It generalizes Bulatov's [21] results for homomorphisms of powers of triangles into graphs that have no edge in two triangles, to statements about homomorphisms of powers of C_α into graphs of odd girth α with the no-overlap property.

We use C_α^k as shorthand for $(C_\alpha)^k$ (recall Definition 5.13). We will start by proving that for all $k \geq 1$ and $\alpha \geq 3$, adding any edge to C_α^k will reduce the odd girth of C_α^k, or introduce an edge that lies on two distinct C_α-subgraphs. To prove this, we start by showing a number of relevant properties of the graphs C_α and C_α^k.

Given a walk $X = (x_0, \ldots, x_k)$, the walk Y is a *subwalk* of X there exist $i_0 < i_1 < \ldots < i_\ell \in \{0, \ldots, k\}$ such that $Y = (x_{i_0}, x_{i_1}, \ldots, x_{i_\ell})$. First of all, we observe for any graph G, every odd closed walk in G, contains an odd closed subwalk that is an odd cycle. Thus, to show that G has an cycle of odd length at most m, it is sufficient to show that there is closed walk of odd length at most m.

Observation 5.22 *Let G be a simple graph. If G contains a closed walk X of length α for some odd $\alpha \geq 3$, then there exists a closed subwalk Y of X, such that Y is an odd cycle in G of length at most α.*

For a cycle of odd length, there are always two edge-disjoint paths between any two points on the cycle. It is easy to observe that one of these paths must have an even length. From this, we have the following observation.

Observation 5.23 *Let $\alpha \geq 3$ be odd and let $x, y \in V(C_\alpha)$, not necessarily distinct. There exists a walk from x to y in C_α of length exactly $\alpha - 1$.*

The following lemma is the key ingredient in the proof of Lemma 5.25.

▶ **Lemma 5.24** *Let $\alpha \geq 3$ be odd and $k \geq 1$. Let $\mathbf{x}, \mathbf{y} \in V(C_\alpha^k)$, not necessarily distinct, with $\{\mathbf{x}, \mathbf{y}\} \notin E(C_\alpha^k)$. There are two distinct walks of length $\alpha - 1$ from $\mathbf{x}$*

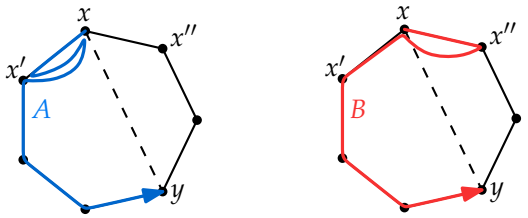


Figure 5.4 Example of two distinct walks of length $\alpha - 1$ from x to y with $k = 1$ and $\alpha = 7$.

to $\mathbf{y}$ *in* C_α^k.

Proof. We will prove the result by induction on k. For $k = 1$, it is easy to verify that there is a simple path P from $\mathbf{x}$ to $\mathbf{y}$ of even length m, with $m < \alpha - 1$. Observe that $m = 0$ is allowed. We use this path P to define the two desired walks A and B of length $\alpha - 1$. Let $x', x'' \in V(C_\alpha)$ be the two neighbors of x in C_α, implying $x' \neq x''$. Define

$$A = (\underbrace{x, x', x, x', \ldots, x, x'}_{\alpha-1-m \text{ vertices}}, P)$$

and define

$$B = (\underbrace{x, x'', x, x'', \ldots, x, x''}_{\alpha-1-m \text{ vertices}}, P).$$

See Figure 5.4 for an example of the paths defined above.

It can be verified that A and B are two walks from x to y of length $\alpha - 1$. They are distinct since P has length at most $\alpha - 3$, and thus the second vertex visited by A is x' and the second vertex visited by B is x'', and $x' \neq x''$. This concludes the base case.

Let $k > 1$ and suppose the statement holds for all smaller values of k. For $i \in [k]$, let $\mathbf{x^{(i)}} := (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k)$ be the vector $\mathbf{x}$ with coordinate i removed. Define $\mathbf{y^{(i)}}$ accordingly. Choose $i \in [k]$ such that $\{\mathbf{x^{(i)}}, \mathbf{y^{(i)}}\} \notin E(C_\alpha^{k-1})$, which exists since $\{\mathbf{x}, \mathbf{y}\} \notin E(C_\alpha^k)$. For ease of notation, we from now on assume $i = k$. By the induction hypothesis, there are two distinct walks from $\mathbf{x^{(k)}}$ to $\mathbf{y^{(k)}}$ in C_α^{k-1} of length $\alpha - 1$, let these be A' and B'. Let $A' = (\mathbf{a_1}', \ldots, \mathbf{a'_\alpha})$ and let $B' = (\mathbf{b_1}', \ldots, \mathbf{b'_\alpha})$, where $\mathbf{x^{(k)}} = \mathbf{a_1}' = \mathbf{b_1}'$ and $\mathbf{y^{(k)}} = \mathbf{a'_\alpha} = \mathbf{b'_\alpha}$. Furthermore, let $S = (x_k = s_1, s_2, \ldots, s_{\alpha-1}, s_\alpha = y_k)$ be a walk from x_k to y_k in C_α of length exactly $\alpha - 1$. Such a walk exists by Observation 5.23.

We define walks $A := (\mathbf{a_1}, \ldots, \mathbf{a_\alpha})$ and $B := (\mathbf{b_1}, \ldots, \mathbf{b_\alpha})$ in C_α^k as follows; see Figure 5.5 for a sketch. Let $\mathbf{a_i} := (a'_{i,1}, \ldots, a'_{i,k-1}, s_i)$, and define $\mathbf{b_i} := (b_{i,1}, \ldots, b_{i,k-1}, s_i)$ for $i \in [\alpha]$. Since A' and B' are distinct walks, it is easy to

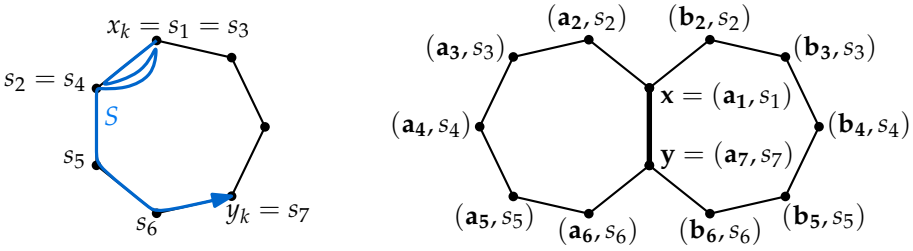


Figure 5.5 (left) Walk of even length from x_k to y_k in C_α. (right) Two distinct closed walks of length $\alpha = 7$ containing the edge $\{\mathbf{x}, \mathbf{y}\}$.

verify that A and B are distinct walks. Since A' and B' are walks in C_α^{k-1} from $\mathbf{x}^{(k)}$ to $\mathbf{y}^{(k)}$ and S is a walk in C_α from x_k to y_k, it follows that A and B are indeed walks from $\mathbf{x}$ to $\mathbf{y}$ in C_α^k of length $\alpha - 1$. ◀

We can now show that adding an edge to C_α^k will decrease the odd girth of C_α^k, or introduce an edge that lies in two distinct C_α subgraphs (such that the no-overlap property no longer holds).

▶ **Lemma 5.25** *Let $\alpha \geq 3$ be odd and $k \geq 1$. Let $\mathbf{x}, \mathbf{y} \in V(C_\alpha^k)$ be two distinct vertices, such that $\{\mathbf{x}, \mathbf{y}\} \notin E(C_\alpha^k)$. Let G be the graph obtained by adding edge $\{\mathbf{x}, \mathbf{y}\}$ to C_α^k. Then*

- *The odd girth of G is smaller than α, or*
- *G contains an edge that is contained in at least two cycles of length α.*

Proof. It follows directly from Lemma 5.24 that the edge $\{\mathbf{x}, \mathbf{y}\}$ lies on two distinct closed walks A and B of length α in G. If one of A and B is *not* a cycle, it follows from Observation 5.22 that the odd girth of G is smaller than α. If both A and B are cycles, then it immediately follows that G contains an edge that lies on two distinct cycles of length α. ◀

Using this result, we will prove that if G has the no-overlap property, then it has a definable subdomain isomorphic to C_α^k. We need the following additional lemmas and definitions, following Bulatov's notation [21].

Definition 5.26 (ker) For a function $f\colon S \to T$, define

$$\ker(f) := \{(s, s') \mid s, s' \in S \wedge f(s) = f(s')\}.$$

Hence $\ker(f)$ is the equivalence relation pairing up all elements with the same image under f, represented as a set of pairs.

Definition 5.27 (π_I) For a domain D, integer $k \geq 1$, and index set $I = \{i_1, \ldots, i_{|I|}\} \subseteq [k]$ with $i_1 < \ldots < i_{|I|}$, we use π_I to denote the function that projects any tuple $\mathbf{x} = (x_1, \ldots, x_k) \in D^k$ onto index set I, so that $\pi_I(\mathbf{x}) := (x_{i_1}, \ldots, x_{i_{|I|}})$.

Observe that by this definition, $\ker(\pi_I) = \{(\mathbf{x}, \mathbf{y}) \mid x_i = y_i \text{ for all } i \in I\}$.

▶ **Proposition 5.28** *Let $\alpha \geq 3$ be odd. For all vertices $b_0, b_\alpha \in V(C_\alpha)$ there exist walks*

$$(b_0, b_1, \ldots, b_{\alpha-1}, b_\alpha) \text{ and } (b_0 = b'_0, b'_1, \ldots, b'_{\alpha-1}, b'_\alpha = b_\alpha)$$

of length α in C_α such that $b_1 \neq b'_1$, and if $b_0 \neq b_\alpha$ then $b_{\alpha-1} = b'_{\alpha-1}$.

Proof. Assume without loss of generality that $b_0 = 1$. In case that $b_0 = b_\alpha$, the walks $(1, 2, 3, \ldots, \alpha, 1)$ and $(1, \alpha, \alpha-1, \ldots, 2, 1)$ suffice.

Assume $b_0 \neq b_\alpha$. Let $P = (p_1, \ldots, p_m)$ be a path of odd length from b_0 to b_α. Observe that such a path exists and has length at most $\alpha - 2$, as $b_0 \neq b_\alpha$. Let b_1 and b'_1 be the neighbors of b_0. Consider the walks given by (b_0, b'_1, P) and (b_0, b_1, P), pad both paths with repetitions of (p_{m-1}, p_m) as needed such that they have length exactly α. It is easy to see that these walks satisfy the requirements. ◀

For $\mathbf{x} = (x_1, \ldots, x_{k-1}) \in V(C_\alpha^{k-1})$ and $c \in V(C_\alpha)$, let $(\mathbf{x}, c)$ denote the tuple formed by $(x_1, \ldots, x_{k-1}, c)$; note that such a tuple represents a vertex in C_α^k. We sometimes omit the brackets for readability.

▶ **Lemma 5.29** *Let G be a nonbipartite graph with odd girth α that has the no-overlap property. Let $\varphi\colon V(C_\alpha^k) \to V(G)$ be a homomorphism for some $k \geq 2$. If there exist $\mathbf{x}, \mathbf{y} \in V(C_\alpha^{k-1})$ and $c, d \in V(C_\alpha)$ such that $\varphi(\mathbf{x}, c) = \varphi(\mathbf{y}, d)$ and $c \neq d$, then*

$$ker(\pi_{[k]\setminus\{k\}}) \subseteq ker(\varphi),$$

that is, $\varphi(\mathbf{x}') = \varphi(\mathbf{y}')$ for all $\mathbf{x}', \mathbf{y}' \in V(C_\alpha^k)$ that agree on the first $k-1$ coordinates.

Proof. Let $\varphi\colon V(C_\alpha^k) \to V(G)$ be a homomorphism. We start with the following claim.

▷ **Claim 5.30** *Let $((\mathbf{x_0}, a_0), \ldots, (\mathbf{x_\alpha}, a_\alpha))$ be a walk in C_α^k such that $a_0 \neq a_\alpha$, $\varphi(\mathbf{x_0}, a_0) = \varphi(\mathbf{x_\alpha}, a_\alpha)$, and $\mathbf{x_i} \in V(C_\alpha^{k-1})$ for all $0 \leq i \leq \alpha$. Let b_1, b'_1 be the neighbors of a_0 in C_α. Then*

$$\varphi(\mathbf{x_1}, b_1) = \varphi(\mathbf{x_1}, b'_1).$$

Proof. Use Proposition 5.28 to obtain walks $(a_0 = b_0, b_1, \ldots, b_\alpha = a_\alpha)$ and $(a_0 = b'_0, b'_1, \ldots, b'_\alpha = a_\alpha)$ in C_α such that $b_{\alpha-1} = b'_{\alpha-1}$. Consider the following walks in C_α^k:

$$W := ((\mathbf{x_0}, b_0), \ldots, (\mathbf{x_\alpha}, b_\alpha)), \text{ and } W' = ((\mathbf{x_0}, b'_0), \ldots, (\mathbf{x_\alpha}, b'_\alpha)).$$

The images of W and W' under φ form walks of length α in G. Let these images be Y and Y' respectively. Observe that Y and Y' are closed walks in G, by the assumption that $\varphi(\mathbf{x_0}, a_0) = \varphi(\mathbf{x_\alpha}, a_\alpha)$. Since the odd girth of G is α, it follows using Observation 5.22 that Y and Y' must be cycles in G.

Since $b_{\alpha-1} = b'_{\alpha-1}$, the last edge of walks W and W' is the same. Thereby, the same holds for Y and Y', such that Y and Y' are cycles of length α in G that share an edge. It follows from the fact that G has the no-overlap property, that thereby $Y = Y'$. Hence $\varphi(\mathbf{x_i}, b_i) = \varphi(\mathbf{x_i}, b'_i)$ for all $0 \leq i \leq \alpha$, implying $\varphi(\mathbf{x_1}, b_1) = \varphi(\mathbf{x_1}, b'_1)$. ◁

We will prove that there exists a $(k-1)$-tuple $\mathbf{y_0} \in V(C_\alpha^{k-1})$ such that for all $a, a' \in V(C_\alpha)$ it holds that $\varphi(\mathbf{y_0}, a) = \varphi(\mathbf{y_0}, a')$. Towards achieving this, we prove the following claim.

▷ **Claim 5.31** *Suppose that $\mathbf{x_0} \in V(C_\alpha^{k-1})$, that $b, b' \in V(C_\alpha)$ are distinct vertices that have a common neighbor a in C_α, and that $\varphi(\mathbf{x_0}, b) = \varphi(\mathbf{x_0}, b')$. Suppose further that $\mathbf{x_1} \in V(C_\alpha^{k-1})$ is such that there exists a walk $(\mathbf{x_0}, \mathbf{x_1}, \ldots, \mathbf{x_0})$ of length α in C_α^{k-1}. Let a and a' be the two neighbors of b; then*

$$\varphi(\mathbf{x_1}, a) = \varphi(\mathbf{x_1}, a').$$

Proof. From Proposition 5.28, there is a walk $W = (b, c, \ldots, b')$ of length α from b to b' in C_α. Consider the walk $U = ((\mathbf{x_0}, b), (\mathbf{x_1}, c), \ldots, (\mathbf{x_0}, b'))$ in C_α^k obtained from combining the walk in the hypothesis and the walk W. By Claim 5.30 applied to U, we obtain the conclusion. ◁

From the hypothesis of the lemma and from Proposition 5.28, there is a length-α walk from a vertex $(\mathbf{x}, c) \in C_\alpha^k$ to a vertex $(\mathbf{y}, d) \in C_\alpha^k$ where both vertices are mapped to the same value under φ and such that $c \neq d$. We can apply Claim 5.30 to this walk to obtain $\mathbf{y_0} \in C_\alpha^{k-1}$, and vertices $b_0, b'_0 \in C_\alpha$ sharing a neighbor (in C_α) such that $\varphi(\mathbf{y_0}, b_0) = \varphi(\mathbf{y_0}, b'_0)$. We assume up to symmetry that $b_0 = \alpha$ and $b'_0 = 2$, so we have $\varphi(\mathbf{y_0}, \alpha) = \varphi(\mathbf{y_0}, 2)$. Let $\mathbf{z_0}$ be such that there exists a walk $W_0 = (\mathbf{y_0}, \mathbf{z_0}, \ldots, \mathbf{y_0})$ of length α in C_α^{k-1}. By applying Claim 5.31, we may obtain that $\varphi(\mathbf{z_0}, 1) = \varphi(\mathbf{z_0}, 3)$. By applying this same claim to the vertices $3, 1 \in C_\alpha$ and the length α walk $(\mathbf{z_0}, \mathbf{y_0}, \ldots, \mathbf{z_0})$ obtained by rotating W_0, we obtain that $\varphi(\mathbf{y_0}, 2) = \varphi(\mathbf{y_0}, 4)$. Repeatedly arguing in this zipper-like fashion, we obtain that $\varphi(\mathbf{y_0}, \alpha) = \varphi(\mathbf{y_0}, 2) = \varphi(\mathbf{y_0}, 4) = \cdots$. Since 2 generates the integers modulo any odd number, we obtain that for all values b, b' it holds that $\varphi(\mathbf{y_0}, b) = \varphi(\mathbf{y_0}, b')$.

We use this to argue that in fact for an arbitrary $\mathbf{x_0} \in V(C_\alpha^{k-1})$ it holds that $\varphi(\mathbf{x_0}, a) = \varphi(\mathbf{x_0}, a')$ for all $a, a' \in V(C_\alpha)$. To show this, we will show that if the statement holds for $\mathbf{x_0}$ and $\mathbf{x_1}$ is a neighbor of $\mathbf{x_0}$, then the statement holds for $\mathbf{x_1}$.

▷ **Claim 5.32** *Let $\mathbf{x_0} \in V(C_\alpha^{k-1})$ and let $\mathbf{x_1}$ be a neighbor of $\mathbf{x_0}$ in C_α^{k-1}. If $\varphi(\mathbf{x_0}, b) = \varphi(\mathbf{x_0}, b')$ for all $b, b' \in V(C_\alpha)$, then for all $b, b' \in V(C_\alpha)$:*

$$\varphi(\mathbf{x_1}, b) = \varphi(\mathbf{x_1}, b').$$

Proof. It is sufficient to prove that $\varphi(\mathbf{x_1}, b) = \varphi(\mathbf{x_1}, b')$ for all b, b' that have a common neighbor in C_α. After all, by using this repeatedly we then obtain

$\varphi(\mathbf{x_1}, 1) = \varphi(\mathbf{x_1}, 3) = \ldots = (\mathbf{x_1}, \alpha) = \varphi(\mathbf{x_1}, 2) = \ldots$ and obtain the result for all $b, b' \in V(C_\alpha)$.

Let such b, b' be given with common neighbor a and let $\mathbf{x_0}$ and $\mathbf{x_1}$ be given. Let a' be the other neighbor of b. It is easy to verify that there exists a walk $(\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_{\alpha-1}}, \mathbf{x_0})$ in C_α^{k-1} for some $\mathbf{x_2}, \ldots, \mathbf{x_{\alpha-1}}$ and furthermore there exists a walk $(a, a_1, a_2, \ldots, a_{\alpha-1}, a')$ in C_α. Consider the walk

$$W = ((\mathbf{x_0}, a), (\mathbf{x_1}, a_1), (\mathbf{x_2}, a_2), \ldots, (\mathbf{x_{\alpha-1}}, a_{\alpha-1}), (\mathbf{x_0}, a')).$$

Given that b and b' are the neighbors of a, we obtain from Claim 5.30 that $\varphi(\mathbf{x_1}, b) = \varphi(\mathbf{x_1}, b')$, as desired. ◁

The fact that $\varphi(\mathbf{x_0}, b) = \varphi(\mathbf{x_0}, b')$ for arbitrary $\mathbf{x_0}, b, b'$ now follows from the existence of $\mathbf{y_0}$ such that $\varphi(\mathbf{y_0}, b) = \varphi(\mathbf{y_0}, b')$ for all $b, b' \in C_\alpha$, the fact that C_α^{k-1} is connected, and Claim 5.32.

