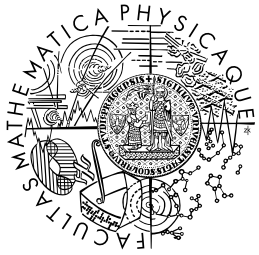

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

DOCTORAL THESIS

Michal Opler

Structural and Algorithmic Properties of Permutation Classes

Computer Science Institute of Charles University

Supervisor of the doctoral thesis: doc. RNDr. Vít Jelínek, Ph.D.

Study programme: Computer Science

Study branch: Theory of Computing, Discrete Models and Optimization

Prague 2022

I declare that I carried out this doctoral thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

I would like to thank my supervisor Vít Jelínek for guiding me throughout my whole doctoral studies. His patience and invaluable keen eye for detail have significantly improved the quality of my writing. I would also like to thank my coauthors, collaborators and fellow doctoral students for all the inspiration and stimulating discussions. I was lucky to meet many interesting people and even more so that I can call some of them my friends. I especially enjoyed discussing anything and everything with Miloš Chromý.

Finally, I would like to thank my family for their endless support and love. My grandfather, in particular, has been my staunchest supporter. Most of all, I thank my beloved wife Kristýnka for inspiring me and for always believing in me, even when I don't believe in myself.

Title: Structural and Algorithmic Properties of Permutation Classes

Author: Michal Opler

Institute: Computer Science Institute of Charles University

Supervisor: doc. RNDr. Vít Jelínek, Ph.D., Computer Science Institute of Charles University

Abstract: In this thesis, we study the relationship between the structure of permutation classes and the computational complexity of different decision problems. First, we explore the structure of permutation classes through the lens of various parameters, with a particular interest in tree-width. We define novel structural properties of a general permutation class $\mathcal{C}$, the most notable being the long path property. Using these properties, we infer lower bounds on the maximum tree-width attained by a permutation of length n in $\mathcal{C}$. For example, we prove that any class with the long path property has unbounded tree-width.

The main decision problem we consider is known as Permutation Pattern Matching (PPM). The input of PPM consists of a pair of permutations τ (the 'text') and π (the 'pattern'), and the goal is to decide whether τ contains π as a subpermutation. After briefly considering general PPM, we focus on its pattern-restricted variant known as $\mathcal{C}$-Pattern PPM where we additionally require that the pattern π comes from a fixed class $\mathcal{C}$. We derive both classical and parameterized hardness results assuming different structural properties of $\mathcal{C}$. For example, we show that $\mathcal{C}$-Pattern PPM is NP-complete whenever $\mathcal{C}$ has the long path property.

Furthermore, we focus on an even more restricted variant of PPM where the text is also assumed to come from a fixed class $\mathcal{C}$; this restriction is known as $\mathcal{C}$-PPM. We present a new hardness reduction which allows us to show, in a uniform way, that $\mathrm{Av}(\sigma)$-PPM, where $\mathrm{Av}(\sigma)$ is the class of σ-avoiding permutations, is NP-complete for any σ of length at least four that is not symmetric to one of 3412, 3142, 4213, 4123 or 41352.

Permutations can also be viewed as models over a signature consisting of two binary relation symbols. We investigate the expressive power of monadic second-order (MSO) logic and the complexity of MSO model checking in this setting. Among other results, we show that MSO model checking is hard not just for general permutations, but also within any class with the long path property.

Finally, we consider, for fixed permutation classes $\mathcal{C}$ and $\mathcal{D}$, the complexity of determining whether a given permutation π can be colored red and blue so that the red elements induce a permutation from $\mathcal{C}$ and the blue ones a permutation from $\mathcal{D}$; this problem is known as generalized coloring. As a consequence of more general results, we can provide nontrivial instances of both tractable and intractable generalized coloring involving commonly studied permutation classes.

Keywords: permutations, pattern matching, monadic second-order logic, grid classes, generalized coloring

Contents

Introduction **5**

1 Preliminaries **11**
1.1 Permutations . . . 11
1.1.1 Symmetries of permutations . . . 12
1.1.2 Basic operations acting on permutations . . . 13
1.1.3 Classical permutation classes . . . 14
1.2 Generalized permutation patterns . . . 15
1.2.1 Vincular patterns . . . 15
1.2.2 Bivincular patterns . . . 16
1.2.3 Mesh patterns . . . 17
1.2.4 Partially ordered patterns. . . . 18
1.2.5 Other notions of patterns . . . 18
1.3 Grid classes . . . 19
1.3.1 Building gridded permutations . . . 20
1.3.2 Geometric grid classes . . . 22

2 Structural properties **23**
2.1 Width parameters . . . 23
2.1.1 Twin-width . . . 23
2.1.2 Tree-width . . . 25
2.1.3 Grid-width . . . 28
2.1.4 Clique-width . . . 31
2.1.5 Horizontal and vertical grid-width . . . 38
2.1.6 Modular-width . . . 39
2.2 Containment of grid subclasses . . . 42
2.2.1 Long path property . . . 43
2.2.2 Cycle property . . . 45
2.2.3 Deep tree property . . . 48
2.2.4 Bicycle property . . . 51
2.3 Principal classes . . . 59
2.3.1 Classes without the deep tree property . . . 59
2.3.2 Classes with the bicycle property . . . 63

3 Logic of permutations **67**
3.1 First-order logic . . . 67
3.1.1 Expressibility of first-order logic . . . 68
3.1.2 First-order model checking . . . 69
3.2 Monadic second-order logic . . . 71
3.2.1 Properties expressible in MSO . . . 72
3.2.2 Properties inexpressible in MSO . . . 78
3.2.3 Monadic second-order model checking . . . 79

4 Permutation pattern matching **89**
4.1 Classical complexity . . . 89
4.2 Parameterized complexity . . . 94
4.2.1 Left-aligned patterns . . . 96
4.2.2 Pattern parameters . . . 99
4.2.3 Counting patterns . . . 100
4.2.4 Generalized patterns . . . 100
4.2.5 Patterns as CSPs . . . 101
4.3 Pattern from a given permutation class . . . 103
4.3.1 Classical complexity . . . 103
4.3.2 Parameterized complexity . . . 111

5 Pattern matching inside a fixed permutation class **123**
5.1 Current state of the art . . . 123
5.2 Monotone-griddable classes . . . 124
5.3 Hardness results . . . 125
5.3.1 Overview of the proof of Theorem 5.6 . . . 126
5.3.2 Details of the hardness reduction . . . 130
5.3.3 Principal classes . . . 142

6 Grid classes **145**
6.1 Monotone grid classes . . . 145
6.1.1 Tree-width . . . 145
6.1.2 Pattern matching . . . 151
6.2 General grid classes . . . 152
6.2.1 Classes with bumper-ended paths . . . 152
6.2.2 Classes without bumper-ended paths . . . 153

7 Generalized coloring **161**
7.1 Previous results . . . 161
7.2 Different recognition frameworks . . . 162
7.2.1 NLOL-recognizable classes . . . 162
7.2.2 BD-recognizable classes . . . 165
7.2.3 GT-recognizable classes . . . 168
7.3 Combining BD- with GT-recognizability . . . 171
7.4 Hardness results . . . 175

Conclusion **183**

Bibliography **185**

List of publications **195**

Funding. This research was supported by the GAUK project 1766318, by project Impuls of the Neuron Fund for Support of Science, and by projects 18-19158S and 21-32817S of the Czech Science Foundation.

Introduction

Permutations can be represented as combinatorial objects in many ways. The most common one is to define permutation as a sequence $\pi = \pi_1, \ldots, \pi_n$ in which each number of the set $\{1, \ldots, n\}$ appears exactly once. The origins of the study of permutation classes can be traced to the famous book *The Art of Computer Programming, Volume 1* by Knuth [93]. Among a plethora of other topics, Knuth studied sorting with stacks, which lead him naturally to the notion of permutation patterns. He showed that a permutation π can be sorted by a stack if and only if it does not contain a subsequence $\pi_{i_1}, \pi_{i_2}, \pi_{i_3}$ such that $\pi_{i_3} < \pi_{i_1} < \pi_{i_2}$. Nowadays, we would say that these are exactly the 231-avoiding permutations since such a sequence $\pi_{i_1}, \pi_{i_2}, \pi_{i_3}$ is in the same relative order as the permutation 231. In general, a permutation $\pi = \pi_1, \ldots, \pi_n$ *contains* a permutation $\sigma = \sigma_1, \ldots, \sigma_k$ if there is a subsequence of π of length k in the same relative order as σ. Otherwise, we say that π *avoids* σ.

The study of pattern-avoiding permutations is today an active field that spans over many different areas, such as enumeration ("How many permutations of length n avoid σ?"), structural questions ("Which permutation classes are well-quasi-ordered?"), probability ("What does the typical σ-avoiding permutation look like?") or computational complexity ("How hard it is to decide if π contains σ?"). In this thesis, we focus on computational problems and related structural questions.

Permutation pattern matching. The most fundamental decision problem involving permutations is undisputedly the Permutation Pattern Matching problem (PPM). In PPM, the input consists of two permutations: τ, referred to as the 'text', and π, referred to as the 'pattern'. The goal is to determine whether the text τ contains the pattern π.

Bose, Buss and Lubiw [39] have shown that the PPM problem is NP-complete. Thus, most recent research into the complexity of PPM focuses either on parametrized or superpolynomial algorithms, or on restricted instances of the problem.

For a pattern π of size k and a text τ of size n, several fast algorithms exist, each suitable for different relative difference between k and n. Berendsohn, Kozma and Marx [23] designed an algorithm running in time $O(n^{k/4+o(k)})$. The current fastest exponential algorithm with running time $O(1.415^n)$ is due to Gawrychowski and Rzepecki [73]. And last but not least, Guillemot and Marx [77] have shown, perhaps surprisingly, that PPM can be solved in time $n \cdot 2^{O(k^2 \log k)}$, later improved to $n \cdot 2^{O(k^2)}$ by Fox [68].

The other line of research is to consider restrictions on the input permutations that would allow for an efficient pattern matching algorithm. These restrictions usually take the form of specifying that the pattern must belong to a prescribed class $\mathcal{C}$ of permutations (the $\mathcal{C}$-Pattern PPM problem), or that both the pattern and the text must belong to $\mathcal{C}$ (the $\mathcal{C}$-PPM problem). The most commonly considered choices for $\mathcal{C}$ are the *principal classes* consisting of all the permutations that avoid a fixed pattern σ, denoted by $\mathrm{Av}(\sigma)$.

The complexity of $\mathrm{Av}(\sigma)$-Pattern PPM was completely resolved by Jelínek

and Kynčl [86] who showed that it is polynomial-time solvable for $\sigma \in \{1, 12, 21, 132, 231, 213, 312\}$ and NP-complete otherwise. However, not much is known beyond the principal classes and we explore this uncharted territory in Chapter 4. The situation is far less clear in the case of the text-restricted variant $\mathcal{C}$-PPM, even for the principal classes. Clearly, Av(σ)-PPM is polynomial-time solvable whenever Av(σ)-Pattern PPM is. Moreover, Guillemot and Vialette [78] proved that Av(321)-PPM is polynomial-time solvable as well. On the other hand, Jelínek and Kynčl [86] showed that Av(4321)-PPM is already NP-complete and that, in fact, Av(σ)-PPM is NP-complete for any σ of length at least 10. We narrow the existing gap in Chapter 5.

Structural parameters. It is natural to ask how the structure of permutation influences the complexity of pattern matching. Ahal and Rabinovich [2] discovered a connection between the complexity of PPM and a structural parameter called *tree-width*. Later, Berendsohn, Kozma and Marx [23] noticed that PPM can be phrased as the so-called constraint satisfaction problem, which leads to an algorithm running in time $O(n^{\mathrm{tw}(\pi)+1})$ where $\mathrm{tw}(\pi)$ is the tree-width of the pattern π and n is the length of the text. This motivates the study of the maximum tree-width attained by permutations of length n inside a given class $\mathcal{C}$, called the *tree-width growth of* $\mathcal{C}$. If the tree-width growth of $\mathcal{C}$ is bounded by a constant, then $\mathcal{C}$-Pattern PPM is polynomial-time solvable, and if it is sublinear, e.g., of order n^c for some $c < 1$, then we obtain a subexponential algorithm for $\mathcal{C}$-Pattern PPM. Describing the asymptotic tree-width growth of (not only) principal classes forms a large part of Chapter 2.

Grid classes. The grid classes played a prominent role in recent developments of the permutation pattern research [115, 44, 25, 113]. Indeed they also form the backbone of many results presented in this thesis. When dealing with grid classes, it is useful to represent a permutation $\pi = \pi_1\pi_2\cdots\pi_n$ by its *diagram*, which is the set of points $\{(i, \pi_i) \mid i = 1, \ldots, n\}$. A grid class is defined in terms of a *gridding matrix* $\mathcal{M}$, whose entries are (possibly empty) permutation classes. We say that a permutation π has an *$\mathcal{M}$-gridding*, if its diagram can be partitioned, by horizontal and vertical cuts, into an array of rectangles, where each rectangle induces in π a subpermutation from the permutation class in the corresponding cell of $\mathcal{M}$. The permutation class $\mathrm{Grid}(\mathcal{M})$ then consists of all the permutations that have an $\mathcal{M}$-gridding.

Griddings have been previously used, sometimes implicitly, in the analysis of special cases of PPM, where they were applied both in the design of polynomial algorithms [10, 78], and in NP-hardness proofs [86]. In fact, all the known NP-hardness arguments for special cases of both $\mathcal{C}$-Pattern PPM and $\mathcal{C}$-PPM are based on the existence of suitable grid subclasses of the class $\mathcal{C}$. Our reductions in Chapters 4 and 5 are no exceptions to this rule.

Generalized coloring. The second decision problem that we consider is that of generalized coloring. A permutation π is a *merge* of a permutation σ and a permutation τ, if we can color the elements of π red and blue so that the red elements have the same relative order as σ and the blue ones as τ. The problem of generalized coloring is to determine, for fixed permutation classes $\mathcal{C}$ and $\mathcal{D}$,

whether a given permutation π can be obtained as a merge of an element of $\mathcal{C}$ with an element of $\mathcal{D}$.

Previously, only very little has been known about the complexity of generalized coloring of permutations. The only existing algorithms have been stated in the language of permutation graphs [46, 62] and no NP-hard instances have been known. We initiate an exploration of this area in Chapter 7.

Outline

Structural properties. In Chapter 2, we focus on various *permutation parameters* such as tree-width or twin-width which can be interpreted as measures of structural complexity. We consider several different parameters, some of them used often in previous works and some introduced by us. We unravel all the mutual relationships between them.

In the second part of Chapter 2, we focus our attention solely on tree-width. We define several structural properties of a general permutation class $\mathcal{C}$, called the *long path property*, the *cycle property*, the *deep tree property* and the *bicycle property*. All these properties can be viewed as stating that $\mathcal{C}$ contains monotone grid subclasses (whose all entries are either increasing, decreasing or empty) of a particular type. We show that these properties impose different lower bounds on the tree-width growth of $\mathcal{C}$. Finally, we relate these properties to the principal classes providing us with lower bounds on their tree-width growth. In particular, it follows that the class $\mathrm{Av}(\sigma)$ contains permutations with tree-width linear in their length for any σ of length at least four that is not symmetric to one of 3412, 3142, 4213, 4123 or 41352.

Logic of permutations. Chapter 3 explores permutations as structures of a logical theory called Theory of Two Orders (TOTO). We build on previous results of Albert et al. [9] on first-order logic of permutations. Our attention is, however, fixed on monadic second-order logic. We uncover natural properties inexpressible in first-order logic but expressible in monadic second-order logic. As a specific example, there is no first-order sentence expressing that a permutation can be obtained as a merge of two 3142-avoiding patterns. Furthermore, we focus on the computational complexity of monadic second-order model checking. We show its tractability for permutations of bounded tree-width but on the other hand, we prove that it remains hard in any class with the long path property.

Permutation pattern matching. In Chapters 4 and 5, we focus on the PPM problem. We start Chapter 4 by considering the general unrestricted PPM problem through the lenses of both classical and parameterized complexity. This part contains both alternative proofs of previously known facts as well as a few new results. Afterwards, we focus on the pattern-restricted variant $\mathcal{C}$-Pattern PPM. We prove its NP-hardness and impose different conditional lower bounds on the running time of $\mathcal{C}$-Pattern PPM under the assumption that $\mathcal{C}$ has either the long path or the deep tree property. Note that most of the previous results only concerned the principal classes of permutations and thus, this can be seen as a major step towards understanding the connection between the structure of a

class $\mathcal{C}$ and the complexity of $\mathcal{C}$-Pattern PPM. As a noteworthy byproduct, we prove that the tree-width-based algorithm solving PPM in time $O(n^{\mathrm{tw}(\pi)+1})$ is asymptotically optimal under the exponential time hypothesis (ETH).

In Chapter 5, we go on to study the more restricted variant $\mathcal{C}$-PPM where we also require that the text τ belongs to the class $\mathcal{C}$. We show that $\mathcal{C}$-PPM can be solved in polynomial time for any monotone grid class $\mathcal{C}$. On the other hand, we develop a general type of hardness reduction, applicable to any permutation class containing a suitable family of grid subclasses. We then verify that for most choices of σ large enough, the principal class $\mathrm{Av}(\sigma)$ indeed satisfies this condition. Specifically, we can prove that $\mathrm{Av}(\sigma)$-PPM is NP-complete for any pattern σ of length at least four that is not symmetric to one of 3412, 3142, 4213, 4123 or 41352.

Grid classes. In Chapter 6, we restrict our attention solely to grid classes. We extend the results of previous chapters to show that monotone grid classes exhibit a sharp trichotomy in their tree-width growth and the complexity of $\mathcal{C}$-Pattern PPM. Specifically for a monotone gridding matrix $\mathcal{M}$, there are three distinct regimes depending on the properties of the associated cell graph. In each of them, we know the exact asymptotic behavior of the tree-width growth of $\mathrm{Grid}(\mathcal{M})$ and the complexity of $\mathrm{Grid}(\mathcal{M})$-Pattern PPM together with almost sharp conditional lower bounds, if applicable.

We are able to extend this to a weaker dichotomy for general grid classes. In particular, we can precisely characterize the grid classes of unbounded tree-width and also when the $\mathrm{Grid}(\mathcal{M})$-Pattern PPM problem is NP-hard, albeit the latter applies only for the gridding matrices whose every entry has bounded tree-width.

Generalized coloring. Finally, Chapter 7 deals with the problem of generalized coloring of permutations. We design three novel general frameworks for recognizing permutation classes. One is based on the concept of non-deterministic online computations with logarithmic memory, while the other two are based on tree automata. Combining them, we obtain polynomial time algorithms recognizing permutations that are merges of permutations $\pi \in \mathcal{C}$ and $\tau \in \mathcal{D}$ for various interesting pairs of permutation classes $\mathcal{C}$ and $\mathcal{D}$. For example, we can decide in polynomial time whether a given permutation can be obtained as a merge of a 213-avoiding pattern with a 321-avoiding pattern. We also provide a complementary NP-hardness result which implies, among other examples, that the recognition of merges of two 3142-avoiding permutations is NP-hard.

This work is based on the following papers whose contents have been modified, extended and interleaved to achieve an easier and more logical presentation[1]:

- Vít Jelínek, Michal Opler, and Jakub Pekárek. Long paths make pattern-counting hard, and deep trees make it harder. [P6] – Section 2.2, part of Section 2.3, Subsection 4.3.2 and Subsection 6.1.1.
- Vít Jelínek, Michal Opler, and Jakub Pekárek. Griddings of Permutations and Hardness of Pattern Matching. [P5] – Chapter 5.

[1] the citation references are into List of Publications

- Vít Jelínek, Michal Opler, and Jakub Pekárek. A complexity dichotomy for permutation pattern matching on grid classes. [P7] – the rest of Chapter 6, parts of Section 2.2 and Subsection 4.3.1.
- Vít Jelínek, Michal Opler, and Pavel Valtr. Generalized coloring of permutations. [P9] – Chapter 7.

Moreover, Chapter 3 is based on (yet) unpublished work with Vít Jelínek.

1. Preliminaries

1.1 Permutations

A *permutation of length* n is a sequence $\pi = \pi_1, \ldots, \pi_n$ in which each element of the set $[n] = \{1, 2, \ldots, n\}$ appears exactly once. When writing out short permutations explicitly, we shall omit all punctuation and write, e.g., 15342 for the permutation $1, 5, 3, 4, 2$. However, it is much more beneficial for our purposes to view permutations as geometric objects. To that end, the *permutation diagram* of π is the set of points $S_\pi = \{(i, \pi_i);\ i \in [n]\}$ in the plane. Observe that no two points from S_π share the same x- or y-coordinate. We say that such a set is in *general position*.

For a point p in the plane, we let $p.x$ denote its horizontal coordinate, and $p.y$ its vertical coordinate. Two finite sets $S, R \subseteq \mathbb{R}^2$ in general position are *isomorphic* if there is a bijection $f\colon S \to R$ such that for any pair of points $p \neq q$ of S we have $f(p).x < f(q).x$ if and only if $p.x < q.x$, and $f(p).y < f(p).y$ if and only if $p.y < q.y$. The *reduction* of a finite set $S \subseteq \mathbb{R}^2$ in general position is the unique permutation π such that S is isomorphic to S_π. We write $\pi = \mathrm{red}(S)$.

We say that a permutation τ *contains* a permutation π if the diagram of τ contains a subset that is isomorphic to the diagram of π. See Figure 1.1. The witnessing bijection $f : S_\pi \to S_\tau$ is called an *embedding of* π *into* τ. If τ does not contain π, we say that it *avoids* π. In this context, we call π the *pattern*. Observe that this definition of containment is equivalent to the definition via subsequences in the same relative order used in Introduction. A *permutation class* is a set $\mathcal{C}$ of permutations which is *hereditary*, i.e., for every $\sigma \in \mathcal{C}$ and every permutation π contained in σ, we have $\pi \in \mathcal{C}$. For a permutation π, we let $\mathrm{Av}(\pi)$ denote the set of all the permutations that avoid π; this is clearly a permutation class. More generally, for a set B of permutations, called *basis*, we let $\mathrm{Av}(B)$ be the set of permutations that avoid all the elements of B.

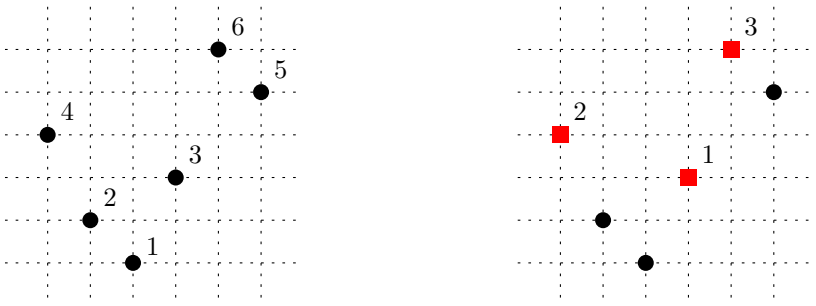

Figure 1.1: A permutation diagram of the permutation 421365 on the left, an occurrence of the pattern 213 in the same permutation on the right.

For a permutation σ, we define four neighbors of a point $(i, j) \in S_\sigma$ as

$$N^R((i,j)) = (i+1, \sigma_{i+1}),$$
$$N^L((i,j)) = (i-1, \sigma_{i-1}),$$
$$N^U((i,j)) = (\sigma^{-1}_{j+1}, j+1),$$
$$N^D((i,j)) = (\sigma^{-1}_{j-1}, j-1).$$

Note that the superscripts R, L, U and D signify the direction in which we reach the corresponding neighbor by sweeping with axis-parallel line. We additionally define the values when some index is out of bounds as one of two 'virtual neighbors' as follows: $N^R((n,i)) = N^U((i,n)) = (\infty, \infty)$ and $N^L((1,i)) = N^D((i,1)) = (0,0)$ for all $i \in [n]$. Note that we can define these neighbors for arbitrary point set S in general position, simply by looking at the reduction of S.

This notion allows for a different, albeit equivalent, definition of permutation τ containing a permutation π. Instead of verifying the correct relative positions of images for every pair of points $p, q \in S_\pi$, it is sufficient to verify just the correct relative positions for every pairs of neighbors. The correctness of the relative positions of other pairs is then implied via transitivity. This fact has been observed by Berendsohn et al. [23].

Observation 1.1. *A function $f : S_\pi \to S_\tau$ is an embedding of π into τ if for every point $p \in S_\pi$*

- $f(N^L(p)).x < f(p).x < f(N^R(p)).x$, *and*
- $f(N^D(p)).y < f(p).y < f(N^U(p)).y$,

whenever the corresponding neighbor $N^\alpha(p)$ exists, i.e., it is not a virtual point.

1.1.1 Symmetries of permutations

We will frequently refer to symmetries that transform permutations into other permutations. For our purposes, it is convenient to describe these symmetries geometrically, as transformations of the plane acting on permutation diagrams. We define the *m-box* to be the set $(\frac{1}{2}, m+\frac{1}{2}) \times (\frac{1}{2}, m+\frac{1}{2})$. Observe that for every permutation π of length at most m, the permutation diagram S_π is a subset of the m-box. We view permutation symmetries as bijections acting on the whole m-box. There are eight such symmetries, generated by:

reversal which reflects the m-box horizontally, i.e. the image of point p is $(m+1-p.x, p.y)$,

complement which reflects the m-box vertically, i.e. the image of point p is $(p.x, m+1-p.y)$,

inverse which reflects the m-box through its main diagonal, i.e. the image of point p is $(p.y, p.x)$.

In particular, the reversal of a permutation $\pi = \pi_1, \ldots, \pi_n$ is the permutation $\pi^r = \pi_n \pi_{n-1}, \ldots, \pi_1$, the complement of π is the permutation $\pi^c = n+1-\pi_1, n+1-\pi_2, \ldots, n+1-\pi_n$, and the inverse π^{-1} is the permutation $\sigma = \sigma_1, \ldots, \sigma_n$ such

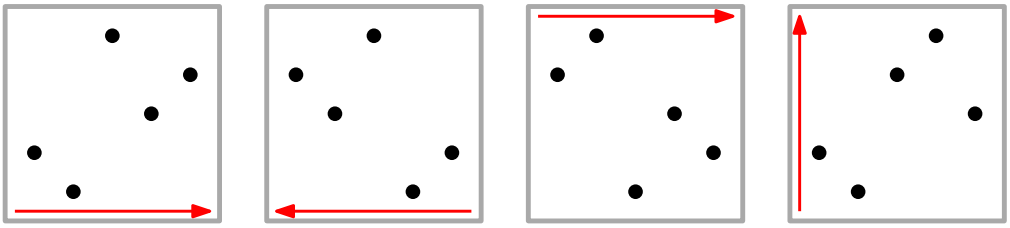

Figure 1.2: From left to right: the permutation 21534, its reverse, complement and inverse. We added a red arrow to help visualize the transformations at play.

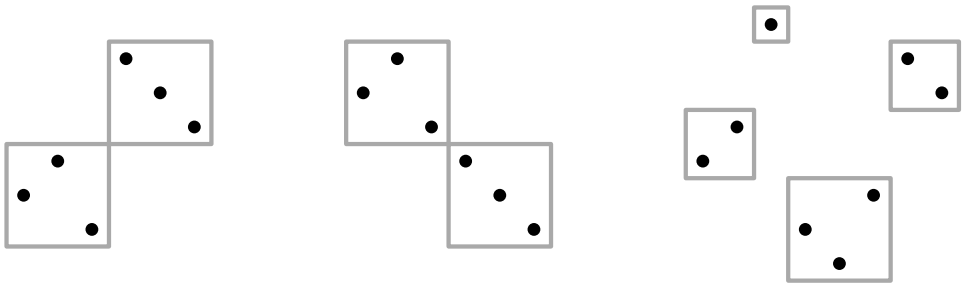

Figure 1.3: The direct sum $213 \oplus 321$ (left), the skew sum $213 \ominus 321$ (center) and the inflation $2413[12, 1, 213, 21]$ (right).

that $\sigma_i = j \iff \pi_j = i$. See Figure 1.2. We also apply these symmetries to sets of permutations, in an obvious way: if Ψ is one of the eight symmetries defined above and $\mathcal{C}$ is a permutation class, we define $\Psi(\mathcal{C})$ as $\{\Psi(\pi) \mid \pi \in \mathcal{C}\}$. Specifically, we denote by $\mathcal{C}^\alpha$ for $\alpha \in \{r, c, -1\}$ the class $\{\pi^\alpha \mid \pi \in \mathcal{C}\}$.

1.1.2 Basic operations acting on permutations

We shall now introduce several standard operations acting on permutations.

Direct sum $\sigma \oplus \tau$, skew sum $\sigma \ominus \pi$. Consider a pair of permutations σ of length k and τ of length ℓ. The *direct sum* of σ and τ, denoted $\sigma \oplus \tau$, is the reduction of a point set obtained as a union of the diagram of σ with the diagram of τ translated by the vector (k, k), i.e.

$$\sigma \oplus \tau = \operatorname{red}\left(S_\sigma \cup (S_\tau + (k, k))\right).$$

Similarly, their *skew sum*, denoted $\sigma \ominus \tau$, is the reduction of a point set obtained as a union of the diagram of σ translated by the vector $(0, \ell)$ with the diagram of τ translated by the vector $(k, 0)$, i.e.

$$\sigma \ominus \tau = \operatorname{red}\left((S_\sigma + (0, \ell)) \cup (S_\tau + (k, 0))\right).$$

See Figure 1.3. A permutation π is called *sum-decomposable* if it can be obtained as a direct sum of two shorter permutations, otherwise we say that π is *sum-indecomposable*. Similarly, a permutation π is called *skew-decomposable* if it can be obtained as a skew sum of two shorter permutations and *skew-indecomposable* otherwise.

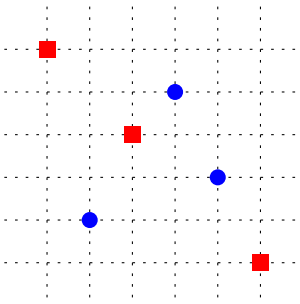

Figure 1.4: Permutation 624531 can be obtained as a merge of 321 (red squares) and 132 (blue disks).

$\mathcal{C} \oplus \mathcal{D}$, $\mathcal{C} \ominus \mathcal{D}$, sum-closure $\mathcal{C}^{\oplus}$, skew-closure $\mathcal{C}^{\ominus}$. For a pair of permutation classes $\mathcal{C}$ and $\mathcal{D}$, we let $\mathcal{C} \oplus \mathcal{D}$ be the set $\{\sigma \oplus \tau \mid \sigma \in \mathcal{C}, \tau \in \mathcal{D}\}$; note that this is again a permutation class. The class $\mathcal{C} \ominus \mathcal{D}$ is defined analogously. The *sum-closure* of a class $\mathcal{C}$, denoted $\mathcal{C}^{\oplus}$, is the class of all the permutations that can be obtained as a direct sum of finitely many members of $\mathcal{C}$; the *skew-closure* $\mathcal{C}^{\ominus}$ is again defined analogously.

Inflation, simple permutation. Given a permutation σ of length n and non-empty permutations $\tau_1, \ldots, \tau_n$, where τ_i is of length m_i, the *inflation* of σ by $\tau_1, \ldots, \tau_n$, denoted $\sigma[\tau_1, \ldots, \tau_n]$, is the permutation obtained by replacing each point (i, σ_i) of the diagram of σ with a scaled-down copy of the diagram of τ_i

$$\sigma[\tau_1, \ldots, \tau_n] = \operatorname{red}\left(\bigcup_{i=1}^{n} \frac{1}{m_i} \cdot S_{\tau_i} + (i, \sigma_i)\right).$$

See again Figure 1.3. A permutation is *simple*, if it cannot be obtained from strictly smaller permutations by an inflation. For instance, the permutation 25314 is simple, while 25341 is not, since it can be obtained, e.g., as $231[1, 312, 1]$.

Merge, $\mathcal{C} \odot \mathcal{D}$. A permutation π is a *merge* of permutations σ and τ if we can color the points of S_π with colors red and blue so that the red points are isomorphic to S_σ and the blue ones to S_τ. See Figure 1.4. The merge of a class $\mathcal{C}$ and a class $\mathcal{D}$ is the class $\mathcal{C} \odot \mathcal{D}$ of permutations that can be obtained by merging an element of $\mathcal{C}$ with an element of $\mathcal{D}$.

1.1.3 Classical permutation classes

We will meet many different permutation classes, several of whom have standard names in the literature. The class Av(21) of all the *increasing permutations* and the class Av(12) of all the *decreasing permutations* will emerge by far the most often and therefore, we use the symbols $\boxslash$ and $\boxbslash$, respectively, as shorthands. Another frequently encountered class are the *layered permutations*, which is the class obtained as a sum-closure of decreasing permutations. Layered permutations can also be characterized by finite basis as the class $\mathrm{Av}(231, 312)$. The complements of layered permutations are known as the *co-layered permutations*; they can be obtained as a skew-closure of the increasing permutations, or alternatively they form the class $\mathrm{Av}(132, 213)$.

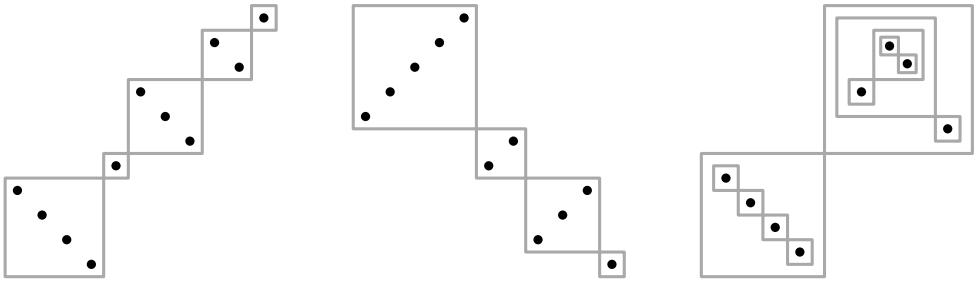

Figure 1.5: A layered permutation (left), a co-layered permutation (center), and a separable permutation (right).

Finally, the *separable permutations* are the permutations that can be created from the singleton permutation of size 1 by direct sums and skew sums.; it is known [39] that these are precisely the permutations avoiding the patterns $2413, 3142$. It follows that the separable permutations are the smallest infinite class which is both sum-closed and skew-closed. See Figure 1.5.

1.2 Generalized permutation patterns

In this section, we introduce several generalizations of the notion of patterns that appear in the literature.

1.2.1 Vincular patterns

Recall that a permutation τ contains a pattern π if there is a subset of the diagram of τ that is isomorphic to the diagram of π. In particular, we care only about the relative positions of the points in the pattern without considering the rest of the permutation τ. It, however, might make sense to require that certain elements in the occurrence of π are consecutive in the left-to-right order. That is exactly the additional constraint carried by vincular patterns.

Formally, a *vincular pattern* is a pair (π, C) where π is a permutation of length k and C is an arbitrary subset of $\{0, \dots k\}$. A permutation τ is said to contain the vincular pattern (π, C) if there is an embedding f of π into τ such that additionally for every $i \in C$

- if $1 \leq i < k$ then $N^R(f((i, \pi_i))) = f((i+1, \pi_{i+1}))$,
- if $i = 0$ then $N^L(f((1, \pi_1))) = (0, 0)$, and
- if $i = k$ then $N^R(f((k, \pi_k))) = (\infty, \infty)$.

To represent a vincular pattern (π, C) visually, we simply take the permutation diagram of π and shade the strip $(i, i+1) \times \mathbb{R}$ (the set of points whose x-coordinates are between i and $i+1$) for every $i \in C$. See Figure 1.6.

Vincular patterns first appeared under the name of *generalized patterns* in the work of Babson and Steingrímsson [18] where the authors showed that many permutation statistics can be expressed as linear combinations of vincular patterns. Let us remark that vincular patterns are usually denoted in the literature either by underlining the consecutive entries or separating the non-consecutive entries with dashes. For example, the vincular pattern $(132, \{1\})$ would be written as

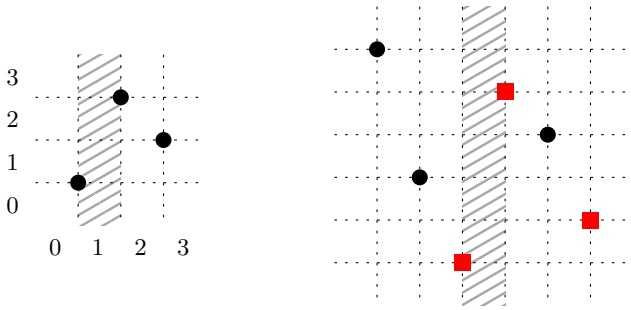

Figure 1.6: The vincular pattern $(132, \{1\})$ (left) and its occurrence in the permutation 631542 (right).

either $\underline{1}32$ or $13-2$. We, however, chose this more technical description in order to ease the upcoming more complicated notions of patterns.

Moreover, a special case of vincular patterns are so-called consecutive patterns. As the name suggests, we require that any embedding of a consecutive pattern must have all its entries consecutive in the left-to-right order. Formally, a *consecutive* pattern π of length k is simply the vincular pattern (π, C) where C is exactly the set $\{1, \dots, k-1\}$.

Covincular patterns.

Let us note that a symmetric notion to that of vincular patterns has appeared in the literature under the name of covincular patterns [21]. Informally, a covincular pattern is obtained by taking a vincular pattern together with its restrictions and rotating it by 90°. Formally, a *covincular pattern* is a pair (π, R) where π is again a permutation of length k and R is an arbitrary subset of $\{0, \dots k\}$. Now a permutation τ is said to contain the covincular pattern (π, R) if there is an embedding f of π into τ such that for every $i \in R$

- if $1 \leq i < k$ then $N^U(f((\pi_i^{-1}, i))) = f((\pi_{i+1}^{-1}, i+1))$,
- if $i = 0$ then $N^D(f((\pi_1^{-1}, 1))) = (0, 0)$, and
- if $i = k$ then $N^U(f((\pi_k^{-1}, k))) = (\infty, \infty)$.

We would naturally represent a covincular pattern (π, R) in the same way as vincular patterns, only this time we shade horizontal strips between elements required to be adjacent. However, from the perspective of pattern matching complexity it is sufficient to consider only vincular patterns as covincular patterns are their mere symmetry.

1.2.2 Bivincular patterns

It is only natural to further generalize the notion of patterns by combining the restrictions of vincular and covincular patterns. Indeed, a *bivincular pattern* is a triple (π, C, R) where π is a permutation of length k and C, R are arbitrary subsets of $\{0, \dots, k\}$. A permutation τ then contains the bivincular pattern (π, C, R) if there is an embedding f of π into τ defining simultaneously an occurrence of the vincular pattern (π, C) and the covincular pattern (π, R). See Figure 1.7.

Bivincular patterns were first introduced by Bousquet-Mélou et al. [41]. The particular advantage of bivincular patterns over vincular patterns is that bivincular

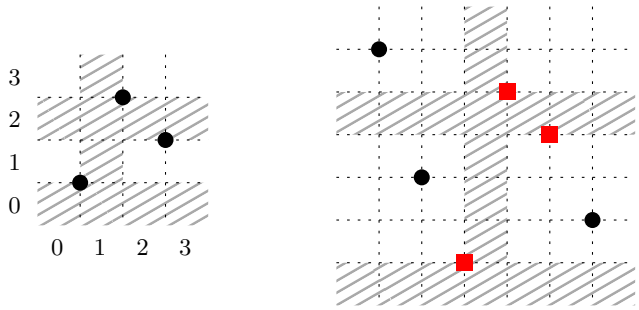

Figure 1.7: The bivincular pattern $(132, \{1\}, \{0, 2\})$ (left) and its occurrence in the permutation 631542 (right).

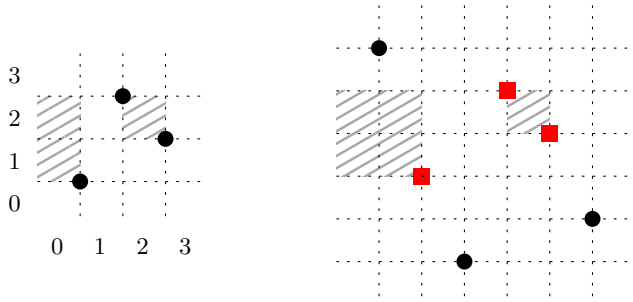

Figure 1.8: The mesh pattern $(132, \{(0, 1), (0, 2), (2, 2)\})$ (left) and its occurrence in the permutation 631542 (right).

patterns are closed under all the symmetries generated by reverse, complement and inverse whereas vincular patterns are not closed under taking inverses.

1.2.3 Mesh patterns

Going one step further, one might impose restrictions not only on the strips between two (horizontally or vertically) adjacent points in the occurrence but rather on individual boxes defined by quadruples of points. These so-called mesh patterns were introduced by Brändén and Claesson [42] who showed their capability to encode a plethora of permutation statistics.

A *mesh pattern* is a pair (π, B) where π is again a permutation of length k but B is now an arbitrary subset of $\{0, \dots, k\} \times \{0, \dots, k\}$. A permutation τ contains the mesh pattern (π, B) if there is an embedding $f : S_\pi \to S_\tau$ of π into τ such that for every $(i, j) \in B$ there is no point of S_τ in the interior of the box

$$[f((i, \pi_i)).x, f((i + 1, \pi_{i+1})).x] \times [f((\pi_j^{-1}, j)).y, f((\pi_{j+1}^{-1}, j + 1)).y]$$

In order to simplify the definition, we additionally assume here that both S_π and S_τ contain the points $(0, 0)$, (∞, ∞) and that $f(p) = p$ for both $p \in \{(0, 0), (\infty, \infty)\}$. It is, however, again much clearer when represented visually. In a mesh pattern (π, B), the points of S_π separate the plane into $(k + 1) \times (k + 1)$ grid and we simply shade the unit square with bottom left corner (i, j) for every $(i, j) \in B$. An occurrence of (π, B) in τ is then any occurrence of π such that all the shaded squares are empty. See Figure 1.8.

There is also a special type of mesh patterns that have been introduced by Avgustinovich, Kitaev and Valyuzhenich [17]. The *boxed mesh pattern* is a mesh

pattern (π, B) where π is classical permutation of length k and $B = [k-1] \times [k-1]$. In other words, all but the boundary boxes are shaded.

1.2.4 Partially ordered patterns.

A different generalization of classical patterns was introduced by Kitaev [92] and have since been subject of a decent amount of research [72, 91, 119]. Informally, we relax the condition that values of the occurrence must have a prescribed linear order and replace it with an arbitrary partial order. Formally, a *partially ordered pattern* p of length k is a pair $(<_P, k)$ where $<_P$ is a partial order on the set $[k]$. An embedding of $p = (<_P, k)$ into a permutation τ is a mapping $f : [k] \to S_\tau$ such that $f(i).x < f(j).x$ for every $i < j$, and $f(i).y < f(j).y$ whenever $i <_P j$.

Example. The partially ordered pattern $p =$  (defined by the Hasse diagram of $<_P$) requires that the first element in the left-to-right order is larger than the last one. The permutation 41253 therefore contains four occurrences of p, namely, as the subsequences 412, 413, 423, and 453.

We can furthermore generalize this concept to the so-called *doubly partially ordered patterns* that replace the linear order on x-coordinates with a second partial order [52].

1.2.5 Other notions of patterns

We conclude this section with a brief overview of various other types of patterns that have made their appearances in the literature. Note that as we shall see in Chapter 3, all these different notions of patterns can be expressed in the powerful framework of first-order logic.

Barred patterns. A quite different generalization of classical patterns was introduced by West [118] as part of the effort to understand sets of permutations that can be sorted by two passes through a stack. Formally, a *barred pattern* is a classical pattern π of length k with bars over some of its entries. A permutation τ avoids a barred pattern π if every occurrence of the unbarred subpattern of π is actually a part of an occurrence of the underlying classical pattern of π.

Marked mesh and decorated patterns. Mesh patterns prescribe that certain regions defined by the embedding of the underlying classical pattern must be empty. *Marked mesh patterns* generalize mesh patterns by specifying how many elements may be contained in certain regions. *Decorated patterns* go even further by prescribing that certain regions must avoid given patterns (which themselves might be decorated). We refer an interested reader to their full and formal introduction by Úlfarsson [110].

2-avoidance. Very recently, a generalization of barred patterns was introduced by Elder and Goh [63]. For two sets of permutations F and G, we say that a permutation τ *2-avoids* (F, G) if any occurrence of $\pi \in F$ in τ is actually a part of an occurrence of some $\sigma \in G$. In other words, we allow the patterns of F but

$$\mathcal{M} = \begin{pmatrix} \mathrm{Av}(231) & \mathrm{Av}(21) \\ \mathrm{Av}(132) & \mathrm{Av}(12) \end{pmatrix}$$

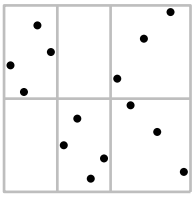

Figure 1.9: A gridding matrix $\mathcal{M}$ on the left and a permutation equipped with an $\mathcal{M}$-gridding on the right. Empty entries of $\mathcal{M}$ are omitted and the edges of $G_{\mathcal{M}}$ are displayed inside $\mathcal{M}$.

only under the condition that each occurrence is "saved" by being contained in some pattern from G.

1.3 Grid classes

An important type of permutation classes are the so-called grid-classes, which we we introduce in this section. A *gridding matrix of size* $k \times \ell$ is a matrix $\mathcal{M}$ with k columns and ℓ rows, whose every entry is a permutation class. A *monotone gridding matrix* is a gridding matrix whose every entry is one of the three classes $\emptyset$, $\boxslash$ or $\boxbslash$. Note that to be consistent with the Cartesian coordinates that we use to describe permutation diagrams, we will number the rows of a matrix from bottom to top, and we give the column coordinate as the first one. In particular, $\mathcal{M}_{i,j}$ denotes the entry in column i and row j of the matrix $\mathcal{M}$, with $i \in [k]$ and $j \in [\ell]$.

Let π be a permutation of length n. A $(k \times \ell)$-*gridding* of π is a pair of weakly increasing sequences $1 = c_1 \leq c_2 \leq \cdots \leq c_{k+1} = n+1$ and $1 = r_1 \leq r_2 \leq \cdots \leq r_{\ell+1} = n+1$. For $i \in [k]$ and $j \in [\ell]$, the (i,j)-*cell* of the gridding of π is the set of points $p \in S_\pi$ satisfying $c_i \leq p.x < c_{i+1}$ and $r_j \leq p.y < r_{j+1}$. Note that each point of the diagram S_π belongs to a unique cell of the gridding. A permutation π together with a gridding (c, r) forms a *gridded permutation*.

Let $\mathcal{M}$ be a gridding matrix of size $k \times \ell$. We say that the gridding of π is an $\mathcal{M}$-*gridding* if for every $i \in [k]$ and $j \in [\ell]$, the subpermutation of π induced by the points in the (i,j)-cell of the gridding of π belongs to the class $\mathcal{M}_{i,j}$.

We let $\mathrm{Grid}(\mathcal{M})$ denote the set of permutations that admit an $\mathcal{M}$-gridding. This is clearly a permutation class. A *monotone grid class* is any permutation class $\mathrm{Grid}(\mathcal{M})$ for a monotone gridding matrix $\mathcal{M}$.

As we shall later see, many important properties of the grid class $\mathrm{Grid}(\mathcal{M})$ can be characterized using properties of certain graph associated to $\mathcal{M}$. The *cell graph* of a gridding matrix $\mathcal{M}$, denoted $G_{\mathcal{M}}$, is the graph whose vertices are all the pairs (i,j) for which $\mathcal{M}_{i,j}$ is an infinite permutation class. Two vertices are adjacent if they appear in the same row or the same column of $\mathcal{M}$, and there is no other cell containing an infinite class between them. See Figure 1.9. We shall sometime prescribe the properties of a cell graph to the matrix itself, e.g., we might say that a gridding matrix $\mathcal{M}$ is acyclic meaning that, in fact, its cell graph $G_{\mathcal{M}}$ is acyclic.

1.3.1 Building gridded permutations

Let π be a permutation of length n with a $(k \times \ell)$-gridding (c, r), where $c = (c_1, \ldots, c_{k+1})$ and $r = (r_1, \ldots, r_{\ell+1})$. The *reversal of the i-th column* of π is the operation that transforms π into a new permutation π' by taking the rectangle $[c_i, c_{i+1} - 1] \times [1, n]$ and flipping it along its vertical axis, thus producing the diagram of a new permutation π'. Equivalently, π' is created from π by reversing the order of the entries of π at positions $c_i, c_i + 1, \ldots, c_{i+1} - 1$. We view π' as a gridded permutation, with the same gridding (c, r) as π.

Similarly, the *complementation of the j-th row* transforms the diagram of π by flipping the rectangle $[1, n] \times [r_j, r_{j+1} - 1]$ along its horizontal axis, producing the diagram of a new gridded permutation π'.

We may similarly apply reversals to the columns of a gridding matrix $\mathcal{M}$ and complements to its rows. Reversing the i-th column of $\mathcal{M}$ produces a new gridding matrix, in which all the classes in the i-th column of $\mathcal{M}$ are replaced by their reversals. Row complementation of a gridding matrix is defined analogously. Note that a column reversal or a row complementation in a gridded permutation or in a gridding matrix is an involution, i.e., repeating the same operation twice restores the original permutation or matrix. Note also that when we perform a sequence of column reversals and row complementations, then the end result does not depend on the order in which the operations were performed.

To describe succinctly a sequence of row and column operations, we introduce the notion of $(k \times \ell)$-*orientation*, which is a pair of functions $\mathcal{F} = (f_c, f_r)$ with $f_c \colon [k] \to \{-1, 1\}$ and $f_r \colon [\ell] \to \{-1, 1\}$. *Applying* the orientation $\mathcal{F}$ to a $(k \times \ell)$-gridded permutation π produces a new gridded permutation $\mathcal{F}(\pi)$ with the same gridding as π, obtained by reversing each column i such that $f_c(i) = -1$ and complementing each row j such that $f_r(j) = -1$. The application of $\mathcal{F}$ to a gridding matrix $\mathcal{M}$ is defined analogously, and produces a gridding matrix denoted $\mathcal{F}(\mathcal{M})$. Note that (c, r) is an $\mathcal{M}$-gridding of π if and only if it is an $\mathcal{F}(\mathcal{M})$-gridding of $\mathcal{F}(\pi)$.

We are especially interested in orientation whose application turns every non-empty entry of $\mathcal{M}$ into the class of increasing permutations. Formally, an orientation $\mathcal{F}$ is a *consistent orientation* of a monotone gridding matrix $\mathcal{M}$, if every nonempty entry of $\mathcal{F}(\mathcal{M})$ is equal to $\boxslash$.

Example. The matrix $\begin{pmatrix} \boxbslash & \boxslash \\ \boxslash & \boxbslash \end{pmatrix}$ has a consistent orientation $\mathcal{F} = (f_c, f_r)$ acting by reversing the first column and complementing the first row. Formally,

$$\begin{matrix} f_c(1) = -1 \\ f_c(2) = 1 \end{matrix} \quad \text{and} \quad \begin{matrix} f_r(1) = -1 \\ f_r(2) = 1. \end{matrix}$$

On the other hand, the matrix $\begin{pmatrix} \boxbslash & \boxslash \\ \boxslash & \boxslash \end{pmatrix}$ has no consistent orientation, since applying any orientation to this matrix yields a matrix with an odd number of $\boxbslash$-entries.

It is quite crucial that any acyclic monotone gridding matrix has a consistent orientation as shown by Vatter and Waton [116].

Lemma 1.2 ([116, Proposition 2.1])**.** *Every monotone gridding matrix whose cell graph is acyclic has a consistent orientation.*

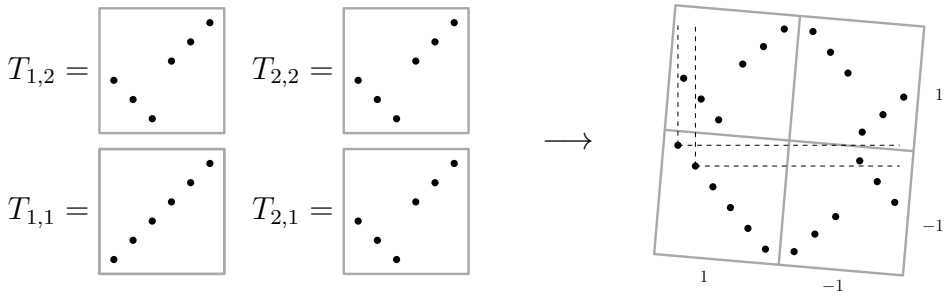

Figure 1.10: A 2×2 family of tiles $\mathcal{T}$ on the left and its $\mathcal{F}$-assembly on the right for a 2×2 orientation $\mathcal{F}$ given next to each row and column on the right. General position is attained by rotating the resulting point set clockwise. The dashed lines indicate relative positions of two particular points.

Tile assembly

Throughout the thesis, we frequently need to construct permutations whose diagrams have a natural $k \times \ell$ grid-like structure. We describe such a diagram by taking each cell individually and describing the points inside it. For such a description, it is often convenient to assume that each cell has its own coordinate system whose origin is near the bottom-left corner of the cell. This allows us to describe the coordinates of the points inside the cell without referring to the position of the cell within the whole permutation diagram. In effect, we describe the diagram of the gridded permutation by first constructing a set of independent 'tiles' $\mathcal{T}_{i,j}$ for $i \in [k]$ and $j \in [\ell]$ of the same size, and then translating each tile $\mathcal{T}_{i,j}$ to column i and row j of the diagram. On top of that, we often need to apply an orientation to the gridded permutation whose diagram we constructed.

We now describe the whole procedure more formally. Fix an integer m and recall that an m-box is a square of the form $(\frac{1}{2}, m + \frac{1}{2}) \times (\frac{1}{2}, m + \frac{1}{2})$. An *$m$-tile* is an arbitrary finite set of points inside the m-box. Note that the coordinates of the points in the tile may not be integers. A $(k \times \ell)$-family of m-tiles is a collection $(\mathcal{T}_{i,j} \mid i \in [k], j \in [\ell])$ where each $\mathcal{T}_{i,j}$ is an m-tile. For a $(k \times \ell)$-orientation $\mathcal{F}$, the *$\mathcal{F}$-assembly* of the family $(\mathcal{T}_{i,j} \mid i \in [k], j \in [\ell])$ is the gridded permutation obtained as follows.

First, we translate each tile $\mathcal{T}_{i,j}$ by adding $m(i-1)$ to each horizontal coordinate and $m(j-1)$ to each vertical coordinate. Thus, the m-tiles will be disjoint and occupying their respective cells of the final gridding. If the union of the translated tiles is not in general position, we rotate it slightly clockwise to reach general position. Notice that we can do so without changing the relative position of any pair of points that were already in general position. This yields a point set isomorphic to a unique permutation π. Additionally, π has a natural gridding whose cells correspond to the translated tiles. To finish the construction, we apply the orientation $\mathcal{F}$ to π, obtaining the gridded permutation $\mathcal{F}(\pi)$, which is the $\mathcal{F}$-assembly of the family of tiles $(\mathcal{T}_{i,j} \mid i \in [k], j \in [\ell])$. See Figure 1.10.

Observation 1.3. *Let $(\mathcal{T}_{i,j};\ i \in [k], j \in [\ell])$ be a family of tiles, let $\mathcal{F}$ be an orientation, and let $\mathcal{M}$ be a gridding matrix such that $\mathcal{T}_{i,j}$ is isomorphic to a permutation from the class $\mathcal{M}_{i,j}$. Then the $\mathcal{F}$-assembly of the family of tiles $(\mathcal{T}_{i,j};\ i \in [k], j \in [\ell])$ is a permutation from the class* $\mathrm{Grid}(\mathcal{F}(\mathcal{M}))$.

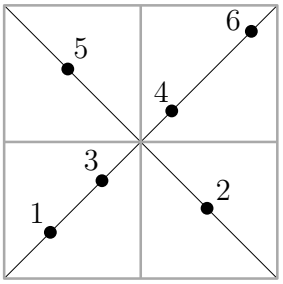

Figure 1.11: The permutation 153426 can be obtained as a subset of the standard figure $\Lambda_{\mathcal{M}}$ of the monotone gridding matrix $\mathcal{M} = \begin{pmatrix} \boxminus & \boxslash \\ \boxslash & \boxminus \end{pmatrix}$ and thus, it belongs to the geometric grid class $\mathrm{Geom}(\mathcal{M})$.

1.3.2 Geometric grid classes

There also exists a stricter geometrical analogue to the monotone grid classes, called geometric grid classes, which we now introduce. Let $\mathcal{M}$ be a monotone gridding matrix. The standard figure of $\mathcal{M}$, denoted by $\Lambda_{\mathcal{M}}$, is the point set in the plane consisting of:

- the open segment from $(i-1, j-1)$ to (i, j) whenever $\mathcal{M}_{i,j} = \boxslash$, and
- the open segment from $(i-1, j)$ to $(i, j-1)$ whenever $\mathcal{M}_{i,j} = \boxminus$.

The *geometric grid class* of $\mathcal{M}$, denoted by $\mathrm{Geom}(\mathcal{M})$, is the set of all permutations isomorphic to finite subsets of $\Lambda_{\mathcal{M}}$ in general position. See Figure 1.11.

Clearly, $\mathrm{Geom}(\mathcal{M}) \subseteq \mathrm{Grid}(\mathcal{M})$ for any monotone gridding matrix $\mathcal{M}$. Moreover, it is not hard to convince oneself that equality holds for acyclic gridding matrices. This fact was first proved by Albert et al. [7].

Proposition 1.4 ([7, Theorem 3.2])**.** *If $\mathcal{M}$ is an acyclic monotone gridding matrix then* $\mathrm{Geom}(M) = \mathrm{Grid}(\mathcal{M})$*. In fact, given any $\mathcal{M}$-gridding of a permutation π we can find a subset of $\Lambda_{\mathcal{M}}$ that is isomorphic to S_π and respects the given $\mathcal{M}$-gridding.*

Note that Albert et al. [7] additionally show that this holds only for the acyclic matrices. For every monotone gridding matrix $\mathcal{M}$ with a cycle in its cell graph, we have $\mathrm{Geom}(\mathcal{M}) \subsetneq \mathrm{Grid}(\mathcal{M})$.

2. Structural properties

In this chapter, we investigate different ways how to describe the structural complexity of a permutation class. We start by considering various "width" parameters that can be defined for permutations in Section 2.1. Some of these, like tree-width, have been previously studied for permutations. Others, like modular-width, have been studied for graphs and were considered only briefly and implicitly in the case of permutations. And finally, we introduce a new parameter called grid-width which turns out to be very suitable for designing dynamic programming algorithms. Moreover, we unravel all relationships between these parameters. See Figure 2.1.

In Section 2.2, we define structural properties of permutation classes using the containment of monotone grid subclasses of a particular type. These properties then allow us to impose lower bounds on the maximal tree-width attained by a permutation of a given class. As an example, we are able to show that any class with the so-called bicycle property contains permutations whose tree-width is linear in their length.

Finally in Section 2.3, we focus our attention on the principal permutation classes, i.e. the classes defined by avoiding a single pattern. We are able to almost fully characterize which properties are attained by which classes. In particular, we show that almost all principal classes have the aforementioned bicycle property and therefore, contain permutations of linear tree-width.

2.1 Width parameters

2.1.1 Twin-width

Twin-width was introduced as a graph parameter by Bonnet et al. [36], who generalized previous decomposition of permutations designed by Guillemot and Marx [77] in their ground-breaking fpt-algorithm for PPM. It turned out to unify many previous structural and algorithmic results and is subject to a fruitful ongoing research, e.g. [19, 32, 33, 34, 35]. Note that we shall only introduce twin-width as a parameter of permutations but it should be clear how to define it for graphs, or even more generally for arbitrary binary relational structures. We refer an interested reader, e.g., to [37]. In fact, together with twin-width, we also define its two variants introduced in [34], that we will later link to other permutation parameters.

A *partition sequence* of a permutation π of length n is a sequence $S = \mathcal{P}_n, \dots, \mathcal{P}_1$ of partitions of the permutation diagram S_π where $\mathcal{P}_n = \{\{p\} \mid p \in S_\pi\}$ is the partition of S_π into singletons, $\mathcal{P}_1 = \{S_\pi\}$ contains a single part equal to the whole diagram S_π, and each $\mathcal{P}_i$ is obtained by merging two parts of $\mathcal{P}_{i+1}$. Notice that each $\mathcal{P}_i$ consists of exactly i disjoint parts.

We say that two disjoint subsets X, Y of the permutation diagram S_π are *horizontally separated* if the whole set X lies to the left of Y, i.e. we have $p.x < q.x$ for every $p \in X$ and $q \in Y$, or vice versa. Similarly, X and Y are *vertically separated* if the whole set X lies below Y, i.e. we have $p.y < q.y$ for every $p \in X$ and $q \in Y$, or vice versa. Finally, we say that X, Y are *homogeneous* if they are

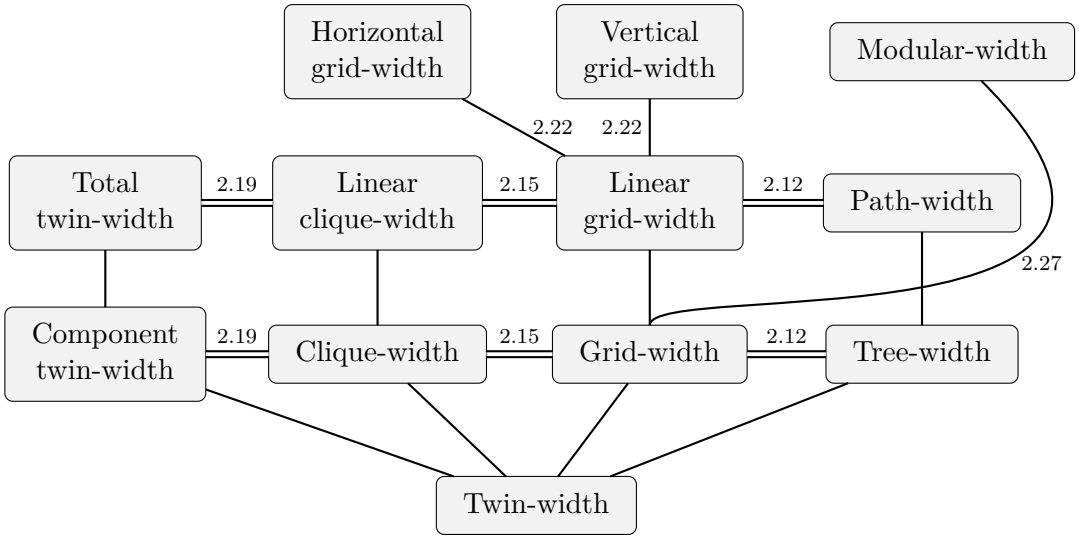

Figure 2.1: The hierarchy of permutation parameters considered in Section 2.1. Solid edges connect pairs of parameters where the lower is bounded from above by a function of the upper, double edges connect pairs for which this holds in both ways. We omitted edges implied by transitivity, all remaining pairs of parameters are incomparable.

separated both horizontally and vertically.

Additionally, we define that a set X is homogeneous to itself if and only if X is a singleton. It is perhaps easier to think about the notion of homogeneous sets visually. For a set of points X, let $\mathcal{R}(X)$ be the smallest axis-parallel rectangle that contains the whole set X. Disjoint sets X and Y are homogeneous if and only if the projections of the rectangles $\mathcal{R}(X)$ and $\mathcal{R}(Y)$ onto both x- and y-axis are disjoint.

Given a permutation π and a partition $\mathcal{P}_i$ of S_π, we consider an auxiliary graph R_i, called *red graph*, with vertices the parts of $\mathcal{P}_i$ and (red) edges all pairs of parts X, Y that are not homogeneous. Note that by our definition of homogeneity, the red graph contains a loop (X, X) for every part X that is not a singleton. See Figure 2.2 for an example.

We can define different width measures based on partition sequences. An arbitrary function w from a partition $\mathcal{P}$ of a permutation diagram into the non-negative integers is called a width. The *partition-width associated to* w of a permutation π is the minimum integer t such that there exists a partition sequence $\mathcal{P}_n, \ldots, \mathcal{P}_1$ of π such that $w(\mathcal{P}_i) \leq t$ for every $i \in [n]$.

We consider three natural width parameters defined using the red graph R_i: w_d is the maximum degree of R_i, w_c is the maximum number of vertices in a connected component of R_i, and w_t is the total number of vertices adjacent to red edge in R_i. Observe that $w_d \leq w_c \leq w_t$. These widths define associated width parameters:

- The *twin-width* of π as the partition-width $\mathrm{tww}(\pi)$ associated to w_d.
- The *component twin-width* of π as the partition-width $\mathrm{ctww}(\pi)$ associated to w_c.
- The *total twin-width* of π as the partition-width $\mathrm{ttww}(\pi)$ associated to w_t.

We must add that Bonnet et. al. [34] define the total twin-width via the total

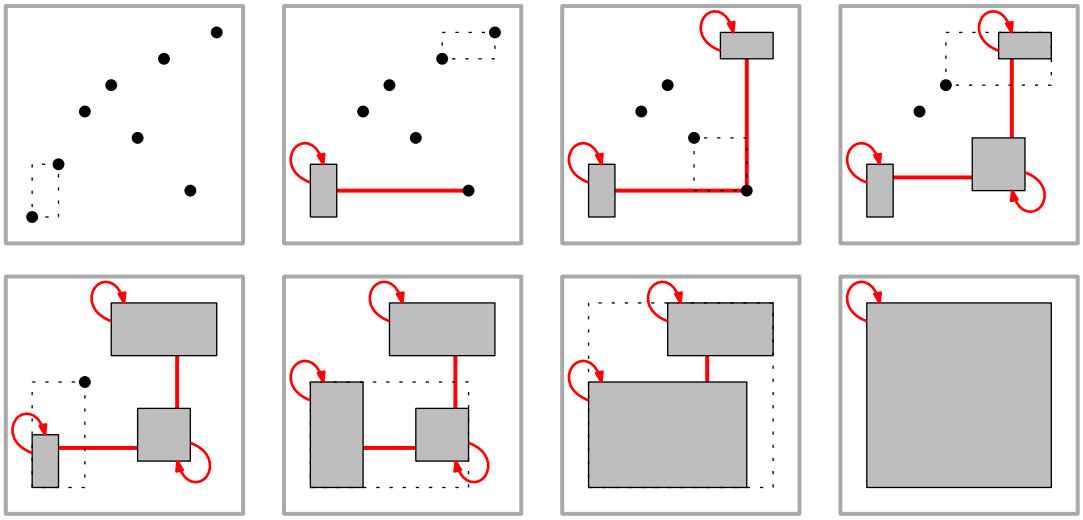

Figure 2.2: A partition sequence of the permutation 13564728 with each part X visually represented by the rectangle $\mathcal{R}(X)$. The edges of the red graph are drawn in each partition including the loops at every non-singleton part.

number of red edges. We chose to alter this definition as it is more suitable for our needs while measuring almost the same thing. Denoting by w'_t the total number of red edges in the red graph, it is easy to see that $w_t(\mathcal{P})/2 \leq w'_t(\mathcal{P}) \leq (w_t(\mathcal{P}))^2$ for any partition $\mathcal{P}$.