To see that this indeed implies that $\ker(\pi_{[k]\setminus\{k\}}) \subseteq \ker(\varphi)$, suppose $(\mathbf{x}', \mathbf{y}') \in \ker(\pi_{[k]\setminus\{k\}})$. Let $\mathbf{x}'' = \pi_{[k]\setminus\{k\}}(\mathbf{x}')$ and similarly let $\mathbf{y}'' = \pi_{[k]\setminus\{k\}}(\mathbf{x}')$, such that $\mathbf{x}'' = \mathbf{y}''$. It now follows from the claim above that $\varphi(\mathbf{x}') = \varphi(\mathbf{x}'', x_k) = \varphi(\mathbf{y}'', x_k) = \varphi(\mathbf{y}'', y_k) = \varphi(\mathbf{y}')$ and thus $(\mathbf{x}', \mathbf{y}') \in \ker(\varphi)$, as desired. ◀

Definition 5.33 Let H be a graph and $k \geq 1$. A homomorphism φ from H^k to a graph G is *projective*, if there exists a nonempty index set $I \subseteq [k]$ such that $\varphi(\mathbf{x}) = \varphi(\mathbf{y})$ if and only if $\pi_I(\mathbf{x}) = \pi_I(\mathbf{y})$.

Hence in a projective homomorphism, the image of $\mathbf{x} \in V(H^k)$ is determined uniquely by the coordinates in I, and distinct values for those coordinates lead to distinct images. The lemma above allows us to prove the following.

▶ **Lemma 5.34** *Let G be a nonbipartite graph with odd girth α that has the no-overlap property. Let $k > 0$ and let $\varphi\colon V(C_\alpha^k) \to V(G)$ be a homomorphism from C_α^k to G. Then φ is projective.*

Proof. By symmetry of the graph C_α^k, Lemma 5.29 implies that for any coordinate $i \in [k]$, if there are vertices $\mathbf{x}, \mathbf{y} \in V(C_\alpha^k)$ that disagree on coordinate i but have the same image under φ, then any $\mathbf{x}', \mathbf{y}' \in V(C_\alpha^k)$ that agree on all coordinates except i satisfy $\varphi(\mathbf{x}') = \varphi(\mathbf{y}')$.

Let $I' \subseteq [k]$ consist of those indices i for which there exist $\mathbf{x}, \mathbf{y} \in V(C_\alpha^k)$ that differ on coordinate i but have the same image under φ. We can show that any two vectors that differ only in coordinates of I' have the same image under φ. Suppose $\mathbf{x}, \mathbf{y} \in V(C_\alpha^k)$ only differ on coordinates in I'. Then there exist $\mathbf{x_1}, \ldots, \mathbf{x_m} \in V(C_\alpha^k)$ such that $\mathbf{x} = \mathbf{x_1}$, $\mathbf{y} = \mathbf{x_m}$, and for all $j \in [m-1]$ we have that $\mathbf{x_j}$ and $\mathbf{x_{j+1}}$ only differ on a single coordinate, and this coordinate is in I'. We then obtain that $\varphi(\mathbf{x_j}) = \varphi(\mathbf{x_{j+1}})$ for all j by the reasoning above, and thus $\varphi(\mathbf{x}) = \varphi(\mathbf{y})$.

By definition of I', for any index $i \in [k] \setminus I'$, vectors that differ on coordinate i map to distinct images. Hence for $I := [k] \setminus I'$ we have $\varphi(\mathbf{x}) = \varphi(\mathbf{y})$ if and only if $\pi_I(\mathbf{x}) = \pi_I(\mathbf{y})$, which proves that φ is projective. ◀

Using this result, we can prove the following lemma.

► **Lemma 5.35** *Let G be a nonbipartite graph with odd girth α that has the no-overlap property, and let $k > 0$ be an integer. Let $\varphi\colon V(C_\alpha^k) \to V(G)$ be a homomorphism from C_α^k to G. Let $B := \{\varphi(\mathbf{v}) \mid \mathbf{v} \in V(C_\alpha^k)\} \subseteq V(G)$. Then the induced subgraph $G[B]$ is isomorphic to C_α^m for some $m \in [k]$.*

Proof. Note first that if C_α^m is a spanning subgraph of $G[B]$, then C_α^m must in fact be isomorphic to $G[B]$: by Lemma 5.25, if C_α^m is a spanning strict subgraph of $G[B]$, then $G[B]$ can be obtained from C_α^m by adding one or more edges and therefore G has odd girth less than α, or the no-overlap property is violated. Hence to complete the proof, it suffices to show that C_α^m is a spanning subgraph of $G[B]$ for some m.

By Lemma 5.34 there is a nonempty $I \subseteq [k]$ such that $\varphi(\mathbf{x}) = \varphi(\mathbf{y})$ if and only if $\pi_I(\mathbf{x}) = \pi_I(\mathbf{y})$. Let $m := |I|$. Hence the image $B := \{\varphi(\mathbf{x}) \mid \mathbf{x} \in V(C_\alpha^k)\}$ consists of exactly α^m points. For $\mathbf{y} \in V(C_\alpha^m)$, let $g(\mathbf{y}) \in V(C_\alpha^k)$ denote the vector that agrees with $\mathbf{y}$ on the coordinates indexed by I, and has value 1 at all other coordinates. By the projectivity of φ, the function $\varphi'(\mathbf{y}) := \varphi(g(y))$ is a bijection from $V(C_\alpha^m)$ to B. To show that C_α^m is isomorphic to a spanning subgraph of $G[B]$, it remains to show that for all edges $\{\mathbf{x}, \mathbf{y}\} \in E(C_\alpha^m)$ we have $\{\varphi'(\mathbf{x}), \varphi'(\mathbf{y})\} \in E(G)$. Consider such a pair $\{\mathbf{x}, \mathbf{y}\} \in E(C_\alpha^m)$. There exist $\mathbf{x}', \mathbf{y}' \in V(C_\alpha^k)$ such that $\{\mathbf{x}', \mathbf{y}'\} \in E(C_\alpha^k)$ while $\pi_I(\mathbf{x}') = \mathbf{x}$ and $\pi_I(\mathbf{y}') = \mathbf{y}$. For example, such a pair can be obtained by padding $\mathbf{x}$ with the value 1 and padding $\mathbf{y}$ with the value 2 on all positions not included in I. Projectivity of φ ensures $\varphi(\mathbf{x}') = \varphi'(\mathbf{x})$ and $\varphi(\mathbf{y}') = \varphi'(\mathbf{y})$. Since φ is a homomorphism and $\{\mathbf{x}', \mathbf{y}'\}$ is an edge of C_α^k, it follows that $\{\varphi(\mathbf{x}'), \varphi(\mathbf{y}')\}$ is an edge of G. Hence C_α^m is isomorphic to a spanning subgraph of $G[B]$, which completes the proof. ◄

Lemma 5.35 is the key ingredient in the proof of the main result of this subsection, given below. The statement of the next lemma is a generalization of [21, Claim 4] to general values of α; our proof strategy is similar now that we have obtained the necessary lemmata.

► **Lemma 5.36** *Let G be a graph with odd girth α that has the no-overlap property. Then G^* has a definable subdomain B such that $G[B] \leftrightarrow C_\alpha$.*

Proof. We will construct a sequence of subsets $B_1, B_2, \ldots$ of $V(G)$ such that $B_i \subseteq B_{i+1}$ for all i, and furthermore each $G[B_i]$ is isomorphic to $C_\alpha^{k_i}$ for some $k_i \geq 1$. Since $C_\alpha \leftrightarrow C_\alpha^k$ for all $k \geq 1$ (projecting onto any fixed coordinate gives a valid C_α-coloring), this yields the desired result. Let B_1 be the vertex set of an arbitrary cycle of length α in G, and start from $i = 1$. If B_i is a definable subdomain of G^*, we are done.

Now suppose B_i has been constructed, but B_i is not a definable subdomain of G^*. We show how to construct B_{i+1}. Since B_i is not a definable subdomain of G^*, it follows from Corollary 5.16 that there exists a polymorphism f of G^*

of some arity n, and vertices $u_1, \ldots, u_n \in B_i$, such that $f(u_1, \ldots, u_n) \notin B_i$. Note that, since f is a polymorphism of G^*, which includes singleton relations forcing a variable to a constant value $v \in V(G)$ for all $v \in V(G)$, it must hold that f is idempotent (i.e., $f(v, \ldots, v) = v$).

We can consider f as a homomorphism from G^n to G, mapping vertex $(v_1, \ldots, v_n) \in V(G^n)$ to $f(v_1, \ldots, v_n) \in V(G)$. Verify that f is indeed a homomorphism: if $\{(u_1, \ldots, u_n), (v_1, \ldots, v_n)\} \in E(G^n)$, then it follows that $\{u_i, v_i\} \in E(G)$ for all $i \in [n]$ by the definition of G^n. Since f is a polymorphism for G, it thus follows that $\{f(u_1, \ldots, u_n), f(v_1, \ldots, v_n)\} \in E(G)$, as desired.

Let $f' := f|_{B_i}$ be defined as f restricted to B_i, such that f' is a homomorphism from $G[B_i]^n$ to G. Since $G[B_i]$ is isomorphic to $C_\alpha^{k_i}$, it follows that f' is a homomorphism from $G[B_i]^n = C_\alpha^{n \cdot k_i}$ to G.

Define B_{i+1} as the image of f' in G. Since f' is idempotent, we obtain that B_i is a subset of B_{i+1}. Furthermore it is easy to see from the definition that $B_i \subsetneq B_{i+1}$. It follows from Lemma 5.35 that $G[B_{i+1}]$ is isomorphic to C_α^m for some m.

Since G is finite and B_i is a strict subset of B_{i+1} for every i, there exists an i such that B_i is a definable subdomain of G^*. Since by definition, $G[B_i]$ is isomorphic to C_α^m for some m, this concludes the proof. ◀

5.3.2 Obtaining a definable subdomain when an edge is in two α-cycles

The results of Section 5.3.1 can only be applied if G has the no-overlap property, so that no edge of G is contained in two distinct minimum-length odd cycles. In this section we develop constructions that can be used to define strict subdomains that induce nonbipartite graphs, if G does not have the no-overlap property. We start by defining a relevant binary relation on the vertex set of a graph, using $E(u, v)$ as the edge predicate of the graph.

Definition 5.37 (R_α) Let R_α be defined as follows:

$$R_\alpha(u, v) =$$
$$\exists x_1, \ldots, x_\alpha, y_1, \ldots, y_\alpha : \left(\bigwedge_{i \in [\alpha-1]} E(x_i, x_{i+1}) \right) \wedge \left(\bigwedge_{i \in [\alpha-1]} E(y_i, y_{i+1}) \right)$$
$$\wedge E(x_\alpha, x_1) \wedge E(y_\alpha, y_1) \wedge (y_1 = x_1) \wedge (y_\alpha = x_\alpha) \wedge (x_2 = u) \wedge (y_2 = v).$$

Intuitively, R_α relates vertices u and v that lie on closed walks of length α sharing a common edge $\{x_1, x_\alpha\}$, when u and v lie at distance one from vertex x_1 of this shared edge along the closed walk. Note that such vertices must receive the same color in any C_α-coloring of G. Using this relation, we can generalize Bulatov's [21] analysis of chains of rhombuses to graphs of odd girth larger than 3. See Figure 5.6 for an illustration. Note that R_α is symmetric and that

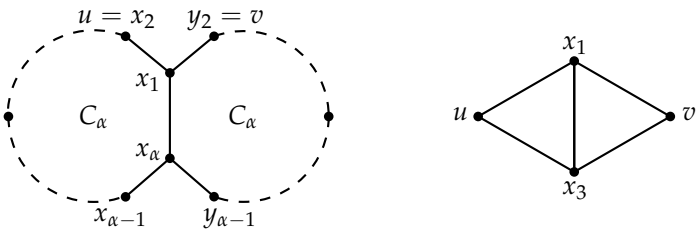


Figure 5.6 A general depiction of $uR_\alpha v$ (left) and a depiction of the $uR_3 v$ (right).

any vertex that lies on a cycle of length α is related to itself. The following property of R_α, R_α^n, and R_α^+ (recall Definition 2.31) will be essential in a number of proofs in this section.

▶ **Lemma 5.38** *Let G have odd girth α, such that each vertex of G is contained in a cycle of length α. For any $n \geq 0$, the relations R_α, R_α^n, and R_α^+ are pp-definable over G^*.*

Proof. The definition of R_α straightforwardly translates into a pp-definition by using the constant relations of G^* to enforce $x_2 = u$ and $y_2 = v$, and identifying variables that the formula requires to be equal.

Using this definition, we can define R_α^n using the same process on the expression

$$R_\alpha^n(u,v) := \exists x_1, \ldots, x_{n+1} : \left(\bigwedge_{i \in [n]} x_i R_\alpha x_{i+1} \right) \wedge (x_1 = u) \wedge (x_{n+1} = v).$$

Here applications of R_α can obviously be replaced by the pp-definition of R_α to obtain a pp-definition for R_α^n. Now observe that R_α is reflexive since each vertex of G lies on a cycle of length α. Hence for $n := |V(G)|$ we have that $R_\alpha^n = R_\alpha^+$, showing that the transitive closure can also be pp-defined by a finite expression. ◀

Before presenting further lemmata, let us discuss the overall approach and the relevance of the relation R_α. To obtain a lower bound via Lemma 5.21, we want to show that for any (core) graph H with a triangle, H^* has a definable subdomain B such that $H[B] \leftrightarrow C_\alpha$ for some odd $\alpha \geq 3$. If H has the no-overlap property, then we can obtain such a subdomain using Lemma 5.36. If H has an edge contained in two distinct triangles, then the private vertices of these overlapping triangles are related under R_3. Hence in the quotient graph $H' := H/_{R_3^+}$, these two private vertices are identified into the same equivalence class, effectively reducing the number of overlapping triangles. Repeating this process, we eventually arrive at a graph with the no-overlap property to which Lemma 5.36 can be applied. The challenge that then arises is the following:

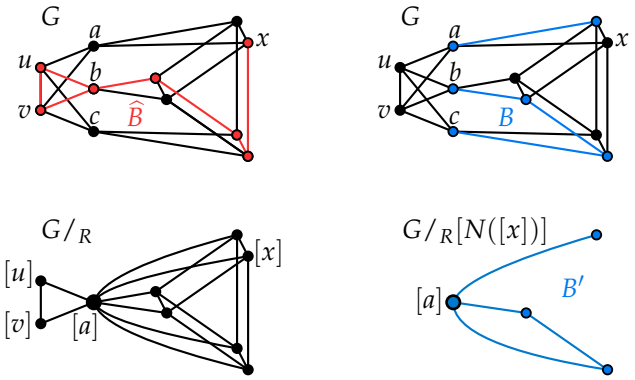


Figure 5.7 Illustration of the potential difficulties in lifting definable subdomains from quotient graphs, attacked in Lemma 5.40. Here $R = R_3^+$. Graph G on the top has overlapping triangles through the edge $\{u, v\}$. In the quotient graph $G/_R$ (bottom-left), the definable subdomain B' consisting of all neighbors of vertex $[x]$ induces a graph (homomorphically) equivalent to C_3, highlighted on the bottom-right. However, the pre-image B of B' in graph G (highlighted in blue on the top-right) induces a bipartite subgraph of G, not equivalent to C_3. Using the pp-definability of the set B we can define a subdomain $\hat{B}$ (highlighted in red on the top-left) of G^* for which $G[\hat{B}] \leftrightarrow C_3$ via $\hat{B}(y) := \exists x_1, \exists x_2 \colon x_1 \in B \wedge x_2 \in B \wedge E_G(c, x_1) \wedge E_G(x_1, x_2) \wedge E_G(x_2, y)$.

if B' is a definable subdomain of $(H')^*$ such that $H'[B'] \leftrightarrow C_{\alpha'}$ where α' is the odd girth of H', then we want to lift B' to a definable subdomain B of H^* such that $H[B] \leftrightarrow C_{\alpha'}$. This can fail in two significant ways:

- It is possible that while $H'[B']$ has a triangle, the vertices B that form the pre-image of B' induce a bipartite subgraph of H. This behavior is illustrated in Figure 5.7. In this case, we show that the subdomain B' can be exploited in a novel way to provide a definable subdomain of H equivalent to an odd cycle.
- The quotient graph H' may have strictly smaller odd girth than the odd girth α of H, which happens when H' has a self-loop while H does not. Since $C_{\alpha'}$ for odd $\alpha' < \alpha$ does not admit a homomorphism into a graph of odd girth α, there exists no induced subgraph of H homomorphically equivalent to $C_{\alpha'}$. This issue is resolved by showing that, rather than lifting B' to a definable subdomain of B, we directly define a strict nonbipartite subdomain of H using the fact that the odd girth decreases when taking the quotient graph. Therefore we get closer to the no-overlap case in a different manner.

Note that, by using induction, it is sufficient if our constructions yield a strict subdomain that induces a graph of the same odd girth, rather than one that induces a graph equivalent to an odd cycle. We therefore start by recalling

a simple condition under which a nonbipartite subdomain can be defined (cf. [21, §3.1]).

► **Lemma 5.39** *Let G have odd girth $\alpha > 1$. If some $v \in V(G)$ is not on any cycle of length α, then G^* has a strict definable subdomain B such that $G[B]$ has odd girth α.*

Proof. Let B be the set of vertices that lie on at least one cycle of length α. Clearly, if G has a vertex v that does not lie on a cycle of length α, it holds that $B \subsetneq V(G)$. Furthermore from this definition it follows $G[B]$ has odd girth α. It remains to show that B is indeed a definable subdomain of G^*. Define B as follows

$$B(x) := \exists x_0, \dots, x_\alpha : \left(\bigwedge_{i \in [\alpha]} E(x_{i-1}, x_i) \right) \wedge (x_0 = x_\alpha = x).$$

Observe that $B(x)$ holds exactly if x lies on a closed walk of length α in G. Since G has odd girth α, it follows from Observation 5.22 that thereby x lies on a cycle of length α. ◄

Using the property that every vertex lies on a minimum-length odd cycle, we now deal with the first type of lifting issue described above.

► **Lemma 5.40** *Let G be a graph of odd girth 3, such that every vertex of G is contained in a triangle. Let $G' := G/_{R_3^+}$. Assume G' has odd girth 3. If $(G')^*$ has a strict definable subdomain B' such that $G'[B']$ has odd girth 3, then G^* has a strict definable subdomain B such that $G[B]$ has odd girth 3.*

Proof. Let $B'' \subseteq V(G)$ denote the pre-image of B'. Proposition 5.18 shows that B'' is a definable subdomain of G^*, since R_3^+ is pp-definable by Lemma 5.38. Since $B' \subsetneq V(G')$ we also have $B'' \subsetneq V(G)$. If furthermore $G[B'']$ contains a triangle, the lemma statement follows. For the remainder of the proof, we therefore assume $G[B'']$ does not contain a triangle.

Choose n_1, n_2, n_3 satisfying the following, such that $n_1 + n_2 + n_3$ is minimized:

1. $n_1 \geq n_2 \geq 0$, and $n_1 \geq n_3 \geq 0$.
2. There exist $v_1, v_1', v_2, v_2', v_3, v_3' \in B''$ such that $v_i R_3^{n_i} v_i'$ for all $i \in [3]$, edge $\{v_i', v_{i+1}\} \in E(G)$ for $i \in [2]$, and $\{v_3', v_1\} \in E(G)$.

To see that such values exist, consider a triangle $\{b_1', b_2', b_3'\}$ in $G'[B']$. By Definition 5.14, the fact that $G'[B']$ contains an edge $\{b_i', b_j'\}$ for $(i,j) \in \{(1,2),(2,3),(3,1)\}$ implies that in G there is an edge between a vertex $v_i' \in [b_i']$ and a vertex $v_j \in [b_j']$. For the vertices defined in this way, appropriate values of n_i

5

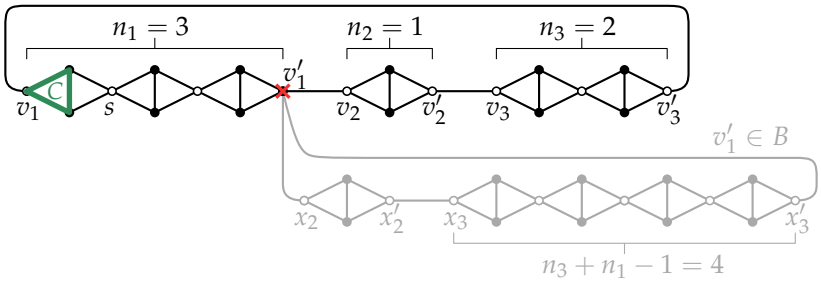


Figure 5.8 This figure describes the situation in the proof of Lemma 5.40, it depicts a subgraph of G. Vertices in B'' are marked in white. The triangle C that is part of B is indicated in bold green. Be aware that not all depicted vertices and edges are necessarily distinct.

exist since R_3^+ is the transitive closure of R_3: any two elements $v_i, v'_i \in [b'_i]$ related under R_3^+ are related by a finite number of applications of R_3.

Now take vertices $v_1, v'_1, v_2, v'_2, v_3, v'_3$ witnessing the minimal value of $n_1 + n_2 + n_3$ in the statement above. Observe that $n_1 > 0$, as otherwise the above statement shows that vertices $v_1, v_2, v_3 \in B''$ form a triangle in G, contradicting our assumption on $G[B'']$.

We now give a definable subdomain B of G^* that contains a triangle. Since B'' is a definable subdomain of G^*, we can write statements like $x \in B''$ in this definition. We can also use R_3 and powers of this relation in our pp-expression, by Lemma 5.38.

$$B(y) := \exists x_2, x_3, x'_1, x'_2, x'_3 \in B'' : \; x'_1 = v'_1 \; \wedge E_G(x'_1, x_2) \; \wedge x_2 R_3^{n_2} x'_2 \wedge \\ E_G(x'_2, x_3) \; \wedge x_3 R_3^{n_1+n_3-1} x'_3 \; \wedge E_G(x'_3, y).$$

Observe that, by the above definition, a vertex $y \in B$ may or may not lie in B'' but all vertices that are quantified over do lie in B''. Refer to Figure 5.8 for a sketch of the situation. It remains to show that $G[B]$ contains a triangle and that $B \subsetneq V(G)$. Let $s \in V(G)$ such that $v_1 R_3 s$ and $s R_3^{n_1-1} v'_1$. Note that such a vertex exists by the assumption that $v_1 R_3^{n_1} v'_1$.

We start by showing that $G[B]$ has a triangle containing v_1. Since $v_1 R_3 s$, there exist vertices a_1, a_2 such that the edge $\{a_1, a_2\}$ lies on a triangle with s and on a triangle with v_1. By letting $x_i := v_i$ and $x'_i := v'_i$ for $i \in [3]$, it is easy to verify that $v_1 \in B$. Furthermore, letting $x'_1 := v'_1$, $x_2 := x'_2 := v_2$, $x_3 := v'_1$ and $x'_3 := s$ shows that any vertex at distance one from s is in B, implying a_1 and a_2 are in B. Thereby, $G[B]$ contains a triangle.

It remains to show that $B \subsetneq V(G)$. To this end we will prove that $v'_1 \notin B$. Assume for a contradiction that $v'_1 \in B$. Thus, there exist witnesses $x_2, x_3, x'_1, x'_2, x'_3$ for this containment. Let $x_1 := x'_1 = v'_1$. Observe that hereby, $x_1 R_3^0 x'_1$, $x_2 R_3^{n_2} x'_2$,

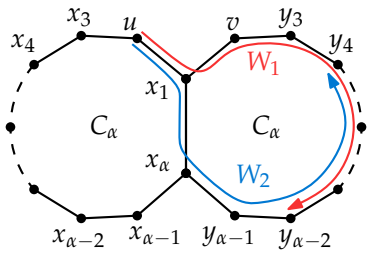


Figure 5.9 Depiction of the situation in the proof of Lemma 5.41. The figure shows vertices u and v such that $uR_\alpha v$ and a sketch of the proof that all vertices in the figure other than u and v can be reached by a walk of length exactly $\alpha - 2$ from u. Note that not all vertices are necessarily distinct, even if they are shown as such.

$x_3 R_3^{n_1+n_3-1} x_3'$, and $\{x_1', x_2\}, \{x_2', x_3\}, \{x_3', x_1\} \in E(G)$. As such, substituting the values $n_1' = 0, n_2' := n_2, n_3' := n_1 + n_3 - 1$ for n_1, n_2, n_3 satisfies statement 2 and $n_1' + n_2' + n_3' < n_1 + n_2 + n_3$. Renaming the variables to ensure that $n_1' \geq n_2'$ and $n_1' \geq n_3'$ (note that this can be done; the definition is entirely symmetric) ensures that also statement 1 is satisfied, which contradicts the minimality of our choice for n_1, n_2 and n_3. ◀

The next lemma gives a relevant property of the set of vertices that witnesses $uR_\alpha v$ in a graph G, that will be useful when proving the inclusion of these vertices in a definable subdomain.

▶ **Lemma 5.41** *Let G be a graph with odd girth α, let $u, v \in V(G)$ such that $uR_\alpha v$ (recall Definition 5.37) and let $S = \{x_1, \ldots, x_\alpha, y_1, \ldots, y_\alpha\}$ with $x_2 = u$ and $y_2 = v$ be vertices witnessing that $uR_\alpha v$. If $x \in S \setminus \{u, v\}$, then there is a walk in G of length $\alpha - 2$ from u to x.*

Proof. For a sketch of the situation and the proof, refer to Figure 5.9.

Consider any other vertex x_i for $i \in [\alpha] \setminus \{2\}$. Consider the walks $P := (u, x_3, x_4, \ldots, x_{i-1}, x_i)$ and $P' := (u, x_1, x_\alpha, x_{\alpha-1}, \ldots, x_{i+1}, x_i)$, which both have length at most $\alpha - 2$. Since the combined length of these walks is α, one of these two walks is odd and has length less than α. Thereby, there is a walk of length $\alpha - 2$ from u to x_i.

It remains to show that there is a walk of length $\alpha - 2$ from u to y_i for all $i \neq 2$. For $i = 1$ and $i = \alpha$ the result was shown above. We do a case distinction on the value of i. If $1 < i < \alpha$ is odd, consider the walk

$$W_1 := (x_2, x_1, y_2, y_3, \ldots, y_{i-1}, y_i),$$

observe that it has length exactly $i < \alpha$. Since i is odd and α is odd, it follows that there is a walk of length exactly $\alpha - 2$ from $u = x_2$ to y_i by padding W_1 as needed.

If $1 < i < \alpha$ is even, implying $i \geq 4$ as $i \neq 2$, consider the walk

$$W_2 := (x_2, x_1, x_\alpha, y_{\alpha-1}, \ldots, y_{i+1}, y_i),$$

which has length $2 + \alpha - i \leq \alpha - 2$. Observe that since α is odd, while 2 and i are even, the walk has odd length and thus there is a walk of length exactly $\alpha - 2$ from u to y_i. ◀

The next lemma will be used to solve the second type of lifting issue we described, by giving the subdomain construction used when taking the quotient graph reduces the odd girth. Observe that the lemma statement supports graphs of any odd girth $\alpha \geq 3$, meaning that the first type of lifting issue truly is the obstacle to proving a more general lower bound for H-COLORING.

▶ **Lemma 5.42** *Let G be a graph of odd girth $\alpha > 1$ such that each vertex of G lies on a cycle of length α. If $G/_{R_\alpha^+}$ has smaller odd girth than G, then there is a strict definable subdomain B of G^* such that $G[B]$ contains a cycle of length α.*

Proof. Suppose $G/_{R_\alpha^+}$ has smaller odd girth than G. Take *minimal* $n_1, \ldots, n_{\alpha-2} \in \mathbb{N}$ with $n_1 \geq n_i$ for all i such that there exist $v_1, v'_1, \ldots, v_{\alpha-2}, v'_{\alpha-2}$ with $v_i R_\alpha^{n_i} v'_i$ for all $i \in [\alpha - 2]$ and $\{v'_i, v_{i+1}\} \in E(G)$ for all $i \in [\alpha - 3]$ and $\{v'_{\alpha-2}, v_1\} \in E(G)$. Note that some of the n_i may be zero. We say a choice for $n_1, \ldots, n_{\alpha-2}$ is minimal simply if their sum is minimized along all possible valid choices.

Similarly as in the proof of Lemma 5.40, such values always exist. They are obtained by considering a closed walk of length $\alpha - 2$ in $G/_{R_\alpha^+}$, which exists since $G/_{R_\alpha^+}$ has odd girth at most $\alpha - 2$. We now show how to define a subdomain for G, using a case distinction on n_1.

(n_1 is zero) Since $n_1 \geq n_i$ for all $i \in [\alpha - 2]$ it follows that all n_i are zero. One may verify that in this case, there is a closed walk of length $\alpha - 2$ in the graph, contradicting that the odd girth is α by Observation 5.22.

($n_1 > 0$ is odd) Refer to Figure 5.10 for an illustration of this case. Let $n'_1 := \lfloor n_1/2 \rfloor$ and pick a vertex u such that $u R_\alpha^{n'_1} v'_1$ and $v_1 R_\alpha^{n'_1+1} u$. Note that such a vertex exists by definition since $n'_1 + (n'_1 + 1) = n_1$. Using Lemma 5.38, define B as

$$B(y) := \exists x_2, \ldots, x_{\alpha-2}, x'_1, \ldots, x'_{\alpha-2} : u R_\alpha^{n'_1} x'_1 \wedge \bigwedge_{1 \leq i \leq \alpha-3} E_G(x'_i, x_{i+1}) \wedge \bigwedge_{2 \leq i \leq \alpha-2} x_i R_\alpha^{n_i} x'_i \wedge E_G(x'_{\alpha-2}, y).$$

We show that $G[B]$ has a cycle of length α, and that some vertex of $V(G)$ is not included in B. We start by showing that $v'_1 \notin B$. Suppose towards a contradiction that $B(v'_1)$. Then by definition, there exist vertices $x_2, \ldots, x_{\alpha-2}, x'_1, \ldots, x'_{\alpha-2}$

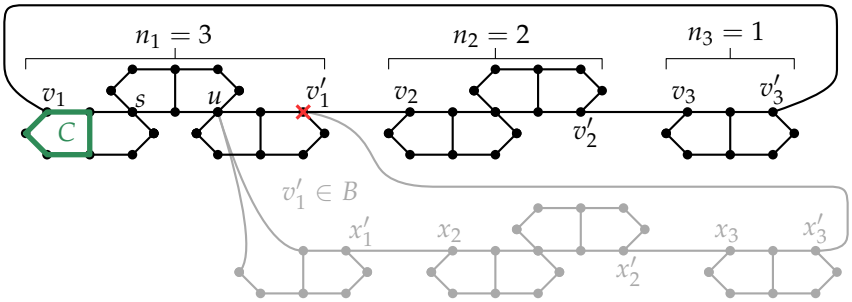


Figure 5.10 Illustration of the proof of Lemma 5.42, where $\alpha = 5$, $n_1 = 3$, $n_2 = 2$, and $n_3 = 1$. The gray part depicts the situation when $v'_1 \in B$, which leads to a contradiction. Be aware that not all depicted vertices and edges are necessarily distinct.

witnessing $B(v'_1)$. As such, $uR_\alpha^{n'_1}x'_1$ and by symmetry of the relation it follows that $x'_1R_\alpha^{n'_1}u$. Furthermore u was defined such that $uR_\alpha^{n'_1}v'_1$. Thereby, $x'_1R_\alpha^{2n'_1}v'_1$ with $2n'_1 = n_1 - 1$. Defining $x_1 := v'_1$ for convenience, it follows that $x_iR_\alpha^{n_i}x'_i$ for all $i \geq 2$, $x_1R_\alpha^{n_1-1}x'_1$, $\{x'_i, x_{i+1}\} \in E(G)$ for all i, and $\{x'_{\alpha-2}, x_1\} \in E(G)$ (as $x_1 = v'_1$ and $v'_1 \in B$). But this contradicts the minimality of the choice for $n_1, \ldots, n_{\alpha-2}$, since it follows that the sequence $n_1 - 1, n_2, \ldots, n_{\alpha-2}$ (potentially re-ordered to put the largest value at the front) satisfies all requirements while it has a strictly smaller sum.

It remains to show that $G[B]$ has a cycle of length α. Let s be a vertex such that $v_1R_\alpha s$ and $sR_\alpha^{n'_1}u$ (note that if $n_1 = 1$ then $s = v'_1$). Thereby, there exist vertices $c_1, \ldots, c_\alpha$ and $c'_1, \ldots, c'_\alpha$ witnessing $v_1R_\alpha s$, such that $v_1 = c_2$. Let C be the closed walk given by $(c_1, \ldots, c_\alpha, c_1)$ and observe that since the odd girth of G is α, C is a cycle in G by Observation 5.22. We will show that $V(C) \subseteq B$.

One may verify that B includes all vertices that can be reached via a walk of length $\alpha - 2$ from s by choosing $x'_1 = s$ and $x_i = x'_i$ for all $i \geq 2$ in the definition of B, using that R_α is reflexive. It follows from Lemma 5.41 that all vertices of C except v_1 are in B. It remains to show $v_1 \in B$, which is straightforward from the definition of B and the choice of $n_1, \ldots, n_{\alpha-2}$. Thus, $G[B]$ contains all α vertices of cycle C.

($n_1 > 0$ is even) Note that $n_1 > 1$. Refer to Figure 5.11 for a sketch of the situation. Let $n'_1 := n_1/2$. Pick v, v' such that $v_1R_\alpha^{n'_1}v$, $vR_\alpha v'$, and $v'R_\alpha^{n'_1-1}v'_1$. Since $v'R_\alpha v$, there are vertices $a_1, \ldots, a_\alpha$ and $a'_1, \ldots, a'_\alpha$ with $v = a_2$ and $v' = a'_2$ witnessing this. We define $u := a_1 = a'_1$ and $u' := a_\alpha = a'_\alpha$, such that $\{u, u'\}$ is the edge in which the cycles through v and v' overlap. Given u and u', we

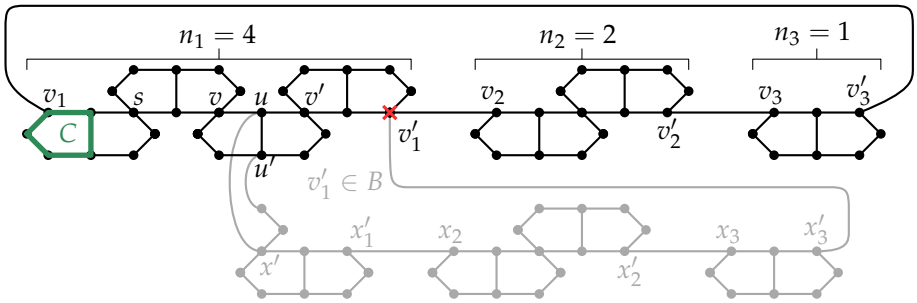


Figure 5.11 Illustration of the proof of Lemma 5.42 for $\alpha = 5$, where $n_1 = 4$, $n_2 = 2$, and $n_3 = 1$. Depicted vertices need not always be distinct. Indicated in gray is the situation when $v'_1 \in B$, which leads to a contradiction.

now define B as

$$B(y) := \exists x', x_2, \dots, x_{\alpha-2}, x'_1, \dots, x'_{\alpha-2} : x' R_\alpha^{n'_1-1} x'_1 \wedge \bigwedge_{1 \le i \le \alpha-3} E_G(x'_i, x_{i+1}) \wedge \bigwedge_{2 \le i \le \alpha-2} x_i R_\alpha^{n_i} x'_i \wedge E_G(x'_{\alpha-2}, y) \wedge (\exists y_1, \dots, y_\alpha : y_1 = u \wedge y_\alpha = u' \wedge y_2 = x' \wedge E_G(y_\alpha, y_1) \wedge \bigwedge_{1 \le i < \alpha} E_G(y_i, y_{i+1})).$$

We start by showing that $v'_1 \notin B$, such that indeed $B \subsetneq V(G)$. Suppose for contradiction that $v'_1 \in B$ and let $x', x_2, \dots, x_{\alpha-2}, x'_1, \dots, x'_{\alpha-2}$ and $y_1, \dots, y_\alpha$ be the vertices witnessing this. By definition, $v' R_\alpha^{n'_1-1} v'_1$. Furthermore, $x' R_\alpha^{n'_1-1} x'_1$ by definition. We start by showing $x' R_\alpha v'$ to obtain that $x'_1 R_\alpha^{2n'_1-1} v'_1$. By definition, we have that $y_1 = u$, $y_\alpha = u'$ and $y_2 = x'$, furthermore $a'_1 = u$, $a'_\alpha = u'$ and $a'_2 = v'$. It follows from the definitions that hereby $x' R_\alpha v'$ and thus $x'_1 R_\alpha^{2n'_1-1} v'_1$ with $2n'_1 - 1 = n_1 - 1$. Similar to the argument given in the case for n_1 is odd, we observe that introducing $x_1 := v'_1$ gives $x_1, \dots, x_{\alpha-2}$ and $x'_1, \dots, x'_{\alpha-2}$ satisfying $x_i R_\alpha^{n_i} x'_i$ for all $i \ge 2$, $\{x'_i, x_{i+1}\} \in E(G)$ for all i, and $\{x'_{\alpha-2}, x_1\} \in E(G)$. This contradicts the minimality of the choice for $n_1, \dots, n_{\alpha-2}$, since the sequence $n_1 - 1, n_2, \dots, n_{\alpha-2}$ (possibly after reordering) has a smaller total sum and satisfies all requirements.

It remains to show that $G[B]$ has a cycle of length α. The proof will follow from similar arguments as in the case where n was odd. Observe that $B(v_1)$ follows from substituting x' with v', x_i with v_i, and x'_i with v'_i. Choose s such that $v_1 R_\alpha s$ and $s R_\alpha^{n'_1-1} v$. Thus, there exist vertices $c_1, \dots, c_\alpha$ and $c'_1, \dots, c'_\alpha$ witnessing $v_1 R_\alpha s$, such that $v_1 = c_2$. Let C be the closed walk given by $(c_1, \dots, c_\alpha, c_1)$ and observe that since the odd girth of G is α, C is a cycle in G

by Observation 5.22. We will show that $V(C) \subseteq B$. Letting $x' := v$, $x'_1 := s$, and $x'_i = x_i$ for $i = 2, \ldots, \alpha - 2$ shows that B contains all vertices reachable by a walk of length $\alpha - 2$ from vertex s. It now follows from Lemma 5.41 that all vertices in $C \setminus \{v_1\}$ are in B and thus $V(C) \subseteq B$, concluding the proof. ◀

Using the two results above, we can not prove that when G contains a triangle, then it has a definable subdomain B such that $G[B]$ is homomorphically equivalent to a triangle. This is the last ingredient we will need to prove the main lower bound result of this chapter.

▶ **Lemma 5.43** *Let G be a graph with odd girth 3. Then G^* has a definable subdomain $B \subseteq V(G)$ such that $G[B]$ is homomorphically equivalent to C_3, that is, $G[B] \leftrightarrow C_3$.*

Proof. Proof by induction on $|V(G)|$. If G has the no-overlap property, then by applying Lemma 5.36 we obtain a definable subdomain B such that $G[B] \leftrightarrow C_3$.

In the remainder, we assume G has an edge that is contained in two distinct triangles. We first deal with an easy case. If there is a vertex of G that is not contained in any triangle, then Lemma 5.39 yields a strict definable subdomain B of G^* such that $G' := G[B]$ contains a triangle. Since $B \subsetneq V(G)$, we may apply induction to obtain a definable subdomain B' of $(G')^*$ such that $G'[B'] \leftrightarrow C_3$. By Proposition 5.17 the set B' is also a definable subdomain of G^*, and since $G[B'] = G'[B'] \leftrightarrow C_3$, this concludes the proof for this case.

It remains to deal with graphs in which some edge belongs to distinct triangles, in which every vertex lies on a triangle. Note that if some edge $\{u, v\}$ is in two triangles, say $\{u, v, x_1\}$ and $\{u, v, x_2\}$ for distinct $x_1, x_2 \in V(G)$, then $x_1 R_3 x_2$. Additionally, since every vertex lies on a triangle we have $v R_3 v$ for all $v \in V(G)$. It follows that R_3 is reflexive on G and therefore that R_3^+ is an equivalence relation on $V(G)$ that has strictly fewer than $|V(G)|$ equivalence classes. Consider $G/_{R_3^+}$. Since any closed walk in G projects to a closed walk in $G/_{R_3^+}$, the odd girth of $G/_{R_3^+}$ is at most that of G, so 3. If $G/_{R_3^+}$ has a self-loop, which occurs when some equivalence class of R_3^+ is not an independent set, then $G/_{R_3^+}$ has odd girth 1. We distinguish these two cases.