Let us focus just on twin-width for now. It turns out that many permutations have small twin-width. For example, the separable permutations are exactly the permutations of twin-width 1 (see [77, Proposition 3.5]). Bonnet et al.[36] showed that, in fact, every permutation class has bounded twin-width as long as it is not the class of all permutations. This stems from the same observation by Guillemot and Marx [77] about their permutation decomposition even before the term twin-width was coined.

Theorem 2.1 ([36]). *A permutation class $\mathcal{C}$ has bounded twin-width if and only if it does not contain all permutations.*

Another useful property of twin-width is that it can be approximately computed in polynomial time. The following algorithm was designed by Marx and Guillemot [77], with later improvement in the upper bound on twin-width due to Fox [68].

Theorem 2.2 ([77, Theorem 4.1]). *There exists an algorithm that, given a permutation π of length n and an integer k, runs in time $O(n)$ and either outputs that the twin-width of π is larger than k, or returns a partition sequence of π witnessing that* $\mathrm{tww}(\pi) \leq 2^{O(k)}$.

2.1.2 Tree-width

In recent years, a parameter called tree-width has proven to be very useful in investigating the hardness of permutation pattern matching.

Let us first define tree-width as a standard graph parameter. A *tree decomposition* of a graph G is a pair (T, β), where $\beta : V(T) \to 2^{V(G)}$ assigns a *bag* $\beta(p)$ to each vertex of T, such that

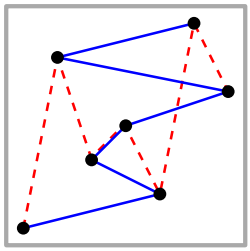

Figure 2.3: The permutation 1634275 and its incidence graph. The blue solid edges connect horizontal neighbors and the red dashed edges connect vertical neighbors. Note that the incidence graph is a simple graph and in particular, there is only one edge between the points $(3, 3)$ and $(4, 4)$.

- for every vertex v of G, there exists $p \in V(T)$ such that $v \in \beta(p)$,
- for every edge uv in G, there exists $p \in V(T)$ such that $u, v \in \beta(p)$, and
- for every vertex v of G, the set $T_v = \{p \in V(T) : v \in \beta(p)\}$ induces a connected subtree of T.

The *width* of the tree decomposition is the maximum of $\beta(p) - 1$ over all $p \in V(T)$. The *tree-width* $\mathrm{tw}(G)$ of a graph G is the minimum width of a tree decomposition of G.

The tree-width of a permutation π is then defined by taking a tree-width of a certain graph encoding the structure of π. The *incidence graph* G_π of a permutation π of length n is the graph whose vertices are the n points of S_π, and each point p is connected to its (at most four) neighbors, i.e. a point p is connected to $N^\alpha(p)$ for each $\alpha \in \{R, L, U, D\}$ as long as the given neighbor exists. In particular, the graph G_π is a union of two paths, one of them visiting the points of π in left-to-right order, and the other in top-to-bottom order. See Figure 2.3. The *tree-width* of π, denoted by $\mathrm{tw}(\pi)$, is simply the tree-width of G_π.

There is also a standard and well-known parameter related to tree-width called path-width. There are several different ways how to define path-width. One of them is to define the *path-width* $\mathrm{pw}(G)$ of a graph G as the minimum width of a tree decomposition (T, β) of G such that T is a path. Similar to before, the path-width of a permutation π, denoted by $\mathrm{pw}(\pi)$, is the path-width of its incidence graph G_π.

The incidence graph and the importance of its tree-width was first observed by Ahal and Rabinovich [2]. They defined their own width parameter called tree-complexity and showed that it is up to multiplication by a constant factor equivalent to the tree-width of G_π. Therefore, we shall omit the definition of tree complexity.

We state without a proof that separable permutations have bounded tree-width. This fact was observed also by Ahal and Rabinovich [2, p. 643] but we will later obtain it as a simple corollary of the fact that separable permutations have grid-width at most 1.

Since tree-width is perhaps the most famous and the most studied graph parameter, we can make use of many available power tools. In particular, there exist fpt-algorithms for approximating the tree-width of a graph G up to a multiplicative constant. The best approximation to date is due to a recent 2-approximation algorithm of Korhonen [95] that superseded the previous 5-

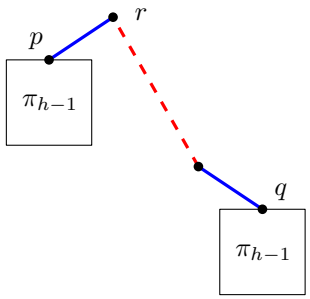

Figure 2.4: The inductive construction of π_h in the proof of Proposition 2.5. The blue solid edges correspond to neighbors in the vertical direction and the red dashed edge corresponds to a pair of horizontal neighbors.

approximation algorithm of Bodlaender et al. [28].

Theorem 2.3 ([95, Theorem 1.1])**.** *There exists an algorithm that, given a graph G on n vertices and an integer k, in time $2^{O(k)} \cdot n$ either outputs that the treewidth of G is larger than k, or constructs a tree decomposition of G of width at most $2k + 1$.*

Later, when we have more tools available, we will show that bounded tree-width implies bounded twin-width but not the other way round. Instead, we conclude this section by showing that there are permutation classes that have bounded tree-width but unbounded path-width.

Proposition 2.4 (folklore)**.** *The complete rooted binary tree of height h has path-width exactly $\lceil \frac{h}{2} \rceil$.*

Proposition 2.5. *The class* Av(132) *has bounded tree-width and unbounded path-width.*

Proof. Observe that Av(132) has bounded tree-width since the pattern 132 is contained in both 2413 and 3142 and thus, every 132-avoiding permutation is separable.

In order to show that there are 132-avoiding permutations of arbitrarily large path-width, we first examine two closure properties of the class Av(132). Firstly, the class Av(132) is closed under taking skew sums. And secondly, the direct sum $\pi \oplus 1$ of an arbitrary 132-avoiding permutation π with the singleton 1 also avoids 132. With this knowledge, we inductively define for each $h \in \mathbb{N}$ a permutation π_h such that its incidence graph G_{π_h} contains the complete rooted binary tree of height h as a minor. We set $\pi_1 = 1$ and for every $h > 1$ we define π_h as follows

$$\pi_h = (\pi_{h-1} \oplus 1) \ominus 1 \ominus \pi_{h-1}.$$

By the discussion of the previous paragraph, we see that π_h avoids 132 for every h. We shall prove by induction that the incidence graph G_{π_h} contains a subdivision of the complete rooted binary tree of depth h rooted in the topmost point of π_h. The path-width of π_h is then at least $\lceil h/2 \rceil$ due to Proposition 2.39 and the result follows.

The claim holds trivially for π_1. Suppose that $h > 1$, let r be the topmost point in π_h and let p, q be the topmost points in the two copies of π_{h-1} contained

in π_h where p belongs to the left copy and q to the right copy. Observe that the bottom neighbor of r is precisely the point p, i.e. $N^D(r) = p$. Furthermore, the point q is the bottom neighbor of the right neighbor of r, i.e. $N^D(N^R(r)) = q$. Thus, G_{π_h} contains a subdivision of the complete binary tree of height h rooted in r since r is connected to the subdivided tree in p by an edge and to the subdivided tree rooted in q by a path of length two. See Figure 2.4. □

2.1.3 Grid-width

An *interval family* $\mathcal{I}$ is a set of pairwise disjoint integer intervals. The *intervalicity* of a set $A \subseteq [n]$, denoted by $\text{int}(A)$, is the size of the smallest interval family whose union is equal to A. For a point set S in the plane, let $\Pi_x(S)$ denote its projection on the x-axis and similarly $\Pi_y(S)$ its projection on the y-axis. For a subset S of the integer grid $\mathbb{Z} \times \mathbb{Z}$, the *grid-complexity* of S is the maximum of $\text{int}(\Pi_x(S))$ and $\text{int}(\Pi_y(S))$.

A *grid tree* of a permutation π of length n is a rooted binary tree T with n leaves, each leaf being labeled by a distinct point of the permutation diagram. Let π_v^T denote the point set of the labels on the leaves in the subtree of T rooted in v. The *grid-width* of a vertex v in T is the grid-complexity of π_v^T, and the *grid-width* of T, denoted by $\text{gw}^T(\pi)$, is the maximum grid-width of a vertex of T. Finally, the *grid-width* of a permutation π, denoted by $\text{gw}(\pi)$, is the minimum of $\text{gw}^T(\pi)$ over all grid trees T of π. See Figure 2.5.

Observation 2.6. *For a permutation π and any permutation σ contained in π, we have* $\text{gw}(\sigma) \leq \text{gw}(\pi)$.

One useful property of grid-width is that it is not increased by inflations.

Lemma 2.7. *Suppose a permutation π is an inflation $\sigma[\alpha_1, \ldots, \alpha_m]$. Then*

$$\text{gw}(\pi) = \max(\text{gw}(\sigma), \text{gw}(\alpha_1), \ldots, \text{gw}(\alpha_m)).$$

Proof. Clearly the left-hand side of the inequality must be at least as big as the right one since grid-width is monotone, i.e., it cannot be increased by removing points. For the other direction, let T_σ be the optimal grid tree of σ and T_{α_i} be the optimal grid tree of α_i for each i. We can obtain a grid tree T of π by taking the grid tree T_σ, replacing its i-th leaf with the grid tree T_{α_i} and then relabeling the leaves of T_{α_i} with the corresponding points of the copy of α_i in π. It is easy to see that the grid-complexity of any vertex in T_{α_i} is at most $\text{gw}(\alpha_i)$ while the grid-complexity of any vertex in the upper part of the tree is at most $\text{gw}(\sigma)$. □

It follows that any separable permutation has grid-width equal to 1 as it can be obtained by repeated direct and skew sums from single points. However, it is easy to see that conversely any permutation of grid-width 1 must be separable.

Observation 2.8. *A permutation π has grid-width exactly 1 if and only if π is separable.*

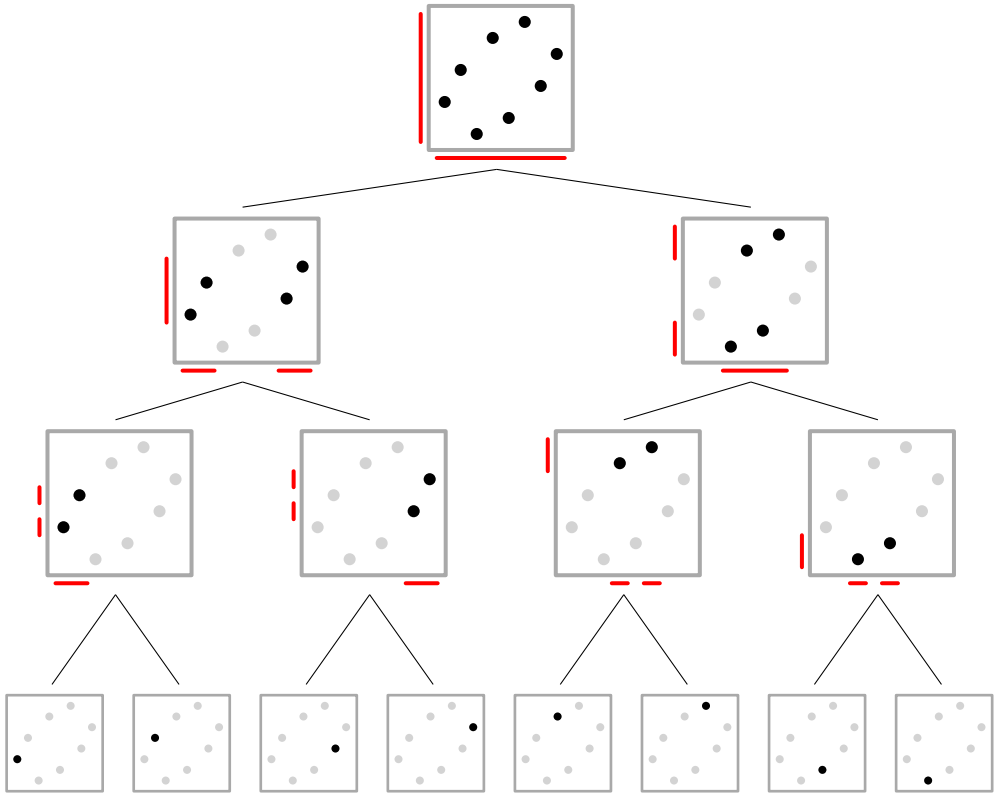

Figure 2.5: A grid tree of the permutation 35172846 with grid-width 2. To illustrate the grid-complexity at each vertex, the red segments along the borders of the diagrams show the intervalicity of each projection. We omit these in the leaves as the grid-complexity of every singleton is trivially equal to 1.

Linear grid-width

It is natural to consider a linear version of grid-width. To that end, we say that a rooted binary tree T is a *caterpillar* if each vertex is either a leaf or has at least one leaf as a child. The *linear grid-width* of a permutation π, denoted by $\mathrm{lgw}(\pi)$, is the minimum of $\mathrm{gw}^T(\pi)$ over all caterpillar grid trees T of π.

It is often convenient to use an alternative equivalent definition of linear grid-width. For permutations π and σ of length n, the *linear grid-width of π in σ-ordering*, denoted by $\mathrm{lgw}^\sigma(\pi)$ is the maximum grid-complexity attained by a set $\{(\sigma_1, \pi_{\sigma_1}), \ldots, (\sigma_i, \pi_{\sigma_i})\}$ for some $i \in [n]$.

Lemma 2.9. *The linear grid-width of a permutation π of length n is equal to the minimum value of $\mathrm{lgw}^\sigma(\pi)$ over all permutations σ of length n.*

Proof. Suppose that $\mathrm{lgw}(\pi) = k$ as witnessed by a caterpillar grid tree T. Observe that all leaves of T except for the deepest pair lie in different depths. Define σ to simply order the labels by the depths of their leaves, defining arbitrarily the order of the two deepest leaves. Then every set $\{(\sigma_1, \pi_{\sigma_1}), \ldots, (\sigma_i, \pi_{\sigma_i})\}$ corresponds exactly to the set π_v^T for some vertex v of T.

In order to prove the other direction, we define a sequence of caterpillar trees $T_1, \ldots, T_n$ in the following way. Let T_1 be a single vertex labeled by $(\sigma_1, \pi_{\sigma_1})$. For $i > 1$, let T_i be the binary rooted tree with left child a leaf labeled by $(\sigma_i, \pi_{\sigma_i})$

and right child the tree T_{i-1}. The tree T_n is a caterpillar grid tree of π and for every inner vertex v the set π_v^T is equal to $\{(\sigma_1, \pi_{\sigma_1}), \ldots, (\sigma_i, \pi_{\sigma_i})\}$ for some i. The claim follows. □

Comparison with tree-width and path-width

It turns out that grid-width is bounded by a linear function of tree-width and vice versa. Moreover, we can efficiently switch between grid trees of a permutation π and tree decompositions of G_π both ways. This allows us, in particular, to use the powerful tree-width approximation algorithms to obtain a good approximation of optimal grid trees. On top of that, the same arguments show that the same relationship holds between linear grid-width and path-width.

Proposition 2.10. *For any permutation π and a tree decomposition (T, β) of width k for G_π, we can compute in linear time a grid tree $T^\star$ of π such that $gw^{T^\star}(\pi) \leq k + 2$. Moreover, if T is a path then $T^\star$ is a caterpillar.*

Proof. We start by rooting the tree T. If T is a path, then we root it in one of its endpoints. Otherwise, we pick an arbitrary node r of T as its root. We perform a few local modifications that do not increase the width of this decomposition with the goal to obtain a tree decomposition (T', β') such that every internal node of T' has at most two children, and for every vertex $p \in V(G_\pi)$, there is a leaf v of T' with $\beta(v) = \{p\}$. To achieve this, we choose for each vertex p of G_π an arbitrary node w of T whose bag contains p, and add a new child v_p to w with bag $\{p\}$. We call v_p the *representing leaf* of p. Next, we replace any node v with more than two children with a sufficiently long path whose every node has its bag equal to $\beta(v)$ and hang its children along this path. We let (T', β') be the resulting tree-decomposition.

As the next step, we repeatedly remove from T' any leaf that does not belong to the set $\{v_p;\ p \in V(G_\pi)\}$ of representing leaves. We repeat these deletions until we are left with a tree whose only leaves are the n representing leaves. In other words, we remove every node w of T' such that the subtree of T' rooted at w does not contain any representing leaf. Let T'' be the resulting tree, and β'' be the restriction of β' to $V(T'')$.

Note that (T'', β'') remains a valid tree decomposition of G_π; indeed, if p and q are adjacent vertices of G_π, the tree T' contains a unique path from v_p to v_q, and at least one node of this path contains both p and q in its bag. Since the path also belongs to T'', it follows that T'' contains a node with both p and q in its bag, and (T'', β'') satisfies the second property of the definition of tree-decomposition. The other two properties are obvious.

We now transform the tree decomposition (T'', β'') into a grid tree $T^\star$ for π as follows: each leaf v_p of T'' will be labeled by p, and for every internal node w of T'' that has only one child, we contract one of the edges incident to w, to obtain a binary tree $T^\star$. We view the nodes of $T^\star$ as a subset of the nodes of T''.

Let us estimate the grid-width of $T^\star$. Fix a node v of $T^\star$ and consider $\Pi_x(\pi_v^{T^*})$ with a minimal decomposition $\mathcal{I}$ into intervals. For each interval $I \in \mathcal{I}$ except the rightmost one, let i be the largest element of I, and consider the two vertices $p_I = (i, \pi_i)$ and $q_I = (i+1, \pi_{i+1})$ of G_π. Note that $p_I q_I$ is an edge of G_π, and that p_I is in $\pi_v^{T^\star}$ while q_I is not. We claim that at least one of p_I and q_I must belong

to $\beta''(v)$. Suppose for a contradiction that neither p_I nor q_I is in $\beta''(v)$. But then v separates all the nodes of T'' that contain p_I in their bags from the nodes of T'' containing q_I in their bags, and thus the edge $p_I q_I$ cannot be contained in any bag.

Moreover, all the vertices p_I and q_I are distinct for different choices of $I \in \mathcal{I}$. Therefore, $\operatorname{int}(\Pi_x(\pi_v^{T^*})) \leq |\beta''(v)| + 1$ and using the same argument for $\Pi_y(\pi_v^{T^\star})$, we see that $\operatorname{gw}^{T^\star}(\pi) \leq k + 2$. It is easy to check that if we started with a path T, we ended up with a caterpillar $T^\star$ as claimed. □

We need one additional definition before showing the reduction of grid tree to a tree decomposition. For a set A of vertices in the incidence graph G_π, the *boundary of* A in G_π, denoted by ∂A, is the set of all elements in A that have a neighbor in G_π outside of A.

Proposition 2.11. *For any permutation π and its grid tree T, we can compute in linear time a tree decomposition (T', β) of G_π of width at most $8\operatorname{gw}^T(\pi)$. Moreover, if T is a caterpillar then T' is a path.*

Proof. Let T' be the tree obtained from T by removing all leaves. Observe that T' is indeed a path whenever T was a caterpillar. We define a tree decomposition (T', β) in the following way: for a vertex u of T' that originally had children v and w in T, we set $\beta(v) = \partial\pi_v^T \cup \partial\pi_w^T$.

Let us verify that (T', β) is indeed a tree decomposition. We denote by v_p the leaf in T that is labeled by a vertex p of G_π, and we denote by u_p the parent of v_p. A vertex p of G_π is clearly in the boundary of $\pi_{v_p}^T = \{p\}$ and thus, it is contained in the bag of u_p. Moreover, every p lies precisely in the bags of the vertices on a path from u_p to the first vertex v where $p \notin \partial\pi_v^T$. And for p and q neighbors in G_π, let u be the least common ancestor of the leaves labeled by p and q, and let v and w be its children. Trivially, p must lie in $\partial\pi_v^T$ and q in $\partial\pi_w^T$ or vice versa, and thus both p and q are contained in $\beta(u)$.

It remains to upper-bound the width of (T', β). The only vertices in the boundary $\partial\pi_v^T$ can be the vertices corresponding to the endpoints of intervals in the interval decomposition of $\Pi_x(\pi_v^T)$ and $\Pi_y(\pi_v^T)$. Therefore, we can bound the size of the boundary as $|\partial\pi_v^T| \leq 2\operatorname{int}(\Pi_x(\pi_v^T)) + 2\operatorname{int}(\Pi_y(\pi_v^t)) \leq 4\operatorname{gw}^T(\pi)$. Now for a vertex u with children v and w, we have a bound on the size of its bag $|\beta(u)| \leq |\partial\pi_v^T| + |\partial\pi_w^T| \leq 8\operatorname{gw}^T(\pi)$. □

Corollary 2.12. *For any permutation π, we have*

$$\tfrac{1}{8} \cdot \operatorname{tw}(\pi) \leq \operatorname{gw}(\pi) \leq \operatorname{tw}(\pi) + 2, \text{ and}$$
$$\tfrac{1}{8} \cdot \operatorname{pw}(\pi) \leq \operatorname{lgw}(\pi) \leq \operatorname{pw}(\pi) + 2.$$

One consequence of these equivalences is that Proposition 2.5 provides a separation between permutation classes of bounded grid-width and bounded linear grid-width.

2.1.4 Clique-width

Perhaps the second most famous graph parameter following tree-width is clique-width. Permutation graphs[1] have played their part in the research of clique-width, see [16, 97]. However, we refrain from introducing clique-width as a

[1] graphs obtained from permutations by adding an edge for each copy of 21.

graph parameter and instead we define it in terms of permutations as relational structures.

A permutation π of length n can be viewed as a structure consisting of a set together with two total orders. We shall see a more in-depth analysis of this approach in Chapter 3. For now, it is sufficient to accept that we can view a permutation of length n as a triple $(A, \prec_x, \prec_y)$ where $|A| = n$ and both $\prec_x$ and $\prec_y$ are linear orders on A. In one way it is clear since we can take as a representation of a permutation π a triple $(S_\pi, \prec_x, \prec_y)$ where $p \prec_x q$ if and only if $p.x < q.x$ and $p \prec_y q$ if and only if $p.y < q.y$.

Taking this idea one step further, we can interpret the triple $(A, \prec_x, \prec_y)$ as a graph with vertex set A and two sets of directed edges (that happen to encode total orders). We say that such graph (V, E_x, E_y) (not restricting E_x and E_y to total orders anymore) is an *xy-digraph* and we call the directed edges in E_x and E_y the x-arcs and y-arcs respectively.

The *clique-width* of an xy-digraph $G = (V, E_x, E_y)$, denoted by $\mathrm{cw}(G)$, is the minimum number of labels needed to construct G using the following five operations:

1. Creation of a single vertex with label i (denoted $\mathbf{i}$).
2. Disjoint union of two labeled xy-digraphs G and H (denoted $G \oplus H$[2]).
3. Renaming label i to j (denoted $\rho_{i\to j}$).
4. Adding an x-arc from every i-labeled vertex to every j-labeled vertex (denoted $\eta^x_{i,j}$), where $i \neq j$.
5. Adding an y-arc from every i-labeled vertex to every j-labeled vertex (denoted $\eta^y_{i,j}$), where $i \neq j$.

A construction of an xy-digraph G using the above operations with at most k distinct labels can be thought of as an algebraic term composed of $\mathbf{i}$, $\oplus$, $\rho_{i\to j}$, $\eta^x_{i,j}$ and $\eta^y_{i,j}$. Such a term is a called a *k-expression* of G. We define the clique-width of a permutation π, denoted by $\mathrm{cw}(\pi)$, as the clique-width of the xy-digraph $(S_\pi, \prec_x, \prec_y)$. Note that we shall also refer to a k-expression Φ of the xy-digraph $(S_\pi, \prec_x, \prec_y)$ as the k-expression of π.

As with most parameters, we can also define its "linear" version. The definition is almost the same, except we allow disjoint unions only with singletons. Formally, we replace the operation 2. with the following operation

$2^\star$. Disjoint union of a labeled xy-digraph G with $\mathbf{i}$ (denoted $G \oplus \mathbf{i}$).

Any k-expression that uses the operation $2^\star$ instead of the operation 2 is called a *linear k-expression*. The *linear clique-width* of an xy-digraph G, denoted by $\mathrm{lcw}(G)$, is then the smallest interger k such that G can be defined by a linear k-expression. As before, we define the linear clique-width of a permutation π, denoted by $\mathrm{lcw}(\pi)$, as the linear clique-width of the xy-digraph $(S_\pi, \prec_x, \prec_y)$.

Example. Suppose a permutation G is an xy-digraph defined by a k-expression Φ and H is an xy-digraph defined by a k-expression Ψ. Moreover, assume that all vertices in G have label 1 and all vertices in H have label 2. We can define the xy-digraph corresponding to the operation of direct sum by the expression

$$\eta^y_{1,2}(\eta^x_{1,2}(\Phi \oplus \Psi)).$$

[2]not to be confused with the direct sum of permutations.

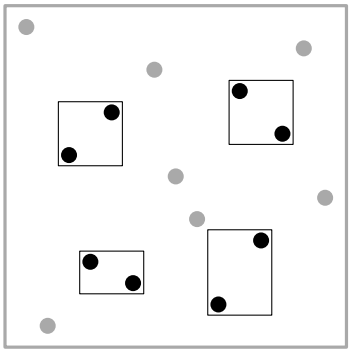

Figure 2.6: A box partition of a subset (in black) of a permutation diagram.

The skew sum is obtained only by replacing the operation $\eta^y_{1,2}$ with $\eta^y_{2,1}$. In this way, we can easily obtain a 2-expression defining any separable permutation π. For example, the following linear 2-expression defines the xy-digraph associated to the permutation $213 = (1 \ominus 1) \oplus 1$

$$\eta^y_{1,2}(\eta^x_{1,2}(\rho_{2\to1}(\eta^y_{2,1}(\eta^x_{1,2}(\mathbf{1} \oplus \mathbf{2}))) \oplus \mathbf{2})).$$

Comparison with grid-width

We shall show that clique-width is again describing similar structural properties as it can be bounded from both sides by a function of grid-width (and thus of tree-width). Note that the equivalence between bounded clique-width and bounded tree-width can be obtained (with worse bounds) as a consequence of more general results by Courcelle and Engelfriet [56].

Proposition 2.13. *For any permutation π and its grid tree T, we can compute a $(2 \cdot \mathrm{gw}^T(\pi)^2)$-expression of π. Furthermore, if T is a caterpillar then it is in fact a linear $(2 \cdot \mathrm{gw}^T(\pi)^2)$-expression.*

Proof. Set k to be $(2 \cdot \mathrm{gw}^T(\pi)^2)$. We show how to transform the grid tree T into an k-expression whose recursive structure is almost identical to T. We do it by inductively defining a k-expression Φ_v for every vertex v of T. We interpret labels as pairs (α, i) where $\alpha \in [2]$ and $i \in [\frac{k}{2}]$ for more clarity.

But first, we define a canonical partition of a subset S of the permutation diagram S_π that does not interfere with the remaining points. Let $\mathcal{I}$ be the smallest interval family whose union is equal to the $\Pi_x(S)$ and similarly, let $\mathcal{J}$ be the smallest interval family with union equal to $\Pi_y(S)$. The *box partition* of S is the partition of S into sets $S \cap (I \times J)$ for every $I \in \mathcal{I}$ and $J \in \mathcal{J}$. See Figure 2.6. Observe that the size of the box partition of S is at most the square of the grid-complexity of S.

We construct an expression Φ_v, which defines an xy-digraph G_v with the following properties. The x-arcs and the y-arcs both form total orders and in particular, G_v with omitted labels represents precisely the permutation π^T_v. And moreover, the labels of G_v induce exactly the box partition of π^T_v and they belong to the set $\{1\} \times [\frac{k}{2}]$.

For a leaf v, we set $\Phi_v = (\mathbf{1}, \mathbf{1})$, i.e., the graph G_v is a singleton with the label $(1, 1)$. Otherwise, v is an internal vertex of T and let u and w denote its children.

We would like to start by taking the disjoint union of Φ_u and Φ_w. However, that might cause trouble by conflating the same labels in G_u and G_w. Therefore, we first create a k-expression Φ'_w by changing every label $(1,i)$ to $(2,i)$ in G_w and only then we take the disjoint union $\Phi_u \oplus \Phi'_w$.

Afterwards, we add x-arcs and y-arcs between every pair of (existing) labels $(1,i)$ and $(2,j)$. Observe that this is always possible with the operations $\eta^x_{i,j}$ and $\eta^y_{i,j}$ since the relative positions of all points with the label $(1,i)$ are the same with respect to all points with the label $(2,j)$. Or put in different terms, these pairs of sets are homogeneous.

As the next step, we need to merge some labels together as the box-partition of π^T_v is not necessarily equal to the union of box partitions of π^T_u and π^T_w. Thus, we add relabeling operations that unify the respective labels $(1,i)$ and $(2,j)$. And to conclude the construction of Φ_v, we rename the labels such that they all belong to the set $\{1\} \times [\frac{k}{2}]$. This is possible since the box partition of π^T_v contains at most $\mathrm{gw}^T(\pi)^2 = \frac{k}{2}$ parts.

It remains to inspect what happens when T is a caterpillar. For an internal vertex v, it is sufficient to choose w in the construction above to be the non-leaf child. One of the expressions participating in the disjoint union is then $\Phi_u = (\mathbf{1}, \mathbf{1})$ since u is a leaf and thus, Φ_v is a linear k-expression for every vertex v of T. □

Proposition 2.14. *For any permutation π and a k-expression Φ of π, we can compute a grid tree T such that $\mathrm{gw}^T(\pi) \leq k$. Moreover, if Φ is a linear k-expression then T is a caterpillar.*

Proof. We define the grid tree T in a straightforward manner using Φ. For any sub-expression Ψ of Φ, we define a rooted tree T_Ψ in the following way. For an expression $\Psi = \mathbf{i}$, we set T_Ψ to be the single vertex (leaf) labeled by the point of π obtained by tracking this vertex throughout the k-expression Φ. If $\Psi = \gamma(\Psi')$ where γ is one of $\rho_{i\to j}$, $\eta^x_{i,j}$ and $\eta^y_{i,j}$ then we set $T_\Psi = T_{\Psi'}$. And finally, for any expression $\Psi = \Psi_1 \oplus \Psi_2$, we set T_Ψ to be the tree obtained by joining T_{Ψ_1} and T_{Ψ_2} with a common root. Observe that T_Φ is a grid tree and moreover, it is a caterpillar whenever Φ is a linear k-expression. We set $T = T_\Phi$.

Let v be a vertex of T. There exits a sub-expression Ψ of Φ such that the subtree of T rooted in v is precisely T_Ψ. Let G_Ψ be the xy-digraph defined by Ψ. Let $\mathcal{I}$ be the smallest interval family whose union is equal to the projection $\Pi_x(\pi^T_v)$ and assume for contradiction that $|\mathcal{I}| > k$. Then there exist two vertices in G_Ψ with the same label such that the x-coordinates of their associated points of the permutation diagram S_π lie in two different intervals of $\mathcal{I}$. Let us denote these two vertices of S_π as p, q and we assume without loss of generality that $p \prec_x q$. But since we chose $\mathcal{I}$ to be the smallest possible, there must be a point r in S_π that separates p and q, i.e we have $p \prec_x r \prec_x q$. Moreover, r is defined outside of the expression Ψ and therefore, the x-arcs (p,r) and (r,q) must be realized later in the expression Φ. That is clearly not possible as p and q already share the same label.

The very same argument works for the projection $\Pi_y(\pi^T_v)$ and we conclude that the grid-complexity of the set π^T_v is at most k for every vertex v of T. □

Corollary 2.15. *For every permutation π, we have*

$$\mathrm{gw}(\pi) \le \mathrm{cw}(\pi) \le 2 \cdot (\mathrm{gw}(\pi))^2, \text{ and}$$
$$\mathrm{lgw}(\pi) \le \mathrm{lcw}(\pi) \le 2 \cdot (\mathrm{lgw}(\pi))^2.$$

Comparison with twin-width

Next, we bring the component and total twin-width into the picture. Bonnet et al. [34] showed that component twin-width of a graph is bounded from both sides by a function of its clique-width and the same for total twin-width and linear clique-width. We include the proof of this fact tailored to the permutation setting. As a byproduct, we obtain significantly tighter bounds.

Recall that w_c denotes the maximum size of a connected component in the red graph associated with a given partition and w_t denotes the total number of vertices in the red graph that are incident to a red edge.