($G/_{R_3^+}$ has odd girth 1) By Lemma 5.42, G^* has a definable strict subdomain B such that $G' := G[B]$ contains a triangle, and therefore has odd girth 3. By induction, there is a definable subdomain B' of $(G')^*$ such that $G'[B'] \leftrightarrow C_3$. As before, Proposition 5.17 ensures B' is also a definable subdomain of G^*, and $G[B'] = G'[B'] \leftrightarrow C_3$.

($G/_{R_3^+}$ has odd girth 3) Let $G' := G/_{R_3^+}$, so that $|V(G')| < |V(G)|$. By induction on G', we find a definable subdomain B' of $(G')^*$ such that $G'[B'] \leftrightarrow C_3$. We distinguish two cases, depending on whether B' is a strict subdomain of $(G')^*$ or not.

5

1. If B' is a strict subdomain of $(G')^*$, then Lemma 5.40 guarantees that G^* has a strict definable subdomain $\hat{B}$ such that $G[\hat{B}]$ has a triangle and therefore has odd girth 3. As before, we may now apply induction to find a definable subdomain $\widetilde{B}$ of $(G[\hat{B}])^*$ such that $G[\widetilde{B}] \leftrightarrow C_3$. By Proposition 5.17 we know $\widetilde{B}$ is also a definable subdomain of G, which concludes the proof.
2. If B' is not strict, then $B' = V(G')$ and hence $G' \leftrightarrow C_3$. Since the quotient graph G' has odd girth 3, it does not have a self-loop, and therefore all equivalence classes of R_3^+ in G are independent sets. It follows that $G \to G'$ by mapping each vertex of G to the vertex representing its equivalence class in G', and therefore $G \to G' \to C_3$. Since G has a triangle by assumption, we also have $C_3 \to G$, so that $G \leftrightarrow C_3$. Hence the desired subdomain is $B := V(G)$, which is trivially definable.

This concludes the proof of Lemma 5.43. ◀

5.3.3 Sparsification lower bound for H-Coloring

The constructions for definable subdomains given by Lemmata 5.36 and 5.43 now allow us to prove the H-Coloring lower bound when H has the no-overlap property or contains a triangle, via Lemma 5.21. The main step of the proof is to show that graphs with a triangle have cores with a triangle, and that nonbipartite graphs with the no-overlap property have nonbipartite cores with the no-overlap property.

▶ **Theorem 5.44** *Let H be a simple nonbipartite graph whose shortest odd cycle has length α. If $\alpha = 3$, or no edge of H is contained in two distinct C_α-subgraphs, then H-Coloring parameterized by the number of vertices n does not admit a generalized kernel of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless* NP $\subseteq$ coNP/poly.

Proof. Let H be a graph satisfying the preconditions. Let H' be the core of H, that is, the unique (up to isomorphism) minimal induced subgraph H' of H for which $H \to H'$, so that in fact $H \leftrightarrow H'$. We claim that H' is nonbipartite and that the odd girth of H' equals α. If $C = (x_0, x_1, \ldots, x_\alpha)$ with $x_0 = x_\alpha$ is a cycle of length α in H, then the image of C under any homomorphism is a closed walk of length α. Since $H \to H'$, the graph H' has a closed walk of length α, and hence an odd cycle of length at most α by Observation 5.22. Since H' is a subgraph of H, this cycle cannot be shorter than α, which proves that H' has odd girth α.

We show that there is a definable subdomain B of $(H')^*$ such that $H'[B] \leftrightarrow C_\alpha$. If H has the no-overlap property, then no edge is in two distinct minimum-length odd cycles. Since the odd girth of H' equals that of H, and H' is an induced subgraph of H, it follows that H' also has the no-overlap property, so that the existence of such a subdomain follows from Lemma 5.36. If H has odd girth 3, then H' also has odd girth 3 as shown above, and Lemma 5.43 implies the existence of the desired subdomain.

It follows from Lemma 5.21, that the definable subdomain B of $(H')^*$ shows that H'-Coloring does not have a generalized kernel of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$ unless NP $\subseteq$ coNP/poly. Now, since $H' \leftrightarrow H$ it follows that a graph G has an H-coloring if and only if it has an H'-coloring, so that the decision problems H'-Coloring and H-Coloring are equivalent. Hence the same lower bound applies to the latter problem. ◀

5.4 Conclusion

For a large class of nonbipartite graphs H, we proved that H-Coloring is unlikely to admit a non-trivial polynomial-time sparsification algorithm. Our results imply that for the more general H-List Coloring problem, where the input graph G is given along with a list $\mathcal{L}(v) \subseteq V(H)$ for each $v \in V(G)$ and the question is whether G has a homomorphism to H mapping each vertex to a color on its list, there is *no* simple nonbipartite graph for which H-List Coloring has a non-trivial polynomial-time sparsification (unless NP $\subseteq$ coNP/poly). This follows from the fact that, by using the lists to force all vertices of G to be mapped to the vertex set of an induced odd cycle in H, one obtains a linear-parameter transformation from C_α-Coloring to H-List-Coloring. Hence the sparsification lower bound for the former problem, given in Theorem 5.4, also applies to the latter.

Our main open problem is to determine whether our sparsification lower bound for the cases of H-Coloring described by Theorem 5.44, can be generalized to all nonbipartite graphs H. One route to such a proof would be to show that if H is a simple nonbipartite core graph, then H^* has a definable subdomain that induces a subgraph homomorphically equivalent to an odd cycle; this would imply the lower bound via Lemma 5.21.

Chapter 6

An Optimal Kernel for H-Coloring

In this chapter, we will study the kernelization complexity of the q-Coloring problem and its generalization H-Coloring. Since 3-Coloring is already NP-hard, studying the problem parameterized by the number of colors does not yield interesting results from a parameterized perspective. As such, coloring problems are often studied under parameters that capture the complexity of the input graph. For example, Fiala *et al.* [38] compared the parameterized complexity of several coloring problems when parameterized by vertex cover, to the complexity when parameterized by treewidth. Jansen and Kratsch [57] studied the q-Coloring problem by a hierarchy of different parameters.

In this earlier work [57], Jansen and Kratsch provided a kernel for q-Coloring parameterized by the size of a vertex cover with $\mathcal{O}(k^q)$ vertices that can be encoded in $\mathcal{O}(k^q)$ bits. Furthermore they showed that for $q \geq 4$, a kernel of bitsize $\mathcal{O}(k^{q-1-\varepsilon})$ does not exist, unless NP $\subseteq$ coNP/poly. In Chapter 5, it was shown that also for $q = 3$ a kernel of bitsize $\mathcal{O}(k^{q-1-\varepsilon})$ is unlikely.

Unfortunately, there is a gap of a factor k between these bounds, and it remained unclear whether the upper or the lower bound had to be strengthened. In this chapter, we close this gap by improving the kernel. We show that q-Coloring has a kernel of bitsize $\mathcal{O}(k^{q-1}\log k)$ when parameterized by vertex cover, completely settling the kernelization size for the q-Coloring problem.

We further generalize the kernelization result by using the parameter twin-cover, which is smaller or equal to the size of a vertex cover. We obtain a kernel for the H-Coloring problem parameterized by the size of a twin-cover of size

$\mathcal{O}(k^{\Delta(H)} \log k)$. Since K_q-Coloring corresponds to the normal q-Coloring problem, this indeed generalizes the size-$\mathcal{O}(k^{q-1} \log k)$ kernel for q-Coloring.

Related work Ganian introduced twin-cover as a new parameter [44] and gave relations to existing parameters. For example, a minimum twin-cover is not larger than a minimum vertex cover, but twin-cover is incomparable to treewidth. The paper also gives an FPT algorithm for Precoloring Extension parameterized by the size of a twin-cover, and studies a number of other problems using this parameter.

Overview We start by providing the general idea towards obtaining a kernel of size $\mathcal{O}(k^2 \log k)$ for the 3-Coloring problem parameterized by vertex cover in Section 6.1. We then show how to use the ideas presented for the 3-Coloring kernel, to obtain the kernel for H-Coloring parameterized by twin-cover in Section 6.2.

6.1 Kernel for 3-Coloring parameterized by Vertex Cover

To get a better feeling for the general idea behind the kernel for H-Coloring presented in Section 6.2, we start by showing the kernel for 3-Coloring parameterized by the size of a vertex cover. First, we will quickly recap the existing kernel for 3-Coloring given by Jansen and Kratsch [57], which has size $\mathcal{O}(k^3)$. Then, we will show how using the results from Chapter 3 allows us to improve the size of this kernel to $\mathcal{O}(k^2 \log k)$.

6.1.1 Existing kernel of cubic size

Let us start by looking at the kernel by Jansen and Kratsch [57]. While their kernel works for general q-Coloring, we focus on the case of 3-Coloring here.

As an input for the kernelization algorithm, we assume we are given a graph G with vertex cover S of size k. We can assume this without loss of generality, since it is easy to obtain a 2-approximate vertex cover if no vertex cover is given. Let us start by some general observations. First of all, once the vertices in S are colored, it is easy to verify if this coloring can be extended to the entire graph. This is the case since the vertices in $V(G) \setminus S$ form an independent set. As such, by coloring S, their entire neighborhood is colored. All that needs to be verified, is that in each neighborhood there is at least one color that remains unused.

By looking at the problem this way, we see that each vertex $v \in G \setminus S$ puts a constraint on the coloring of the vertices in S: the vertices in the neighborhood of v should not use all three available colors. The reverse also holds; any coloring of S satisfying all such constraints can be extended to color the entire

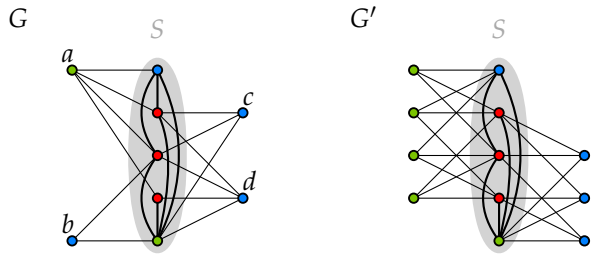


Figure 6.1 The result of applying the simple $\mathcal{O}(k^3)$-size kernel for 3-Coloring to the graph G depicted on the left, with the corresponding colorings. The fact that in this case G' is larger than G is caused by the choice of this (too small) example, observe that we did in fact make some progress. In particular, vertex b has been removed. Furthermore, vertex c is made redundant by the vertices added due to the existence of vertex d.

graph. In fact, when trying to extend the coloring of S to a vertex v in $V(G) \setminus S$, one may verify that it suffices to check for each size-3 subset of $N(v)$ that it does not use all three available colors. After all, if all three colors *are* used by $N(v)$, then there is a size-3 subset of $N(v)$ that witnesses this.

Kernel Given a 3-Coloring instance G with vertex cover S, we may obtain a kernelized instance G' as follows. Initialize G' as $G[S]$. For every size-3 subset S' of S, check if there exists a vertex $v \in V(G) \setminus S$ such that $S' \subseteq N(v)$. If so, add a new vertex v' to G' and connect v' to all vertices in S'. Refer to Figure 6.1 for an illustration of this procedure. We will not argue the correctness here in full detail, it can be found in Lemma 1 in [57]. The general idea is to prove that when G' is 3-colorable, we can color G by taking the same 3-coloring on S and extending this coloring to the remainder of the graph G.

Size The graph G' has at most $|S| + |V(G') \setminus S| \leq k + \binom{k}{3} = \mathcal{O}(k^3)$ vertices. The nice thing about this kernel is that it can even be stored in $\mathcal{O}(k^3)$ bits, by storing the graph structure of $G'[S]$ in $\mathcal{O}(k^2)$ bits, together with a Boolean vector of size $\mathcal{O}(k^3)$ that indicates for every size-3 subset of S, whether these vertices have a common neighbor in $V(G') \setminus S$.

6.1.2 Improved kernel

To improve the existing kernel, we use some similar ideas. Again, consider a graph G with vertex cover S of size k. For each size-3 subset $S' \subseteq S$ that has a common neighbor in $V(G) \setminus S$, we want to express that S' does not use all three colors. This can be seen as a constraint. In Section 6.1.1, we modeled this constraint by adding a vertex that was connected to all vertices in S'. Doing this for all size-3 subsets of S, resulted in a kernel with $\mathcal{O}(k^3)$ vertices. To improve

the size of the kernel, we will first try to reduce the number of constraints from $\mathcal{O}(k^3)$ to $\mathcal{O}(k^2)$, and then remove vertices and edges that corresponded to redundant constraints.

To reduce the number of constraints, we will use the method for sparsifying constraint satisfaction problems where the constraints are given as equalities of low-degree polynomials, that we described in Chapter 3. The first step is thus to model the relevant constraints as an instance of 2-Polynomial root CSP. To do this, we start by introducing three Boolean variables for each vertex in S, such that we have variable set

$$\{y_{v,c} \mid v \in S, c \in [3]\}.$$

Let $\mathbf{y}$ be a vector containing all these variables in arbitrary order. A coloring of the vertices corresponds to an assignment to the variables by letting $y_{v,c} = 1$ if vertex v has color c, and $y_{v,c} = 0$ otherwise. To formalize this, we will say that $\mathbf{y}$ is given a *choice assignment* if for all $v \in S$, there is exactly one $c \in [3]$ such that $y_{v,c} = 1$, meaning that each vertex in S is assigned exactly one color.

For distinct vertices $S' = \{u, v, w\} \subseteq S$, consider the following polynomial over the integers modulo two:

$$\begin{aligned} p_{S'}(\mathbf{y}) := {} & y_{u,1} \cdot y_{v,1} + y_{u,1} \cdot y_{w,1} + y_{v,1} \cdot y_{w,1} + \\ & y_{u,2} \cdot y_{v,2} + y_{u,2} \cdot y_{w,2} + y_{v,2} \cdot y_{w,2} + \\ & y_{u,3} \cdot y_{v,3} + y_{u,3} \cdot y_{w,3} + y_{v,3} \cdot y_{w,3}. \end{aligned}$$

We now observe the following. If $\mathbf{y}$ is given a choice assignment, corresponding to a coloring where u, v, and w receive distinct colors, then $p_{S'}(\mathbf{y}) \equiv_2 0$. If however $\mathbf{y}$ is given a choice assignment, corresponding to a coloring where u, v, and w use at most two distinct colors, then $p_{S'}(\mathbf{y}) \equiv_2 1$. A generalization of this result will be formally proven in Lemma 6.10. Informally, the idea is as follows.

- If u, v, and w each have a different color, then none of the terms of the polynomial evaluate to 1. After all, each term corresponds to two vertices receiving the same color. As such, we trivially obtain $p_{S'}(\mathbf{y}) \equiv_2 0$.
- If u, v, and w do not receive distinct colors, we consider two options. If all of them have the same color, it is easily verified that there will be exactly three terms of $p_{S'}(\mathbf{y})$ that evaluate to one. For example, if u, v, and w all receive color 1, we see that $y_{u,1} \cdot y_{v,1} = y_{u,1} \cdot y_{w,1} = y_{v,1} \cdot y_{w,1} = 1$. Thereby, we obtain $p_{S'}(\mathbf{y}) = 3 \equiv_2 1$. In the remaining case, exactly two vertices receive the same color, and the remaining vertex receives a different color. It is easy to verify that hereby exactly one term of the polynomial evaluates to 1, and again $p_{S'}(\mathbf{y}) \equiv_2 1$.

Kernel Given a graph G with vertex cover S, the kernel is now obtained as follows. We create a 2-Polynomial root CSP instance on the variable set

$\{y_{v,c} \mid v \in S, c \in [3]\}$. For each vertex $v \in V(G) \setminus S$, for each $S' \subseteq N(v)$ of size three, we add the constraint $p_{v,S'}(\mathbf{y}) \equiv_2 1$, or equivalently $p_{v,S'}(\mathbf{y}) - 1 \equiv_2 0$, to obtain a set L of polynomial equalities. The additional subscript v is only used to keep track of which polynomial was added for which vertex. Using Theorem 3.1, we obtain $L' \subseteq L$ with $|L'| \leq (3k)^2 + 1$, such that any assignment satisfying all equations in L', satisfies all equations in L.

We now obtain G' from G by removing every vertex in $V(G) \setminus S$ for which all corresponding equalities are in $L \setminus L'$, implying they were redundant. Furthermore, we remove an edge $\{u, v\}$ with $v \in V(G) \setminus S$ if there is no equality $p_{v,S'}(\mathbf{y}) - 1 \equiv_2 0$ in L' with $u \in S'$.

Correctness We will informally argue the correctness of the above procedure. Since G' is a subgraph of G, it is easy to verify that G' is 3-colorable if G is 3-colorable. The opposite direction is more interesting. Suppose G' is 3-colorable, we show that it is possible to 3-color G. For every vertex in S, use the same coloring as in G'. Observe that for every vertex in $V(G) \setminus S$, its neighborhood is now completely colored. We show that none of these neighborhoods uses all three available colors, to conclude the proof. Suppose towards a contradiction that there is a vertex $v \in V(G) \setminus S$ to which the coloring of S cannot be extended. As such, there are neighbors $v_1, v_2, v_3 \in N(v)$ such that v_i has color i for $i \in [3]$. Let $S' := \{v_1, v_2, v_3\}$. Hereby, $p_{v,S'}(\mathbf{y}) \equiv_2 0$, as we have seen before. Observe that $p_{v,S'}(\mathbf{y}) - 1 \equiv_2 0$ is one of the equations in the set L. We show that all equalities in L' are satisfied by the coloring of S. By choice of L', this implies that all equalities in L are satisfied, which is a contradiction with the fact that this equality is not.

Let $p_{v,S'}(\mathbf{y}) - 1 \equiv_2 0$ be an arbitrary equality in L'. Since this equality exists, v is a vertex of G' and v is connected to all three vertices in S'. Since we started from a valid 3-coloring of G', this equality must be satisfied by the coloring, as otherwise the neighborhood of v uses all three available colors. This would contradict that there is a proper 3-coloring of G', assigning the considered coloring to the vertices in S.

We conclude that we can extend the coloring of S to color all vertices in G.

Size The graph $G'[S]$ has k vertices by definition, and thereby has at most k^2 edges. Additionally, G' contains at most one vertex and at most three edges for every equality in L', leading to a total of $\mathcal{O}(k^2)$ edges and vertices. By using an adjacency-list representation, this kernel can be stored in $\mathcal{O}(k^2 \log k)$ bits.

Extensions Extending the idea described in this section to obtain a kernel for q-Coloring parameterized by vertex cover of size $\mathcal{O}(k^{q-1} \log k)$ is in principle straightforward, the only necessary ingredient is to find a degree-$(q-1)$ polynomial expressing the constraint that "these q vertices do not all receive distinct colors". We will see how this constraint can be formulated as a polynomial in

Lemma 6.10. To further extend the results to H-Coloring, some additional work is needed as not only the *number* of colors that is used in the neighborhood of a vertex matters, but also which colors specifically. Due to these additional constraints on the coloring, the kernel for H-Coloring will use two types of polynomial equalities, as we will see in Section 6.2.3.

While in this section we assumed that a vertex cover of the input graph is given, we show in the next section that this assumption is not needed. The overall strategy is as follows. Instead of only adding the necessary polynomial equalities corresponding to vertices not in the vertex cover to L, we simply consider the relevant polynomial equalities for *all* vertices in the graph, meaning we also introduce variables for all vertices in the input graph. Then, we iteratively check if one of the edges of the graph is redundant, by verifying whether the constraints in the graph obtained by removing this edge, imply the original set of constraints. If this is the case, we update the graph accordingly and recompute the set of constraints. Of course, now we can no longer straightforwardly apply our results from Chapter 3 to obtain a size bound on our kernel, as we have $\mathcal{O}(n)$ variables instead of $\mathcal{O}(k)$ variables. However, with some additional effort we can show that as long as the set of constraints is still large, the above reduction must be applicable. In this way, we obtain the same size bound.

Furthermore, we will extend the ideas described in this section to obtain a kernel parameterized by twin-cover, instead of vertex cover, but this does not add too many technical complications.

6.2 Kernel for H-Coloring parameterized by Twin-Cover

In this section, we give a kernel for H-Coloring parameterized by the size of a twin-cover. We start by giving some additional preliminaries and showing how to partition the graph into vertex sets that are twins in Section 6.2.1. We introduce some of the polynomial equalities that we use and their properties in Section 6.2.2, and use them in Section 6.2.3 to define the set of equalities that is constructed for a given input graph. In Section 6.2.4 we define the three reduction rules our kernel will use and prove that they are safe. Finally, in Section 6.2.5 we give the kernel.

6.2.1 Twin-Cover

To define the parameter twin-cover, we first need to define the notion of twins. We say vertices u and $v \in V(G)$ are *(true) twins* whenever $N_G[u] = N_G[v]$. Note that this relation is transitive.

Definition 6.1 We say $X \subseteq V(G)$ is a *twin-cover* [44] of G, if for every edge $\{u,v\} \in E(G)$, vertex $u \in X$, or $v \in X$, or u and v are twins.

We will regularly use the following two observations about H-colorings of a graph. Both follow straightforwardly from the definition of H-coloring.

Observation 6.2 *Let $S \subseteq V(G)$ such that $G[S]$ is a clique and let f be an H-coloring of G. Define $X := \{f(v) \mid v \in S\}$. Then $H[X]$ is a clique in H and all vertices in S receive a different color, so that $|S| = |X|$.*

Observation 6.3 *Let $v \in V(G)$ and let f be an H-coloring of G. Then the number of colors used to color $N_G(v)$ is bounded by $\Delta(H)$.*

Computing a minimum Twin-Cover is NP-hard, since Vertex Cover is NP-hard on graphs where no two vertices are twins[7]. We will therefore construct the kernel for H-Coloring without knowing a twin-cover of the input graph. In order to do this, we decompose the graph into vertex sets consisting of twins. Recall that throughout this chapter, twins are vertices with the same *closed* neighborhood.

Definition 6.4 A *partial twin decomposition* of a graph G is a partition $\Pi = \{P_1, \ldots, P_m\}$ of $V(G)$, such that any two vertices in the same partite set are twins. Partition Π is a *twin decomposition* if furthermore any two vertices in different partite sets are not twins.

To be able to use the twin decomposition for the kernelization procedure, we show how it can efficiently be computed.

► **Lemma 6.5** *A twin decomposition of a graph G can be computed in $\mathcal{O}(|V(G)| + |E(G)|)$ time.*

Proof. This is for example stated in [86, Exercise 2.17] for the case of finding false twins, which are vertices such that $N_G(u) = N_G(v)$. Finding (true) twins is similar. An example solution uses the adjacency-list representation, and adds each vertex to its own adjacency list. Then we efficiently sort the adjacency lists by bucket sort. We partition these sorted lists into sets of duplicates by a recursive process, which splits the sets into groups based on the first element of each adjacency list, which is removed from the recursively partitioned lists. The running time is linear since the sum of the lengths of all adjacency lists is $\mathcal{O}(|E(G)|)$, while the work done in each iteration is proportional to the decrease in total volume of the lists. ◄

The next lemma shows how the twin decomposition and a minimal twin-cover may intersect.

► **Lemma 6.6** *Let G be a graph with twin decomposition Π and a minimal twin-cover S. Then for any partite set $P \in \Pi$ it holds that either $P \subseteq S$ or $P \cap S = \emptyset$.*

[7] In a graph where no two vertices are twins, vertex covers and twin-covers are the same.

Proof. Let $P \in \Pi$. Suppose $P \cap S \neq \emptyset$ and $P \setminus S \neq \emptyset$. Let $u \in S \cap P$ and $v \in P \setminus S$. We show that $S \setminus \{u\}$ is a twin-cover of G, which contradicts the assumption that S is minimal.

Let $\{u, w\}$ be any edge in G. If $w = v$, then since $u, v \in P$, this is an edge between twins. If $w \neq v$, then since u and v are twins, it follows that $\{v, w\} \in E(G)$. Thereby, either $w \in S$ and thus edge $\{u, w\}$ is covered by w, or w and v are twins. In this case, by transitivity of being twins, u and w are also twins. This proves that $S \setminus \{u\}$ is indeed a twin-cover of G, which is a contradiction. ◀

6.2.2 Modeling constraints as polynomial equalities

As explained in Section 6.1.2, the kernelization is based on a connection to constraint satisfaction problems. To find the kernel, we represent the constraints that a vertex set puts on the coloring of its neighborhood, as polynomial equalities. We then use this representation to find redundant vertices and edges in the graph. Recall that a monomial of degree d is the product of d variables, with the unique monomial of degree zero being the constant 1. For example, $x_1 \cdot x_3 \cdot x_3$ is a monomial of degree three. A monomial is multilinear if each variable occurs at most once (refer to Section 2.6 for more details).

To find redundant vertices and edges, we will need to find which polynomial equalities are redundant. To be able to do this, we relate polynomials and polynomial equalities to the vectors that we will use to represent them.

Definition 6.7 (vect) Let $p(x_1, \ldots, x_n)$ be a multivariate polynomial in (a subset of) the variables $x_1, \ldots, x_n$, evaluated over the integers modulo 2, of degree at most d for some fixed d. Hence p is a weighted sum of monomials of degree at most d over $x_1, \ldots, x_n$. For some fixed ordering of the monomials of degree d over $x_1, \ldots, x_n$, let $\text{vect}(p)$ denote the vector containing the coefficients of the corresponding monomials in p. We may use this notation as well for polynomial equalities where the right-hand side is zero; for a polynomial equality $p(x_1, \ldots, x_n) \equiv_2 0$ over the integers modulo 2, such that p has degree at most d, we let $\text{vect}(p(x_1, \ldots, x_n) \equiv_2 0)$ be defined as $\text{vect}(p)$.

Let P be a set of multivariate polynomials in variables $x_1, \ldots, x_n$, we use $\text{vect}(P)$ to denote $\{\text{vect}(p) \mid p \in P\}$. Similarly, if L is a set of polynomial equalities in variables $x_1, \ldots, x_n$ for which the right-hand sides are zero, we let $\text{vect}(L)$ be defined as $\{\text{vect}(p) \mid (p(x_1, \ldots, x_n) \equiv_2 0) \in L\}$.

The following lemma follows from the definition above.

▶ **Lemma 6.8** *Let P be a set of polynomials of degree at most d over a tuple of variables $\mathbf{y}$, and let q be a polynomial of degree at most d over $\mathbf{y}$. If $\text{vect}(q) \in \text{span}_2(\text{vect}(P))$, then any assignment to $\mathbf{y}$ that satisfies $p(\mathbf{y}) \equiv_2 0$ for all $p \in P$, satisfies $q(\mathbf{y}) \equiv_2 0$.*

Proof. Choose $\alpha_p \in \{0, 1\}$ for all $p \in P$ such that $\text{vect}(q) \equiv_2 \sum_{p \in P} \alpha_p \, \text{vect}(p)$.

Consider an assignment to the variables $\mathbf{y}$ with $p(\mathbf{y}) \equiv_2 0$ for all $p \in P$. Let $\mathbf{y}'$ be the vector containing the evaluation of the monomials of degree at most d over $\mathbf{y}$, for the values assigned to $\mathbf{y}$. List them in the same order in which the coefficients for these monomials are listed in $\text{vect}(\cdot)$. Since a polynomial is a weighted sum of monomials, the value of a polynomial p of degree most d in $\mathbf{y}$ for the assigned values, equals the inner product of $\text{vect}(p)$ and $\mathbf{y}'$. So:

$$q(\mathbf{y}) = \text{vect}(q) \cdot \mathbf{y}' \equiv_2 \sum_{p \in P} \alpha_p \, \text{vect}(p) \cdot \mathbf{y}' \equiv_2 \sum_{p \in P} \alpha_p \cdot p(\mathbf{y}) \equiv_2 0. \qquad \blacktriangleleft$$

To utilize polynomials over Boolean variables to represent solutions of graph H-coloring problems, we represent the color of a vertex v in a graph G by $|V(H)|$ Boolean variables, indicating whether v has the corresponding color. We now define a partial choice assignment, which reflects that any vertex receives at most one color.

Definition 6.9 Let $y_{i,k} \in \{0,1\}$ for $i \in [n]$, $k \in [q]$ be a set of Boolean variables and let $\mathbf{y}$ be the vector containing all these variables. We say $\mathbf{y}$ is given a *partial choice assignment* if for all $i \in [n]$:

$$\sum_{k=1}^{q} y_{i,k} \leq 1.$$

Note that a partial choice assignment sets at most n variables to *true*. By this definition, a partial choice assignment can be seen as a partial coloring in the following way: $y_{i,k} = 1$ means vertex i has color k. Note that the coloring of some vertices may remain undefined.

The following lemma gives a polynomial that can be used to express the constraint that out of exactly q neighbors of a given vertex u, there are at least two that have the same color. By combining multiple such constraints, we can ensure that at most $q-1$ different colors are used in the neighborhood of vertex u, leaving one color free for u itself in the q-coloring problem. When evaluating the polynomial for $\mathbf{y}$ that is given a partial choice assignment, the polynomial has the following two essential properties. (1) It equals 1 modulo 2 when the q vertices all receive a distinct color, and (2) it equals 0 modulo 2 whenever two vertices have the same color, or when two vertices have no color defined[8].

► **Lemma 6.10** *Let $q > 0$ be an integer and let $y_{i,k}$ for $i \in [q], k \in [q]$ be Boolean variables. Then there exists a polynomial p of degree $q-1$ over the integers modulo 2, such that whenever the variables in $\mathbf{y}$ are given a partial choice assignment, it holds that $p(\mathbf{y}) \equiv_2 1$ if and only if*

[8] One may note that the polynomial described in Section 6.1.2 does *not* satisfy this. It evaluated to 0 when all vertices receive distinct colors, or when two colors where not defined. It evaluated to 1 whenever two vertices receive the same color. The fact that the role of 1 and 0 is swapped is easy to resolve, but the fact that it evaluates to 0 both for desirable and undesirable partial choice assignments, is problematic in our new setting.

- *there exist no* $i, j, k \in [q]$ *with* $i \neq j$ *such that* $y_{i,k} = y_{j,k} = 1$, *and*
- *for all* $k \in [q-1]$ *there exists* $i \in [q]$ *such that* $y_{i,k} = 1$.

We have seen an example of a polynomial used for the case where $q = 3$, in Section 6.1.2. However, the ideas behind this particular polynomial appear difficult to extend to larger values of q. Furthermore, while it has all desirable properties with respect to choice assignments, when considering partial choice assignments, problems arise. Considering partial choice assignments is however necessary to deal with H-COLORING, instead of q-COLORING.

Therefore, the proof of Lemma 6.10 will use an entirely different polynomial, that is based on different ideas. Before proving Lemma 6.10, let us therefore give the polynomial p corresponding to $q = 3$ that we will use in this proof.

$$\begin{aligned} p(\mathbf{y}) &:= \sum_{i_1 \neq i_2 \in [3]} \prod_{k=1}^{2} y_{i_k,k} \\ &= y_{1,1} \cdot y_{2,2} + y_{1,1} \cdot y_{3,2} + y_{2,1} \cdot y_{1,2} + y_{2,1} \cdot y_{3,2} + y_{3,1} \cdot y_{1,2} + y_{3,1} \cdot y_{2,2}. \end{aligned}$$

One may verify for this example that letting $y_{1,1} = y_{2,2} = y_{3,3} = 1$ and all other variables be zero, gives $p(\mathbf{y}) = 1 \equiv_2 1$, as desired. Setting $y_{1,1} = y_{2,2} = y_{3,2} = 1$ and all other variables to zero, gives $p(\mathbf{y}) = 2 \equiv_2 0$, which explains why the modulus is used. Letting $y_{1,1} = y_{2,3} = y_{3,3} = 1$ and all other variables zero, results in $p(\mathbf{y}) = 0 \equiv_2 0$. Furthermore, letting $y_{1,1} = y_{2,2} = 1$ and all other variables be zero, also results in $p(\mathbf{y}) \equiv_2 1$. We now proceed with the general construction.

Proof of Lemma 6.10. Define the multivariate polynomial p as

$$p(\mathbf{y}) := \sum_{\substack{i_1,\ldots,i_{q-1} \in [q] \\ \text{distinct}}} \prod_{k=1}^{q-1} y_{i_k,k}.$$

We prove that p has the desired properties. It is easy to see that the degree of p is $q-1$. It remains to prove the claim on the values of $p(\mathbf{y})$ for partial choice assignments. So let $\mathbf{y}$ be given a partial choice assignment, and for each $i \in [q]$ let $x_i := k$ exactly when $y_{i,k} = 1$. Let $x_i := 0$ when there is no such $y_{i,k}$.

We now show that $p(\mathbf{y}) \equiv_2 1$ if there exist no $i, j, k \in [q]$ with $i \neq j$ such that $y_{i,k} = y_{j,k} = 1$, and for all $k \in [q-1]$ there exists $i \in [q]$ such that $y_{i,k} = 1$. In terms of the values for x_i, this implies that they are all distinct, and that $[q-1] \subseteq \{x_1, \ldots, x_q\}$. Thereby, we have $\{x_1, \ldots, x_q\} = [q]$ or $\{x_1, \ldots, x_q\} = [q-1] \cup \{0\}$.

For $k \in [q-1]$, let j_k be the unique index such that $x_{j_k} = k$, implying that $y_{j_k,k} = 1$. Note that this is well defined, since all values from $[q-1]$ are used exactly once. Then, $\prod_{k=1}^{q-1} y_{j_k,k} = 1$. For any other choice of distinct indices

$i_1, \ldots, i_{q-1} \in [q]$, there exists $m \in [q-1]$ such that $i_m \neq j_m$. This implies that $y_{i_m,m} = 0$ and thereby $\prod_{k=1}^{q-1} y_{i_k,k} = 0$. Thus, $p(\mathbf{y}) = 1 \equiv_2 1$.

Now suppose there exist $i, j \in [q]$, such that $x_i = x_j \neq 0$, or there exists $k \in [q-1]$ such that $y_{i,k} = 0$ for all $i \in [q]$. We show that $p(\mathbf{y}) \equiv_2 0$ by a case distinction.

- Suppose there exists $k \in [q-1]$ such that $\sum_{i=1}^{q} y_{i,k} = 0$, or equivalently there is no $i \in [q]$ such that $x_i = k$. Thereby, for any choice of $i_1, \ldots, i_{q-1} \in [q]$, we have that $\prod_{\ell=1}^{q-1} y_{i_\ell,\ell} = 0$, since $y_{i_k,k} = 0$. Thus, $p(\mathbf{y}) \equiv_2 0$.
- Suppose there exists no $k \in [q-1]$ such that $\sum_{i=1}^{q} y_{i,k} = 0$. Thereby, for each $k \in [q-1]$ there exists $i \in [q]$ such that $x_i = k$. It follows from our earlier assumption that there must exist $i, j, k \in [q]$ with $i \neq j$ such that $x_i = x_j = k$, which implies $k < q$. For all $c \in [q-1]$ with $c \neq k$, let i_c be the unique index such that $x_{i_c} = c$ and thus $y_{i_c,c} = 1$. Then

$$y_{i,k} \cdot \prod_{\substack{c=1 \\ c \neq k}}^{q-1} y_{i_c,c} = y_{j,k} \cdot \prod_{\substack{c=1 \\ c \neq k}}^{q-1} y_{i_c,c} = 1.$$

  However, $\prod_{c=1}^{q-1} y_{i_c,c} = 0$ for any other choice of $i_1, \ldots, i_{q-1}$. Thereby, $p(\mathbf{y}) = 2 \equiv_2 0$. ◀

There is another way to look at the proof of Lemma 6.10, which may give some intuition. Consider the $q \times q$ matrix A given by $a_{i,j} = y_{i,j}$ for all $i \in [q], j \in [q-1]$, and $a_{i,q} = 1$ for all $i \in [q]$. Then $p(\mathbf{y})$ is equal to the determinant of the matrix A over the integers modulo 2. It follows from this that $p(\mathbf{y}) \equiv_2 1$ if and only if the columns of A are linearly independent. One may verify that if $\mathbf{y}$ is given a partial choice assignment, the columns of A are linearly independent if and only if the two conditions of the lemma statement hold for $\mathbf{y}$.

6.2.3 Construction of polynomial equalities

We continue to define the polynomial equalities that will be constructed for a subset P of the vertices of G. These are necessary constraints on the coloring of $N_G(P)$, such this coloring can be extended to H-color P. In this construction, P will be a partite set of the twin decomposition of G, and hence a clique.

Let G be a graph with $P \subseteq V(G)$. We create variables $c_{v,i}$ for each $v \in V(G)$ and $i \in V(H)$, denoting whether vertex v has color i. Let $\mathbf{C}$ be the set containing all constructed variables. Let $L(P, G)$ be the set of polynomial equalities produced by the following procedure, which results in two types of constraints. The first type will ensure that the neighborhood of P uses at most $\Delta(H)$ colors. The second type of constraints will ensure that the coloring of the

neighborhood of P can indeed be extended to color P: we ensure that there is a clique in H of size at least $|P|$, such that the colors used for $N_G(P)$ are in the neighborhood of this clique in H. The reason for having these two types of constraints is to obtain a good bound on the degree of the relevant polynomials.

Consider each set $S \subseteq N_G(P)$ with $|S| = \Delta(H)+1$ and each set $X \subseteq V(H)$ with $|X| = |S|$. Pick an arbitrary ordering on X such that $X = \{x_1, \ldots, x_{|S|}\}$ and use Lemma 6.10 to find a polynomial $p_{P,S,X}$ such that $p_{P,S,X}(\mathbf{C}) \equiv_2 1$ if and only if the following two statements hold:

1. there exist no $u \neq v \in S$, $k \in X$ such that $c_{u,k} = c_{v,k} = 1$, and
2. for all $k \in [|S|-1]$ there exists $u \in S$ such that $c_{u,x_k} = 1$.

When considering a partial choice assignment that corresponds to a mapping from $V(G)$ to $V(H)$, the above two statements together imply that S is colored with $|S| = \Delta(H)+1$ distinct colors (which is what we want to avoid). More precisely, when $p_{P,S,X}(\mathbf{C}) \equiv_2 1$, the coloring of S would use colors $x_1, \ldots, x_{|S|-1}$ and one other color. Add the following constraint to $L(P,G)$:

$$p_{P,S,X}(\mathbf{C}) \equiv_2 0.$$

For the second type of constraints, consider each set $S \subseteq N_G(P)$ of $\ell \in [\Delta(H)]$ elements. Pick an arbitrary ordering of S, such that $S = \{s_1, \ldots, s_\ell\}$. Consider each sequence $\mathbf{x} = (x_1, \ldots, x_\ell) \in V(H)^\ell$ (of not necessarily distinct elements). Let $Y := \bigcap_{i=1}^{\ell} N_H(x_i)$ be the common neighborhood of all vertices from $\mathbf{x}$ in H. If $H[Y]$ does not contain a clique of size at least $|P|$ (i.e., if $\omega(H[Y]) < |P|$), add the following polynomial equality to $L(P,G)$:

$$q_{P,S,\mathbf{x}}(\mathbf{C}) := \prod_{i=1}^{\ell} c_{s_i,x_i} \equiv_2 0.$$

The computation of $\omega(H[Y])$ for some $Y \subseteq V(H)$ can be done in constant time, since H is considered constant. This concludes the construction of $L(P,G)$. Note that the constraints $L(P,G)$ are defined solely in terms of the variables that describe the coloring of the *open neighborhood* of P.

Next, we define a complete list of equations for G.

Definition 6.11 ($L_\Pi(G)$) Let G be a graph and let Π be a partition of its vertex set. Let $L_\Pi(G)$ be defined as follows.

$$L_\Pi(G) := \bigcup_{P \in \Pi} L(P,G).$$

Since the polynomials for the first type of constraints have degree at most $|S|-1 = \Delta(H)$ by Lemma 6.10, while the polynomials for the second type of constraints are the product of $\ell \leq \Delta(H)$ variables, we observe the following.

Observation 6.12 *Let G be a graph with Π a partition of its vertex set. The polynomials in $L_\Pi(G)$ have degree at most $\Delta(H)$.*

We now prove two useful properties for this choice of constraints.