Proposition 2.16. *For any permutation π and its k-expression Φ, we can compute a partition sequence $\mathcal{P}_n, \ldots, \mathcal{P}_1$ of π such that $w_c(\mathcal{P}_i) \le k$ for each i. Furthermore, if Φ is a linear k-expression then $w_t(\mathcal{P}_i) \le k$.*

Proof. Observe that labels in any k-expression behave similarly to parts in a partition sequence in that they only merge to create larger sets. The main idea of this proof is to utilize this observation and track evolution of the labels throughout Φ in a partition sequence.

For a sub-expression Ψ of Φ, we let A_Ψ denote the set of points in S_π that originate in Ψ when constructing π using Φ. We start by constructing sets $\mathcal{S}_1, \ldots, \mathcal{S}_m$ of sub-expressions of Φ such that for each i the set $\{A_\Psi \mid \Psi \in \mathcal{S}_i\}$ forms a partition of π. We define the initial set $\mathcal{S}_1$ as the set of all singletons $\mathbf{i}$ in Φ and we stop the construction once $\mathcal{S}_m$ contains only Φ. To obtain $\mathcal{S}_{i+1}$ from $\mathcal{S}_i$, we pick arbitrary sub-expression $\Psi = \gamma(\Psi')$ of Φ such that $\Psi' \in \mathcal{S}_i$, and we set

$$\mathcal{S}_{i+1} = \mathcal{S}_i \setminus \{\Psi'\} \cup \{\Psi\}.$$

If there is no such sub-expression then we can find a sub-expression $\Psi = \Psi_1 \oplus \Psi_2$ where $\Psi_1, \Psi_2 \in \mathcal{S}_i$ and we set

$$\mathcal{S}_{i+1} = \mathcal{S}_i \setminus \{\Psi_1, \Psi_2\} \cup \{\Psi\}.$$

For a sub-expression Ψ of Φ, let $\mathcal{P}_\Psi$ be the partition of A_Ψ induced by the labels of the labeled xy-digraph G_Ψ defined by Ψ. Using the sequence $\mathcal{S}_1, \ldots, \mathcal{S}_m$, we define a sequence of partitions $\mathcal{P}'_1, \ldots \mathcal{P}'_m$ where

$$\mathcal{P}'_i = \bigcup_{\Psi \in \mathcal{S}_i} \mathcal{P}_\Psi.$$

We claim that for every i, the partition $\mathcal{P}'_{i+1}$ is either equal to $\mathcal{P}'_i$ or it is obtained by merging two parts of $\mathcal{P}'_i$ together. Let us examine the possible options depending on $\Psi \in \mathcal{S}_i \setminus \mathcal{S}_{i-1}$. If $\Psi = \eta^\alpha_{i,j}(\Psi')$ for α equal to x or y then the labels remain the same and $\mathcal{P}'_i = \mathcal{P}'_{i+1}$. If $\Psi = \rho_{i \to j}(\Psi')$ then either two non-empty parts corresponding to labels i and j merge or the partition remains unchanged when at most one of these labels is present in $G_{\Psi'}$. Otherwise we have $\Psi = \Psi_1 \oplus \Psi_2$.

If there were a label i used in both G_{Ψ_1} and G_{Ψ_2} then there would be no way to define an x-arc between such vertices in future, which is impossible. Thus, $\mathcal{P}'_{i+1} = \mathcal{P}'_i$. This allows us to define a valid partition sequence $\mathcal{P}_n, \ldots, \mathcal{P}_1$ simply by reversing the sequence $\mathcal{P}'_1, \ldots, \mathcal{P}'_m$ and removing any duplicate entries.

Let Ψ and $\mathcal{X}$ be a pair of sub-expressions of Φ such that the sets A_Ψ and $A_\mathcal{X}$ are disjoint. We claim that any pair of sets $P \in \mathcal{P}_\Psi$ and $Q \in \mathcal{P}_\mathcal{X}$ is homogeneous. Recall that by definition, all the points of P share the same label in G_Ψ, and the same holds for Q in $G_\mathcal{X}$. The x- and y-arcs between points of P and Q can be defined only later in Φ and that would not be possible if they were not homogeneous.

As a consequence, if we take two different sub-expressions $\Psi, \mathcal{X} \in \mathcal{S}_i$ the subgraphs of the red graph of the partition $\mathcal{P}'_i$ induced by $\mathcal{P}_\Psi$ and $\mathcal{P}_\mathcal{X}$ must be disjoint. The upper bound on $w_c(\mathcal{P}'_i)$ follows since the partition $\mathcal{P}_\Psi$ for any sub-expression Ψ contains at most k parts. Moreover, if Φ is a linear k-expression then any set $\mathcal{S}_i$ contains at most one sub-expression Ψ that is not a singleton. Therefore, all the parts of the partition $\mathcal{P}'_i$ adjacent to red edges belong to $\mathcal{P}_\Psi$ and the upper bound on $w_t(\mathcal{P}'_i)$ follows. □

Proposition 2.17. *For any permutation π and its partition sequence $\mathcal{P}_n, \ldots, \mathcal{P}_1$, we can compute a k-expression of π where $k = \max_i(w_c(\mathcal{P}_i)) + 1$.*

Proof. For each $i \in [n]$ and for each connected component C of the red graph R_i associated to the partition $\mathcal{P}_i$, we define a k-expression Φ^i_C with the following properties. The xy-digraph defined by Φ^i_C describes exactly the permutation induced by the points contained in $\bigcup C$ and labels are in one-to-one correspondence with the partition of $\bigcup C$ induced by C. It should be obvious that the k-expression Φ^1_C for the only connected component C in the red graph R_1 gives the desired result.

At the beginning, every connected component C in the red graph R_n is a singleton and we set $\Phi^n_C = \mathbf{1}$. For $i \in [n-1]$, let C be a connected component in the red graph R_i. We start by defining an intermediate expression Ψ^i_C. If C has already existed as a connected component in R_{i+1}, we set $\Psi^i_C = \Phi^{i+1}_C$. Otherwise, C emerged as a union of connected components $C_1, \ldots, C_t$ of the red graph R_{i+1} and a merge of two parts of $C_1 \cup \cdots \cup C_t$ into a single part. We would like to take a disjoint union of the expressions $\Phi^{i+1}_{C_1}, \ldots, \Phi^{i+1}_{C_t}$. However, there the same label might be used across different expressions so we first make the labels distinct. We select pairwise disjoint sets $A_1, \ldots, A_t$ such that $|A_j| = |C_j|$. For each $j \in [t]$, we define a k-expression $\mathcal{X}_j$ by relabeling the xy-digraph defined by $\Phi^{i+1}_{C_j}$ using the set A_j. And only then we take their disjoint union $\Psi^i_C = \oplus^t_{j=1}\mathcal{X}_j$. The total number of labels used by Ψ^i_C is

$$\sum_{j=1}^{t} |A_j| = \sum_{j=1}^{t} |C_j| \leq |C| + 1 \leq k$$

where the first inequality is due to the fact that C is obtained by merging two parts from some components C_j and $C_{j'}$ and therefore, the sum of the sizes of all components is bounded by the size of C plus one.

Finally, we turn the k-expression Ψ^i_C into the final desired k-expression Φ^i_C. We first add all x- and y-arcs between points originating from different components in R_{i+1}. That is always possible since each such part has distinct label and any pair

of parts from different components is by definition homogeneous. To conclude the proof, if the newly merged part in $\mathcal{P}_i$ belongs to the component C we add a relabel operation that merges the corresponding pair of labels. □

Proposition 2.18. *For any permutation π and its partition sequence $\mathcal{P}_n, \ldots, \mathcal{P}_1$, we can compute a linear k-expression of π where $k = \max_i(w_t(\mathcal{P}_i)) + 1$.*

Proof. We reuse most of the ideas from the proof of Proposition 2.17. We say that a connected component C of the red graph R_i is *non-trivial* if $\bigcup C$ contains more than one point. In other words, the trivial components are exactly the singleton parts of the partition not incident to a red edge. In the case of linear clique-width, we shall for each $i \in [n]$ define a single linear k-expression Φ^i with the following properties. The xy-digraph defined by Φ^i describes exactly the permutation induced by all the points contained in non-trivial components and labels are in one-to-one correspondence with the parts of all non-trivial components. Observe that in any partition $\mathcal{P}_i$ there are at most $w_t(\mathcal{P}_i)$ parts contained in non-trivial components. Again, the linear k-expression Φ^1 is exactly the result we are after.

Naturally, we set Φ^n to be empty as there are no non-trivial connected components in R_n. For $i \in [n-1]$, two scenarios can happen when going from $\mathcal{P}_{i+1}$ to $\mathcal{P}_i$. If the permutation induced by the non-trivial components remains the same, we obtain Φ^i from Φ^{i+1} simply by using the relabel operation to unify the two merged parts. Otherwise, there are singleton parts $\{p_1\}, \ldots, \{p_t\}$ that become part of a non-trivial component for the first time. We let $j_1, \ldots, j_t$ be labels that are not used in the xy-digraph $G_{\Phi^{i+1}}$ defined by Φ^{i+1}. We first define an intermediate linear expression Ψ^i as

$$\Psi^i = (((\Phi^{i+1} \oplus \mathbf{j_1}) \oplus \mathbf{j_2}) \cdots) \oplus \mathbf{j_t}.$$

Afterwards, we add all the x-arcs between the newly added vertices and every other point contained in a non-trivial component. That is again possible since every $\{p_i\}$ is homogeneous to any other part. And finally, we define Φ^i by using the relabel operation to unify the two parts that were merged in $\mathcal{P}_i$. The number of labels in Φ^i is bounded by $w_t(\mathcal{P}_i) + 1$ using the same argument as in the proof of Proposition 2.17. □

Corollary 2.19. *For every permutation π, we have*

$$\begin{aligned} \mathrm{ctww}(\pi) &\leq \mathrm{cw}(\pi) \leq \mathrm{ctww}(\pi) + 1, \text{ and} \\ \mathrm{ttww}(\pi) &\leq \mathrm{lcw}(\pi) \leq \mathrm{ttww}(\pi) + 1. \end{aligned}$$

Finally, we get two large groups of functionally equivalent parameters by combining Corollaries 2.12, 2.15 and 2.19.

Corollary 2.20. *For both of the following groups of parameters, a permutation class $\mathcal{C}$ either has all of them bounded by a constant, or all of them unbounded*

1. *tree-width, clique-width, grid-width and component twin-width;*
2. *path-width, linear clique-width, linear grid-width and total twin-width.*

Moreover, we can efficiently translate between decompositions of any two parameters from the same group.

2.1.5 Horizontal and vertical grid-width

One natural way to introduce a more restricted parameter than the previous ones is to consider linear grid-width in a fixed prescribed ordering.

For permutation π, the *horizontal grid-width* of π denoted by $\mathrm{hgw}(\pi)$ is $\mathrm{lgw}^{\sigma}(\pi)$ where $\sigma_i = i$, and the *vertical grid-width* of π, denoted by $\mathrm{vgw}(\pi)$ is $\mathrm{lgw}^{\sigma}(\pi)$ where $\sigma_i = \pi_i^{-1}$. The horizontal grid-width was introduced independently by Ahal and Rabinovich [2] and Albert et al. [6] in the context of designing permutation pattern matching algorithms. Moreover, there is a connection to the so-called insertion-encodable classes which appear often in the area of permutation classes enumeration, see e.g. [11, 20, 112].

Example. As an example, consider the class of layered permutations. Clearly, any prefix of a layered permutation has grid-complexity at most 2. To be more precise, it has grid-complexity equal to 1 if every layer is either fully contained in the prefix or it is completely disjoint from the prefix. In the other case when the prefix intersects some layer, its grid-complexity is exactly 2. Therefore, the horizontal grid-width of any layered permutation is at most 2. And so is, in fact, its vertical grid-width as layer permutations are closed under taking inversions.

It turns out that the obstructions to small horizontal (vertical) grid-width are fairly straight-forward and easy to determine. However, we first need to introduce a couple of new definitions. A permutation π of length n is a *horizontal alternation* if all the even entries of π precede all the odd entries of π, i.e., there are no indices $i < j$ such that π_i is odd and π_j is even. A permutation π is a *vertical alternation* if π^{-1} is a horizontal alternation. A *horizontal monotone juxtaposition* is a monotone grid class $\mathrm{Grid}(\mathcal{C}\ \mathcal{D})$ where both $\mathcal{C}$ and $\mathcal{D}$ are non-empty. Similarly, a *vertical monotone juxtaposition* is a monotone grid class $\mathrm{Grid}\left(\begin{smallmatrix}\mathcal{C}\\ \mathcal{D}\end{smallmatrix}\right)$.

We are now ready to characterize the obstructions to small horizontal grid-width (and via symmetry for its vertical counterpart). Note that this behavior was first observed, albeit in a different form, by Albert et al. [11] in the context of regular insertion encodings.

Lemma 2.21. *For a permutation class $\mathcal{C}$ the following are equivalent:*

(a) $\mathcal{C}$ has unbounded horizontal grid-width,
(b) $\mathcal{C}$ contains arbitrarily large horizontal alternations, and
(c) $\mathcal{C}$ contains a horizontal monotone juxtaposition as a subclass.

Proof. Suppose (a) holds and for every integer k there is a permutation $\pi^k \in \mathcal{C}$ such that the horizontal grid-width of π^k is at least k. Then there is i such that the set $S = \{\pi_1^k, \ldots, \pi_i^k\}$ has grid-complexity at least k. Each pair of neighboring intervals if S is separated by π_j^k for some $j > i$. Therefore, π^k contains a horizontal alternation of length at least $2k - 1$ which proves (b). On the other hand, a horizontal alternation of length $2k$ must have a horizontal grid-width at least k and thus (b) implies (a).

Now suppose that (b) holds. A *monotone horizontal alternation* is a horizontal alternation whose set of odd entries and set of even entries both form monotone sequences. We claim that every horizontal alternation π of length $2k^4$ contains a monotone horizontal alternation of length $2k$. By applying the Erdős–Szekeres theorem [65] on the odd entries of π, we obtain a horizontal alternation π' of

length at least $2k^2$ whose odd entries form a monotone sequence. Applying the Erdős–Szekeres theorem again on the even entries of π' yields a monotone horizontal alternation π'' of length at least $2k$. Therefore, $\mathcal{C}$ contains arbitrarily large monotone horizontal alternations. There are only four possible types of such alternations depending on the type of the monotone sequences. Therefore, $\mathcal{C}$ also contains arbitrarily large alternations belonging to a horizontal monotone juxtaposition $\mathrm{Grid}(\mathcal{D}_1\ \mathcal{D}_2)$ for some choice of $\mathcal{D}_1, \mathcal{D}_2 \in \{\boxslash, \boxbslash\}$. Since every $\sigma \in \mathrm{Grid}(\mathcal{D}_1\ \mathcal{D}_2)$ is contained in a sufficiently large monotone alternation, the class $\mathcal{C}$ must in fact contain the whole class $\mathrm{Grid}(\mathcal{D}_1\ \mathcal{D}_2)$ as a subclass.

On the other hand, if (c) holds then $\mathcal{C}$ contains arbitrarily large monotone horizontal alternations which trivially implies (b). □

Comparison with other parameters

It follows directly from definitions and Lemma 2.9 that linear grid-width is upper-bounded by both horizontal and vertical grid-width.

Observation 2.22. *For every permutation π, we have*

$$\mathrm{lgw}(\pi) \leq \min(\mathrm{hgw}(\pi), \mathrm{vgw}(\pi)).$$

It remains to relate horizontal and vertical grid-widths to each other. Unsurprisingly, they are incomparable in the sense that one of them can be arbitrarily large while the other is equal to 1.

Proposition 2.23. *For every k, there are permutations π, σ such that* $\mathrm{vgw}(\pi) = \mathrm{hgw}(\sigma) = 1$ *and simultaneously* $\mathrm{hgw}(\pi), \mathrm{vgw}(\sigma) \geq k$.

Proof. It is sufficient to prove the existence of π such that $\mathrm{vgw}(\pi) = 1$ and $\mathrm{hgw}(\pi) \geq k$, the existence of σ follows from symmetry. Consider the horizontal monotone juxtaposition $\mathrm{Grid}(\boxbslash\, \boxslash)$. It follows from Lemma 2.21 that we can find $\pi \in \mathrm{Grid}(\boxbslash\, \boxslash)$ with horizontal grid-width greater than k. On the other hand, any subset of form $\{(\pi_1^{-1}, 1), \ldots, (\pi_i^{-1}, i)\}$ clearly has grid-complexity 1 and thus, $\mathrm{vgw}(\pi) = 1$. □

2.1.6 Modular-width

We continue with introducing a graph parameter that has been previously considered for permutations only implicitly. On the other hand, the corresponding decomposition, called substitution decomposition, has appeared many times in the study of permutations, both structural and enumerative.

An *interval* in the permutation π is a set of points of the permutation diagram S_π whose horizontal and vertical projection both form an integer interval. In other words, intervals are exactly the subsets of the permutation diagram with grid-complexity at most 1. Recall that a simple permutation is one that cannot be obtained as an inflation from strictly smaller permutations. Equivalently, a permutation of length n is simple if it has only the trivial intervals of sizes 0, 1 and n. It is easy to observe that every non-trivial permutation π is an inflation of a unique simple permutation.

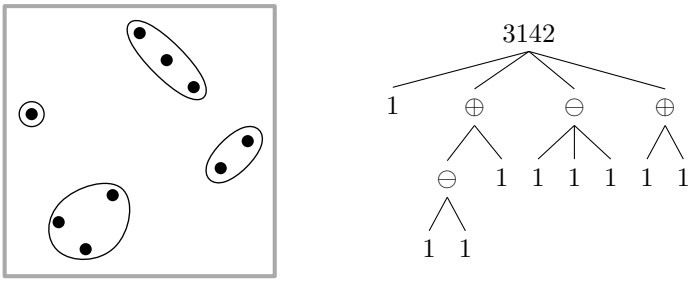

Figure 2.7: The permutation diagram of the permutation 621398745 and its substitution decomposition tree.

Proposition 2.24 ([8, Proposition 2])**.** *Every permutation may be written as the inflation of a unique simple permutation. Moreover, if π can be written as $\sigma[\alpha_1, \ldots, \alpha_m]$ where σ is simple and not equal to 12 or 21, then each α_i is unique. Otherwise, either π is sum-decomposable and there is a unique sequence of sum-indecomposable permutations $\alpha_1, \ldots, \alpha_m$ such that $\pi = \alpha_1 \oplus \cdots \oplus \alpha_m$, or π is skew-decomposable and the same holds with sum replaced by skew sum.*

Naturally, we can use Proposition 2.24 to define a permutation decomposition. A *substitution decomposition tree* of a permutation π is a labelled rooted plane tree where every leaf is labelled with the singleton permutation 1, and the inner nodes are of two different kinds. There are vertices, called *linear nodes*, that are labelled by $\oplus$ or $\ominus$ and can have arbitrary many children but at least 2. The other vertices are called *prime nodes*, they are labelled by a simple permutation of length $k \geq 4$ and they have exactly k children. Furthermore, we require that a linear node cannot have another linear node of the same type as its child. See Figure 2.7.

There is an obvious way of recursively translating a substitution decomposition tree into a permutation. A leaf represents the singleton permutation, a linear node get translated as a direct or as a skew sum of the permutations corresponding to its children and a prime node labeled with a simple permutation σ is translated as an inflation of σ by its children. Conversely, there is a unique substitution decomposition tree for every permutation π due to Proposition 2.24. The *modular-width* of π, denoted by $\mathrm{mw}(\pi)$, is the size of the largest prime node, measured in the length of its label, in the substitution decomposition tree of π. Additionally, we define the modular-width of any separable permutation (that does not contain any prime nodes in their substitution decomposition trees) to be 1.

Observation 2.25. *For a permutation π, $\mathrm{mw}(\pi) = 1$ if and only if π is separable.*

Observation 2.26. *For every non-separable permutation π, the modular-width of π is equal to the length of the largest simple permutation contained in π.*

Therefore, a permutation class $\mathcal{C}$ has bounded modular-width if and only if it contains finitely many simple permutations. This serves as a major motivation for considering modular-width, since classes with finitely many simple permutations have played an important role in many structural and enumerative results. We refer an interested reader to [114, Section 3.2] for a survey on the topic.

Finally, let us discuss the complexity of obtaining substitution decomposition trees. It is not hard to see that the substitution decomposition tree (and, thus, the

modular-width) of π can be computed in polynomial time. In fact, it is possible to compute the substitution decomposition tree without labels in linear time by an algorithm by Xuan, Habib and Paul [51]. We can then label each prime node with d children in time $O(d \log d)$ by sorting its children.

Comparison with other parameters

We first show that the modular-width of a permutation is always lower-bounded by its grid-width.

Proposition 2.27. *For every permutation π, we have* $\mathrm{mw}(\pi) \geq \mathrm{gw}(\pi)$.

Proof. The substitution decomposition tree of π describes how is π obtained via inflations. The linear nodes correspond to inflations of increasing or decreasing sequences while the prime nodes correspond to inflation of simple permutations. For a node v of the tree, let π^v be the permutation obtained by performing the inflations in the subtree rooted in v. Let us prove that $\mathrm{mw}(\pi) \geq \mathrm{gw}(\pi^v)$ by induction on the depth of v.

The grid-width of π^v for a leaf v is 1 which is clearly a lower bound for $\mathrm{mw}(\pi)$. For an inner node v, we have $\pi^v = \sigma[\pi^{w_1}, \ldots, \pi^{w_m}]$ where $w_1, \ldots, w_m$ are the children of v. Therefore due to Lemma 2.7, the grid-width of π^v is equal to $\max(\mathrm{gw}(\sigma), \mathrm{gw}(\pi^{w_1}), \ldots, \mathrm{gw}(\pi^{w_m}))$. We have the bound $\mathrm{mw}(\pi) \geq \mathrm{gw}(\pi^{w_i})$ due to induction. If v is a linear node then σ is a monotone permutation and $\mathrm{gw}(\sigma) = 1$. Otherwise v is a prime node and it follows from the definition of modular-width that $\mathrm{mw}(\pi) \geq |\sigma|$. And finally, $|\sigma|$ is a trivial upper bound for $\mathrm{gw}(\sigma)$.

Observe that the same reasoning guarantees that we can almost use the substitution decomposition tree of π directly as a grid tree for π. It suffices to replace every inner node v that has more than two children with a sufficiently large binary tree and attach the original children of v to its leaves. □

On the other hand, modular-width is incomparable with all the introduced parameters larger than grid-width. Proposition 2.5 shows that permutations of bounded modular-width can have arbitrarily large path-width since every separable permutation has modular-width 1.

Observation 2.28. *For every k, there is a permutation π such that* $\mathrm{mw}(\pi) = 1$ *and* $\mathrm{pw}(\pi) = k$.

On the other hand, we can exhibit permutations that have bounded both horizontal and vertical grid-width while having arbitrarily large modular-width.

We define the *increasing oscillating point set* as $\{(n, a_n) \mid n \in \mathbb{N}\}$ where

$$a_n = \begin{cases} n+3 & \text{if } n \text{ is odd} \\ n-1 & \text{if } n \text{ is even.} \end{cases}$$

See Figure 2.8. The *increasing oscillation* of length m for $m \geq 4$ is the permutation obtained as the reduction of

- the leftmost m points of the increasing oscillating point set if m is even, or
- the bottommost m points of the increasing oscillating point set if m is odd.

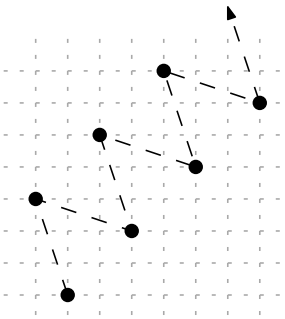

Figure 2.8: The beginning of the increasing oscillating point set. The increasing oscillation of length m is formed by the first m points along the dashed line.

Proposition 2.29. *For every k, there exists a permutation π such that* $\mathrm{hgw}(\pi) = \mathrm{vgw}(\pi) = 3$ *and* $\mathrm{mw}(\pi) \geq k$.

Proof. Consider the increasing oscillation π of length k. Observe that π is a simple permutation as long as $k \geq 4$. It follows that $\mathrm{mw}(\pi) = k$.

On the other hand, let us show that π has small both horizontal and vertical grid-width. Observe that for every $i < k$, the set $S_i = \{(j, \pi_j) \mid j \leq i\}$ is the reduction of $\{(j, a_j) \mid j \leq i\}$, regardless of whether k is even or odd. If i is odd, then the y-coordinates of S_i induce 3 contiguous intervals – the interval $\{1, \ldots, i-2\}$ and two singletons $\{i\}$ and $\{i+2\}$. Otherwise if i is even, then the y coordinates of S_i induce only 2 contiguous intervals – the interval $\{1, \ldots, i-1\}$ and the singleton $\{i+1\}$. The grid-complexity of S_i is thus at most 3 and the horizontal grid-width of π must be exactly 3. We omit a proof of the bound on vertical grid-width as it is a symmetric argument to that above. □

2.2 Containment of grid subclasses

In the following section, our goal is to investigate how the structure of a given permutation class $\mathcal{C}$ affects how large tree-width can be achieved by a permutation of a given length. We have found that lower bounds as these are achieved by showing that $\mathcal{C}$ contains a certain kind of monotone grid subclasses. We have already caught a glimpse of such result in Lemma 2.21.

First, let us formalize what we mean by saying that a permutation class $\mathcal{C}$ has a large tree-width. The *tree-width growth function* of a class $\mathcal{C}$ is defined as

$$\mathrm{tw}_{\mathcal{C}}(n) = \max\{\mathrm{tw}(\pi);\ \pi \in \mathcal{C} \wedge |\pi| = n\}.$$

We could similarly define the growth functions for any other parameter introduced in Section 2.1 using the same notation. However, we choose to investigate tree-width as it carries implications towards algorithms for permutation pattern matching. Let us just briefly remark that $\mathrm{gw}_{\mathcal{C}}(n)$ behaves asymptotically identically to $\mathrm{tw}_{\mathcal{C}}(n)$ by Corollary 2.12 and the same asymptotic lower bounds holds for $\mathrm{cw}_{\mathcal{C}}(n)$ and $\mathrm{ctww}_{\mathcal{C}}(n)$ due to Corollaries 2.15 and 2.19.

The study of fine-grained tree-width bounds was initiated by Berendsohn et al. [23] who have shown that $\mathrm{tw}_{\mathcal{C}}(k) = O(\sqrt{n})$ when $\mathcal{C}$ is the class of 2-monotone permutations (i.e., the permutations merged from two monotone sequences),

and that consequently these classes allow sub-exponential algorithms for pattern matching. They show, however, that for the class Av(4321), the tree-width growth is of order $\Omega(n/\log n)$. Later, Berendsohn [22] formally introduced the tree-width growth function and proved that most classes defined by avoiding a single pattern contain permutations with tree-width almost linear in their length.

Theorem 2.30 ([22, Theorem 3.3])**.** *We have* $\mathrm{tw}_{\mathrm{Av}(\sigma)}(n) = \Omega(n/\log n)$ *for all permutations* σ *such that* $|\sigma| \geq 4$ *and* σ *is not symmetric to any of 3412, 3142, 4213, 4123, 42153, 41352 and 42513.*

Unfortunately, the theorem is proved by showing $\mathrm{tw}_{\mathcal{C}} \in O(n/\log n)$ separately for six different choices of $\mathcal{C}$ and therefore, it does not provide much insight into the structural reasons for large tree-width. Our aim is to provide tools that describe structural properties forcing large tree-width; as a result, we will be able to unify and strengthen all the aforementioned lower bounds.

2.2.1 Long path property

The first property of interest is defined by the containment of monotone grid subclasses with arbitrary long paths in their cell graph. We say that a permutation class $\mathcal{C}$ has the *long path property* if for every k the class $\mathcal{C}$ contains a monotone grid subclass whose cell graph is a path of length k. Additionally, we say that $\mathcal{C}$ has the *computable long path property* if there exists an algorithm that given a positive integer k outputs a monotone gridding matrix $\mathcal{M}$ such that $\mathrm{Grid}(\mathcal{M}) \subseteq \mathcal{C}$ and the cell graph of $\mathcal{M}$ is a path of length k. Finally, $\mathcal{C}$ has the *poly-time computable long path property* if the algorithm runs in time polynomial in k.

Example. Let us show that the class Av(321) has the long path property. In order to do that, we introduce a general way how to build staircase-like grid classes. The *k-step increasing $(\mathcal{C},\mathcal{D})$-staircase*, denoted by $\mathrm{St}_k(\mathcal{C},\mathcal{D})$ is a grid class $\mathrm{Grid}(\mathcal{M})$ of a $k \times (k+1)$ gridding matrix $\mathcal{M}$ such that the only non-empty entries in $\mathcal{M}$ are $\mathcal{M}_{i,i} = \mathcal{C}$ and $\mathcal{M}_{i,i+1} = \mathcal{D}$ for every $i \in [k]$. In other words, the entries on the main diagonal are equal to $\mathcal{C}$ and the entries of the adjacent lower diagonal are equal to $\mathcal{D}$. The *increasing $(\mathcal{C},\mathcal{D})$-staircase*, denoted by $\mathrm{St}(\mathcal{C},\mathcal{D})$, is the union of $\mathrm{St}_k(\mathcal{C},\mathcal{D})$ over all $k \in \mathbb{N}$. See Figure 2.9. Observe that any class $\mathrm{St}(\mathcal{C},\mathcal{D})$ for infinite classes $\mathcal{C}$ and $\mathcal{D}$ has the long path property. It remains to notice that the increasing staircase $\mathrm{St}(\boxslash,\boxslash)$ avoids 321 and thus, Av(321) has the long path property. In fact, Waton [117, Lemma 3.2.7] noticed that Av(321) is equal to $\mathrm{St}(\boxslash,\boxslash)$.

The reason of our particular interest in the long path property is that it is the minimal condition we found to enforce unbounded tree-width. In fact, we conjecture that a permutation class $\mathcal{C}$ has unbounded tree-width if and only if it has the long path property.

Conjecture 2.31. *A permutation class $\mathcal{C}$ has unbounded tree-width if and only if it has the long path property.*

The lower bound builds upon the ideas of Berendsohn et al. [23], who proved a similar result for the class Av(321) using the fact that this class contains a staircase-shaped grid path of arbitrary length. As we shall see later, this lower

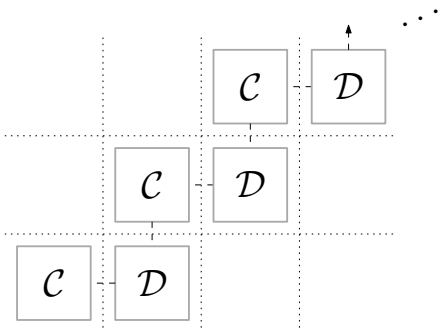

Figure 2.9: The increasing $(\mathcal{C}, \mathcal{D})$-staircase.

bound is tight in general as there exist classes with the long path property and tree-width $O(\sqrt{n})$.

Before proving the lower bound of interest, we need to introduce the most prominent example of graphs with unbounded tree-width. For a positive interger k, the $k \times k$ *grid graph* is the graph with vertex set $[k] \times [k]$ such that vertices (i, j) and (i', j') are joined with an edge if and only if $|i - i'| + |j - j'| = 1$.

Proposition 2.32 (folklore)**.** *The $k \times k$ grid graph has tree-width exactly k.*

Proposition 2.33. *If a permutation class $\mathcal{C}$ has the long path property then*

$$\mathrm{tw}_{\mathcal{C}}(n) \in \Omega(\sqrt{n}).$$

Proof. First, we show that $\mathcal{C}$ contains for every k a grid subclass whose cell graph is a proper-turning path of length k, i.e. a path in which no three consecutive vertices are in the same row or column of the gridding. For contradiction, assume that there is ℓ such that $\mathcal{C}$ does not contain such path of length ℓ. The long path property then implies that $\mathcal{C}$ contains for every t a class $\mathrm{Grid}(\mathcal{M})$ where $\mathcal{M}$ is either a $1 \times t$ or $t \times 1$ matrix without empty entries. However, any such matrix of dimensions $1 \times n$ or $n \times 1$ contains all permutations of length n and thus, $\mathcal{C}$ must actually be the class of all permutations that contains all possible proper turning paths.

So we can suppose that there is a monotone gridding matrix $\mathcal{M}$ such that $\mathcal{M}$ is a proper-turning path $v_1, \ldots, v_{2m-1}$ of length $2m - 1$ and $\mathrm{Grid}(\mathcal{M})$ is contained in $\mathcal{C}$. We explicitly construct a permutation $\pi \in \mathrm{Grid}(\mathcal{M})$ such that G_π contains the $m \times m$ grid graph as a subgraph. See Figure 2.10. The claim then follows by Proposition 2.32.

For $i \in [m]$ and $j \in [i]$, let

$$p_{i,j} = (m + 2j - i - 1, m + 2j - i - 1), \quad p_{2m-i,j} = p_{i,j}.$$

Less formally, the individual tiles along the path contain monotone sequences of lengths $1, 2, \ldots, m - 1, m, m - 1, m - 2, \ldots, 1$ and for two consecutive tiles, their points sets alternate when ordered by either coordinate. We define a family of $2m$-tiles $\mathcal{P}$ by setting $\mathcal{P}_{v_i}$ to be the set of points $p_{i,j}$ for all possible choices of j.