▶ **Lemma 6.13** *Let G be a graph with some partial twin decomposition Π. Let $f\colon V(G) \to V(H)$ be a mapping. If f is an H-coloring of G, then the Boolean assignment to $\mathbf{C} := \{c_{v,i} \mid v \in V(G), i \in V(H)\}$ given by $c_{v,i} = 1 \Leftrightarrow f(v) = i$ satisfies all constraints in $L_\Pi(G)$.*

Proof. Let f be given and the value of any $c_{v,i} \in \mathbf{C}$ be defined by $c_{v,i} = 1 \Leftrightarrow f(v) = i$. We show that this assignment satisfies all constraints in $L_\Pi(G)$, by showing that it satisfies both types of constraints in $L(P,G)$ for all $P \in \Pi$. Consider some $P \in \Pi$. Since it consists of twins, it is a clique in G. As H has no self-loops, the vertices in P all receive distinct colors by Observation 6.2, and the colors used on P form a clique in H. The fact that P consists of twins also implies that $\{u,v\} \in E(G)$ for all $u \in P, v \in N_G(P)$. Thereby, any color used in P is not used in the coloring of $N_G(P)$.

Consider a constraint $p_{P,S,X}(\mathbf{C}) \equiv_2 0 \in L(P,G)$ for $S \subseteq N_G(P)$ of size $|\Delta(H)|+1$ and $X \subseteq V(H)$ of the same size. By Observation 6.3, the vertices in S use at most $\Delta(H) = |S|-1$ colors. Hence at least one color $d \in V(H)$ appears twice on a vertex of S. If $d \in X$ then some color of X is used twice on S, violating the first condition of Lemma 6.10. If $d \notin X$ then at least two vertices $u,v \in S$ do not receive a color from X and hence $\sum_{i\in X} c_{u,i} = \sum_{i\in X} c_{v,i} = 0$. Since $|S| = |X| = q$, there are at most $q-2$ vertices $w \in S$ for which there exists $i \in X$ with $c_{w,i} = 1$. As such, there exist distinct colors $d_1, d_2 \in X$ such that $\sum_{s\in S} c_{s,d_1} = \sum_{s\in S} c_{s,d_2} = 0$, so that the lowest-indexed of d_1 and d_2 violates the second condition of Lemma 6.10. It then follows from Lemma 6.10 that $p_{P,S,X}(\mathbf{C}) \equiv_2 0$ as required.

Consider a constraint $q_{P,S,\mathbf{x}}(\mathbf{C}) \equiv_2 0 \in L(P,G)$ for $S \subseteq N(P)$ and $\mathbf{x} = (x_1,\ldots,x_{|S|}) \in V(H)^{|S|}$. Suppose this constraint is not satisfied. Then the coloring of S is given by $\mathbf{x}$ and furthermore, $H[Y]$ where $Y := \bigcap_{i=1}^{|S|} N_H(x_i)$ does not contain a clique on $|P|$ vertices. But any H-coloring (for H without self-loops) colors any clique in G with an equally-sized clique in H, and the colors used on the clique P must be adjacent in H to all the colors used on the neighbors S of P in G. Since $H[Y]$ contains no clique on $|P|$ vertices, f cannot be an H-coloring of G. It follows that for any H-coloring, all constraints are satisfied. ◀

Let $S \subseteq V(G)$ and let $f\colon S \to V(H)$ be an H-coloring of $G[S]$. We say that f can be *extended* to color G, if there exists $f'\colon V(G) \to V(H)$ such that f' is an H-coloring of G and furthermore $f'(v) = f(v)$ for all $v \in S$. The next lemma shows that an H-coloring of a part of the graph can be extended to color the entire graph, if it satisfies certain constraints.

▶ **Lemma 6.14** *Let G be a graph with $P' \subseteq V(G)$. Let $f\colon V(G) \setminus P' \to V(H)$ be an H-coloring of $G - P'$, such that the Boolean assignment to $\mathbf{C} := \{c_{v,i} \mid v \in V(G), i \in V(H)\}$ given by $c_{v,i} = 1 \Leftrightarrow (v \notin P' \wedge f(v) = i)$ satisfies all constraints in $L(P', G)$. Then f can be extended to H-color G.*

Proof. Let f be given and $\mathbf{C}$ be defined by $c_{v,i} = 1 \Leftrightarrow (v \notin P' \wedge f(v) = i)$. We start by showing that $N_G(P')$ uses at most $\Delta(H)$ different colors. Suppose not, then there is a set $S \subseteq N_G(P')$ of size $\Delta(H)+1$ using $\Delta(H)+1$ distinct colors. Let X be the set of colors used in S. It follows from Lemma 6.10, that regardless of which ordering of X was chosen when constructing $p_{P',S,X}(\mathbf{C})$, we have $p_{P',S,X}(\mathbf{C}) \equiv_2 1$. By definition, $L(P', G)$ contains the equation $p_{P',S,X}(\mathbf{C}) \equiv_2 0$. This contradicts the assumption that all constraints in $L(P', G)$ are satisfied.

Let X be the set of colors used by $N_G(P')$, suppose $|X| = \ell$. We have shown above that $\ell \leq \Delta(H)$. Let S be a size-ℓ subset of $N_G(P')$ such that for every color $x \in X$, there exists exactly one $s \in S$ such that $f(s) = x$. Let $Y := \bigcap_{x \in X} N_H(x)$. Suppose towards a contradiction that $H[Y]$ contains no clique of size $|P'|$. As such, for some ordering of S as $S = \{s_1, \ldots, s_\ell\}$ and for $\mathbf{x} = (x_1, \ldots, x_\ell)$ such that $f(s_i) = x_i$ for all i, the constraint $q_{P',S,\mathbf{x}}(\mathbf{C}) \equiv_2 0$ was added to $L(P', G)$. However, by definition, $q_{P',S,\mathbf{x}}(\mathbf{C}) \equiv_2 1$, contradicting the fact that all constraints in $L(P', G)$ are satisfied. Thereby, $H[Y]$ contains a clique K of size $|P'|$, where $Y := \bigcap_{i=1}^{|X|} N_H(x_i)$. To extend f to color P', assign each vertex in P' a distinct color from K.

It remains to verify that f is indeed a valid H-coloring of G. Any edge between two vertices in $V(G) \setminus P'$ remains properly colored. Any edge in P' is properly colored, because its endpoints have a different color and K is a clique in H. Any edge between P' and $V(G) \setminus P'$ is properly colored, because all vertices in K are common neighbors of the vertices in X, and $K \cap X = \emptyset$. ◀

6.2.4 Reduction rules

We now present the three reduction rules that will be used to obtain the kernel, and prove that they are safe. The first checks whether the graph is trivially not H-colorable, the second removes sets of edges from the graph, and the third removes sets of vertices from the graph.

Reduction rule 6.1 Let G be a graph with twin decomposition Π. If there exists $P \in \Pi$ with $|P| > \omega(H)$, return a trivial no-instance.

It is easy to see that Reduction rule 6.1 preserves the answer to the problem, since in this case G cannot have an H-coloring by Observation 6.2.

Reduction rule 6.2 Let G be a graph with twin decomposition Π. Let $P' \neq P'' \in \Pi$ such that $E_G(P', P'') \neq \emptyset$. If

$$\operatorname{vect}\big(L(P', G)\big) \subseteq \operatorname{span}_2\Big(\operatorname{vect}\big(L_\Pi(G \setminus E_G(P', P''))\big)\Big),$$

remove all edges in $E_G(P', P'')$ from graph G.

Reduction rule 6.2 is the key rule for our kernelization. It simplifies the graph by removing all edges between two distinct sets of twins P' and P'', if the constraints $L(P', G)$ are implied by the constraints generated by the remaining graph $G \setminus E_G(P', P'')$. Observe that it is essential for the effectiveness of Reduction rule 6.2 that Π is a twin decomposition, since applying the rule to partial twin decompositions that are not twin decompositions may increase the size of an optimal twin-cover in the considered graph. The following lemma proves that the reduction rule is safe.

▶ **Lemma 6.15** *If G' is obtained from G by applying Reduction rule 6.2, then G is H-colorable if and only if G' is H-colorable.*

Proof. Let G' be $G \setminus E_G(P', P'')$. Clearly, if G is H-colorable, then G' is also H-colorable, since G' is a subgraph of G.

In the other direction, let f' be an H-coloring of G'. It follows from Lemma 6.13 and the fact that Π is a partial twin decomposition of $G \setminus E_G(P', P'')$ that the derived setting of the Boolean variables $\mathbf{C}$ satisfies the constraints in $L_\Pi(G \setminus E_G(P', P''))$. Since $\text{vect}(L(P', G)) \subseteq \text{span}_2(\text{vect}(L_\Pi(G \setminus E_G(P', P''))))$ it follows from Lemma 6.8 that this setting of $\mathbf{C}$ also satisfies all constraints in $L(P', G)$. Let f be defined as f' restricted to the vertices in $G' - P'$. Note that $G' - P'$ equals $G - P'$ by definition. It is easy to see that f is indeed an H-coloring of $G - P'$ since $G - P'$ is a subgraph of G' and f' is an H-coloring of G'. Furthermore, f satisfies the constraints in $L(P', G)$ since it colors the relevant vertices the same as f'. It now follows from Lemma 6.14 that we can extend f to color all vertices in G. Thereby, G is H-colorable. ◀

The final rule effectively removes isolated cliques from the graph, when H has a sufficiently large clique to allow them to be colored properly.

Reduction rule 6.3 Let G be a graph with twin decomposition Π. If there exists $P' \in \Pi$ with $N_G(P') = \emptyset$ and $|P'| \leq \omega(H)$, then remove P' from G.

▶ **Lemma 6.16** *If G' is obtained from G by applying Reduction rule 6.3, then G is H-colorable if and only if G' is H-colorable.*

Proof. Let P' be such that $G' = G - P'$. Clearly, if G is H-colorable, G' remains H-colorable. In the other direction, suppose G' is H-colorable. We show how to extend this coloring to G. Since we assumed that $|P'| \leq \omega(H)$, graph H has a clique X of size $|P'|$. Use the colors of X to assign a different color to each vertex in P'. Since $N_G(P') = \emptyset$, this gives an H-coloring of G. ◀

▶ **Lemma 6.17** *Let G' be the graph resulting from applying Reduction rule 6.2 or 6.3 to a graph G. Then the size of a minimum twin-cover in G' is at most the size of a minimum twin-cover in G.*

Proof. Let $P \subseteq V(G)$. It is easy to see that if S is a twin-cover of G, then it is also a twin-cover of $G' = G - P$. Thereby, the statement holds for Reduction rule 6.3.

Let Π be the twin decomposition of G and let $P' \neq P'' \in \Pi$. Let $F := E_G(P', P'')$ be the set of edges between P' and P''. Let S be a twin-cover of G. We show that S is a twin-cover of $G' = G \setminus F$. Clearly, any edges that were previously covered, are still covered. We show that all vertices that were twins in G, are also twins in $G \setminus F$ to conclude the proof. Let u, v be twins in G, and let $P \in \Pi$ such that $u, v \in P$. If $P \neq P'$ and $P \neq P''$, it is obvious that u and v remain twins in $G \setminus F$. Suppose u, v lie in P' or in P''; without loss of generality, suppose $u, v \in P'$. Note that the edge $\{u, v\}$ belongs to $E(G) \setminus F$. Then $N_{G \setminus F}[u] = N_G[u] \setminus P''$. Since u and v are twins in G, we have $N_G[u] \setminus P'' = N_G[v] \setminus P'' = N_{G \setminus F}[v]$. Thereby, u and v are twins in $G \setminus F$. It follows that the lemma statement also holds for Reduction rule 6.2. ◀

6.2.5 Analysis of the kernelization

Having described all reduction rules, we can now formally prove the kernelization result for H-Coloring.

▶ **Theorem 6.18** *For any fixed non-bipartite graph H (without self-loops), H-Coloring parameterized by the size k of a twin-cover has a kernel with $\mathcal{O}(k^{\Delta(H)})$ vertices and edges, which can be encoded in $\mathcal{O}(k^{\Delta(H)} \log k)$ bits. Furthermore, the kernelized instance is a subgraph of the original input graph.*

Proof. Let G be a graph. Apply Reduction rules 6.1, 6.2, and 6.3 exhaustively. Let the resulting graph be G'. We show that this is a correct kernelization.

▷ **Claim 6.19** *Reduction rules 6.1–6.3 can exhaustively be applied in time $|V(G)|^{\mathcal{O}(\Delta(H))}$.*

Proof. We can compute a twin decomposition of G in linear time by Lemma 6.5. Computing $\omega(H)$ can be done in $\mathcal{O}(1)$ time for fixed H. Hence Reduction rule 6.1 can be applied in polynomial time.

The set $L_\Pi(G)$ contains at most $m := 2|V(G)| \cdot |V(G)|^{\Delta(H)+1} \cdot |V(H)|^{\Delta(H)+1}$ polynomial equalities (the number of ways to pick S, X, and P as for the definition of $p_{P,S,X}$ and $q_{P,S,X}$), over $|V(G)| \cdot |V(H)|$ variables. All polynomials we employ are multilinear. This can be verified directly from their construction, and explained by noting that squaring a number does not change it, when working modulo 2. By Lemma 2.27, we therefore only have to consider $(|V(G)| \cdot |V(H)|)^{\Delta(H)} + 1$ coefficients for the polynomials. Constructing the required polynomial equalities can be done in time $|V(G)|^{\mathcal{O}(\Delta(H))}$ for fixed H. We can test if one vector lies in the span of a set of other vectors by comparing the ranks of matrices of dimensions at most $m \times ((|V(G)| \cdot |V(H)|)^{\Delta(H)} + 1)$. Thereby, Reduction rule 6.2 can be applied in polynomial time. Note that the

twin decomposition has to be recomputed after each application of Rule 6.2. Reduction rule 6.3 can trivially be applied in polynomial time. Since $|\Pi| \leq |V(G)|$, checking for all $P \in \Pi$ whether any of the reduction rules can be applied takes polynomial time.

Each rule can be applied at most $|V(G)|^2$ times, as it always removes at least one edge or vertex. The claim follows. ◁

Let G' be the result of applying Reduction rules 6.1, 6.2, and 6.3 exhaustively. We use the following claim to prove a bound on the size of G'.

▷ **Claim 6.20** *The resulting graph G' has $\mathcal{O}\left(|V(H)|^{\Delta(H)} \cdot \Delta(H)^2 \cdot k^{\Delta(H)}\right)$ vertices and $\mathcal{O}\left(|V(H)|^{\Delta(H)} \cdot \Delta(H)^3 \cdot k^{\Delta(H)}\right)$ edges.*

Proof. When Reduction rule 6.1 has been applied at any point, G' trivially has constant size. Otherwise, since G has a twin-cover of size k, it follows from Lemma 6.17 that G' has a twin-cover of size at most k. Let Y be a minimum twin-cover of G', such that $|Y| \leq k$. Let Π be the twin decomposition of G'. By Lemma 6.6, every P in Π is either fully contained in Y, or disjoint from Y. Let $\Pi' := \{P \in \Pi \mid P \cap Y = \emptyset\}$. Define $L_{tc} := \bigcup_{P \in \Pi'} L(P, G')$, and note that $N_{G'}(P) \subseteq Y$ for all $P \in \Pi'$. This implies the polynomial equalities in L_{tc} only involve the variables controlling the coloring of Y. By Observation 6.12, the polynomials in L_{tc} have degree at most $\Delta(H)$ and they use at most $|V(H)|$ variables for each of the k vertices in Y. Define

$$\alpha := (k \cdot |V(H)|)^{\Delta(H)} + 1.$$

Let $V_{tc} := \text{vect}(L_{tc})$ be the vectors of coefficients corresponding to the polynomials in L_{tc}. Compute a basis $V'_{tc} \subseteq V_{tc}$ of V_{tc} over the integers modulo 2. Let $L'_{tc} \subseteq L_{tc}$ contain all polynomial equalities whose corresponding vector is contained in V'_{tc}. Since all employed polynomials are multilinear, it follows that the vectors in V_{tc} only have nonzero entries in positions corresponding to multilinear monomials over $|Y||V(H)|$ distinct variables, of which there are at most α by Lemma 2.27. As the size of the basis V'_{tc} equals the rank of the matrix containing the (row)vectors V_{tc}, which is upper-bounded by the number of columns that contain a nonzero entry, it follows that $|L'_{tc}| = |V'_{tc}| \leq \alpha$.

We define a set of meta-edges $F \subseteq (\Pi' \times (\Pi \setminus \Pi'))$ based on the constraints in L'_{tc}. For each constraint Z in L'_{tc}, do the following.

- Suppose $Z = (p_{P',S,X}(\mathbf{C}) \equiv_2 0)$ for some $P' \in \Pi'$, $S \subseteq N_G(P')$, and $X \subseteq V(H)$. Since P' is a partite set of twins that is disjoint from Y, we have $N_G(P') \subseteq Y$ since Y is a twin-cover. So each $v \in S$ belongs to a partite set P_v of twins with $P_v \in \Pi \setminus \Pi'$. For each $v \in S$, add (P', P_v) to F.
- Otherwise, $Z = (q_{P',S,\mathbf{x}}(\mathbf{C}) \equiv_2 0)$ for some $P' \in \Pi'$, $S \subseteq N_G(P')$, and sequence $\mathbf{x} = (x_1, \ldots, x_{|S|}) \in V(H)^{|S|}$. Similarly as above, for each $v \in S$ take $P_v \in \Pi \setminus \Pi'$ such that $v \in P_v$ and add (P', P_v) to F.

The above procedure adds at most $\Delta(H)+1$ meta-edges for each constraint in L'_{tc}. Thereby,

$$|F| \leq \alpha(\Delta(H)+1). \tag{6.1}$$

We now argue that for any $(P', P'') \notin F$ with $P' \in \Pi'$ and $P'' \in \Pi \setminus \Pi'$, the following holds:

$$L'_{tc} \subseteq L_{\Pi}(G' \setminus E_{G'}(P', P'')). \tag{6.2}$$

To see this, consider a constraint in L'_{tc}. It is of one of two possible types, and it was added to $L_{tc} = \bigcup_{P \in \Pi'} L(P, G') \supseteq L'_{tc}$ because it satisfied the criteria described in Section 6.2.3. Effectively, the constraint was created because some set $P \in \Pi'$ contains a certain vertex set S of size at most $\Delta(H)+1$ in its open neighborhood in G'. But by our choice of meta-edges F, the set P still has S in its neighborhood in $G' \setminus E_{G'}(P', P'')$, so that all constraints of L'_{tc} are also contained in $L_{\Pi}(G' \setminus E_{G'}(P', P''))$.

Using this, we show that for all $P' \in \Pi'$ and $P'' \in \Pi \setminus \Pi'$:

$$E_{G'}(P', P'') \neq \emptyset \Rightarrow (P', P'') \in F. \tag{6.3}$$

Suppose there exist $P' \in \Pi', P'' \in \Pi \setminus \Pi'$ such that $E_{G'}(P', P'') \neq \emptyset$ but $(P', P'') \notin F$. It follows from Equation 6.2 that $L'_{tc} \subseteq L_{\Pi}(G' \setminus E_{G'}(P', P''))$. Thereby,

$$\begin{aligned} \operatorname{span}_2(\operatorname{vect}(L_{\Pi}(G' \setminus E_{G'}(P', P'')))) &\supseteq \operatorname{span}_2(\operatorname{vect}(L'_{tc})) \\ &= \operatorname{span}_2(V'_{tc}) \supseteq V_{tc} = \operatorname{vect}(L_{tc}) \supseteq \operatorname{vect}(L(P', G')). \end{aligned}$$

Thereby, Reduction rule 6.2 could be applied to G', which is a contradiction. It follows that $P' \in \Pi'$ and $P'' \in \Pi \setminus \Pi'$ can only be connected in G' if there is a corresponding meta-edge in F. We can now use Equations 6.1 and 6.3 to bound the number of vertices and edges in G'.