Let $\mathcal{F}$ be a consistent orientation of $\mathcal{M}$ guaranteed by Lemma 1.2 and let π be the $\mathcal{F}$-assembly of $\mathcal{P}$. The sets P_{v_i} were defined in such a way that for every i the points in $P_{v_{2i}}$ have both coordinates odd whereas the points in $P_{v_{2i+1}}$ have both coordinates even. Since $\mathcal{M}$ is a proper turning path, there are always at

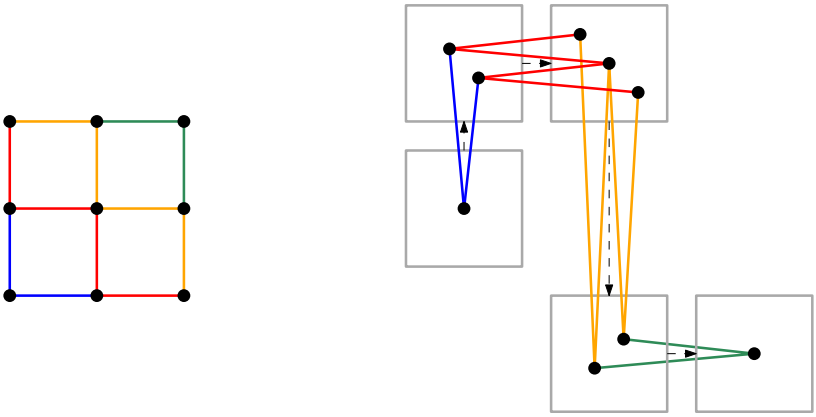

Figure 2.10: Illustration of the proof of Proposition 2.33. Embedding a 3×3 grid graph (left) into a permutation from a monotone grid class whose cell graph is a path of length 5 (right).

most two non-empty tiles sharing the same row or column in π and in such case they correspond to neighboring vertices of the path. Moreover, if they share a common row, then the y-coordinates of their points are interleaved, and if they share a common column, the same holds for the x-coordinates.

It remains to show that G_π contains the $m \times m$ grid graph as a subgraph. Let $s_{i,j}$ be the image of $p_{i,j}$ under the $\mathcal{F}$-assembly. We claim that we can map consecutive diagonals of the grid to the tiles P_{v_i}. More precisely, for $x, y \in [m]$ set

$$g_{x,y} = \begin{cases} s_{x+y-1,x} & \text{if } x + y \leq m + 1, \\ s_{x+y-1,m-y+1} & \text{otherwise.} \end{cases}$$

We start by showing that for any $i \in [m-1]$, there is an edge between $s_{i,j}$ and $s_{i+1,j}$, and also between $s_{i,j}$ and $s_{i+1,j+1}$. This follows since the points of P_{v_i} and $P_{v_{i+1}}$ have their x- or y-coordinates interleaved and there is no other tile occupying their shared row or column. Due to symmetry, it holds that for $i > m$, there is an edge between $s_{i,j}$ and $s_{i-1,j}$ and also between $s_{i,j}$ and $s_{i-1,j+1}$.

If we take $x, y \in [m]$ such that $x + y \leq m$ (i.e. $g_{x,y}$ lies below the anti-diagonal of the grid), the fact proved in the previous paragraph directly translates to the existence of edges between $g_{x,y}$ and $g_{x+1,y}$ and between $g_{x,y}$ and $g_{x,y+1}$. On the other hand for $x, y \in [m]$ such that $x + y \geq m + 2$, the points $g_{x,y}$, $g_{x-1,y}$ and $g_{x,y-1}$ translate to $s_{x+y-1,x}$, $s_{x+y-2,x-1}$ and $s_{x+y-2,x}$. Therefore, in this case there are edges between $g_{x,y}$ and $g_{x-1,y}$ and between $g_{x,y}$ and $g_{x,y-1}$. This concludes the proof as any edge in the $m \times m$ grid graph is of one of the two types whose existence we proved. □

2.2.2 Cycle property

We shall often show that a permutation class $\mathcal{C}$ has the long path property indirectly through a different property defined by containment of grid classes. Namely, a permutation class $\mathcal{C}$ has the *cycle property* if the class $\mathcal{C}$ contains a monotone grid subclass whose cell graph is a cycle. Notice that there is no reason in defining a computable version of the cycle property.

Example. We put forth as the most prominent example the class of the so-called skew-merged permutations defined as the merge of an increasing and a decreasing sequence. It is easy to see that the class of skew-merged permutations is in fact equal to the grid class $\mathrm{Grid}\left(\begin{smallmatrix}\diagdown & \diagup\\ \diagup & \diagdown\end{smallmatrix}\right)$ whose cell graph is a cycle on four vertices.

Proposition 2.34. *If $\mathcal{C}$ has the cycle property then it also has the poly-time computable long path property.*

Before we prove the proposition, we need to introduce some more terminology. We say that a monotone gridding matrix $\mathcal{M}$ is *increasing* if $\mathrm{Grid}(\mathcal{M}) = \{\diagup\}$, it is *decreasing* if $\mathrm{Grid}(\mathcal{M}) = \{\diagdown\}$, and it is *empty* if $\mathrm{Grid}(\mathcal{M}) = \emptyset$. For instance, $\left(\begin{smallmatrix}\cdot & \cdot & \diagup\\ \diagup & \cdot & \cdot\end{smallmatrix}\right)$ or $\left(\begin{smallmatrix}\diagup & \cdot\\ \cdot & \cdot\end{smallmatrix}\right)$ are both increasing gridding matrices, while $\left(\begin{smallmatrix}\diagup & \cdot\\ \cdot & \diagup\end{smallmatrix}\right)$ is neither increasing nor decreasing.

We shall now introduce a variant of the $\mathcal{F}$-assembly acting on matrices. Suppose that $\mathcal{M}$ is a $k \times \ell$ monotone gridding matrix, and let Λ be a $k \times \ell$ family of monotone gridding matrices $\{\mathcal{L}^{i,j} \mid i \in [k], j \in [\ell]\}$ such that for each $i \in [k]$ and $j \in [\ell]$, $\mathcal{L}^{i,j}$ is an $m \times m$ monotone gridding matrix. We say that Λ is *consistent* with $\mathcal{M}$ if $\mathcal{L}^{i,j}$ is empty whenever $\mathcal{M}_{i,j} = \emptyset$, $\mathcal{L}^{i,j}$ is increasing whenever $\mathcal{M}_{i,j} = \diagup$, and $\mathcal{L}^{i,j}$ is decreasing whenever $\mathcal{M}_{i,j} = \diagdown$. We then define the *assembly of* Λ, denoted by $[\Lambda]$, as the $km \times \ell m$ gridding matrix $\mathcal{N}$ with

$$\mathcal{N}_{(i-1)\cdot m+a,(j-1)\cdot m+b} = (\mathcal{L}^{i,j})_{a,b}$$

for every $i \in [k]$, $j \in [\ell]$ and $a, b \in [m]$. Informally, we replace each cell (i, j) of $\mathcal{M}$ by the matrix $\mathcal{L}^{i,j}$. Observe that if Λ is consistent with $\mathcal{M}$, then $\mathrm{Grid}([\Lambda])$ is a subclass of $\mathrm{Grid}(\mathcal{M})$.

If $\mathcal{F} = (f_c, f_r)$ is a $k \times \ell$ orientation, then *applying* $\mathcal{F}$ to a matrix family Λ as above yields a matrix family

$$\mathcal{F}(\Lambda) = \{\Phi_i(\Psi_j(\mathcal{L}^{i,j}));\ i \in [k], j \in \ell\},$$

where Φ_i is an identity if $f_c(i) = 1$ and reversal otherwise, while Ψ_j is an identity if $f_r(j) = 1$ and complement otherwise. The *$\mathcal{F}$-assembly of* Λ is defined as the assembly of $\mathcal{F}(\Lambda)$. Observe that Λ is consistent with $\mathcal{M}$ if and only if $\mathcal{F}(\Lambda)$ is consistent with $\mathcal{F}(\mathcal{M})$ and moreover, the cell graph of $[\Lambda]$ is isomorphic to the cell graph of $[\mathcal{F}(\Lambda)]$.

Furthermore, we shall use the fact that while a monotone gridding matrix $\mathcal{M}$ might not have a consistent orientation we can always find a monotone gridding matrix $\mathcal{N}$ such that $\mathrm{Grid}(\mathcal{N})$ is a subclass of $\mathrm{Grid}(\mathcal{M})$ and they have similar properties. To that end, the *refinement* $\mathcal{M}^{\times 2}$ of a gridding matrix $\mathcal{M}$ is the matrix obtained from $\mathcal{M}$ by replacing every $\diagup$-entry with the 2×2 increasing matrix $\left(\begin{smallmatrix}\cdot & \diagup\\ \diagup & \cdot\end{smallmatrix}\right)$, every $\diagdown$-entry with the decreasing matrix $\left(\begin{smallmatrix}\diagdown & \cdot\\ \cdot & \diagdown\end{smallmatrix}\right)$ and every empty entry with the empty 2×2 matrix. One can also view the refinement as a special case of assembly. Clearly, $\mathrm{Grid}(\mathcal{M}^{\times 2})$ is contained in $\mathrm{Grid}(\mathcal{M})$. And as it was observed by Albert et al. [7], $\mathcal{M}^{\times 2}$ always admits a consistent orientation. We include the very brief proof for the sake of completeness.

Lemma 2.35 ([7, Proposition 4.1])**.** *For every monotone gridding matrix $\mathcal{M}$, the refinement $\mathcal{M}^{\times 2}$ admits a consistent orientation.*

Proof. Consider the orientation $\mathcal{F} = (f_c, f_r)$ defined as $f_c(i) = (-1)^i$ and $f_r(j) = (-1)^j$. The consistency of $\mathcal{F}$ follows since the entries of every inflated $\boxslash$-entry occupy cells (i, j) such that the sum $i + j$ is even, while it is odd for the cells occupied by the entries of every inflated $\boxbslash$-entry. □

Proof of Proposition 2.34. Let $\mathcal{M}$ be a gridding matrix with the following properties:

(i) $\mathrm{Grid}(\mathcal{M}) \subseteq \mathcal{C}$,
(ii) the cell graph of $\mathcal{M}$ contains a cycle,
(iii) $\mathcal{M}$ has a consistent orientation $\mathcal{F}$,
(iv) $\mathcal{M}$ is minimal with respect to these conditions, i.e., changing any non-empty entry of $\mathcal{M}$ to $\emptyset$ yields a gridding matrix whose cell graph is acyclic.

Let us first show that such $\mathcal{M}$ exists. The cycle property guarantees an existence of a gridding matrix $\mathcal{M}'$ with the properties (i), (ii) and (iv). Note that the property (iv) implies that every vertex of $G_{\mathcal{M}'}$ corresponds to a *corner*, i.e. it has one neighbor in the same row and the other in the same column.

Consider the refinement $\mathcal{M}'^{\times 2}$ which possesses a consistent orientation by Lemma 2.35. We claim that while the cell graph $G_{\mathcal{M}'^{\times 2}}$ might not be connected, its every connected component must be a cycle. This follows since both of the new vertices created by replacing a single non-empty entry with a 2×2 matrix inherit the degree of the original vertex in $G_{\mathcal{M}'}$. Furthermore, there are no vertices of degree one due to property (iv) and therefore, all vertices have degree exactly two and every connected component is a cycle. It suffices to take as $\mathcal{M}$ any connected component of $G_{\mathcal{M}'^{\times 2}}$.

Let $\mathcal{F}$ be a consistent orientation of $\mathcal{M}$ and let x be any vertex on the cycle in $G_{\mathcal{M}}$. For a natural number k, we show how to define a family of matrices Λ that will create upon its $\mathcal{F}$-assembly a gridding matrix whose cell graph is a path of length at least k. We define $\mathcal{L}^x$ as the increasing $k \times k$ matrix with non-empty entries of the form $(i, i+1)$ for $i \in [k-1]$. And for every other vertex v of $G_{\mathcal{M}}$, we simply take as $\mathcal{L}^v$ the increasing matrix with k non-empty entries on its diagonal.

Let $\mathcal{N}$ be the $\mathcal{F}$-assembly of Λ. It follows that $\mathrm{Grid}(\mathcal{N})$ is a subclass of $\mathrm{Grid}(\mathcal{M})$ (and thus of $\mathcal{C}$). Therefore, it suffices to check that $G_{\mathcal{N}}$ is a path. With that in mind, let y be the neighbor of x in $G_{\mathcal{M}}$ sharing a common row with x and let z be the other neighbor of x. We call the path from y to z along the cycle that avoids x the *arc*. Observe that the vertex originating from the (i, i) cell in $\mathcal{L}^y$ is connected by path to the vertex originating from the (i, i) cell in $\mathcal{L}^w$ for every vertex w on the arc, in particular for z. The vertex originating from the (i, i) cell in $\mathcal{L}^z$ is then neighbor of the vertex originating from the $(i, i+1)$ cell in $\mathcal{L}^x$. Finally, this vertex is connected to the vertex originating from the $(i+1, i+1)$ cell in $\mathcal{L}^y$ and we see that all vertices in $\mathcal{N}$ form a single long path. See Figure 2.11. Notice that we actually described how to construct $\mathcal{N}$ in time polynomial in k which proves that $\mathcal{N}$ has the poly-time computable long path property. □

Finally, we exhibit an example of a permutation class that has the long path property but lacks the cycle property.

Proposition 2.36. *The class* $\mathrm{Av}(321)$ *does not have the cycle property.*

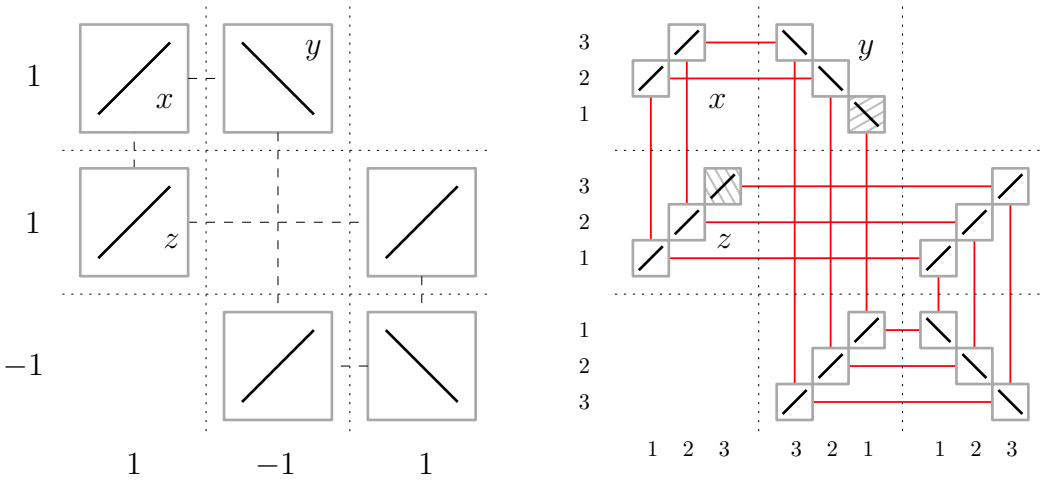

Figure 2.11: Left: a gridding matrix $\mathcal{M}$ whose cell graph is a cycle together with a consistent orientation $\mathcal{F}$ given by the numbers along the edges. Right: a gridding matrix $\mathcal{N}$ whose grid class is contained in $\mathrm{Grid}(\mathcal{M})$ and whose cell graph forms one long path. The numbers along the edges are the coordinates of the original cells under matrix $\mathcal{F}$-assembly and we highlighted the endpoints of the path.

Proof. Assume for a contradiction that there is a monotone gridding matrix $\mathcal{M}$ such that $G_{\mathcal{M}}$ is a cycle and $\mathrm{Grid}(\mathcal{M})$ is contained in $\mathrm{Av}(321)$. We first notice that any non-empty entry in $\mathcal{M}$ must be increasing as the class $\boxdot$ itself contains the forbidden pattern 321.

Let i be the index of the rightmost non-empty column in $\mathcal{M}$ and let j be the largest integer such that the entry $\mathcal{M}_{i,j}$ is non-empty. Since (i, j) is participating in the cycle it must have degree 2 and therefore, there must be $i' < i$ and $j' < j$ such that both entries $\mathcal{M}_{i',j}$ and $\mathcal{M}_{i,j'}$ are non-empty. However, these three entries alone form the class $\mathrm{Grid}\left(\begin{smallmatrix} \boxslash & \boxslash \\ \cdot & \boxslash \end{smallmatrix}\right)$ which contains 321. □

2.2.3 Deep tree property

We continue with a strengthening of the long path property. For integer constants c and d, a *c-subdivided binary tree of depth d* is a graph obtained from a binary tree of depth d by replacing every edge by a path of length at most c. We say that a permutation class $\mathcal{C}$ has the *deep tree property* if there is a constant c such that for every d, the class $\mathcal{C}$ contains a monotone grid subclass whose cell graph is a c-subdivided binary tree of depth d. Similar to before, we say that $\mathcal{C}$ has the *(poly-time) computable deep tree property* if there is a (poly-time) algorithm that outputs monotone grid subclass of $\mathcal{C}$ whose cell graph is a c-subdivided tree of depth d.

Observation 2.37. *If a permutation class $\mathcal{C}$ has the deep tree property then it also has the long path property.*

Example. We claim that the class $\mathrm{Av}(4321)$ has the deep tree property. First, observe that the increasing $(\boxslash, \mathrm{Grid}(\boxslash, \boxslash))$-staircase avoids 4321 and thus, is a subclass of $\mathrm{Av}(4321)$. The deep tree property follows since we can embed arbitrarily deep 3-subdivided tree by using the lower diagonal cells of the staircase to branch. See Figure 2.12.

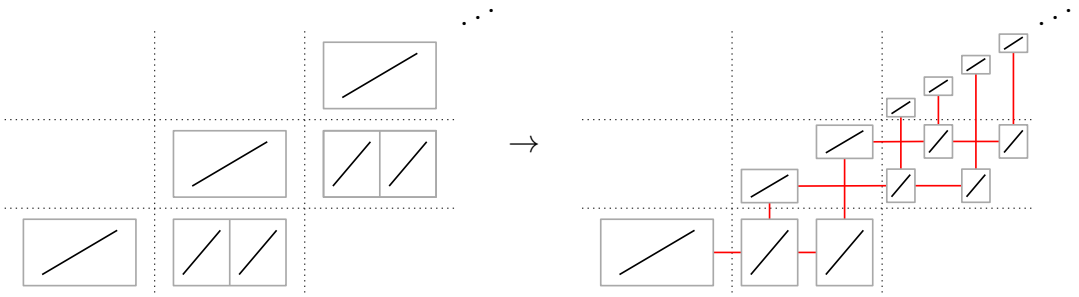

Figure 2.12: Construction of a grid subclass of the class $\mathrm{St}(\boxslash, \mathrm{Grid}(\boxslash, \boxslash))$ whose cell graph is a 3-subdivided tree with four leaves.

The main motivation for the introduction of the deep tree property is that it enforces linear tree-width up to a logarithmic factor. Inspired by the approach of Berendsohn [22], we want to show that for every graph G , we can find a permutation $\pi \in \mathcal{C}$ such that G_π contains G as a minor while the length of π exceeds the size of G by at most a logarithmic factor. The desired bound follows since there are sparse graphs with linear tree-width.

Lemma 2.38. *Let $\mathcal{C}$ be a permutation class with the deep tree property. For every connected graph G with n vertices and m edges, there exists a permutation $\pi \in \mathcal{C}$ of length $O(m \log m)$ such that G is a minor of G_π.*

Proof. Suppose that the vertex set of G is $V_G = [n]$ and we arbitrarily number its edges as $\{e_1, \ldots, e_m\}$ where $e_i = \{a_i, b_i\}$. Let $\mathcal{M}$ be a monotone gridding matrix such that $\mathrm{Grid}(\mathcal{M}) \subseteq \mathcal{C}$ and the cell graph of $\mathcal{M}$ is a c-subdivided binary tree with exactly m leaves. Let r denote the root of this tree. It follows that the tree has maximal depth at most $c(\log m + 1)$. We turn $G_\mathcal{M}$ into an oriented graph by orienting all edges consistently away from r. For any vertex v of the tree, the *descendants of* v, denoted by $D(v)$, are all the out-neighbors of v.

We assign a set $A_w \subseteq V_G$ to each vertex w of the tree. First, we arbitrarily order the m leaves of $G_\mathcal{M}$ as $v_1, \ldots, v_m$. Then we inductively define

$$A_w = \begin{cases} \{a_i, b_i\} & \text{if } w = v_i \text{ for } i \in [m] \text{ where } e_i = \{a_i, b_i\}, \\ \bigcup_{v \in D(w)} A_v & \text{otherwise.} \end{cases} \tag{2.1}$$

We remark that $\sum_v |A_v| \in O(m \log m)$ since each vertex $i \in V_G$ is present in exactly $\deg(i)$ leaves and in the paths of length $O(\log m)$ that connect those leaves to r. We proceed to define a family of m-tiles $\mathcal{P}$ by setting $P_v = \{(i, i) \mid i \in A_v\}$ for every vertex v of the tree, and keeping all the other tiles empty.

Let $\mathcal{F}$ be a consistent orientation obtained from the application of Lemma 1.2 on $\mathcal{M}$ and let π be the $\mathcal{F}$-assembly of $\mathcal{P}$. See Figure 2.13. Since every tile is an increasing point set, it follows that π belongs to $\mathrm{Grid}(\mathcal{M})$.

In order to simplify the rest of the proof, we color π with n colors. We assign a color $i \in V_G$ to a point $p \in \pi$ with preimage (i, i) in $P_{x,y}$. We claim that π satisfies the following conditions:

(a) The subgraph of G_π induced by a single color is connected;
(b) For each edge $e_i = \{a_i, b_i\}$ of G, there is an edge in G_π between a vertex of color a_i and a vertex of color b_i.

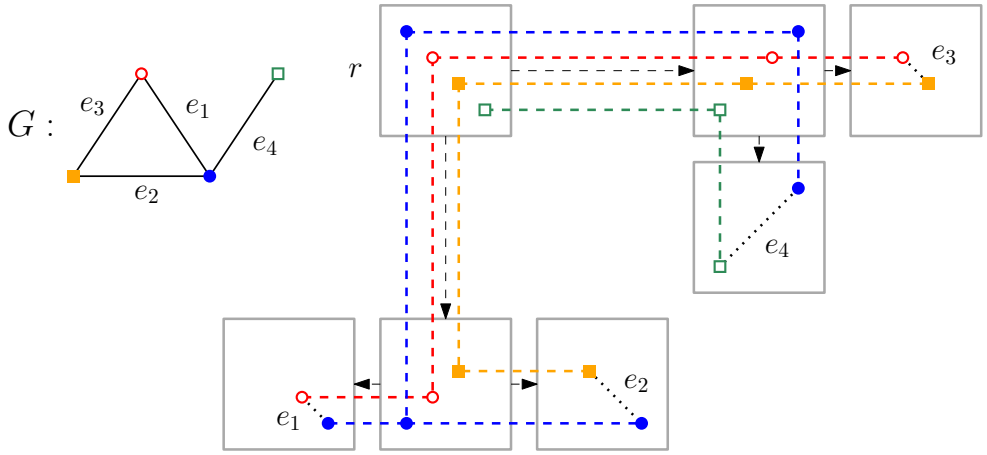

Figure 2.13: Illustration of the proof of Proposition 2.39. Embedding a graph G with four edges (top left) into a permutation from a monotone grid class whose cell graph is a tree with four leaves rooted in vertex r (right).

Fix a color $i \in V_G$. Let Q_i be the set of all vertices v of $G_{\mathcal{M}}$ such that $i \in A_v$. Clearly, Q_i induces a connected subtree of $G_{\mathcal{M}}$. Recall that every point of color i has always the coordinates (i, i) inside any tile. It follows that for points (i, i) in two neighboring tiles, the $\mathcal{F}$-assembly of $\mathcal{P}$ transforms them first to points that share one coordinate and then by rotating slightly clockwise makes them either horizontal or vertical neighbors. Therefore, the subgraph of G_π induced by color i is connected, which proves (a).

Every leaf v_i must be the only non-empty vertex in its row or column. Let us assume the latter case as the other one is symmetric. Therefore, the two points contained in the image of P_{v_i} form an edge in G_π since no other point lies in the vertical strip between them. In particular, the leaf v_i satisfies the condition (b) for the edge e_i.

The conditions (a) and (b) together imply that we can obtain a supergraph of G by contracting every monochromatic subgraph of G_π to a single vertex and thus, G is a minor of G_π. Observe that the total length of π is equal to $\sum_v |A_v|$ which we showed to be $O(m \log m)$. □

Proposition 2.39. *If a permutation class $\mathcal{C}$ has the deep tree property, then*

$$\mathrm{tw}_{\mathcal{C}}(n) \in \Omega(n/\log n).$$

Proof. It is well known that $\mathrm{tw}(H) \leq \mathrm{tw}(G)$ for any minor H of a graph G (see, e.g., [104]). Suppose we have an infinite family of graphs $\mathcal{G}$ such that any $G \in \mathcal{G}$ on n vertices has $O(n)$ edges and moreover, the tree-width of G is $\Theta(n)$. It follows by application of Lemma 2.38 on $\mathcal{G}$ and $\mathcal{C}$ that $tw_{\mathcal{C}}(n) \in \Omega(n/\log n)$. We conclude the proof by stating that these properties are attained by any family of the so-called expander graphs (see [76]). Note that we shall introduce expanders more properly in the upcoming subsection. □

We remark that the deep tree property is not the only cause of near linear tree-width growth. Recall the definition of increasing $(\mathcal{C}, \mathcal{D})$-staircase $\mathrm{St}(\mathcal{C}, \mathcal{D})$ and let $\mathcal{L}$ denote the class of layered permutations. It can be shown that the tree-width growth of the class $\mathrm{St}(\boxslash, \mathcal{L})$ is at least $\Omega(n/\log^2 n)$ even though $\mathrm{St}(\boxslash, \mathcal{L})$ does not have the deep tree property.

And to conclude our investigation of deep tree property, we note that it is incomparable with the cycle property. We shall see later that there are classes with the cycle property that cannot have the deep tree property as their tree-width growth is $\Theta(\sqrt{n})$ (Theorem 6.1). For now, we exhibit an example of the opposite case, i.e. a class $\mathcal{C}$ that has the deep tree property but lacks the cycle property.

Lemma 2.40. *The increasing* $(\boxslash, \mathrm{Grid}(\boxslash, \boxslash))$*-staircase has the deep tree property but lacks the cycle property.*

Proof. We had already shown that the class $\mathrm{St}(\boxslash, \mathrm{Grid}(\boxslash, \boxslash))$ has the deep tree property at the beginning of this subsection. On the other hand, if we assume that $\mathrm{St}(\boxslash, \mathrm{Grid}(\boxslash, \boxslash))$ has the cycle property then it has to contain the grid class $\mathrm{Grid}\left(\begin{smallmatrix}\boxslash & \boxslash \\ \cdot & \boxslash\end{smallmatrix}\right)$ via the same argument as in the proof of Proposition 2.36. It suffices to mechanically check that the permutation 463152 belongs to $\mathrm{Grid}\left(\begin{smallmatrix}\boxslash & \boxslash \\ \cdot & \boxslash\end{smallmatrix}\right)$ but not to the staircase class. □

2.2.4 Bicycle property

As a next step, we introduce an even stronger property than the deep tree property which is often easier to exhibit for a specific class $\mathcal{C}$. A permutation class $\mathcal{C}$ has the *bicycle property* if it contains a monotone grid subclass whose cell graph is connected and contains at least two cycles.

It is perhaps quite unsurprising that the bicycle property implies the deep tree property. However, the proof is quite long and technical since it has to deal with all kinds of possible configurations of two cycles in a cell graph of a monotone grid class.

Proposition 2.41. *If a permutation class* $\mathcal{C}$ *has the bicycle property, then it also has the poly-time computable deep tree property.*

Before the actual proof, let us show that we can assume additional properties of the grid subclass of $\mathcal{C}$ with two cycles in its cell graph. Namely, there are two possible configurations of two cycles in a graph – either two cycles connected by a path, or one cycle with a (possibly subdivided) chord. We show that, in fact, we can without loss of generality assume that the former holds. Note that we consider the degenerate case of two cycles joined via a vertex of degree four as two cycles connected by a path of zero length.

Lemma 2.42. *For every permutation class* $\mathcal{C}$ *with the bicycle property, there exists a monotone gridding matrix* $\mathcal{M}$ *with a consistent orientation* $\mathcal{F}$ *such that* $\mathrm{Grid}(\mathcal{M})$ *is a subclass of* $\mathcal{C}$ *and the cell graph* $G_{\mathcal{M}}$ *consists of two edge-disjoint cycles connected by a path (possibly of zero length).*

Proof. Let $\mathcal{M}$ be a monotone gridding matrix with the following properties:

(i) $\mathrm{Grid}(\mathcal{M}) \subseteq \mathcal{C}$,
(ii) the cell graph of $\mathcal{M}$ contains two connected cycles,
(iii) $\mathcal{M}$ has a consistent orientation $\mathcal{F}$,
(iv) $\mathcal{M}$ is minimal with respect to these conditions, i.e., changing any non-empty entry of $\mathcal{M}$ to $\emptyset$ yields a gridding matrix violating (ii).

The bicycle property guarantees an existence of a monotone gridding matrix $\mathcal{M}'$ with properties (i), (ii) and (iv). Recall that a corner is any vertex of $G_{\mathcal{M}'}$ of degree two which has one neighbor in the same row and the other in the same column, and observe that every vertex of degree two in $G_{\mathcal{M}'}$ must in fact be a corner as a result of property (iv).

Consider the refinement $\mathcal{M}'^{\times 2}$ of $\mathcal{M}'$ which admits a consistent orientation by Lemma 2.35. We claim that while the cell graph $G_{\mathcal{M}'^{\times 2}}$ might not be connected, its every connected component in fact contains at least two cycles. This follows since both of the new vertices created by replacing a single non-empty entry with a 2×2 matrix inherit the degree of the original vertex in $G_{\mathcal{M}'}$. There are no vertices of degree one due to our choice of $\mathcal{M}'$ as a minimal matrix with the property and moreover, there is at least one vertex of degree at least 3. Thus, we can find two connected cycles by walking from a single vertex in three different directions. We take as $\mathcal{M}$ any connected component of $G_{\mathcal{M}'^{\times 2}}$.

We are done if $G_{\mathcal{M}}$ consists of two cycles joined by a path. So let us assume that $G_{\mathcal{M}}$ forms a cycle with a chord and let u and v be the two vertices with degree three. We call the shortest of the three paths between u and v the *chord of* $G_{\mathcal{M}}$ and the other two parts the *arcs of* $G_{\mathcal{M}}$. We orient the arcs such that they together form an oriented cycle. Since the chord is the shortest of the three parts, there exists at least one corner on each arc – let us choose arbitrarily one corner from each arc, and denote them by x and y.

We define a family of matrices Λ where each $\mathcal{L}^{i,j}$ is an increasing 3×3 matrix that is a symmetry of one of the four following types

$$P = \begin{pmatrix} \cdot & \cdot & \cdot \\ \cdot & \boxslash & \cdot \\ \boxslash & \cdot & \cdot \end{pmatrix}, \quad Q = \begin{pmatrix} \cdot & \cdot & \boxslash \\ \cdot & \boxslash & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix}, \quad R = \begin{pmatrix} \cdot & \boxslash & \cdot \\ \boxslash & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix}, \quad S = \begin{pmatrix} \cdot & \cdot & \cdot \\ \cdot & \boxslash & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix},$$

where the dots denote the empty entries. In particular, for every vertex z on the oriented path from x to y (excluding the endpoints), we set $\mathcal{L}^z = P$. For every vertex z on the oriented path from y to x (again excluding the endpoints), we set $\mathcal{L}^z = Q$. For every vertex z on the chord (again excluding x and y), we set $\mathcal{L}^z = S$. We set $\mathcal{L}^x = R$ if x and its predecessor on the cycle share a common row, otherwise we set $\mathcal{L}^x = R^{-1}$. And finally, $\mathcal{L}^y = R^{-1}$ if y and its predecessor on the cycle share a common row, otherwise $\mathcal{L}^y = R$. Let $\mathcal{N}$ be the matrix $\mathcal{F}$-assembly of Λ.

Observe that when we limit our view only to the arcs in $\mathcal{N}$, the non-empty entries actually form two disjoint cycles. One cycle contains what was originally the $(1,1)$ cell of $\mathcal{L}^z$ for every z on the path from x to y and the $(2,2)$ cell of $\mathcal{L}^z$ for every z on the path from y to x. The other cycle contains the $(2,2)$ cells on the path from x and y and the $(3,3)$ cells on the path from y to x. The matrices $\mathcal{L}^x$ and $\mathcal{L}^y$ act as switches. Most importantly, every $\mathcal{L}^z$ for z in the chord has a single non-empty entry in the $(2,2)$ cell which establishes the connection between the two cycles. See Figure 2.14. □

Proof of Proposition 2.41. By Lemma 2.42, there is a gridding matrix $\mathcal{M}$ with a consistent orientation $\mathcal{F}$ such that $\mathrm{Grid}(\mathcal{M})$ is a subclass of $\mathcal{C}$ and $G_{\mathcal{M}}$ contains two cycles joined with a path. Let u and v be the vertices of $G_{\mathcal{M}}$ with degree 3 and let us orient each cycle arbitrarily. If the cell graph consists of two cycles sharing a vertex, we simply set u, v to both be this vertex of degree 4. We pick one corner on each cycle, let x be a corner on the cycle incident with u and y a

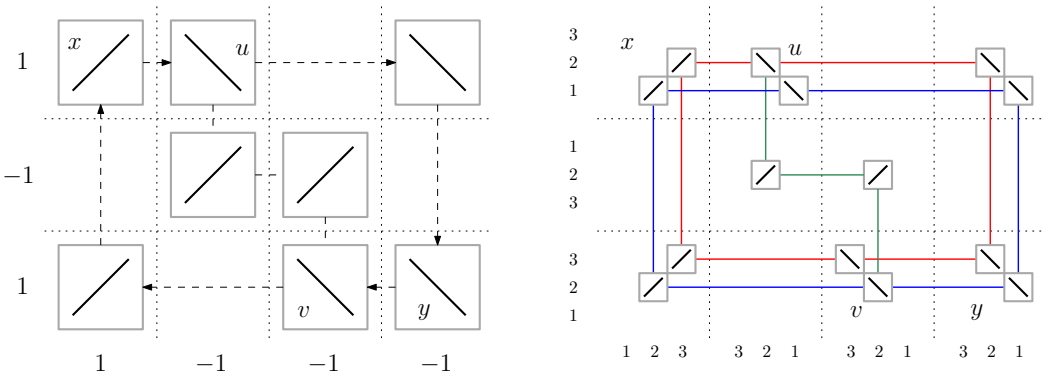

Figure 2.14: Left: a gridding matrix $\mathcal{M}$ whose cell graph is a cycle with a chord together with a consistent orientation $\mathcal{F}$ given by the numbers along the edges. Right: a gridding matrix $\mathcal{N}$ whose grid class is contained in Grid($\mathcal{M}$) and whose cell graph consists of two cycles joined by a path. The numbers along the edges are the coordinates of the original cells under matrix $\mathcal{F}$-assembly.

corner on the cycle incident with v. These corners shall play the role of some sort of switches.

Let T be a complete binary rooted tree of depth d and let $m = 2^d - 1$ denote the number of vertices in T. We identify the vertices of T with the set $[m]$ using a breadth-first search numbering. Formally, the root is the integer 1 and the remaining vertices are numbered by layers and in each layer from left to right. Let $L(i)$ and $R(i)$ denote the left and the right child of a vertex i in T. Observe that we have $L(i) = 2i$ and $R(i) = 2i + 1$.

We now define a family of matrices Λ that will create a subdivided T upon its $\mathcal{F}$-assembly (see Figure 2.15). Each $\mathcal{L}^{i,j}$ is the following $2m \times 2m$ matrix.

- For any vertex z on the path from v to x including u and v but excluding x, we define $\mathcal{L}^z$ as the whole identity matrix with $2m$ non-empty entries on its diagonal.
- For z on the path from v to y excluding v and y, we define $\mathcal{L}^z$ as the matrix with non-empty entries of the form $(2i - 1, 2i - 1)$ for $i \in [m]$.
- For z on the path from y to v (again without its endpoints), we define $\mathcal{L}^z$ as the matrix with non-empty entries of the form $(2i, 2i)$ for $i \in [m]$.
- For the corner y, we set $\mathcal{L}^y$ to contain the non-empty entries $(2i - 1, 2i)$ if y and its predecessor on the cycle share a common row, otherwise its non-empty entries are of the form $(2i, 2i - 1)$ for $i \in [m]$.
- For z on the path from x to u (excluding x and u), we define $\mathcal{L}^z$ as the matrix with non-empty entries of the form $(2i - 1, 2i - 1)$ for $i \in [m]$.
- Finally for the corner x, we set $\mathcal{L}^x$ to contain the non-empty entries $(2 \cdot L(i) - 1, 2i - 1)$ and $(2 \cdot R(i) - 1, 2i)$ for every i that is not a leaf in T if x and its predecessor on the cycle share a common row, otherwise we swap the x- and y-coordinates.

First, we need to verify that every matrix is increasing. Luckily, that is obvious in all cases except for $\mathcal{L}^x$ and in that case it follows from the chosen labeling of

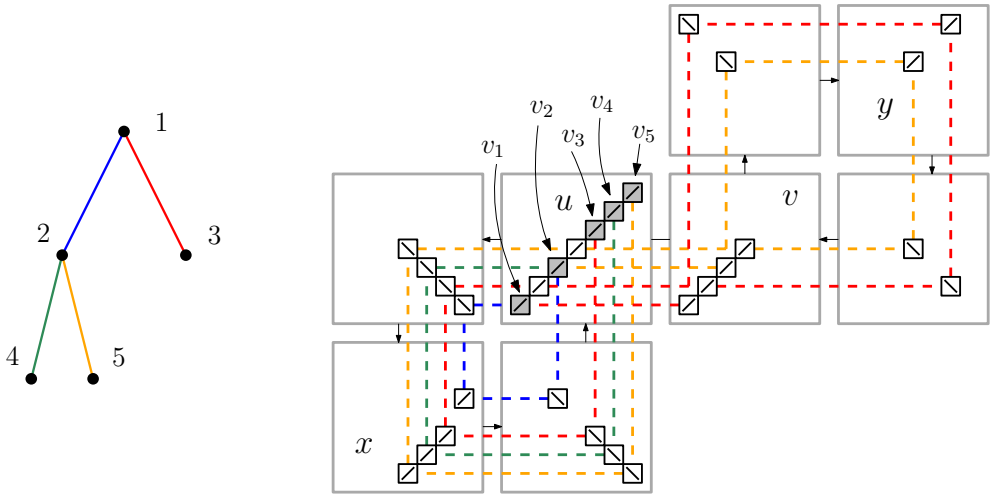

Figure 2.15: Embedding a subdivision of the tree T on the left in a grid class whose cell graph consists of two cycles joined by a path, on the right. The grey cells in the submatrix obtained by replacing u denote precisely the vertices corresponding to vertices of T, the subdivided edges are highlighted using matching colors. The cells not participating in the subdivision of T are omitted.

the vertices in T. Let $\mathcal{N}$ be the $\mathcal{F}$-assembly of Λ and let v_i for $i \in [m]$ be the vertex of $G_{\mathcal{N}}$ originating from the $(2i-1, 2i-1)$ cell in $\mathcal{L}^u$. We claim that for every i that is not a leaf in T, there are paths of constant length from v_i to both $v_{L(i)}$ and $v_{R(i)}$ and moreover, all these paths are disjoint. It then follows that $G_{\mathcal{N}}$ contains the desired subdivision of T.

Let us start with the vertex v_i for an inner vertex i of T. Following the path from u to x in $G_{\mathcal{M}}$, we inductively see that v_i is connected to the vertices originating from the $(2i-1, 2i-1)$ cells. The submatrix corresponding to the original vertex x then acts as a switch using the $(2 \cdot L(i) - 1, 2i - 1)$ cell and we can extend the path to all the vertices originating from the $(2 \cdot L(i) - 1, 2 \cdot L(i) - 1)$ cells on the path from x back to u. Therefore, we found a path of constant length that connects v_i to $v_{L(i)}$.

In the case of the right child, we first follow the path from u to v and subsequently to y using in every step the vertex arising from the $(2i-1, 2i-1)$ cell. The corner y again acts as a switch via the $(2i-1, 2i)$ cell and we extend the path all the way back through v to u using the $(2i, 2i)$ cells. From there, we follow the path to x and then back to u. Using the same arguments as for the left child, we see that we end up in the vertex $v_{R(i)}$. It is easy to see that the paths corresponding to different edges are indeed pairwise disjoint and thus, we verified the deep tree property. □

Improved lower bound on tree-width

We will now show that the bicycle property in fact implies an existence of permutations with linear tree-width. In order to do that, we shall introduce the notion of expanders, connect them to large tree-width and then exhibit that there are expanders that can be embedded in any class with the bicycle property.

Let G be a graph. For a subset of its vertices X, we let the *sphere* around X, denoted by $S_G(X)$, be the set of all vertices outside of X that are adjacent to a

vertex in X. We drop the subscript when it is clear which graph we consider. We define the *vertex expansion of* G as

$$\mathrm{vx}(G) = \min_{\substack{X \subseteq V(G), \\ 0<|X|\leq \frac{1}{2}\cdot|V(G)|}} \frac{|S_G(X)|}{|X|}$$

if $|V(G)| \geq 2$ and $\mathrm{vx}(G) = 0$ otherwise. We say that an infinite set of graphs $\mathcal{G}$ is a *family of (ϵ-)expanders* if there is a constant $\epsilon > 0$ such that for every graph $G \in \mathcal{G}$ we have $\mathrm{vx}(G) \geq \epsilon$.

Expanders exhibit many interesting properties and have found numerous applications in both mathematics and computer science. We refer an interested reader, e.g., to [82] for a survey on this topic. An important property from our point of view is that expanders have large tree-width.

Proposition 2.43 ([76, Proposition 1])**.** *For every n-vertex graph G, we have*

$$\mathrm{tw}(G) \geq \left\lfloor \tfrac{1}{4} \cdot \mathrm{vx}(G) \cdot n \right\rfloor .$$

Our aim is to take previously discovered expanders of very limited structure and transform them into expanding incidence graphs of permutations from a given class with the bicycle property. First, let us introduce the restricted class of graphs which contains expanders. A *partial monotone function on* $[n]$ is a function $f : [n] \to [n] \cup \{\perp\}$ (we interpret $f(x) = \perp$ as f being undefined at x) such that whenever f is defined at two distinct values $x, y \in [n]$ and $x < y$, we have $f(x) < f(y)$. A bipartite graph G with the vertex set $[2] \times [n]$ is *d-monotone* if there exist partial monotone functions $f_1, \ldots, f_d$ such that the edges of G are exactly all the pairs $(1, x), (2, f_i(x))$ for $i \in [d]$ and $x \in [n]$ whenever $f_i(x)$ is defined.

Bourgain and Yehudayoff [40] provided a (perhaps surprising) construction of a family of $O(1)$-monotone expanders. This was later reduced by Dujmović et al. [60] to a family of 3-monotone expanders. Note that this is the best possible since it is well known that 2-monotone graphs are planar [24], but planar graphs have $O(\sqrt{n})$ separators [96] and thus, too small tree-width to be expanders. We have to first define a slightly different notion of expansion for bipartite graphs in order to be compatible with these results.

A bipartite graph G with bipartition A, B is a *two-sided bipartite ϵ-expander* if $|A| = |B|$, and for every $X \subset A$ with $|X| \leq \frac{1}{2} \cdot |A|$ we have $|S(X)| > (1 + \epsilon) \cdot |X|$, and symmetrically for every $Y \subset B$ with $|Y| \leq \frac{1}{2} \cdot |B|$ we have $|S(Y)| > (1+\epsilon) \cdot |Y|$. We are now ready to state the exact result proved by Dujmović et al. [60].

Theorem 2.44 ([60, Theorem 2])**.** *There is an infinite family of two-sided 3-monotone bipartite expanders.*

Unsurprisingly, any two-sided bipartite expander has large vertex expansion. And therefore, Theorem 2.44 provides us an infinite family of expanders even in the sense of vertex expansion.

Lemma 2.45. *For any two-sided bipartite ϵ-expander G we have* $\mathrm{vx}(G) \geq \frac{1}{2} \cdot \epsilon$.

Proof. Let us check that $\mathrm{vx}(G) \geq \frac{1}{2} \cdot \epsilon$. Suppose that A, B is the bipartition of G and let $X \subset V(G)$ be a set of vertices such that $0 < |X| \leq \frac{1}{2} \cdot V(G)$. We may assume without loss of generality that $|A \cap X| \geq |B \cap X|$. Observe that $|A \cap X| \geq \frac{1}{2} \cdot |X|$ while $|B \cap X| \leq \frac{1}{2} \cdot |X|$

First, suppose that $|A \cap X| \leq \frac{1}{2} \cdot |A|$. In this case, we can apply the expansion property of G on $X \cap A$ to obtain a large set of its neighbors in B. We then just need to subtract all the vertices of $B \cap X$ to get a lower bound on the sphere of X. Putting it all together, we get

$$\begin{aligned} |S(X)| &\geq |S(X) \cap B| \geq (1+\epsilon) \cdot |A \cap X| - |B \cap X| \\ &\geq (1+\epsilon) \cdot \tfrac{1}{2} \cdot |X| - \tfrac{1}{2} \cdot |X| \geq \tfrac{1}{2} \cdot \epsilon \cdot |X|. \end{aligned}$$

Otherwise, we must have $|A \cap X| > \frac{1}{2} \cdot |A|$. Notice that now $|B \cap X| \leq \frac{1}{2} \cdot |B| = \frac{1}{2} \cdot |A|$. Let X' be an arbitrary subset of $X \cap A$ of size $\frac{1}{2} \cdot |A|$. We can now use the expansion property of G on X' and see that

$$\begin{aligned} |S(X)| &\geq |S(X) \cap B| \geq (1+\epsilon) \cdot |X'| - |B \cap X| \\ &\geq (1+\epsilon) \cdot \tfrac{1}{2} \cdot |A| - \tfrac{1}{2} \cdot |A| \geq \tfrac{1}{2} \cdot \epsilon \cdot |A| \geq \tfrac{1}{2} \cdot \epsilon \cdot |X|. \end{aligned}$$ □

Theorem 2.46. *Let $\mathcal{C}$ be a permutation class with the bicycle property. Then there exists an infinite set of permutations $\{\pi_i\}_{i \in \mathbb{N}} \subset \mathcal{C}$ such that their incident graphs $\{G_{\pi_i}\}_{i \in \mathbb{N}}$ form an infinite family of expanders.*

Proof. Due to Lemma 2.42, there exists a monotone gridding matrix $\mathcal{M}$ with a consistent orientation $\mathcal{F}$ such that $\mathrm{Grid}(\mathcal{M}) \subseteq \mathcal{C}$ and the cell graph $G_{\mathcal{M}}$ consists of two cycles joined with a path. Let u, v be the two vertices that connect each cycle to the path, or both equal to the one vertex shared by the two cycles. By taking the minimal $\mathcal{M}$ we are guaranteed that every vertex of degree two is a corner. Moreover, any cycle in a cell graph has even length and therefore, there is a vertex x on the cycle with u such that both paths from x to u are of the same length. Similarly, let y be the vertex on the cycle with v such that both paths from y to v are of the same length. Let s_1, s_2 be arbitrary corners, one on each path from x to u, and let t_1, t_2 be arbitrary corners, one on each path from y to v. Finally, we define an orientation of $G_{\mathcal{M}}$ by orienting both paths from x to u towards u, orienting the path between u and v towards v and finally orienting both paths from v to y towards y. Observe that any oriented path from x to y contains the same number of vertices — let us denote this quantity by m.

Before we delve into the construction of expanders, we need to describe a representation of the partial monotone functions as permutations. Let f be a partial monotone function on $[n]$. An *order encoding of f* is a bijection $g : [2] \times [n] \to [2n]$ with two additional properties. First, the image of each copy of $[n]$ is in the correct order, i.e. for every $\alpha \in [2]$ and two distinct $i, j \in [n]$ such that $i < j$, we have $g(\alpha, i) < g(\alpha, j)$. And moreover, we have $g(2, f(i)) = g(1, i) + 1$ for every $i \in [n]$ where $f(i)$ is defined. In other words, the images of $(1, i)$ and $(2, f(i))$ form a pair of successive numbers. Observe that every partial monotone function has at least one order encoding due to the monotonicity. For example, the partial monotone function f on $[3]$ defined by $1 \mapsto 2, 3 \mapsto 3$ has an order encoding g which induces the following linear order of $[2] \times [n]$ (the pairs corresponding to defined values of f are underlined)

$$(2,1), \underline{(1,1), (2,2)}, (1,2), \underline{(1,3), (2,3)}.$$

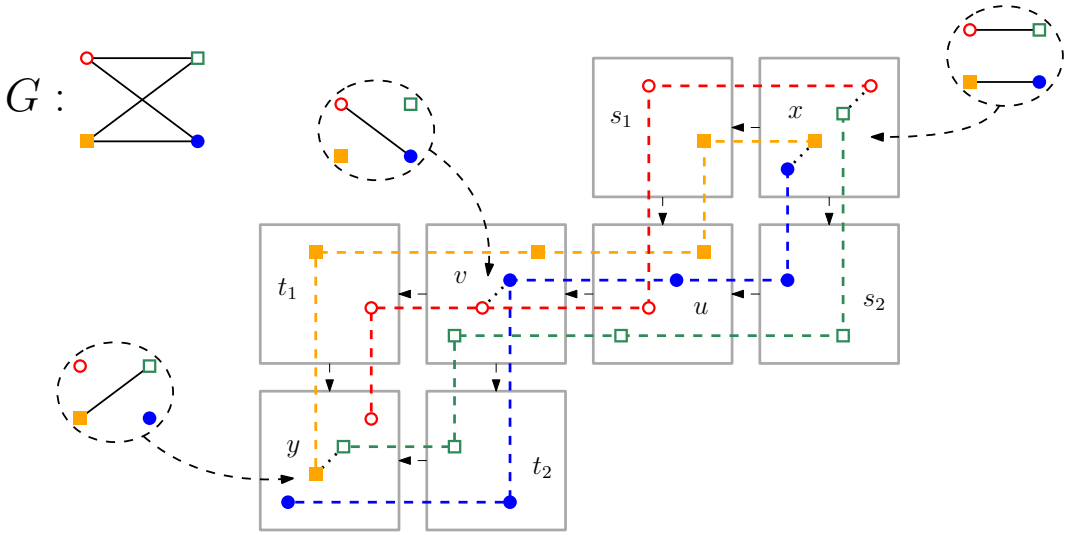

Figure 2.16: Illustration of the proof of Theorem 2.44. Embedding a 3-monotone bipartite graph G (top left) into a permutation from a monotone grid class whose cell graph contains two cycles joined with a path (right).

Let G be a two-sided 3-monotone bipartite ϵ-expander on $2n$ vertices. We describe how to transform it into a permutation $\pi \in \mathrm{Grid}(\mathcal{M})$ with large vertex expansion. Let f_1, f_2 and f_3 be the partial monotone functions encoding the edges of G and let g_1, g_2 and g_3 be their respective order encodings.

Let $P[w_1, w_2]$ be the set of inner vertices of the shortest path from w_1 to w_2 in the cell graph $G_{\mathcal{M}}$. We proceed to define a family of $2n$-tiles $\mathcal{P}$ as

- $P_x = \{(g_1(\alpha, i), g_1(\alpha, i)) \mid i \in [n], \alpha \in [2]\}$,
- $P_y = \{(g_3(\alpha, i), g_3(\alpha, i)) \mid i \in [n], \alpha \in [2]\}$,
- $P_w = \{(g_2(\alpha, i), g_2(\alpha, i)) \mid i \in [n], \alpha \in [2]\}$ for $w \in P[u, v] \cup \{u, v\}$,
- $P_w = \{(g_1(\alpha, i), g_1(\alpha, i)) \mid i \in [n]\}$ for $w \in P[x, s_\alpha]$ and $\alpha \in [2]$,
- $P_w = \{(g_2(\alpha, i), g_2(\alpha, i)) \mid i \in [n]\}$ for $w \in P[s_\alpha, u] \cup P[v, t_\alpha]$ and $\alpha \in [2]$,
- $P_w = \{(g_3(\alpha, i), g_3(\alpha, i)) \mid i \in [n]\}$ for $w \in P[t_\alpha, y]$ and $\alpha \in [2]$,
- $P_{s_\alpha} = \{(g_1(\alpha, i), g_2(\alpha, i)) \mid i \in [n]\}$ for each $\alpha \in [2]$ if s_α and its predecessor share the same column, otherwise its inverse (we swap the x- and y-coordinates), and
- $P_{t_\alpha} = \{(g_2(\alpha, i), g_3(\alpha, i)) \mid i \in [n]\}$ for each $\alpha \in [2]$ if t_α and its predecessor share the same column, otherwise its inverse.

The definition might seem a bit perplexing at first, however there is a fairly simple logic behind it. In order to make more sense of it, notice that every point in every tile is of the form $(g_k(\alpha, i), g_\ell(\alpha, i))$ for some choice of possibly different k, ℓ but the same arguments α and i. For every $i \in [n]$ and $\alpha \in [2]$, let $A_{\alpha,i}$ denote precisely the points of this form.

Observe that every tile is an increasing point set – this is trivial for every tile except for s_1, s_2, t_1, t_2 and for them it follows from the properties of order

encodings. Let π be the $\mathcal{F}$-assembly of $\mathcal{P}$ and let $B_{\alpha,i}$ denote the image of $A_{\alpha,i}$ under the $\mathcal{F}$-assembly. See Figure 2.16. Note that we shall slightly abuse the notation and index the sets A_w and B_w directly by a vertex w of the graph G. We first show that G_π contains G as a minor by verifying the following conditions:

(a) The subgraph of G_π induced by B_w is connected for every vertex w of G;
(b) For each edge $e = \{w_1, w_2\}$ of G, there is an edge in G_π between a vertex of B_{w_1} and a vertex of B_{w_2}.

Observe that for any $i \in [n]$ and $\alpha \in [2]$, the set $A_{\alpha,i}$ contains exactly one point in each tile on the path going from x through s_α, u, v and then continuing through t_α to y. It follows that $A_{\alpha,i}$ contains exactly m points and $|\pi| = 2 \cdot m \cdot n$. Moreover, both coordinates of this point are equal to $g_1(\alpha, i)$ on the path from x to s_α where they change in P_{s_α} and continue equal to $g_2(\alpha, i)$ on the path from s_α through u and v to t_α. In P_{t_α} they change again and then they continue equal to $g_3(\alpha, i)$ on the path from t_α to y. Therefore, every set $B_{\alpha,i}$ induces a connected subgraph by the same argument as in the proof of Proposition 2.41 and hence, (a) is proved.

In order to prove (b), consider an edge e of G. It must be of the form $e = \{(1, i), (2, f_k(i))\}$ for some $k \in [3]$ and $i \in [n]$. First, assume that $k = 1$. Let us focus on the column of the $\mathcal{M}$-gridding of π that contains the image of the tile P_x. When reading points in this column from left to right, we will meet the sets $B_{\alpha,i}$ in the order given by g_1 or its reverse. And the properties of g_1 guarantee that there will be neighboring pair of points from $B_{1,i}$ and $B_{2,f_1(i)}$ which enforces the desired edge. If $k = 3$ then we apply the same argument to the column that contains the image of the tile P_y and obtain a neighboring pair of points from $B_{1,i}$ and $B_{2,f_3(i)}$. And finally for $k = 2$, we use the same argument for the image of any tile P_w on the path between u and v and obtain a neighboring pair of points from $B_{1,i}$ and $B_{2,f_2(i)}$, which concludes the proof of (b).

Our only remaining task is to provide a lower bound for the vertex expansion of G_π. Let X be a non-empty subset of its vertices such that $|X| \leq \frac{1}{2} \cdot |V(G_\pi)|$. We say that a vertex w of G is *heavy* if X contains the whole set B_w and we say that w is *light* if w is not heavy but $X \cap B_w$ is non-empty. Let H denote the set of all heavy vertices and L the set of all light vertices. We have either $|H| \geq \frac{1}{2m} \cdot |X|$ or $|L| \geq \frac{1}{2m} \cdot |X|$ since $\{B_w \mid w \in V(G_\pi)\}$ forms a partition of $V(G_\pi)$ and its every part B_w contains exactly m points.

First suppose that there are at least $\frac{1}{2m} \cdot |X|$ light vertices. For a light vertex w, the set B_w induces a connected subset of G_π and contains both a point that lies in X and a point outside of X. Thus, every such B_w contains at least one point of the sphere $S_{G_\pi}(X)$ and

$$|S_{G_\pi}(X)| \geq |L| \geq \frac{1}{2m} \cdot |X|.$$

Otherwise, there are at least $\frac{1}{2m} \cdot |X|$ heavy vertices. Observe that $|H| \leq n$ as otherwise X would have to contain more than $m \cdot n = \frac{1}{2} \cdot |V(G_\pi)|$ points. Since H contains at most half of all vertices in G, we can use the expanding property of G to see that it has a large sphere $S_G(H)$. For each vertex $w \in S_G(H)$, there exist points p, q in the graph G_π sharing an edge such that $p \in B_{w'}$ for some $w' \in H$ and $q \in B_w$. We know that $p \in X$ since w' is heavy. And now either $q \notin X$

and thus, q belongs to the sphere $S_{G_\pi}(X)$. Or $q \in X$ but there must be some $r \in B_w \setminus X$ as w is not heavy and therefore, B_w must contain some point of the sphere $S_{G_\pi}(X)$ nevertheless. Either way, we have

$$|S_{G_\pi}(X)| \geq |S_G(H)| \geq \mathrm{vx}(G) \cdot |H| \geq \frac{1}{2m} \cdot \mathrm{vx}(G) \cdot |X|.$$

Therefore, the vertex expansion of G_π is at least $\frac{1}{2m} \cdot \mathrm{vx}(G)$. The desired result follows since m is a constant depending only on the class $\mathcal{C}$. □

Finally by combining together Theorem 2.46 together with Proposition 2.43, we obtain a linear lower bound on the tree-width growth of any permutation class with the bicycle property. This bound is clearly asymptotically tight as the tree-width of any permutation cannot exceed its length.

Corollary 2.47. *If a permutation class $\mathcal{C}$ has the bicycle property, then*

$$\mathrm{tw}_{\mathcal{C}}(n) \in \Theta(n).$$

2.3 Principal classes

In this section, we investigate the principal permutation classes, i.e. classes defined by a single avoidance pattern, through the lens of their structural properties and (un)boundedness of the previously defined parameters. Since most properties and parameters of interest are invariant under the symmetries generated by reverse, complement and inverse, we always investigate a single pattern from each equivalence class. We call these equivalence classes *symmetry types*.

Recall that by a result of Berendsohn [22], which we stated as Theorem 2.30, the class $\mathrm{Av}(\pi)$ has tree-width growth of order $\Omega(n/\log n)$ for any π of length at least four that is not symmetric to one of $\{3412, 3142, 4213, 4123, 42153, 41352, 42513\}$. We are able to reproduce and extend this result in a concise way with the tools that we have built up. See Table 2.1 for a summary.

2.3.1 Classes without the deep tree property

Patterns of length at most three

Let us start with the simplest of cases. We can skip the classes defined by forbidden patterns of length one and two as $\mathrm{Av}(1)$ is empty and both classes $\mathrm{Av}(12)$ and $\mathrm{Av}(21)$ contain single permutation of each length n. There are 6 permutations of length three in total but only 2 symmetry types. We pick 132 and 321 as their representatives.

First, we note that the class $\mathrm{Av}(132)$ is quite structured and well-behaved. By Proposition 2.5, it has bounded tree-width and unbounded path-width. Moreover, every 132-avoiding permutation is separable and thus, $\mathrm{Av}(132)$ has also bounded modular-width.

The situation is fairly different with $\mathrm{Av}(321)$. As we previously mentioned, Waton [117, Lemma 5.7.2] first observed that the class $\mathrm{Av}(321)$ is identical to the staircase $\mathrm{St}(◹, ◹)$. Therefore, $\mathrm{Av}(321)$ has the (poly-time computable) long path property and the tree-width growth of $\mathrm{Av}(321)$ is at least $\Omega(\sqrt{n})$ due to

σ	LPP	CP	DTP/BP	$\mathrm{tw}_{\mathrm{Av}(\sigma)}$	Comment
1, 21, 132	✗	✗	✗	$\Theta(1)$	All separable permutations.
321	✓	✗	✗	$\Theta(\sqrt{n})$	LPP: Av(321) = St(◹,◹); CP: Proposition 2.36; DTP: small tree-width contradicts Proposition 2.39.
3142, 4213	✓	?	✗	$\Omega(\sqrt{n})$	LPP: first contains the clockwise spiral, second contains Av(321); DTP: both contained in 41352.
3412, 4123, 41352	✓	✓	✗	$\Omega(\sqrt{n})$	CP: all contain specific cyclic grid subclasses; DTP: Observation 2.48, Propositions 2.49 and 2.52.
All other	✓	✓	✓	$\Theta(n)$	Corollary 2.57.

Table 2.1: We list the properties and the implied tree-width growth of all principal classes, i.e. classes of form $\mathrm{Av}(\sigma)$. The abbreviations LPP, CP, DTP and BP stand for the long path property, cycle property, deep tree property and bicycle property, respectively. Only one pattern σ of each symmetry type is listed.

Proposition 2.33. On the other hand, we have shown in Proposition 2.36 that Av(321) lacks the cycle property. Berendsohn et al. [23, Theorem 5] observed that the incidence graph of any 321-avoiding permutation is planar which implies that $\mathrm{tw}_{Av(321)}(n) = O(\sqrt{n})$ and that Av(321) cannot have the deep tree property. In summary, we have an exact asymptotic behavior of the tree-width growth

$$\mathrm{tw}_{Av(321)}(n) = \Theta(\sqrt{n}).$$

Patterns of length four

There are 24 permutations of length four in total but only 7 symmetry types (see sequence A000903 of [107]) represented by 3412, 4123, 3142, 4213, 4321, 4231 and 4312. We will later show that 4321, 4231 and 4312 all have the bicycle property so for now we focus on the first 4 patterns.

The class of 3412-avoiding permutations contains the class of skew-merged permutations since a permutation is skew-merged if and only if it avoids both 3412 and 2143 [108, Theorem 2.9]. Therefore, it contains as a subclass the monotone grid class $\mathrm{Grid}\left(\begin{smallmatrix} ◺ & ◹ \\ ◹ & ◺ \end{smallmatrix}\right)$ and it has the cycle property.

The class Av(4123) clearly has the long path property since it contains as a subclass the class Av(123), which is symmetric to Av(321). However, it also has the cycle property as we verified computationally that it contains the following monotone cyclic grid class

$$\mathrm{Grid}\left(\begin{smallmatrix} . & ◹ & ◺ \\ ◹ & ◺ & . \\ ◺ & . & ◺ \end{smallmatrix}\right) \subseteq \mathrm{Av}(4123).$$

Next, we focus on the class Av(3142). It turns out that Av(3142) contains a certain spiral-shaped monotone grid subclass which has the long path property. For $k > 0$, a *clockwise spiral of k turns* is the monotone grid class $\mathrm{Grid}(\mathcal{M})$ of a

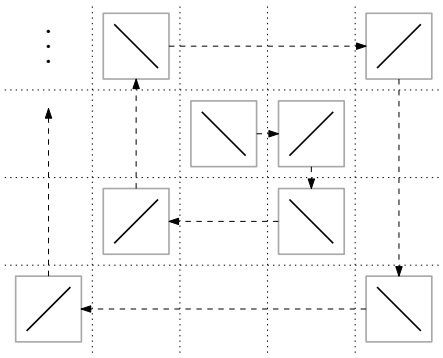

Figure 2.17: The clockwise spiral.

$(2k+1) \times 2k$ gridding matrix $\mathcal{M}$ defined as

$$\mathcal{M}_{i,j} = \begin{cases} \boxslash & \text{if } i \leq k \text{ and } j = i, \\ \boxslash & \text{if } i \geq k+2 \text{ and } j = i-1, \\ \boxbslash & \text{if } i \geq 2 \text{ and } i = 2k+2-i, \text{ and} \\ \emptyset & \text{otherwise.} \end{cases}$$

The *clockwise spiral* is defined as the union of clockwise spirals of k turns over all $k \in \mathbb{N}$. See Figure 2.17. Jelínek and Kynčl [86, Lemma 4.2] proved that the clockwise spiral avoids the pattern 3142. Trivially the clockwise spiral has the (poly-time computable) long path property and thus, so does the class Av(3142). We leave as an open question whether Av(3142) also has the cycle property.

Finally, we are left with the pattern 4213. Notice that the class Av(4213) has the long path property as it contains Av(321). As with the pattern 3142, we leave as an open question whether it also has the cycle property.

Patterns of length five

In total, there are 23 symmetry types of permutations of length 5 (again, see sequence A000903 of [107]). Out of these 23, there is only a single symmetry type defining classes without the deep tree property represented by 41352. As before, the class Av(41352) clearly has the long path property since it contains Av(321) as a subclass. However, it is easy to check that it also contains the grid class $\mathrm{Grid}\left(\begin{smallmatrix} \boxslash & \boxbslash \\ \boxbslash & \boxslash \end{smallmatrix}\right)$ as a subclass and thus, it also has the cycle property.

Proving the absence of the deep tree property

We now prove that all classes $\mathrm{Av}(\sigma)$ for $\sigma \in \{3412, 3142, 4213, 4123, 41352\}$ lack the deep tree property. Hereby, we know that our current tools cannot improve the lower bounds on the tree-width growth of these classes any further. Note that Av(3142) and Av(4213) are, up to symmetry, subclasses of Av(41352), so we only need to focus on the patterns 3412, 4123 and 41352.

Let us say that a graph G is *representable* in a class $\mathcal{C}$ if $\mathcal{C}$ contains a monotone grid subclass whose cell graph is G.

Observation 2.48. *A graph with a vertex of degree three is not representable in* Av(3412)*. In particular,* Av(3412) *does not have the deep tree property.*

Proposition 2.49. *If G is a graph with a vertex of degree 3 whose every neighbor has degree at least 2, then G is not representable in* Av(4123).

Proof. Suppose G is a graph with a vertex v of degree 3, and let x, y and z be the three neighbors of v. Suppose each of the three neighbors has degree at least 2. For contradiction, suppose that Av(4123) contains a monotone grid subclass $\mathcal{C} = \mathrm{Grid}(\mathcal{M})$ whose cell graph is G.