First of all, for all $P' \in \Pi'$ there must exist some $P'' \in \Pi \setminus \Pi'$ such that $(P', P'') \in F$, otherwise it follows from Equation 6.3 that $N_{G'}(P') = \emptyset$ and P' would have been removed by Reduction rule 6.3. Thereby $|\Pi'| \leq |F|$. Since $|P| \leq \omega(H) \leq \Delta(H)+1$ for all $P \in \Pi$ by Reduction rule 6.1, the number of vertices of G' can be bounded as follows.

$$\begin{aligned} |V(G')| \leq |F| \cdot (\Delta(H)+1) + |Y| &\leq \left((k|V(H)|)^{\Delta(H)} + 1\right) \cdot (\Delta(H)+1)^2 + k \\ &= \mathcal{O}\left(|V(H)|^{\Delta(H)} \cdot \Delta(H)^2 \cdot k^{\Delta(H)}\right). \end{aligned}$$

If edge $\{u, v\} \in G'$ with $u \in Y$ and $v \notin Y$, then there exist $(P', P'') \in F$ such that $u \in P'$, $v \in P''$. Since $|P| \leq \Delta(H)+1$ for any $P \in \Pi$, there are at most $|F| \cdot (\Delta(H)+1)^2$ such edges. Furthermore, there are at most $\binom{|Y|}{2} \leq k^2$ edges between vertices in Y, and at most $|F| \cdot (\Delta(H)+1)^2$ edges between vertices in

$V(G) \setminus Y$. Thereby, the total number of edges can be bounded by:

$$\begin{aligned}|E(G')| &\leq |F| \cdot (\Delta(H)+1)^2 + |Y|^2 + |F| \cdot (\Delta(H)+1)^2 \\ &\leq 2\alpha(\Delta(H)+1)^3 + k^2 \\ &= 2\left((k \cdot |V(H)|)^{\Delta(H)} + 1\right) \cdot (\Delta(H)+1)^3 + k^2\end{aligned}$$

(note that $\Delta(H) \geq 2$ for non-bipartite H)

$$= \mathcal{O}\left(|V(H)|^{\Delta(H)} \cdot \Delta(H)^3 \cdot k^{\Delta(H)}\right).$$

This concludes the proof of Claim 6.20. ◁

It follows from the correctness of Reduction rules 6.1, 6.2, and 6.3 that G' is H-colorable if and only if G is H-colorable. It follows from Claims 6.19 and 6.20 that we have given a kernel for H-coloring with $\mathcal{O}(k^{\Delta(H)})$ vertices and edges for constant-size H that can be computed in polynomial time. By encoding the graph using adjacency lists, it can be encoded in $\mathcal{O}(k^{\Delta(H)} \cdot \log k)$ bits. ◀

The following corollary shows that Theorem 6.18 generalizes the result obtained for q-Coloring parameterized by vertex cover in the extended abstract of this work.

▶ **Corollary 6.21** *For any constant $q \geq 3$, q-Coloring parameterized by the size of a twin-cover has a kernel with $O(k^{q-1})$ vertices, which can be encoded in $O(k^{q-1} \log k)$ bits. Furthermore, the resulting instance is a subgraph of the original input graph.*

Proof. Since q-Coloring is equivalent to K_q-Coloring, and $\Delta(K_q) = q-1$ and K_q has q vertices, the result now follows directly from Theorem 6.18. ◀

6.3 Conclusion

We have given a kernel for H-Coloring parameterized by twin-cover with $\mathcal{O}(k^{\Delta(H)})$ vertices and bitsize $\mathcal{O}(k^{\Delta(H)} \log k)$. This kernel can be obtained without using information about (an approximation of) the minimum twin-cover of the input graph. It follows that q-Coloring parameterized by vertex cover has a kernel of bitsize $\mathcal{O}(k^{q-1} \log k)$, improving on the previously known kernel by almost a factor k. Furthermore, this kernel is optimal up-to $k^{o(1)}$ factors, as Jansen and Kratsch showed for $q \geq 4$, there is no kernel of size $\mathcal{O}(k^{q-1-\varepsilon})$ for the problem, unless NP $\subseteq$ coNP/poly [57]. For $q = 3$, the bound follows from Theorem 5.44 and the fact that the size of a minimum vertex cover is at most the number of vertices.

It is easy to see that the kernel lower bounds also hold for q-List Coloring, where every vertex v in the graph has a list $\mathcal{L}(v) \subseteq [q]$ of allowed colors.

Furthermore, we can apply our kernel by first reducing an instance of q-LIST COLORING to an instance of q-COLORING using q additional vertices, and adding these to the twin-cover of the graph. This only changes the size of the obtained kernel by a constant factor. We cannot apply the same method to extend the kernel to the general H-LIST COLORING problem, since the gadget to simulate the list constraints only works if H is a clique.

In this chapter we have seen a first example where finding redundant vertices and edges is done using appropriate polynomial equalities. It would be interesting to see if this technique can be applied to obtain smaller kernels for other graph problems as well [17]. To apply this idea, one needs to first identify which constraints should be modeled. When the constraints are found, they need to be written as equalities of low-degree polynomials over a suitably chosen field. This requires the clever construction of polynomials that have a sufficiently low degree, in order to obtain a good bound on the kernel size.

Acknowledgements We thank Tim Hartmann for suggesting the use of twin-cover as a parameter.

Chapter 7

Concluding Remarks

In this thesis we studied kernelization, and more specifically sparsification, and obtained tight bounds for several graph and logic problems. As the title of the thesis suggests, one of the main sparsification techniques was based on representing constraints by low-degree polynomials. We will next discuss the use of this technique and related techniques in the area of kernelization. We then consider the work on polynomial-time sparsification done in this thesis, and the broader area of sparsification research. We conclude this thesis by discussing potential directions for further research and by listing a few specific open problems.

7.1 Linear-algebraic techniques in kernelization

Using techniques based on linear algebra is relatively new in the area of kernelization. Earlier kernels for (for example) graph problems were more often based on sets of reduction rules, based on high- or low-degree rules (as in the well-known $\mathcal{O}(k^2)$-vertex kernel for VERTEX COVER), crown structures [1, 28, 37, 74], and packing arguments [19, 80]. More recently, a number of other techniques are appearing.

One such technique is the matroid-based method that can be used to obtain randomized kernels [40, Chapter 10]. A matroid is a mathematical structure

that can be defined as a pair $(E, \mathcal{I})$ where E is a set and $\mathcal{I}$ is a family of subsets of E representing the independent sets. The set $\mathcal{I}$ is required to satisfy various additional conditions. In a linear matroid, there is a matrix representing the matroid such that there is a one-to-one correspondence between E and the columns of the matrix, and the independent sets in $\mathcal{I}$ correspond to sets of linearly-independent columns. To obtain kernels using matroids, the idea is to reduce the problem to a (linear) matroid and then obtain a (small) representation of this matroid.

While matroids were already well-studied in various settings (being introduced in the 1930's [89]), they were only studied in parameterized complexity much more recently (cf. [78]).

Matroid-based methods were used in the area of kernelization to obtain a first (randomized) polynomial kernel for Odd Cycle Transversal [69]. An input to the Odd Cycle Transversal problem is a graph G and an integer k, and the goal is to decide whether G can be made bipartite by removing at most k vertices. Whether or not the problem admits a polynomial kernel when parameterized by solution size, was a long-standing open question that has now been positively resolved using a matroid-based method. These methods since turned out to be a helpful tool in more settings [66, 67].

In this thesis, we developed another algebraic tool for sparsification, which applies when we can model a problem using low-degree polynomial inequalities. We then observed that when there are many such equalities over a small number of variables, some of them can be shown to be redundant. In particular, given a set of degree-d polynomial equalities over at most n variables, we provided a method to reduce the number of equalities to $\mathcal{O}(n^d)$, without changing the answer. We further applied this method to obtain an improved kernel for the q-Coloring problem, when parameterized by the size of a vertex cover in the input graph. The method has also been shown to be useful to obtain a polynomial kernel for another graph coloring problem [17].

The method of sparsifying instances by representing them as low-degree polynomial equalities as such has been useful in various contexts, and adds to the available methods to obtain kernelization algorithms. Just like the matroid-based method, it is a data-reduction step based on linear-algebraic dependence.

Recently, the Interval Vertex Deletion problem was shown to have a polynomial kernel [4]. Given a graph G and integer k, the problem asks whether it is possible to remove at most k vertices from G, such that the resulting graph is an interval graph. Part of this kernelization algorithm relies on representing (part of) the structure of the graph by vectors, such that a basis can be computed. The underlying observation that, when given a large number of short vectors, one can find a relatively small basis is the same as the observation we use when finding redundant polynomial equalities. The challenge in both cases lies in actually finding a suitable representation of (part of) the problem.

To conclude, the use of linear-algebraic techniques in kernelization research is increasing and this thesis contributes another useful technique.

7.2 Sparsification

The idea of making a graph or logic problem less dense by reducing the number of edges or clauses compared to the number of vertices or variables, has been widely studied in several settings.

A celebrated result considering sparsification for logic problems is the well-known sparsification lemma by Impagliazzo *et al.* [53], which shows how a d-CNF formula can be rewritten in subexponential time as a subexponential-size disjunction of sparse d-CNF formulas. The lemma has important applications in the study of subexponential-time algorithms [52, 90].

Another research direction is the efficient sparsification of graphs, such that the sparsified graph is close to the original graph by some metric. This generally involves reducing the original (possibly unweighted) graph to a weighted graph. The goal is now that the new (weighted) graph has similar properties to the original one, and can for example be used to obtain approximation results. A well-known example is the sparsification that approximately preserves all cuts in the graph, and for which the reduced instance has only $\mathcal{O}(n \log n)$ edges [10]. Later work improved running times in case the input graph is weighted, and helped to better understand which sparsification procedures maintain the desired cut weights [42]. This type of graph sparsification has been further generalized to preserve even more properties of the original graph [85].

This way of sparsifying graphs can usually be done efficiently; the one given above for preserving cuts even runs in linear time. However, the guarantee is not as strong as for the type of sparsification we considered in this thesis, as solutions to the original problem are only approximately preserved. The bound on the sparsification size depends on the quality of the approximation one wants to achieve.

Polynomial-time sparsification where the sparsified instance is required to have exactly the same yes/no-answer as the original one, was not widely studied before this thesis. One of the earliest results is by Dell and Van Melkebeek [33], showing that d-CNF-SAT and VERTEX COVER do not allow for a non-trivial sparsification. This was followed-up by several impossibility results regarding the sparsification of several classic graph problems [59]. Surprisingly, unlike d-CNF-SAT, the d-NAE-SAT problem did turn out to have a non-trivial sparsification [59].

In this thesis, we further studied this polynomial-time sparsification and showed that H-COLORING is unlikely to have a non-trivial sparsification for a large number of possible choices for H. This adds another problem to the list of graph problems that do not admit non-trivial sparsification [56, 59].

Finally, we studied the sparsification of a large number of logic problems, using the framework of constraint satisfaction problems. We gave a sufficient condition under which $\mathrm{CSP}(\Gamma)$ has a sparsification with $\mathcal{O}(n^d)$ constraints, and showed that this unified known kernelization results for d-NAE-SAT, d-EXACT SAT, and d-CNF-SAT. Furthermore, we fully classified which Boolean CSPs admit non-trivial sparsification.

7.3 Future work

In this thesis we studied the sparsification of constraint satisfaction problems. While we obtained many provably tight sparsification results, numerous open questions remain. The ultimate goal of this line of research would be to fully classify the sparsifiability of (Boolean) $\mathrm{CSP}(\Gamma)$, depending on Γ.

Open problem *Given a finite (Boolean) constraint language Γ, what is the smallest possible kernel for $\mathrm{CSP}(\Gamma)$ parameterized by the number of variables n?*

A first step in this direction could be to further study the sparsifiability of symmetric Boolean CSPs. In this thesis, we have seen a full classification of those symmetric Boolean CSPs that have a sparsification with a linear number of constraints. We also already know for which Boolean CSPs non-trivial sparsification is impossible. Studying the symmetric CSPs whose sparsifiability falls in between these two extremes, could be a next step in obtaining a fine-grained view on the complexity of $\mathrm{CSP}(\Gamma)$ for different Γ. A natural first question would be to classify which symmetric Boolean CSPs allow for a sparsification with quadratically many constraints. Further generalizing this, we would like to know which $\mathrm{CSP}(\Gamma)$ have a sparsification with $\mathcal{O}(n^d)$ constraints for any given value for d.

Open problem *Classify all symmetric Boolean CSPs having a kernel with $\mathcal{O}(n^d)$ constraints, but no generalized kernel of size $\mathcal{O}(n^{d-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.*

We fully classified all Boolean constraint languages for which $\mathrm{CSP}(\Gamma)$ has a non-trivial sparsification. However, this result strongly depends on Γ being Boolean. Suppose d is the arity of the largest-arity relation in Γ and $\mathrm{CSP}(\Gamma)$ is NP-hard. If Γ is Boolean, verifying whether $\mathrm{CSP}(\Gamma)$ has a non-trivial sparsification is as simple as checking whether there is a relation in Γ that contains exactly $2^d - 1$ tuples: if not, then non-trivial sparsification is possible (Theorem 4.14). This simple idea is not effective when Γ it not guaranteed to be Boolean. In this case, a relation with exactly $2^d - 1$ tuples can sometimes be used to give a lower bound (as in the case where Γ is Boolean), but not always: Consider the arity-2 relation $R = \{(0,0),(1,2),(2,1)\}$ over $D = \{0,1,2\}$. Then $(x_1, x_2) \in R$ if and only if $x_1 + x_2 \equiv_3 0$, allowing for a sparsification with only linearly many constraints (assuming the same holds for the remaining relations in Γ). We thus ask the following question.

Open problem *For which (non-Boolean) finite constraint languages* Γ *where each relation in* Γ *has arity at most* d*, does* $\mathrm{CSP}(\Gamma)$ *admit a kernel with* $\mathcal{O}(n^{d-\delta})$ *constraints for some* $\delta > 0$*?*

Finally, we restate the open question that remains from Chapter 5. While it has been partially resolved, in particular in the cases where H contains a triangle, and where H has odd girth α and no edge lies in two C_α-subgraphs, other cases remain open.

Open problem *Prove or disprove that for all simple nonbipartite graphs* H*, the* H-Coloring *problem does not have a generalized kernel of size* $\mathcal{O}(n^{2-\varepsilon})$ *for any* $\varepsilon > 0$*, unless* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

If it turns out that the H-Coloring problem does not allow for a non-trivial sparsification in all NP-hard cases, this raises the question whether there are natural graph problems, defined on general graphs, that *do* have a sparsification with a subquadratic number of edges.

Bibliography

[1] Faisal N. Abu-Khzam, Michael R. Fellows, Michael A. Langston, and W. Henry Suters. Crown structures for vertex cover kernelization. *Theory of Computing Systems*, 41(3):411–430, 2007. doi:10.1007/s00224-007-1328-0. (Cited on page 159.)

[2] Tobias Achterberg. SCIP: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009. doi:10.1007/s12532-008-0001-1. (Cited on page 2.)

[3] Tobias Achterberg and Roland Wunderling. *Mixed Integer Programming: Analyzing 12 Years of Progress*, pages 449–481. Springer Berlin Heidelberg, 2013. doi:10.1007/978-3-642-38189-8_18. (Cited on page 2.)

[4] Akanksha Agrawal, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. Interval vertex deletion admits a polynomial kernel. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 1711–1730, 2019. doi:10.1137/1.9781611975482.103. (Cited on page 160.)

[5] Erling D. Andersen and Knud D. Andersen. Presolving in linear programming. *Mathematical Programming*, 71(2):221–245, 1995. doi:10.1007/BF01586000. (Cited on page 2.)

[6] Nicolas Barnier and Pascal Brisset. Graph coloring for air traffic flow management. *Annals of Operations Research*, 130(1):163–178, 2004. doi:10.1023/B:ANOR.0000032574.01332.98. (Cited on page 8.)

[7] David A. Mix Barrington. Some problems involving Razborov-Smolensky polynomials. In *Boolean Function Complexity*, pages 109–128. Cambridge University Press, 1992. doi:10.1017/CBO9780511526633.010. (Cited on pages 37 and 59.)

[8] David A. Mix Barrington, Richard Beigel, and Steven Rudich. Representing Boolean functions as polynomials modulo composite numbers. *Computational Complexity*, 4(4):367–382, 1994. doi:10.1007/BF01263424. (Cited on page 60.)

[9] Richard Beigel. The polynomial method in circuit complexity. In *Proceedings of the 8th Annual Structure in Complexity Theory Conference (CCC 1993)*, pages 82–95, 1993. doi:10.1109/SCT.1993.336538. (Cited on page 61.)

[10] András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in $\widetilde{O}(n^2)$ time. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC 1996)*, pages 47–55, 1996. doi:10.1145/237814.237827. (Cited on page 161.)

[11] Abhishek Bhowmick and Shachar Lovett. Nonclassical polynomials as a barrier to polynomial lower bounds. In *Proceedings of the 30th Conference on Computational Complexity (CCC 2015)*, volume 33, pages 72–87, 2015. doi:10.4230/LIPIcs.CCC.2015.72. (Cited on pages 37, 60, and 62.)

[12] Robert E. Bixby and Edward Rothberg. Progress in computational mixed integer programming - A look back from the other side of the tipping point. *Annals of Operations Research*, 149(1):37–41, 2007. doi:10.1007/s10479-006-0091-y. (Cited on page 2.)

[13] Hans L. Bodlaender. Kernelization: New upper and lower bound techniques. In *Proceedings of the 4th International Workshop on Parameterized and Exact Computation, (IWPEC 2009)*, volume 5917, pages 17–37, 2009. doi:10.1007/978-3-642-11269-0_2. (Cited on page 16.)

[14] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009. doi:10.1016/j.jcss.2009.04.001. (Cited on page 5.)

[15] Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (Meta) Kernelization. *Journal of the ACM*, 63(5):44:1–44:69, 2016. doi:10.1145/2973749. (Cited on page 4.)

[16] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM Journal on Discrete Mathematics*, 28(1):277–305, 2014. doi:10.1137/120880240. (Cited on pages 5, 19, 21, 22, and 100.)

[17] Hans L. Bodlaender, Sudeshna Kolay, and Astrid Pieterse. Parameterized complexity of conflict-free graph coloring. In *Proceedings of the 16th*

Algorithms and Data Structures Symposium (WADS 2019), 2019. To appear. URL: https://arxiv.org/abs/1905.00305. (Cited on pages 11, 158, and 160.)

[18] Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theoretical Computer Science*, 412(35):4570–4578, 2011. doi:10.1016/j.tcs.2011.04.039. (Cited on pages 4, 5, 19, and 100.)

[19] Hans L. Bodlaender and Thomas C. van Dijk. A cubic kernel for feedback vertex set and loop cutset. *Theory of Computing Systems*, 46(3):566–597, 2010. doi:10.1007/s00224-009-9234-2. (Cited on page 159.)

[20] V. G. Bodnarčuk, L. A. Kalužnin, V. N. Kotov, and B. A. Romov. Galois theory for post algebras, part I and II. *Cybernetics*, 5:243–252, 531–539, 1969. doi:10.1007/BF01070906. (Cited on page 32.)

[21] Andrei A. Bulatov. H-coloring dichotomy revisited. *Theoretical Computer Science*, 349(1):31–39, 2005. doi:10.1016/j.tcs.2005.09.028. (Cited on pages 116, 119, 121, 125, 126, and 129.)

[22] Andrei A. Bulatov. A dichotomy theorem for nonuniform CSPs. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2017)*, pages 319–330, 2017. doi:10.1109/FOCS.2017.37. (Cited on page 9.)

[23] Andrei A. Bulatov, Peter Jeavons, and Andrei A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005. doi:10.1137/S0097539700376676. (Cited on page 33.)

[24] Rafael G. Cano, Kevin Buchin, Thom Castermans, Astrid Pieterse, Willem Sonke, and Bettina Speckmann. Mosaic drawings and cartograms. *Computer Graphics Forum*, 34(3):361–370, 2015. doi:10.1111/cgf.12648. (Cited on page 11.)

[25] Hubie Chen. A rendezvous of logic, complexity, and algebra. *ACM Computing Surveys*, 42(1), 2009. doi:10.1145/1189056.1189076. (Cited on pages 31 and 32.)

[26] Hubie Chen, Bart M. P. Jansen, and Astrid Pieterse. Sparsification lower bounds for H-coloring. Submitted. (Cited on page 11.)

[27] Hubie Chen, Bart M. P. Jansen, and Astrid Pieterse. Best-case and worst-case sparsifiability of Boolean CSPs. In *Proceedings of the 13th International Symposium on Parameterized and Exact Computation (IPEC 2018)*, volume 115, pages 15:1–15:13, 2019. doi:10.4230/LIPIcs.IPEC.2018.15. (Cited on page 10.)