Abusing notation slightly, we will identify the vertices of G with the corresponding cells of the matrix $\mathcal{M}$. Since the pattern 4123 is symmetric with respect to diagonal reflection, we may assume without loss of generality that x and y are in the same column of $\mathcal{M}$ as v, and that z is in the same row as v. Assume further that x is above v and y is below v.

By assumption, x has degree at least 2. In particular, there is a vertex w adjacent to x and different from v. The vertex w is either in the same row as x or in the same column and above x. However, for any possible placement of w, the four cells v, w, x, y will embed the forbidden pattern 4123, a contradiction. □

Corollary 2.50. Av(4123) *does not have the deep tree property.*

Finally, we turn to the pattern 41352. Here the proof is slightly more involved and we begin with a lemma. Note that we assume that the rows in a gridding matrix are numbered bottom to top. A cell in row r and column c of a gridding matrix is referred to as the *cell* (c, r).

Lemma 2.51. *Let $\mathcal{C} = \mathrm{Grid}(\mathcal{M})$ be a monotone grid class not containing the pattern 41352. Suppose that there are two row indices $r_1 < r_2$ and two column indices $c_1 < c_2$, such that the three cells (c_2, r_1), (c_1, r_2) and (c_2, r_2) are all nonempty, and moreover the cell (c_2, r_2) is a ◰-cell. Then the following holds:*

1. *The cell (c_2, r_1) is a ◱-cell.*
2. *Any cell (c, r) satisfying $r_1 \le r \le r_2$ and $c \ge c_2$ is empty, except the cells (c_2, r_1) and (c_2, r_2).*
3. *Any cell (c, r) satisfying $r \le r_1$ and $c_1 \le c \le c_2$ is empty, except the cell (c_2, r_1).*

Proof. See Figure 2.18. If (c_2, r_1) were a ◰-cell, we could embed 41352 by mapping the values $1, 2$ into cell (c_2, r_1), values $3, 5$ into (c_2, r_2) and value 4 into (c_1, r_2). This proves the first claim.

If there were a nonempty cell (c, r) with $r_1 \le r \le r_2$ and $c \ge c_2$, and with $(c, r) \notin \{(c_2, r_1), (c_2, r_2)\}$, we could embed 2 into this cell, 1 into cell (c_2, r_1), 3 and 5 into cell (c_2, r_2), and 4 into (c_1, r_2). This proves the second claim. The third claim is analogous. □

Note that the pattern 41352 is preserved under rotations by a multiple of 90°. Thus, the previous lemma remains valid when the entire gridding matrix $\mathcal{M}$ is rotated in such a way; note that a 90°-rotation transforms a ◱-cell into a ◰-cell and vice versa.

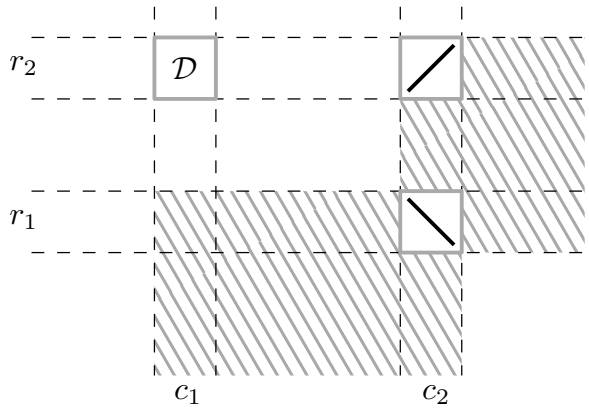

Figure 2.18: The situation of Lemma 2.51. The class $\mathcal{D}$ can be either ◪ or ◹, the shaded regions cannot contain any non-empty cell.

Proposition 2.52. *Let G be a graph containing a vertex v of degree 3, whose three neighbors all have degree 3. Then no subdivision of G is representable in* Av(41352).

Proof. For contradiction, suppose we have a monotone grid class $\mathcal{C} = \mathrm{Grid}(\mathcal{M}) \subseteq \mathrm{Av}(41352)$ whose cell graph G' is a subdivision of G. Let the vertex v correspond to a cell (c, r) of $\mathcal{M}$. By rotational symmetry, we may assume that the three neighbors of v in $\mathcal{M}$ are to the left, to the top and below v. Let x_1 be the neighbor of v situated below v,let y be the neighbor of v situated to its left, and let z be the neighbor of v situated above it.

Note that v must be a ◪-cell, else we could embed values $2, 3$ into v, and the remaining three values into the three neighbors of v.

We will show that the connected component of $G' - v$ containing the vertex x_1 does not contain any vertex of degree greater than 2, contradicting the structure of G'. Applying Lemma 2.51 to the three vertices y, v and x_1, we conclude that x_1 is a ◹-cell, and that is has no neighbor to its bottom or to its right. If x_1 has degree two, then its neighbor different from v is a cell x_2 situated to its left. We may then again apply Lemma 2.51 (rotated 90° clockwise) to the three vertices v, x_1 and x_2, concluding that x_2 is a ◪-cell, and any potential neighbor of x_2 different from x_1 is located above x_2. Denoting such a neighbor x_3, and applying Lemma 2.51 to the vertices x_1, x_2 and x_3, we again conclude that x_3 is a ◹-cell. Continuing in this fashion, we may inductively show that the connected component of $G' - v$ containing the vertex x_1 is a path $x_1, x_2, x_3, \ldots$ arranged into a clockwise spiral and consisting of an alternation of ◪-cells and ◹-cells. See Figure 2.19. In particular, the component does not contain any vertex of degree 3, contradicting the structure of G'. □

Corollary 2.53. *None of the three classes* Av(41352), Av(3142) *and* Av(4213) *has the deep tree property.*

2.3.2 Classes with the bicycle property

We will see that the classes defined using the remaining patterns of lengths 4 and 5 all have the bicycle property. It is a consequence of similar structural properties. The remaining patterns of length 4 all allow embedding of the staircase class

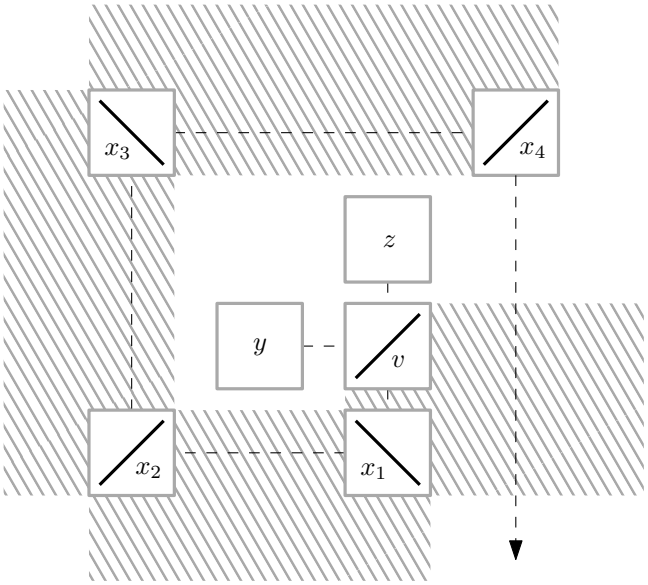

Figure 2.19: Situation in the proof of Proposition 2.52. The shaded regions cannot contain any non-empty cell.

$\mathrm{St}(\boxslash, \mathrm{Av}(\sigma))$ where σ is some permutation of length three. In fact, we prove such claim for any pattern with a suitable structure.

Lemma 2.54. *For any sum-indecomposable permutation σ, $\mathrm{St}(\boxslash, \mathrm{Av}(\sigma))$ is a subclass of $\mathrm{Av}(1 \ominus \sigma)$.*

Proof. Suppose for a contradiction that $\sigma' = 1 \ominus \sigma$ belongs to $\mathrm{St}(\boxslash, \mathrm{Av}(\sigma))$. In particular it belongs to $\mathrm{St}_k(\boxslash, \mathrm{Av}(\sigma))$ for some k and there is a witnessing gridding. If the first element is not mapped to one of the $\boxslash$-entries on the upper diagonal, then the whole σ' must lie in a single $\mathrm{Av}(\sigma)$-entry on the lower diagonal, which is clearly not possible. Therefore, the first element must be mapped to one of the $\boxslash$-entries. Notice that the rest of σ' cannot be mapped to any of the $\boxslash$-entries as it lies below and to the right of the first element. However, it cannot lie in more than one $\mathrm{Av}(\sigma)$-entry; otherwise, we could express σ as a direct sum of two shorter permutations. Hence, there must be an occurrence of σ in an $\mathrm{Av}(\sigma)$-entry which is clearly a contradiction. □

A direct consequence of Lemma 2.54 is that taking σ to be 321, 312 or 231, we see that

$$\begin{aligned}
&\mathrm{St}(\boxslash, \mathrm{Av}(321)) \subseteq \mathrm{Av}(4321),\\
&\mathrm{St}(\boxslash, \mathrm{Av}(231)) \subseteq \mathrm{Av}(4231), \text{ and}\\
&\mathrm{St}(\boxslash, \mathrm{Av}(312)) \subseteq \mathrm{Av}(4312).
\end{aligned}$$

Note that the first inclusion is rather trivial and the latter two have been previously observed by Berendsohn [22].

We verified by computer that there are only five symmetry types of patterns of length 5 that do not contain any of 4321, 4213, 4312 or their symmetries — represented by 14523, 24513, 32154, 42513 and 41352. We have already shown that Av(41352) cannot even have the deep tree property, but the remaining four classes contain a specific type of cyclic grid classes, as we now show.

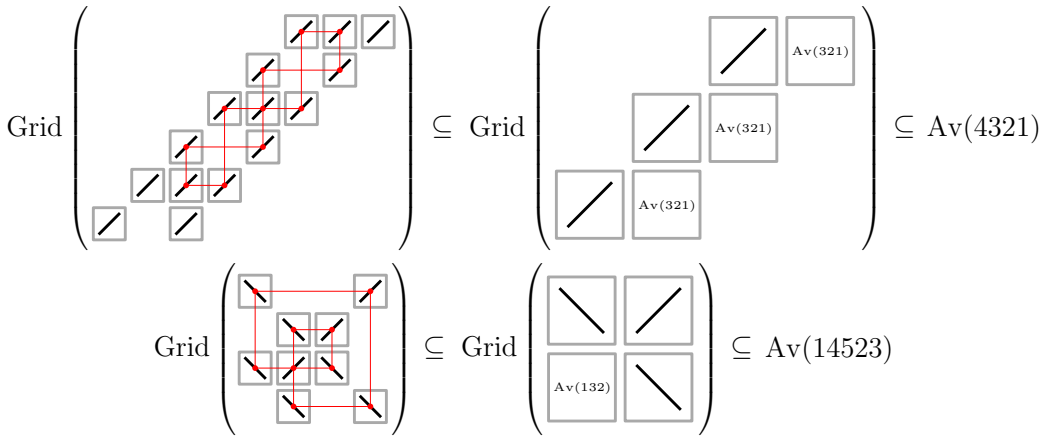

Figure 2.20: Establishing the bicycle property in the proof of Proposition 2.56 with the help of Lemmas 2.54 (top) and 2.55 (bottom). The red lines highlight the two connected cycles in the cell graphs.

Lemma 2.55. *The class* $\mathrm{Av}(\sigma)$ *contains the class* $\mathrm{Grid}(\mathcal{M})$ *for the gridding matrix* $\mathcal{M} = \begin{pmatrix} \searrow & \nearrow\!\searrow \\ \mathrm{Av}(\pi) & \searrow \end{pmatrix}$ *whenever*

- $\pi = 132$ *and* $\sigma = 14523$*, or*
- $\pi = 231$ *and* $\sigma = 24513$*, or*
- $\pi = 321$ *and* $\sigma \in \{32154, 42513\}$.

Proof. Suppose that σ and π are one of the listed cases. Observe that $\mathrm{Grid}(\mathcal{M})$ is a subclass of $\mathrm{Av}(\sigma)$ if and only if σ is not in $\mathrm{Grid}(\mathcal{M})$. For contradiction, suppose that the class $\mathrm{Grid}(\mathcal{M})$ contains σ. Therefore, there exists a witnessing $\mathcal{M}$-gridding $1 = c_1 \le c_2 \le c_3 = 6$ and $1 = r_1 \le r_2 \le r_3 = 6$ of σ.

Let us consider the four choices of σ separately, starting with $\sigma = 14523$: if $c_2 \le 3$ and $r_2 \le 3$, the cell $(2, 2)$ of the gridding contains the pattern 21, if $c_2 \le 4$ and $r_2 \ge 4$, the cell $(2, 1)$ contains 12, if $c_2 \ge 4$ and $r_2 \le 4$, the cell $(1, 2)$ contains 12, and if $c_2 \ge 5$ and $r_2 \ge 5$, the cell $(1, 1)$ contains 132. In all cases we get a contradiction with the properties of the $\mathcal{M}$-gridding. The same argument applies to $\sigma = 14513$, except in the last case we use the pattern 231 instead of 132.

For $\sigma = 32154$, the four cases to consider are $c_2 \le 4 \wedge r_2 \le 4$, $c_2 \ge 5 \wedge r_2 \le 3$, $c_2 \le 3 \wedge r_2 \ge 5$, and $c_2 \ge 4 \wedge r_2 \ge 4$, in each case getting contradiction in a different cell of the gridding. For $\sigma = 42513$, the analogous argument distinguishes the cases $c_2 \le 3 \wedge r_2 \le 3$, $c_2 \ge 4 \wedge r_2 \le 4$, $c_2 \le 4 \wedge r_2 \ge 4$, and $c_2 \ge 5 \wedge r_2 \ge 5$. □

Finally, we show that the structural properties we just proved translate easily into the bicycle property.

Proposition 2.56. *If* σ *is any of* 4321, 4231, 4312, 4123, 14523, 24513, 32154, 42513*, then* $\mathrm{Av}(\sigma)$ *has the bicycle property.*

Proof. We start by proving that every class defined by forbidding a pattern of length 3 must contain a special type of monotone grid subclass. For arbitrary π of length 3, the class $\mathrm{Av}(\pi)$ contains a grid class $\mathrm{Grid}(\mathcal{M})$ such that $\mathcal{M}$ is a 2×2 monotone gridding matrix with three non-empty entries. Since there are only two different symmetry types of permutations of length 3, it is enough to check that

$$\mathrm{Grid}\begin{pmatrix} \nearrow\!\searrow & \nearrow\!\searrow \\ \nearrow\!\searrow & \cdot \end{pmatrix} \subseteq \mathrm{Av}(321) \quad \text{and} \quad \mathrm{Grid}\begin{pmatrix} \searrow & \nearrow\!\searrow \\ \cdot & \searrow \end{pmatrix} \subseteq \mathrm{Av}(132).$$

First, we prove the claim for the patterns that appear in Lemma 2.54. Let $\sigma \in \{4321, 4231, 4312\}$ and take a 3-step increasing staircase $\mathrm{St}_3(\boxslash, \mathrm{Av}(\pi))$ for π of length 3 that is contained in $\mathrm{Av}(\sigma)$. Let $\mathcal{M}'$ be a 6×8 monotone gridding matrix obtained from $\mathrm{St}_3(\boxslash, \mathrm{Av}(\pi))$ by replacing every $\boxslash$-entry by the identity matrix $\left(\begin{smallmatrix} \cdot & \boxslash \\ \boxslash & \cdot \end{smallmatrix}\right)$ and every $\mathrm{Av}(\pi)$-entry with its 2×2 monotone grid subclass which has three non-empty entries. Clearly, $\mathrm{Grid}(\mathcal{M}')$ is a subclass of $\mathrm{Av}(\sigma)$, and it is easy to check that for any π, the cell graph of $\mathcal{M}'$ is connected and contains two cycles. Refer to the top part of Figure 2.20.

We prove the claim for the patterns that appear in Lemma 2.55 in a similar fashion. Let $\sigma \in \{14523, 24513, 32154, 42513\}$ and take $\mathcal{M}$ to be the grid class $\mathrm{Grid}\left(\begin{smallmatrix} \boxbslash & \boxslash \\ \mathrm{Av}(\pi) & \boxbslash \end{smallmatrix}\right)$ for π of length 3 that is contained in $\mathrm{Av}(\sigma)$. Similar to the previous argument, let $\mathcal{M}'$ be the gridding matrix obtained from $\mathcal{M}$ by replacing the $\boxslash$-entry with the matrix $\left(\begin{smallmatrix} \cdot & \boxslash \\ \boxslash & \cdot \end{smallmatrix}\right)$, both $\boxbslash$-entries with the matrix $\left(\begin{smallmatrix} \boxbslash & \cdot \\ \cdot & \boxbslash \end{smallmatrix}\right)$, and $\mathrm{Av}(\pi)$ with its 2×2 monotone grid subclass which has three non-empty entries. Again, $\mathrm{Grid}(\mathcal{M}')$ is a subclass of $\mathrm{Av}(\sigma)$, and it is easy to check that for any π, the cell graph of $\mathcal{M}'$ is connected and contains two cycles. Refer to the bottom part of Figure 2.20. □

It is easy to see that every σ of length at least 6 contains a pattern of length 5 which is not symmetric to 41352. Therefore, we can deduce from Proposition 2.56 that most principal classes have the bicycle property and thus, they contain permutations with linear tree-width due to Corollary 2.47.

Corollary 2.57. *If σ is a permutation of length at least 4 that is not symmetric to any of* $3412, 3142, 4213, 4123$ *or* 41352*, then* $\mathrm{Av}(\sigma)$ *has the bicycle property.*

The results obtained in this section completely resolve the asymptotic tree-width growth of all principal classes except for these five cases (and their symmetries). It is natural to ask whether the remaining classes can also contain permutations of linear tree-width or if their tree-width growth is bounded by $O(n^c)$ for some $c < 1$.

Open problem 2.58. *What is the tree-width growth of* $\mathrm{Av}(\sigma)$*-PPM, when σ is a permutation from the set* $\{3412, 3142, 4213, 4123, 41352\}$*?*

To conclude this section, we remark that the suitable grid subclasses were discovered via computer experiments facilitated by the Permuta [13] and Tilings [98] libraries.

3. Logic of permutations

In this chapter, we interpret permutations from the viewpoint of logic. In particular, we look at permutations as sets equipped with two linear orders, each describing the ordering of points among one of the two axes. This so-called Theory of Two Orders (TOTO) was formally introduced by Albert et al. [9] who were interested in the expressive power of first-order logic formulas. We extend this to monadic second-order logic and moreover, we investigate the complexity of testing whether a given permutation satisfies a given formula.

A *signature* is a set of relation and function symbols, each associated with a non-negative integer, called *arity*. We restrict our attention to signatures consisting purely of relation symbols. The signature $\mathcal{S}_{\mathsf{TO}}$, corresponding to TOTO, contains two binary relation symbols $<_x$ and $<_y$. These symbols intend to convey the ordering of points along the x- and y-axes. A *structure* of a signature $\mathcal{S}$ is a pair $\mathcal{M} = (A, I)$ where A is arbitrary set, called *domain*, and I describes an interpretation of the symbols in $\mathcal{S}$ on A. To be precise, the interpretation $I(R)$ of a relation symbol R with arity k is a subset of A^k. For succinctness, we shall also denote the structures of $\mathcal{S}_{\mathsf{TO}}$ simply as triples $\mathcal{M} = (A, \prec_x^A, \prec_y^A)$ where $\prec_x^A$ and $\prec_y^A$ are interpretations of the symbols $<_x$ and $<_y$. Recall that we have already met these structures when defining the clique-width in Subsection 2.1.4 under the name xy-digraphs.

Observe that any permutation σ can be seen as a structure $(A^\sigma, \prec_x^\sigma, \prec_y^\sigma)$ of $\mathcal{S}_{\mathsf{TO}}$. It suffices to take as the domain A^σ the permutation diagram of σ, and set $\prec_x^\sigma$ and $\prec_y^\sigma$ to be the natural orders given by the x- and y-coordinates of points. From now on, whenever we talk about a permutation σ as a structure of $\mathcal{S}_{\mathsf{TO}}$ (or later as a model of TOTO), we actually mean the structure $(A^\sigma, \prec_x^\sigma, \prec_y^\sigma)$.

3.1 First-order logic

In this section, we consider the first-order logic fragment of TOTO. As a matter of fact, we only survey previous knowledge without providing any substantial new results. We refer an interested reader to [9].

We start by defining first-order formulas over an arbitrary signature $\mathcal{S}$. An *atomic formula* is either an equality predicate $(x = y)$, or a predicate $R(a_1, \ldots, a_k)$ for arbitrary relation symbol R of $\mathcal{S}$ with arity k. *First-order (FO) formulas*, usually denoted by Greek letters, are formed inductively from atomic formulas and logical symbols. In particular, a first-order formula is either (i) an atomic formula, (ii) a negation of a first-order formula $(\neg\varphi)$, (iii) a conjuction $(\varphi_1 \wedge \varphi_2)$, disjunction $(\varphi_1 \vee \varphi_2)$, implication $(\varphi_1 \to \varphi_2)$ or equivalence $(\varphi_1 \leftrightarrow \varphi_2)$ of two first-order formulas, or (iv) an existential $(\exists x\, \varphi)$ or universal $(\forall x\, \varphi)$ quantification of a first-order formula. A first-order (FO) *sentence* is then any FO formula that has no free variables, or put differently, a formula whose variables are all quantified.

A structure $\mathcal{M} = (A, I)$ *satisfies* the sentence φ if φ evaluates to "True" when we interpret the variables as elements of the domain A and the symbols of $\mathcal{S}$ according to I. We denote this by $\mathcal{M} \models \varphi$. In the case of TOTO, we additionally allow ourselves to write $\sigma \models \varphi$ where σ is the permutation associated to $\mathcal{M}$.

Example. The existence of a classical pattern 132 is expressed by the FO sentence

$$\exists\, x, y, z\ \left[(x <_x y) \wedge (y <_x z) \wedge (x <_y z) \wedge (z <_y y)\right].$$

Finally, we can formally define the notion of a theory. A *theory* is just a set of (FO) sentences, which are called the *axioms* of the theory. A *model* of a theory is any structure that satisfies all axioms of the theory. The axioms of TOTO simply state that both $<_x$ and $<_y$ are linear orders. It should be obvious that these axioms can be straight-forwardly stated as FO sentences.

Let us remark that an arbitrary finite model of TOTO can be seen as a permutation. For any linear order $\prec$ on a set A, we define the rank of $a \in A$ as $rank_{\prec}(a) = |\{b \mid b \in A, b < a\}| + 1$, and we set as the permutation associated to the model $(A, \prec_x^A, \prec_y^A)$ of TOTO the permutation isomorphic to the point set

$$\{(rank_{\prec_x^A}(a), rank_{\prec_y^A}(a)) \mid a \in A\}.$$

We say that two structures (A, I) and (B, J) of the same signature $\mathcal{S}$ are *isomorphic* if there is a bijection $f\colon A \to B$ such that for any relation symbol R of $\mathcal{S}$ with arity k and every k-tuple of elements $a_1, \ldots, a_k \in A$ we have $(a_1, \ldots, a_k) \in I(R)$ if and only if $(f(a_1), \ldots, f(b_k)) \in J(R)$. Albert et al. [9] observed that TOTO models permutations as intended since two models are isomorphic if and only if the permutations associated with them are equal.

Proposition 3.1 ([9, Proposition 4])**.** *Two models of TOTO are isomorphic if and only if the permutations associated with them are equal.*

3.1.1 Expressibility of first-order logic

The expressive power of the first-order fragment of TOTO was investigated thoroughly by Albert et al.[9]. First, they showed that virtually every notion of (generalized) pattern containment that has ever appeared in the literature is expressible by a first-order sentence. Therefore, first-order sentences provide in some sense a common generalization of all the different pattern flavors. Furthermore, they obtained first-order sentences capturing whether a permutation is $\oplus$-decomposable, simple or belongs to a fixed grid class. And finally, they showed that first-order sentences can describe permutations that can be sorted by k iterations of arbitrary sorting operator as long as the sorting operator itself can be suitably described in TOTO. We collect all these results in a single proposition.

Proposition 3.2 ([9])**.** *Each of the following properties is expressible by an FO sentence φ. In other words, there exists an FO sentence φ for each property such that $\sigma \models \varphi$ if and only if the property holds for a permutation σ.*

(a) σ contains a fixed pattern π,
(b) σ contains a fixed mesh pattern (π, B),
(c) σ contains a fixed barred pattern π,
(d) σ belongs to the class Grid$(\mathcal{M})$ *for a fixed gridding matrix $\mathcal{M}$,*
(e) σ is $\oplus$-decomposable (and symmetrically for $\ominus$-decomposability),
(f) σ is simple, and
(g) σ is k-stack-sortable for a fixed k.

Let us add that the notion of 2-avoidance and partially ordered patterns is also easily captured by FO sentences even though they were not considered by Albert et al [9]. In the case of a fixed partially ordered pattern, the corresponding sentence is obtained in similar way as for a classical pattern, just omitting the missing requirements in the y-axis. And in the case of 2-avoidance, the reasoning follows similar line to that of barred patterns.

Albert et al. [9] also provide an example of a property that is inexpressible by a first-order sentence in TOTO. A *fixed point* of a permutation π is any point p of the permutation diagram S_π such that $p.x = p.y$.

Proposition 3.3 ([9, Corollary 27])**.** *The property of having a fixed point is not expressible by an FO sentence in TOTO. In other words, there does not exist an FO sentence φ such that $\sigma \models \varphi$ if and only if σ has a fixed point.*

As we shall see later, the property of having a fixed point is in fact not expressible even by a monadic second-order sentence.

3.1.2 First-order model checking

Bonnet et al. [36] proved that an fpt-algorithm for permutation pattern matching designed by Guillemot and Marx [77] can actually be extended to first-order model checking (and transferred to general graphs).

Theorem 3.4 ([36])**.** *Given a permutation π of length n and a first-order sentence φ in TOTO, we can decide whether $\pi \models \varphi$ in time $f(|\varphi|, \mathrm{tww}(\pi)) \cdot n$ for some computable, yet non-elementary, function f.*

Observe that together with Theorem 2.2 this yields a linear time algorithm for deciding whether a given permutation π belongs to a fixed class $\mathcal{C}$ defined by an FO sentence. This is quite obvious for properties like whether a permutation is k-stack-sortable since we can simply simulate the sorting. However, it far less obvious that one can test in linear time whether a permutation is simple even though it alternatively follows from the linear algorithm for substitution decomposition [51]. And we even find it surprising that permutations of any fixed grid class can be recognized in linear time.

Corollary 3.5. *For every permutation class $\mathcal{C}$ defined by an FO sentence, we can decide in linear time whether a given permutation π belongs to $\mathcal{C}$. In particular, we can decide in linear time whether π belongs to the grid class* $\mathrm{Grid}(\mathcal{M})$ *for any fixed gridding matrix $\mathcal{M}$.*

To end this section, we complement Theorem 3.4 by showing that FO model checking on permutations in general is as hard as deciding FO sentences on undirected graphs. But first, we need to formalize the logic of undirected graphs. To that end, we define the signature $\mathcal{S}_G$ which consists of a single binary relation symbol E intended to describe the edges. Our only requirement is that E is a symmetric relation, which can be easily described by a FO sentence. Therefore, we get a valid theory called Theory of Graphs (TOG).

Let us remark that the signature $\mathcal{S}_G$ is usually denoted in the literature by τ_1. However, we chose a different notation to avoid confusion since we typically use small Greek letters (and τ in particular) to denote permutations.

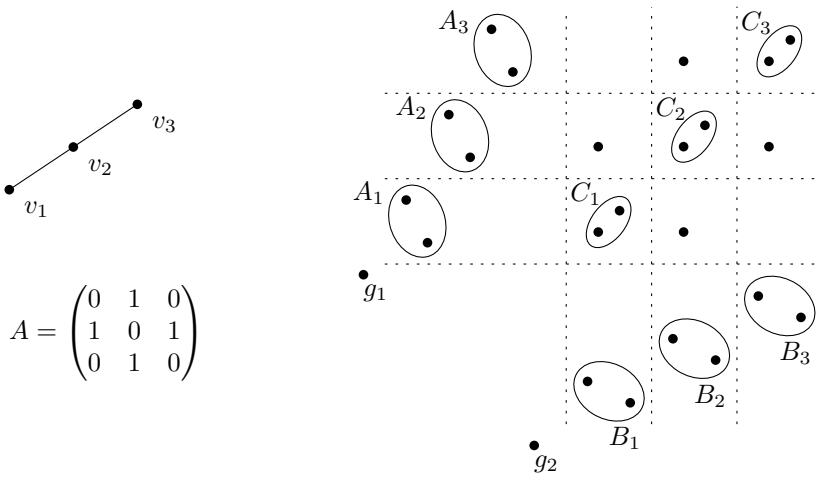

Figure 3.1: Illustration of the proof of Theorem 3.6. Encoding a path on 3 vertices with an adjacency matrix A (left) into a point set P (right).

Theorem 3.6. *There is an algorithm that given a graph G with n vertices and m edges and an FO sentence φ in* TOG*, computes in linear time a permutation π of length $O(n+m)$ and an FO sentence ψ in* TOTO *of length $O(|\varphi|)$ such that $G \models \varphi$ if and only if $\pi \models \psi$.*

Proof. From the given graph G, we first construct a point set P that is not necessarily in general position. Let us number the vertices of G arbitrarily as $V = \{v_1, \ldots, v_n\}$. Our goal is to construct a permutation that would essentially represent the adjacency matrix of G. The set P contains two important points g_1, g_2, called *guards*, defined as

$$g_1 = (1, 2n+2), \qquad g_2 = (2n+2, 1).$$

For each $i \in [n]$, we associate three pairs of points to the vertex v_i defined as

$$\begin{aligned} A_i &= \{(2i, 4i+2n+2), (2i+1, 4i+2n-1)\}, \\ B_i &= \{(4i+2n-1, 2i+1), (4i+2n+2, 2i)\}, \\ C_i &= \{(4i+2n, 4i+2n), (4i+2n+1, 4i+2n+1)\}. \end{aligned}$$

Observe that every pair A_i lies horizontally between g_1 and g_2, every pair B_i lies vertically between g_2 and g_1 and moreover, the pair C_i lies in the intersection of the horizontal strip enclosed by A_i and the vertical strip enclosed by B_i. Moreover, all the pairs A_i form together a layered permutation with each layer of size 2 and therefore, any occurrence of pattern 21 that lies horizontally between g_1 and g_2 must be equal to A_i for some i. The same holds for the pairs B_i.

We interpret the strips defined by sets A_i and B_i as 'columns' and 'rows' of the adjacency matrix, respectively. For each edge $\{i, j\}$ of G, we simply add two points to the corresponding two 'cells', i.e. we add to P the points

$$(4i+2n, 4j+2n), \quad (4j+2n, 4i+2n).$$

See Figure 3.1. Finally, we rotate P slightly to guarantee that it is in general position and take the permutation π as its reduction. Clearly, the length of π is bounded by $O(n+m)$ as promised.

Now given the FO sentence φ, we replace each variable x with four variables $x_1^A, x_2^A, x_1^B, x_2^B$ which are intended to represent the pairs A_i and B_i for some $i \in [n]$. It is sufficient to check that (i) both pairs form an occurrence of 21, (ii) the pair x_1^A, x_2^A lies to the left of g_2, (iii) the pair x_1^B, x_2^B lies below g_1, and (iv) there is an occurrence of 12 in the intersection of the horizontal strip defined by x_1^A, x_2^A and the vertical strip defined by x_1^B, x_2^B. Observe that g_1 is the leftmost point of π while g_2 is the bottommost one. Therefore, it is easy to force two variables x_1^g and x_2^g to be set precisely to g_1 and g_2 throughout the whole sentence Ψ. Using this, we can write the desired conditions as a predicate

$$\begin{aligned} \mathit{vertex}(x_1^A, x_2^A, x_1^B, x_2^B) = 21(x_1^A, x_2^A) \wedge 21(x_1^B, x_2^B) \wedge (x_2^A <_x x_2^g) \wedge (x_1^B <_y x_1^g) \\ \wedge \exists x_1^C, x_2^C \left(4123(x_1^A, x_2^A, x_1^C, x_2^C) \wedge 2341(x_1^B, x_1^C, x_2^C, x_2^B)\right) \end{aligned}$$

where $\sigma(x_1, \ldots, x_k)$ denotes the FO sentence checking that $x_1, \ldots, x_k$ form an occurrence of a permutation σ of length k. Technically, we perform the following replacements

$$\begin{aligned} \exists x\, \rho &\longrightarrow \exists x_1^A, x_2^A, x_1^B, x_2^B \left(\mathit{vertex}(x_1^A, x_2^A, x_1^B, x_2^B) \wedge \rho\right) \\ \forall x\, \rho &\longrightarrow \forall x_1^A, x_2^A, x_1^B, x_2^B \left(\mathit{vertex}(x_1^A, x_2^A, x_1^B, x_2^B) \rightarrow \rho\right). \end{aligned}$$

Finally, checking the existence of an edge is equivalent to testing whether there is a point in the intersection of a given horizontal and vertical strip. Formally, it is sufficient to replace each predicate $E(x, y)$ with

$$\exists z \left(312(x_1^A, x_2^A, z) \wedge 231(y_1^B, z, y_2^B) \wedge \neg \mathit{vertex}(x_1^A, x_2^A, y_1^B, y_2^B)\right). \qquad \square$$

Note that the encoding of graph G into a permutation is similar to previous reductions by Marx and Guillemot [77], or Berendsohn et al. [23].