[28] Zhi-Zhong Chen, Michael R. Fellows, Bin Fu, Haitao Jiang, Yang Liu, Lusheng Wang, and Binhai Zhu. A linear kernel for co-path/cycle packing. In *Proceedings of the 6th International Conference on Algorithmic Applications in Management (AAIM 2010)*, volume 6124, pages 90–102, 2010. doi:10.1007/978-3-642-14355-7_10. (Cited on page 159.)

[29] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2015. (Cited on page 4.)

[30] Holger Dell, Thore Husfeldt, Bart M. P. Jansen, Petteri Kaski, Christian Komusiewicz, and Frances A. Rosamond. The first parameterized algorithms and computational experiments challenge. In *Proceedings of the 11th International Symposium on Parameterized and Exact Computation (IPEC 2016)*, volume 63, pages 30:1–30:9, 2017. doi:10.4230/LIPIcs.IPEC.2016.30. (Cited on page 2.)

[31] Holger Dell, Christian Komusiewicz, Nimrod Talmon, and Mathias Weller. The PACE 2017 parameterized algorithms and computational experiments challenge: The second iteration. In *Proceedings of the 12th International Symposium on Parameterized and Exact Computation (IPEC 2017)*, volume 89, pages 30:1–30:12, 2018. doi:10.4230/LIPIcs.IPEC.2017.30. (Cited on page 2.)

[32] Holger Dell and Dániel Marx. Kernelization of packing problems. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2012)*, pages 68–81, 2012. doi:10.1137/1.9781611973099.6. (Cited on page 22.)

[33] Holger Dell and Dieter van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. *Journal of the ACM*, 61(4):23:1–23:27, 2014. doi:10.1145/2629620. (Cited on pages 5, 7, 19, 20, 59, 64, 70, 103, and 161.)

[34] Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Kernelization lower bounds through colors and IDs. *ACM Transactions on Algorithms*, 11(2):13:1–13:20, 2014. doi:10.1145/2650261. (Cited on page 46.)

[35] Andrew Drucker. New limits to classical and quantum instance compression. *SIAM Journal on Computing*, 44(5):1443–1479, 2015. doi:10.1137/130927115. (Cited on pages 24, 70, and 71.)

[36] Louis Esperet, Mickaël Montassier, Pascal Ochem, and Alexandre Pinlou. A complexity dichotomy for the coloring of sparse graphs. *Journal of Graph Theory*, 73(1):85–102, 2013. doi:10.1002/jgt.21659. (Cited on page 104.)

[37] Michael R. Fellows. Blow-ups, win/win's, and crown rules: Some new directions in FPT. In *Proceedings of the 29th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2003)*, volume 2880, pages 1–12, 2003. doi:10.1007/978-3-540-39890-5_1. (Cited on page 159.)

[38] Jiří Fiala, Petr A. Golovach, and Jan Kratochvíl. Parameterized complexity of coloring problems: Treewidth versus vertex cover. *Theoretical Computer Science*, 412(23):2513 – 2523, 2011. doi:10.1016/j.tcs.2010.10.043. (Cited on page 139.)

[39] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar F-deletion: Approximation, kernelization and optimal FPT algorithms. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2012)*, pages 470–479, 2012. doi:10.1109/FOCS.2012.62. (Cited on page 4.)

[40] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019. (Cited on pages 4 and 159.)

[41] Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *Journal of Computer and System Sciences*, 77(1):91–106, 2011. doi:10.1016/j.jcss.2010.06.007. (Cited on pages 5 and 59.)

[42] Wai Shing Fung, Ramesh Hariharan, Nicholas J. A. Harvey, and Debmalya Panigrahi. A general framework for graph sparsification. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC 2011)*, pages 71–80, 2011. doi:10.1145/1993636.1993647. (Cited on page 161.)

[43] Anna Galluccio, Pavol Hell, and Jařošlav Nešetřil. The complexity of H-colouring of bounded degree graphs. *Discrete Mathematics*, 222(1-3):101–109, 2000. doi:10.1016/S0012-365X(00)00009-1. (Cited on page 104.)

[44] Robert Ganian. Improving vertex cover as a graph parameter. *Discrete Mathematics & Theoretical Computer Science*, 17(2):77–100, 2015. URL: http://dmtcs.episciences.org/2136. (Cited on pages 140 and 144.)

[45] Michael R. Garey, David S. Johnson, and H. C. So. An application of graph coloring to printed circuit testing (working paper). In *Proceedings of the 16th Annual Symposium on Foundations of Computer Science (FOCS 1975)*, pages 178–183, 1975. doi:10.1109/SFCS.1975.3. (Cited on page 8.)

[46] Michael R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Compututer Science*, 1(3):237–267, 1976. doi:10.1016/0304-3975(76)90059-1. (Cited on page 6.)

[47] David Geiger. Closed systems of functions and predicates. *Pacific Journal of Mathematics*, 27:95–100, 1968. (Cited on page 32.)

[48] Mark S. Gockenbach. *Finite-Dimensional Linear Algebra*. Discrete Mathematics and Its Applications. Taylor & Francis, 2011. (Cited on page 25.)

[49] Pavol Hell and Jaroslav Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory, Series B*, 48(1):92–110, 1990. doi:10.1016/0095-8956(90)90132-J. (Cited on page 104.)

[50] Leslie Hogben. *Handbook of Linear Algebra, Second Edition*. Chapman and Hall/CRC, 2014. (Cited on page 39.)

[51] John A. Howell. Spans in the module $(\mathbb{Z}_m)^s$. *Linear and Multilinear Algebra*, 19(1):67–77, 1986. doi:10.1080/03081088608817705. (Cited on page 26.)

[52] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727. (Cited on page 161.)

[53] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774. (Cited on page 161.)

[54] Yoichi Iwata. Linear-time kernelization for feedback vertex set. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80, pages 68:1–68:14, 2017. doi:10.4230/LIPIcs.ICALP.2017.68. (Cited on page 2.)

[55] Lars Jaffke and Bart M. P. Jansen. Fine-grained parameterized complexity analysis of graph coloring problems. In *Proceedings of the 10th International Conference on Algorithms and Complexity (CIAC 2017)*, volume 10236, pages 345–356, 2017. doi:10.1007/978-3-319-57586-5_29. (Cited on page 105.)

[56] Bart M. P. Jansen. On sparsification for computing treewidth. *Algorithmica*, 71(3):605–635, 2015. doi:10.1007/s00453-014-9924-2. (Cited on pages 7, 103, and 161.)

[57] Bart M. P. Jansen and Stefan Kratsch. Data reduction for graph coloring problems. *Information and Computation*, 231:70–88, 2013. doi:10.1016/j.ic.2013.08.005. (Cited on pages 139, 140, 141, and 157.)

[58] Bart M. P. Jansen and Astrid Pieterse. Optimal sparsification for some binary CSPs using low-degree polynomials. In *Proceedings of the 41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*, volume 58, pages 71:1–71:14, 2016. doi:10.4230/LIPIcs.MFCS.2016.71. (Cited on page 11.)

[59] Bart M. P. Jansen and Astrid Pieterse. Sparsification upper and lower bounds for graph problems and not-all-equal SAT. *Algorithmica*, 79(1):3–28, 2017. doi:10.1007/s00453-016-0189-9. (Cited on pages 7, 11, 41, 103, and 161.)

[60] Bart M. P. Jansen and Astrid Pieterse. Optimal data reduction for graph coloring using low-degree polynomials. In *Proceedings of the 12th International Symposium on Parameterized and Exact Computation (IPEC 2017)*, volume 89, pages 22:1–22:12, 2018. doi:10.4230/LIPIcs.IPEC.2017.22. (Cited on page 103.)

[61] Bart M. P. Jansen and Astrid Pieterse. Polynomial kernels for hitting forbidden minors under structural parameterizations. In *Proceedings of the 26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112, pages 48:1–48:15, 2018. doi:10.4230/LIPIcs.ESA.2018.48. (Cited on page 11.)

[62] Bart M. P. Jansen and Astrid Pieterse. Optimal data reduction for graph coloring using low-degree polynomials. *Algorithmica*, 2019. Online First. doi:10.1007/s00453-019-00578-5. (Cited on page 11.)

[63] Tommy R. Jensen and Bjarne Toft. *Graph Coloring Problems*, volume 39. John Wiley & Sons, 2011. (Cited on page 8.)

[64] Ravindran Kannan and Achim Bachem. Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix. *SIAM Journal on Computing*, 8(4):499–507, 1979. doi:10.1137/0208040. (Cited on page 26.)

[65] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. 1972. doi:10.1007/978-1-4684-2001-2_9. (Cited on pages 20 and 46.)

[66] Stefan Kratsch. A randomized polynomial kernelization for vertex cover with a smaller parameter. *SIAM Journal on Discrete Mathematics*, 32(3):1806–1839, 2018. doi:10.1137/16M1104585. (Cited on page 160.)

[67] Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2012)*, pages 450–459, 2012. doi:10.1109/FOCS.2012.46. (Cited on pages 4 and 160.)

[68] Stefan Kratsch and Magnus Wahlström. Two edge modification problems without polynomial kernels. *Discrete Optimization*, 10(3):193–199, 2013. doi:10.1016/j.disopt.2013.02.001. (Cited on page 5.)

[69] Stefan Kratsch and Magnus Wahlström. Compression via matroids: A randomized polynomial kernel for odd cycle transversal. *ACM Transactions on Algorithms*, 10(4):20:1–20:15, 2014. doi:10.1145/2635810. (Cited on page 160.)

[70] Victor Lagerkvist and Magnus Wahlström. Kernelization of constraint satisfaction problems: A study through universal algebra. In *Proceedings of the 23rd International Conference on Principles and Practice of Constraint Programming (CP 2017)*, volume 10416, pages 157–171, 2017. doi:10.1007/978-3-319-66158-2_11. (Cited on pages 64, 70, 71, 91, 92, 94, 99, and 103.)

[71] Victor Lagerkvist and Magnus Wahlström. Kernelization of constraint satisfaction problems: A study through universal algebra. *CoRR*, abs/1706.05941, 2017. arXiv:1706.05941. (Cited on pages 92, 94, and 100.)

[72] Gregory T. Lee. *Abstract Algebra An Introductory Course*. Springer, Cham, 2018. doi:10.1007/978-3-319-77649-1. (Cited on page 24.)

[73] R. M. R. Lewis. *A Guide to Graph Colouring*. Springer International Publishing Switzerland, 2016. doi:10.1007/978-3-319-25730-3. (Cited on page 8.)

[74] Daniel Lokshtanov and Christian Sloper. Fixed parameter set splitting, linear kernel and improved running time. In *Proceedings of the 1st workshop on Algorithms and Complexity in Durham (ACiD 2005)*, pages 105–113, 2005. URL: http://bora.uib.no/handle/1956/1115. (Cited on page 159.)

[75] László Lovász. Chromatic number of hypergraphs and linear algebra. In *Studia Scientiarum Mathematicarum Hungarica 11*, pages 113–114, 1976. URL: http://real-j.mtak.hu/5461/. (Cited on page 41.)

[76] Gary MacGillivray and Mark H. Siggers. On the complexity of H-colouring planar graphs. *Discrete Mathematics*, 309(18):5729–5738, 2009. doi:10.1016/j.disc.2008.05.016. (Cited on page 104.)

[77] Dániel Marx. Graph colouring problems and their applications in scheduling. *Periodica Polytechnica Electrical Engineering (Archives)*, 48(1-2):11–16. URL: https://pp.bme.hu/ee/article/view/926. (Cited on page 8.)

[78] Dániel Marx. A parameterized view on matroid optimization problems. *Theoretical Computer Science*, 410(44):4471–4479, 2009. doi:10.1016/j.tcs.2009.07.027. (Cited on page 160.)

[79] Hermann A. Maurer, Ivan Hal Sudborough, and Emo Welzl. On the complexity of the general coloring problem. *Information and Control*, 51(2):128–145, 1981. doi:10.1016/S0019-9958(81)90226-6. (Cited on page 105.)

[80] Hannes Moser. A problem kernelization for graph packing. In *Proceedings of the 35th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2009)*, pages 401–412, 2009. doi:10.1007/978-3-540-95891-8_37. (Cited on page 159.)

[81] Astrid Pieterse. Sparsification upper and lower bounds for graph problems and not-all-equal SAT. Master's thesis, Eindhoven University of Technology, 2015. URL: https://research.tue.nl/en/studentTheses/sparsification-upper-and-lower-bounds-for-graph-problems-and-not-. (Cited on page 23.)

[82] Astrid Pieterse and Gerhard J. Woeginger. The subset sum game revisited. In *Proceedings of the 5th International Conference on Algorithmic Decision Theory (ADT 2017)*, volume 10576, pages 228–240, 2017. doi:10.1007/978-3-319-67504-6_16. (Cited on page 11.)

[83] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th ACM Symposium on Theory of Computing (STOC 1978)*, pages 216–226, 1978. doi:10.1145/800133.804350. (Cited on pages 9 and 31.)

[84] Mark H. Siggers. Dichotomy for bounded degree H-colouring. *Discrete Applied Mathematics*, 157(2):201–210, 2009. doi:10.1016/j.dam.2008.02.003. (Cited on page 104.)

[85] Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011. doi:10.1137/080734029. (Cited on page 161.)

[86] Jeremy P. Spinrad. *Efficient Graph Representations*. American Mathematical Society, 2003. (Cited on page 145.)

[87] Arne Storjohann and Thom Mulders. *Fast Algorithms for Linear Algebra Modulo N*, pages 139–150. Springer Berlin Heidelberg, 1998. doi:10.1007/3-540-68530-8_12. (Cited on pages 26 and 27.)

[88] Gábor Tardos and David A. Mix Barrington. A lower bound on the mod 6 degree of the OR function. *Computational Complexity*, 7(2):99–108, 1998. doi:10.1007/PL00001597. (Cited on page 62.)

[89] Hassler Whitney. On the abstract properties of linear dependence. *American Journal of Mathematics*, 57(3):509–533, 1935. doi:10.2307/2371182. (Cited on page 160.)

[90] Gerhard J. Woeginger. Exact algorithms for NP-hard problems: A survey. In *Proceedings of the 5th Aussois Combinatorial Optimization Workshop - Eureka, You Shrink!, Papers Dedicated to Jack Edmonds*, volume 2570, pages 185–208, 2001. doi:10.1007/3-540-36478-1_17. (Cited on page 161.)

[91] Chee-Keng Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, 26:287–300, 1983. doi:10.1016/0304-3975(83)90020-8. (Cited on page 18.)

[92] Dmitriy Zhuk. A proof of CSP dichotomy conjecture. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2017)*, pages 331–342, 2017. doi:10.1109/FOCS.2017.38. (Cited on page 9.)

Summary

Tight Parameterized Preprocessing Bounds: Sparsification via Low-Degree Polynomials

Many of the problems we are interested in solving algorithmically are known to be NP-hard. As such, we do not expect to find efficient (polynomial-time) algorithms to solve these problems. However, we are still interested in finding effective ways to (at least) be able to solve certain input instances for such problems. One of the ways to do this, is to first try and preprocess the input instance, without changing the solution. Then, a more expensive algorithm to actually solve the problem can be applied to the (now hopefully easier) preprocessed instance. In this thesis, we will study the power of such preprocessing algorithms. In particular, we will be interested in polynomial-time preprocessing, where the preprocessing algorithm is required to run in polynomial time.

As the problems we consider are NP-hard, we cannot hope that such a preprocessing algorithm will always be guaranteed to decrease the actual size of the input instance. After all, such a preprocessing algorithm could then be used repeatedly to obtain a polynomial-time algorithm that solves the problem, which is unlikely to exist. For this reason, we measure the effectiveness of our preprocessing algorithms by some additional parameter, other than the input size, that measures the complexity of the input. As such, input instances are often denoted as (x, k) where k is the additional parameter.

The type of preprocessing algorithms we study in this thesis are called kernelization algorithms (or kernels). A kernelization algorithm is a polynomial-time algorithm that, given an instance (x, k), outputs an instance (x', k') of the same problem such that both $|x'|$ and k' are bounded by $f(k)$ for some function that may only depend on k. This function is also called the size of the kernel. Naturally, smaller kernels are better. As such, we are interested in what the smallest kernel is that can be obtained for a certain parameterized problem.

In this thesis we obtain several tight upper and lower bound results for such kernelization algorithms. First of all, we study the (very general) CONSTRAINT

SATISFACTION PROBLEM (CSP). Broadly speaking, an input instance for this problem consists of a number of constraints over a set of variables, and the question is whether we can assign values to the variables, such that all constraints are satisfied. We will be particularly interested in Boolean CSPs, where the question is to assign 0 or 1 to each variable such that all constraints are satisfied.

This thesis investigates how the type of constraints that are used, influences the kernelizability of the problem. We give a general method, based on representing constraints by low-degree polynomials, to obtain kernels for CSPs. Furthermore, we show that this method is best-possible in some cases: there are CSPs for which no better kernelization is possible (under commonly used complexity-theoretic assumptions).

As part of our investigation into the kernelization of CSPs, we use universal algebra to describe CSPs for which any n-variable input can be reduced to an input that only has $\mathcal{O}(n)$ constraints. In fact, we show that for these CSPs there is always a subset of the original constraints of size at most $\mathcal{O}(n)$ that directly implies all other constraints. Using this method, we fully classify the symmetric Boolean CSPs that have a kernel with $\mathcal{O}(n)$ constraints.

Furthermore, when considering CSPs over the Boolean domain in which every constraint considers at most d variables, we obtain a dichotomy result. When the constraints can express a size-d logical OR in a certain sense, the best-possible kernel has size $\Theta(n^d)$ (assuming $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$) where n is the number of variables. In all other cases, there is a kernel with at most $\mathcal{O}(n^{d-1})$ constraints and size $\mathcal{O}(n^{d-1} \log n)$.

In addition to the above, we completely classify the kernelizability of Boolean CSPs when every constraint concerns at most three variables. Under standard complexity-theoretic assumptions, it turns out that there is a fairly simple classification into four cases. Either the CSP is in P, or the best-possible kernel has $\mathcal{O}(n)$ constraints, or the best-possible kernel has $\mathcal{O}(n^2)$ constraints, or no better kernel than the trivial one with $\mathcal{O}(n^3)$ constraints exists.

The second problem we study is the H-COLORING problem, which is defined for any fixed graph H. Given a graph G, the problem asks whether there exists a mapping $f\colon V(G) \to V(H)$, such that for any edge $\{u,v\} \in E(G)$, we have $\{f(u),f(v)\} \in E(H)$. Since this is a graph problem, and any n-vertex graph can be stored in $\mathcal{O}(n^2)$ bits, it is easy to see that H-COLORING parameterized by the number of vertices has a kernel of size $\mathcal{O}(n^2)$. In this thesis we show that under some additional restrictions on H, there is no kernel of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

Finally, we study the q-COLORING problem on structurally simple graphs. Given a graph G, the question is whether it is possible to color its vertices with at most q colors, such that any two vertices connected by an edge receive different colors. We obtain a kernel of size $\mathcal{O}(k^{q-1} \log k)$ for the q-COLORING problem when parameterized by the size of a vertex cover. This gives a kernel whose size

is optimal up to polylogarithmic factors under complexity-theoretic assumptions. It improves upon the previously best-known kernel that had size $\mathcal{O}(k^q)$. To obtain this kernel, we apply the methods developed for the kernelization of CSPs in this different setting. We further generalize this result to the H-Coloring problem, parameterized by twin-cover, which is a smaller parameter than vertex cover. We obtain a kernel of size $\mathcal{O}(k^{\Delta(H)} \log k)$, where $\Delta(H)$ is the maximum degree of any vertex in H.

Curriculum Vitae

Astrid Pieterse was born on April 19, 1993 in Weert, The Netherlands. She completed her secondary education at Philips van Horne s.g. (Weert, The Netherlands) in 2010. She received a bachelor in Computer Science and Engineering (Cum Laude) and a bachelor in Industrial and Applied Mathematics (Cum Laude) from Eindhoven University of Technology in 2013. In 2015 she obtained a master's degree in Computer Science and Engineering at Eindhoven University of Technology, which was awarded Cum Laude. She has been a PhD student at Eindhoven University of Technology since September 2015, under the supervision of dr. Bart M. P. Jansen. During her PhD she was awarded the best paper award at the MFCS conference in 2016, and the excellent student paper award at IPEC 2017.