3.2 Monadic second-order logic

Now, we shift our attention to the monadic second-order fragment of TOTO. Our goal is to identify properties of permutations that are inexpressible in first-order logic but expressible in monadic second-order logic. Furthermore, we show that the natural property of having a fixed point is inexpressible even in monadic-second logic. And finally, we investigate the complexity of monadic second-order model checking.

Informally, *monadic second-order (MSO) formulas* differ from FO formulas by allowing set variables (denoted by capital letters) and quantification over them. Formally, MSO formula over a signature $\mathcal{S}$ is formed inductively using the same operations as FO formulas with the addition of existential ($\exists X\, \varphi$) and universal ($\forall X\, \varphi$) set quantifications, and set membership predicates ($x \in X$). As before, a *monadic second-order sentence* is any MSO formula that has all its variables (representing both elements and sets) quantified. The satisfiability of an MSO formula φ by a permutation σ is defined accordingly.

Example. We can define a predicate $\mathit{partition}(X, Y)$ enforcing that X and Y form a partition of the domain, and predicates $\mathit{increasing}(X)$ and $\mathit{decreasing}(X)$

enforcing that X is an increasing, respectively decreasing point set as follows

$$\begin{aligned} \mathit{partition}(X,Y) &= \forall x\ [(x \in X \vee x \in Y) \wedge \neg(x \in X \wedge x \in Y)]\,, \\ \mathit{increasing}(X) &= \forall x, y\,[(x \in X \wedge y \in X) \to (x <_x y \leftrightarrow x <_y y)]\,, \\ \mathit{decreasing}(X) &= \forall x, y\,[(x \in X \wedge y \in X) \to (x <_x y \leftrightarrow y <_y x)]\,. \end{aligned}$$

It is easy to see that using these predicates, we can define skew-merged permutations by the MSO sentence

$$\exists X \exists Y\,(\mathit{partition}(X,Y) \wedge \mathit{increasing}(X) \wedge \mathit{decreasing}(Y))\,.$$

3.2.1 Properties expressible in MSO

Next, we turn our attention to the expressive power of monadic second-order logic. We have already seen that we can define the skew-merged permutations that can be obtained as a merge of an increasing and a decreasing sequence. It is straightforward to extend this to permutations that can be obtained as a merge of k permutations each coming from an arbitrary set defined by a fixed MSO sentence.

Proposition 3.7. *For arbitrary MSO sentences $\varphi_1, \ldots, \varphi_k$ in TOTO, we can construct an MSO sentence ρ such that $\pi \models \rho$ if and only if π can be obtained as a merge of permutations $\pi_1, \ldots, \pi_k$ such that $\pi_i \models \varphi_i$ for every $i \in [k]$.*

Proof. For an MSO sentence φ, we let $\varphi(X)$ denote the predicate that limits the domain of all variables to the set X. Technically, this is done by replacing

$$\begin{aligned} \exists x\,\psi &\longrightarrow \exists x\,(x \in X \wedge \psi) \\ \forall x\,\psi &\longrightarrow \forall x\,(x \in X \to \psi) \\ \exists Y\,\psi &\longrightarrow \exists Y\,(\forall x\,(x \in Y \to x \in X) \wedge \psi) \\ \forall Y\,\psi &\longrightarrow \forall Y\,(\forall x\,(x \in Y \to x \in X) \to \psi). \end{aligned}$$

The desired sentence is then obtained easily using the predicate for testing whether a k-tuple of sets $X_1, \ldots, X_k$ forms a partition of the domain as

$$\rho = \exists X_1 \exists X_2 \cdots \exists X_k \left(\mathit{partition}(X_1, \ldots, X_k) \wedge \bigwedge_{i=1}^{k} \varphi_i(X_i) \right) \qquad \square$$

Furthermore, we can show that monadic second-order logic has greater expressive power than first-order logic. In particular, we show that for any simple permutation α there exists no FO sentence expressing that a permutation is a merge of two α-avoiding permutations. But first we need to introduce a standard tool for proving inexpressibility in logic – the Ehrenfeucht-Fraïssé games.

Let (A, I) and (B, J) be two models of the same theory, and let k be a positive integer. The *k-move Ehrenfeucht - Fraïssé (EF) game* is a game between two players, called Spoiler and Duplicator, on the models (A, I) and (B, J) with the following rules. Spoiler begins and the players alternate in moves until they both had exactly k turns. In the i-th turn, Spoiler chooses either an element $a_i \in A$ or $b_i \in B$ and Duplicator replies by choosing an element of the other model. More precisely, either Spoiler chooses an element $a_i \in A$ and Duplicator responds by

choosing $b_i \in B$, or Spoiler chooses an element $b_i \in B$ and Duplicator responds by choosing $a_i \in A$. At the end of the game, Duplicator wins if the map $a_i \mapsto b_i$ is an isomorphism between the submodels induced by $\{a_i \mid i \in [k]\}$ and $\{b_i \mid i \in [k]\}$. Otherwise, Spoiler wins.

We assume that both players play optimally and we say that Duplicator *wins* a given EF game, if he has a winning strategy. We denote by $(A, I) \sim_k (B, J)$ that Duplicator wins in the k-move EF game on (A, I) and (B, J). It is easily checked that $\sim_k$ is an equivalence for a fixed k.

Before establishing the connection of EF games to FO logic, we need one more definition. The *quantifier depth of an FO formula* φ, denoted by $\mathrm{qd}(\varphi)$, is defined recursively as follows. For any atomic formula φ, we set $\mathrm{qd}(\varphi) = 0$. Otherwise:

$$\begin{aligned} \mathrm{qd}(\neg\varphi) &= \mathrm{qd}(\varphi), \\ \mathrm{qd}(\varphi \wedge \psi) = \mathrm{qd}(\varphi \vee \psi) = \mathrm{qd}(\varphi \to \psi) = \mathrm{qd}(\varphi \leftrightarrow \psi) &= \max(\mathrm{qd}(\varphi), \mathrm{qd}(\psi)), \\ \mathrm{qd}(\exists x\varphi) = \mathrm{qd}(\forall x\varphi) &= \mathrm{qd}(\varphi) + 1. \end{aligned}$$

Theorem 3.8 ([69, 61])**.** *For two models (A, I) and (B, J) of the same theory, we have $(A, I) \sim_k (B, J)$ if and only if (A, I) and (B, J) satisfy the same set of sentences of quantifier depth at most k.*

Let us first provide a simple example of EF games on linear orders. Formally, the *Theory of Linear Orders* (TOLO) is defined on the signature with a single binary relation symbol $<$ where the axioms enforce that $<$ is a linear order. For simplicity, we write models of TOLO as pairs $(A, \prec)$ such that $\prec$ is a linear order on the domain A. It is well-known that we cannot distinguish two sufficiently large linear orders with FO sentences of a fixed quantifier depth. We include the proof since we later use the described strategy for EF games as a building block.

Proposition 3.9 ([75, Theorem 2.3.20])**.** *Let k be a positive integer. If $(A, \prec_A)$ and $(B, \prec_B)$ are finite models of TOLO such that $|A|, |B| \geq 2^k - 1$, then Duplicator wins the k-move EF game played on $(A, \prec_A)$ and $(B, \prec_B)$ and thus, $(A, \prec_A)$ and $(B, \prec_B)$ satisfy the same set of sentences of quantifier depth at most k.*

Proof. Notice that after r rounds of the EF game on $(A, \prec_A)$ and $(B, \prec_B)$, precisely r elements have been chosen in A and they split the linear order $(A, \prec_A)$ into $r+1$ (possibly empty) intervals $I_0^r, \ldots, I_r^r$ where the interval I_s^r contains all elements larger than exactly s elements of the set $\{a_1, \ldots, a_r\}$. Similarly, the linear order $(B, \prec_B)$ is split into $r+1$ intervals $J_0^r, \ldots, J_r^r$.

The goal of Duplicator is to guarantee that after each round, the intervals I_s and J_s are somehow similar for every s. We say that an interval I of a finite linear order $(X, \prec_X)$ is a-long for a given non-negative integer a if it contains at least $2^a - 1$ elements. We describe a strategy for Duplicator which will guarantee that after r steps, the intervals I_s^r and J_s^r for every $s \in \{0, \ldots, r\}$ are either both $(k-r)$-long, or have exactly the same length.

This clearly holds before the game starts by assumption (formally after 0 rounds). Assume that Spoiler picks in the r-th round an element a_r from the interval I_s^{r-1}. The case when he plays in B is symmetrical. The interval I_s^{r-1} gets split into the intervals I_s^r and I_{s+1}^r. Let us assume without loss of generality that I_s^r is at most as long as I_{s+1}^r, the other case being symmetrical. If the interval I_s^{r-1} is not $(k-r+1)$-long then it has the same length as J_s^{r-1} and Duplicator can

exactly replicate the move of Spoiler inside J_s^{r-1}. Otherwise, the larger interval I_{s+1}^r must be $(k-r)$-long. If the interval I_s^r is also $(k-r)$-long, then the job of Duplicator is to simply split J_s^{r-1} into two $(k-r)$-long intervals. Duplicator can achieve this by picking as b_r the median of J_s^{r-1}. Finally, we are left with the case when I_s^r is not $(k-r)$-long. But in that case, Duplicator selects b_r in a way such that the interval J_s^r is of the exact same length as I_s^r. It follows automatically that J_{s+1}^r is $(k-r)$-long.

By playing the described strategy, Duplicator guarantees that the suborders induced by the chosen elements are isomorphic and thus, $(A, \prec_A) \sim_k (B, \prec_B)$. □

Now we posses all the tools necessary to prove that FO sentences in TOTO are not powerful enough to express the property that a permutation can be merged from two smaller permutations avoiding a fixed simple pattern.

Proposition 3.10. *Let α be a simple permutation of length at least 4. The class $\mathrm{Av}(\alpha) \odot \mathrm{Av}(\alpha)$ is not definable by an FO sentence in TOTO. In other words, there does not exist an FO sentence φ such that $\sigma \models \varphi$ if and only if σ can be obtained as a merge of two α-avoiding permutations.*

Proof. Let us start with some observations and assumptions. First, let m be the length of α and observe α must contain either 2413 or 3142, because otherwise it would be separable and there are no separable simple permutations of length larger than 3. Furthermore, we can assume without loss of generality that α contains 3142, as otherwise its reverse α^r would contain 3142 and clearly $\mathrm{Av}(\alpha) \odot \mathrm{Av}(\alpha)$ is definable by an FO sentence if and only if $\mathrm{Av}(\alpha^r) \odot \mathrm{Av}(\alpha^r)$ is.

Recalling the definition of the clockwise spiral from Subsection 2.3.1 (Figure 2.17), we have previously remarked that Av(3142) contains the clockwise spiral and thus, so does also the class $\mathrm{Av}(\alpha)$. Furthermore, the class $\mathrm{Av}(\alpha)$ is closed under inflations, i.e. for any $\tau \in \mathrm{Av}(\alpha)$ of length k and any k permutations $\sigma_1, \ldots, \sigma_k \in \mathrm{Av}(\alpha)$ the inflation $\tau[\sigma_1, \ldots, \sigma_k]$ is also α-avoiding.

We define $\alpha^{\triangleright}$, $\alpha^{\triangleleft}$, $\alpha^{\triangle}$ and $\alpha^{\triangledown}$ to be the permutations obtained from α by removing the rightmost, leftmost, topmost and bottommost element, respectively. We say that a point set P forms a *right arrow* if it is isomorphic to $\alpha^{\triangleright}$. Furthermore, we say that a point p is in the range of the right arrow P, if p lies to the right of P and the point set $P \cup \{p\}$ is isomorphic to α. Similarly, we define *top, left and down arrows* as point sets isomorphic to $\alpha^{\triangle}, \alpha^{\triangleleft}$ and $\alpha^{\triangledown}$, respectively. Their ranges are defined analogously.

An *admissible coloring* of a permutation π is any 2-coloring $\psi : \pi \to \{\text{red}, \text{blue}\}$ such that π does not contain a monochromatic copy of α. Assume we have a permutation π with an admissible coloring. Observe that if π contains a monochromatic arrow of any orientation, say red, then all the points in the range of the arrow must be colored by the other color, i.e. blue.

Our goal is to construct, for each positive k, two permutations that are indistinguishable by FO sentences of quantifier depth k and simultaneously, only one of them belongs to the class $\mathrm{Av}(\alpha) \odot \mathrm{Av}(\alpha)$. To that end, we define a permutation π_ℓ for each positive integer ℓ. We build π_ℓ from $4\ell + 2$ blocks $B_1, B_2, \ldots, B_{4\ell+2}$ forming the clockwise spiral starting with B_1. Note that we could define π_ℓ using a tile assembly but there would be little advantage since we use the same basic structure independent of the simple permutation α.

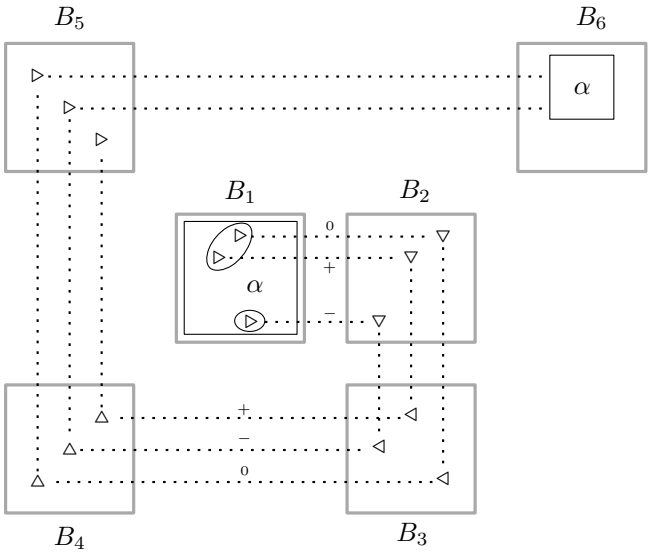

Figure 3.2: Construction of the permutation π_ℓ for $\ell = 1$ in the proof of Proposition 3.10 that contains three tracks of arrows – ground (0), positive (+) and negative (−).

The first block B_1 consists of the permutation α with the topmost point inflated by $\alpha^{\triangleright} \oplus \alpha^{\triangleright}$, the bottommost point inflated by $\alpha^{\triangleright}$, and every other point inflated with α. We will call the inflated topmost and bottommost elements of α the *top chunk* and *bottom chunk*, respectively. Observe that in any admissible coloring of B_1, both the top chunk and the bottom chunk must be monochromatic and moreover, they must use different colors. This is because all the other inflated points of α necessarily contain elements of both colors. Therefore, a pair of elements in the top and bottom chunks sharing the same color would create a monochromatic copy of α. The last block $B_{4\ell+2}$ contains only the permutation α.

Let us now describe the intermediate blocks $B_2, \ldots, B_{4\ell+1}$. Recall that the first block B_1 contains two right arrows in its chunk and one right arrow in its bottom chunk. The block B_2 will contain 3 down arrows, each down arrow placed fully in the range of one of the 3 right arrows in B_1. Similarly, the block B_3 will contain 3 left arrows, each in the range of a distinct down arrow from B_2. Generally, the block B_i for any $i \in \{2, \ldots, 4\ell + 2\}$ will consist of 3 disjoint arrows, all oriented towards B_{i+1}, and each of them inside the range of a distinct arrow in B_{i-1}.

We continue by specifying the relative position of the arrows inside each block B_i. For every even i, the arrows inside the block B_i form an increasing sequence. In other words, B_i is isomorphic to a direct sum of three arrows. For odd i, we distinguish two cases. If $i = 4t + 1$ for some t, the right arrows inside the block B_i form a decreasing sequence. Finally if $i = 4t + 3$ for some t, the block B_i is isomorphic to 231 inflated with three left arrows, i.e. $231[\alpha^{\triangleleft}, \alpha^{\triangleleft}, \alpha^{\triangleleft}]$. We say that B_i is a *monotone block* whenever the arrows in B_i form a monotone sequence. Moreover, we distinguish the arrows in monotone blocks based on the distance form the center of the spiral as *inner*, *middle* and *outer*. In the case when $i = 4t + 3$, it makes no sense to define inner and middle arrows. However, we say that the arrow inflating the element '1' of 231 is also an outer arrow.

Notice that the arrows form three disjoint sequences such that each sequence contains exactly one arrow in each of the blocks $B_1, \ldots, B_{4\ell+1}$ and moreover, the

arrow in the block B_i for $i \geq 2$ lies in the range of the arrow in the block B_{i-1}. We call these sequences of arrows *tracks*. In particular, one track contains all the outer arrows and the remaining two switch between middle and inner arrows whenever they pass through a block B_{4t+3} for some t. The track containing all the outer arrows is called the *ground*, the other track containing an arrow from the top chunk of B_1 is called *positive* and the track starting with the arrow in the bottom chunk of B_1 is called *negative*.

Finally, it remains to describe the relative positions between the points in the blocks $B_{4\ell+1}$ and $B_{4\ell+2}$. We place the topmost point in $B_{4\ell+2}$ in the range of the outer right arrow in $B_{4\ell+1}$ and we place the remaining points in the range of the middle right arrow in $B_{4\ell+1}$. See Figure 3.2.

Claim 3.11. *The permutation π_ℓ belongs to $\mathrm{Av}(\alpha) \odot \mathrm{Av}(\alpha)$ if and only if ℓ is odd.*

Assume that there is an admissible coloring of π_ℓ. We have already noticed that the top and bottom chunk in the block B_1 are both monochromatic and colored by opposite colors. Without loss of generality, we assume that the top chunk receives the color red. As we also observed, the colors of arrows on a single track must alternate and therefore, the rest of the admissible coloring of π_ℓ is uniquely determined. In particular, the outer arrow in the block B_i is colored red if and only if i is odd. The same holds for the arrows contained in the positive track while the arrow in B_i contained in the negative track is red if and only if i is even.

Observe that the middle arrow of a monotone block B_i belongs to the positive track if and only if $\lfloor i/4 \rfloor$ is even. This follows since for every t, the block B_{4t+3} flips the positive and negative track, and the middle arrow of the block B_2 belongs to the positive track. In particular, the middle arrow of the block $B_{4\ell+1}$ belongs to the positive track if and only if $\lfloor (4\ell+1)/4 \rfloor = \ell$ is even.

Let us inspect the case when ℓ is even. The outer arrow of $B_{4\ell+1}$ is red since it belongs to the ground and so is the middle arrow since it belongs to the positive track. But then all the elements in the block $B_{4\ell+2}$ must be colored blue since they all lie in the range of one of these two arrows and in particular, we found a blue copy of α.

It remains to show that for odd ℓ, we have an admissible coloring. To see this, let π_ℓ^R and π_ℓ^B be the subpermutations of π_ℓ formed by the red elements and the blue elements, respectively. We claim that both these permutations avoid α. Let us look at π_ℓ^R, the case of π_ℓ^B being analogous. To check that π_ℓ^R avoids α, we will repeatedly apply the following observation.

Observation 3.12. *Suppose that γ is a permutation which contains an interval I, and suppose that I has no copy of the simple pattern α. Let γ^- be a permutation obtained from γ by 'deflating' the interval I, i.e., by replacing I by a single element. Then γ contains α if and only if γ^- contains α.*

Our goal is to show that by repeatedly deflating α-avoiding intervals, we can transform π_ℓ^R into a permutation from the clockwise spiral.

Consider first the block B_1. The red subset of each inflated point of the copy of α is an interval. This holds vacuously for all points other than the topmost and the bottommost. As we already observed, the whole top chunk is red and the bottom chunk is blue or vice versa. Let us assume that the top chunk is colored

red as the other case is symmetric. Since all the elements in B_2 in the range of the two arrows in the top chunk are blue, the top chunk indeed forms an interval in π_ℓ^R. Thus, the red part of B_1 forms in fact a single interval isomorphic to $\alpha^\triangleright$, which can be deflated to a single point.

Consider now a block B_i for $i \in \{2, \dots, 4\ell + 1\}$. Notice that it contains at most two arrows colored red. The elements in the range of any red arrow are colored blue and therefore, each such arrow can be deflated to a single point. These points form either a monotone pair of the right kind with respect to the clockwise spiral except possibly when $i = 4t + 3$ for some t and the arrows were obtained as inflations of the elements '2' and '3' of 231. However, since these arrows were consecutive and their track contain only blue arrows in both B_{i-1} and B_{i+1}, the two red points again form an interval that can be deflated to a single point.

It remains to deal with the last block $B_{4\ell+2}$. Since ℓ is odd, the outer and middle arrow in the block $B_{4\ell+1}$ are colored differently. If the outer arrow in $B_{4\ell+1}$ is colored blue then there is only a single red point in $B_{4\ell+2}$. Otherwise, the red points in $B_{4\ell+2}$ are exactly the ones lying in the range of the middle arrow in $B_{4\ell+1}$ and thus, they form a copy of $\alpha^\triangle$ and they can be again safely deflated to a single point. This concludes the proof of Claim 3.11.

Claim 3.13. *Let k be a positive integer. For every $n, m \geq 2^{k+1} - 2$, we have $\pi_n \sim_k \pi_m$ and thus, π_n and π_m satisfy the same set of FO sentences of quantifier depth at most k in TOTO.*

We shall define a strategy for the k-move EF game on π_n and π_m using as a building block the strategy for linear orders. In particular, let $A = \{0, \dots, n\}$ and $B = \{0, \dots, m\}$ be sets of numbers equipped with the natural linear order. There exists a winning strategy for Duplicator in the $(k+1)$-move EF game on A and B due to Proposition 3.9. Let us observe a few of its properties. If any two elements a_i and a_j picked in A during the first k rounds are successive, i.e., $a_j = a_i + 1$, then necessarily, so are b_i and b_j. Otherwise, Spoiler could win in the last round by selecting b_{k+1} such that $b_i < b_{k+1} < b_j$. For the same reason, an element a_i for $i \in [k]$ is the minimum element of A if and only if b_i is the minimum element of B; the same holds when we replace minimum with maximum.

We are now ready to define the strategy for Duplicator. At the same time while playing on π_n and π_m, we simulate a virtual $(k+1)$-move EF game on A and B. Suppose Spoiler picks in the i-th round an element p_i from the block B_{c_i} in the permutation π_n (the other case being symmetric). We set its i-th move in the virtual game as picking the element $a_i = \lfloor \frac{c_i}{4} \rfloor$. Let b_i be the response of Duplicator in the virtual game and set $d_i = 4b_i + (c_i \bmod 4)$. Duplicator picks a point q_i in the permutation π_m from the block B_{d_i}. As we discussed, we have $a_i = 1$ if and only if $b_i = 1$ and $a_i = n + 1$ if and only if $b_i = m + 1$. Since moreover c_i and d_i share the same remainder modulo 4, the blocks B_{c_i} and B_{d_i} are isomorphic as point sets and Duplicator can pick q_i inside B_{d_i} mimicking the choice of p_i in B_{c_i}.

We claim that the map $p_i \mapsto q_i$ is an isomorphism between the respective subpermutations of π_n and π_m. It is sufficient to show that the pair p_i, p_j is isomorphic to q_i, q_j for all choices of $i, j \in [k]$. That is trivial if p_i and p_j belong to the same block inside π_n since then $a_i = a_j$ and thus, also $b_i = b_j$ in the virtual game. Otherwise if p_i and p_j lie in different non-successive blocks, it is sufficient

that $a_i \leq a_j$ implies $b_i \leq b_j$ and the blocks B_{c_i} and B_{d_i} are isomorphic. Finally, suppose that p_i and p_j occupy two successive blocks B_t and B_{t+1}, respectively. Then q_i and q_j must also occupy two successive blocks B_s and B_{s+1}, and q_i, q_j is isomorphic to p_i, p_j since $B_s \cup B_{s+1}$ is isomorphic to $B_t \cup B_{t+1}$.

For any positive k, Claims 3.11 and 3.13 together imply that the class $\mathrm{Av}(\alpha) \odot \mathrm{Av}(\alpha)$ cannot be defined by an FO sentence of quantifier depth at most k in TOTO. To see this, consider the permutations $\pi_{2^{k+1}-1}$ and $\pi_{2^{k+1}}$. Only one of them belongs to the class $\mathrm{Av}(\alpha) \odot \mathrm{Av}(\alpha)$ while they are indistinguishable by FO sentences of quantifier depth at most k. □

We remark that this fits well into the bigger picture. We shall see later that deciding whether π belongs to the class $\mathrm{Av}(\alpha) \odot \mathrm{Av}(\alpha)$ is NP-complete by Theorem 7.11. On the other hand, the definition of the class $\mathrm{Av}(\alpha) \odot \mathrm{Av}(\alpha)$ by an FO sentence would imply a linear time algorithm by Corollary 3.5 and thus, it would actually mean that P = NP.

3.2.2 Properties inexpressible in MSO

As our next contribution, we prove that MSO is still not expressive enough to define the property of having a fixed point. We use known results connecting words and languages with monadic second-order logic instead of MSO variant of Ehrenfeucht-Fraïssé games.

Let us start with formal definitions of words and languages. An *alphabet* Σ is an arbitrary finite set and a word over Σ is a finite sequence of elements from Σ. We denote by Σ^* the set of all words over Σ. A *language* L is simply arbitrary subset of Σ^*. Finally, a language L is *regular* if it is the language accepted by some finite automaton.

There exists a standard way of defining words as models of a logic theory. For an alphabet Σ, the signature $\mathcal{S}_\Sigma$ consists of one binary relation symbol $<$ and an unary relation symbol P_a for every $a \in \Sigma$. The symbol $<$ is intended to describe the linear order of positions in a word, while the symbols P_a form a partition of its domain and P_a describes the positions occupied by the letter a. These conditions are again easily described by FO sentences and thus, form a valid theory that we call Theory of Words over Σ (TOW_Σ).

It turns out that the languages definable by MSO sentences in TOW_Σ are easily characterized as they are precisely the regular languages. This result is known as the Büchi-Elgot-Trakhtenbrot theorem.

Theorem 3.14 ([50, 64, 109])**.** *A language $L \subseteq \Sigma^*$ is regular if and only if it is definable by an MSO sentence in TOW_Σ.*

Having these tools, we can at last prove that the property of having a fixed point is inexpressible by an MSO sentence in TOTO. In particular, we show that existence of such sentence would allow us to define an non-regular language by an MSO sentence in TOW_Σ. Note that the proof closely follows a similar argument by Albert et al. [9, Proposition 30].

Proposition 3.15. *The property of having a fixed point is not expressible by an MSO sentence in TOTO. In other words, there does not exist an MSO sentence φ such that $\sigma \models \varphi$ if and only if σ has a fixed point.*

Proof. Let us assume for contradiction that there exists an MSO sentence φ in TOTO expressing the property that a permutation has a fixed point. We will show how to transform φ into an MSO sentence ρ in TOW_Σ for the alphabet $\Sigma = \{a, b\}$ such that ρ defines the language $L = \{a^n\, b\, a^n \mid n \in \mathbb{N}\}$. This proves the intended claim since L is clearly not regular by a standard use of the pumping lemma.

The main idea is that the evaluation of ρ on a word of the form $w = a^k\, b\, a^\ell$ simulates the evaluation of φ on the permutation

$$\pi_{k,\ell} = (\oplus^k 1) \ominus 1 \ominus (\oplus^\ell 1).$$

It is easy to see that $\pi_{k,\ell}$ has a fixed point if and only if $k = \ell$.

In order to transform φ into an MSO sentence in TOW_Σ, we need to replace all atomic formulas of type $x <_x y$ and $x <_y z$. We assume that the variable x_b is set to the position of the only letter b in the word w. We replace $x <_x y$ simply by $x < y$, effectively mapping the word domain to the permutation from left to right. The situation is more complicated with $<_y$ since we need to decide the truth value based on the positions relative to x_b. We replace $x <_y y$ by

$$[x < y \wedge ((x < x_b \wedge y < x_b) \vee (x_b < x \wedge x_b < y))] \vee$$
$$[y < x \wedge ((x < x_b \wedge x_b < y) \vee (x < x_b \wedge y = x_b) \vee (x = x_b \wedge x_b < y))]\,.$$

The first line takes care of the case when $x < y$, since then both x and y must either lie before or after the letter b. The second line concerns the case when $y < x$, in which case either x and y are separated by the letter b or at most one of them is equal to it. Let ρ' be the MSO formula in TOW_Σ obtained by these replacements. We need to additionally check that w contains exactly one occurrence of the letter b and assign its position to the variable x_b.

$$\rho = \exists x_b\, (P_b(x_b) \wedge \forall x\, (P_b(x) \to x = x_b) \wedge \rho'). \qquad \square$$

3.2.3 Monadic second-order model checking

We shift our attention to the complexity of deciding MSO formulas of TOTO. On one hand, we show that there exists an fpt-algorithm for this problem parameterized by the clique-width and the length of the formula. The algorithm reduces the problem to model checking on labeled graphs. On the other hand, we complement this with a negative result showing that checking MSO sentences on permutations from any class with the computable long path property is in some sense as hard as checking MSO sentences on graphs.

Let us start by extending the theory TOG to labeled graphs. For a positive integer p, the signature $\mathcal{S}_G^p$ consists of one binary relation symbol E and p unary relation symbols $U_1, \ldots, U_p$. The symbol E is intended to describe the edges of the graph, while the symbols $U_1, \ldots, U_p$ partition the domain and U_i describes the vertices with label i. We require that E is symmetric relation and that $U_1, \ldots, U_p$ form a partition of the domain. These conditions are easily described by FO sentences and therefore, form a valid theory that we call Theory of p-labeled Graphs (TOG_p). A *p-labeled graph* is for us any finite model $(V, E, U_1, \ldots, U_p)$ of the theory TOG_p.

The *clique-width* of a p-labeled graph $G = (V, E, U_1, \ldots, U_p)$, denoted by $\mathrm{cw}(G)$, is defined similarly to the clique-width of xy-digraphs as introduced in

Subsection 2.1.4. In particular, it is the minimum number of labels needed to construct G using the following five operations:

1. Creation of a single vertex with label i (denoted $\mathbf{i}$).
2. Disjoint union of two labeled graphs G and H (denoted $G \oplus H$).
3. Renaming label i to j (denoted $\rho_{i\to j}$).
4. Adding an edge between every i-labeled vertex and every j-labeled vertex (denoted $\eta_{i,j}$), where $i \neq j$.

Note that we treat the final labels $\{1, \ldots, p\}$ as part of the set of labels used during the construction. Again, an algebraic term using the above operations with at most k distinct labels is called a *k-expression* of G.

Courcelle, Makowsky and Rotics [57] constructed an fpt-algorithm for deciding whether a p-labeled graph satisfies a given formula. Note that the algorithm required a k-expression together with the graph. Later, this requirement was removed due to a series of polynomial-time approximation algorithms for clique-width (see [103, 80, 102]). Luckily, we do not have to concern ourselves with this issue since the clique-width of permutations can be approximated in linear time. It suffices to obtain an approximate tree decomposition of a given permutation by Theorem 2.3 and then transform it to an approximate k-expression by Propositions 2.10 and 2.13.

Theorem 3.16 ([57, Theorem 4])**.** *Given a p-labeled graph G on n vertices together with its k-expression and an MSO sentence φ in TOG_p, we can decide $G \models \varphi$ in time $f(|\varphi|, k) \cdot n$ for some computable, yet non-elementary, function f.*

We reduce the problem of deciding MSO sentences in TOTO over permutations to deciding MSO sentences in TOG_4 over 4-labeled graphs. Note that the main idea of the proof closely follows a similar reduction for directed graphs given by Ganian et al. [71, Theorem 4.2].

Theorem 3.17. *Given a permutation π of length n and a monadic second-order sentence φ in TOTO, we can decide $\pi \models \varphi$ in time $f(|\varphi|, \mathrm{cw}(\pi)) \cdot n$ for some computable, yet non-elementary, function f.*

Proof. Given the input permutation π, we construct a 4-labeled graph G such that $\mathrm{cw}(G) \leq 4 \cdot \mathrm{cw}(\pi)$ in the following way. For more clarity, we use as labels the set $L = \{head_x, head_y, tail, center\}$ instead of the set $\{1, 2, 3, 4\}$. For each point p of the permutation diagram S_π, we create a set of 4 vertices $A_p = \{h_p^x, h_p^y, t_p, c_p\}$ forming a star with the vertex c_p in its center. Moreover, we set the labels of h_p^x, h_p^y, t_p, c_p to $head_x$, $head_y$, $tail$ and $center$, respectively. Finally for every pair of different points p and q such that $p.x < q.x$, we add an edge $t_p h_q^x$ and similarly for every p and q such that $p.y < q.y$, we add an edge $t_p h_p^y$. See Figure 3.3.

Let us verify that the clique-width of G is bounded as promised. In fact, we show that from a k-expression of π we can construct in linear time a $(4 \cdot k)$-expression of G. We interpret the set of labels as the set $[k] \times L$. We replace every creation of a single point bearing label i (operation $\mathbf{i}$) with the expression defining the creation of the labeled graph on 4 vertices that is isomorphic to A_p where each vertex receives the respective label from the set $\{i\} \times L$. Every operation $\eta_{i,j}^\alpha$ for $\alpha \in \{x, y\}$ is then replaced with the operation $\eta_{(i,tail),(j,head_\alpha)}$. And finally, every renaming operation $\rho_{i\to j}$ is replaced with the composition of $\rho_{(i,\ell)\to(j,\ell)}$ over all

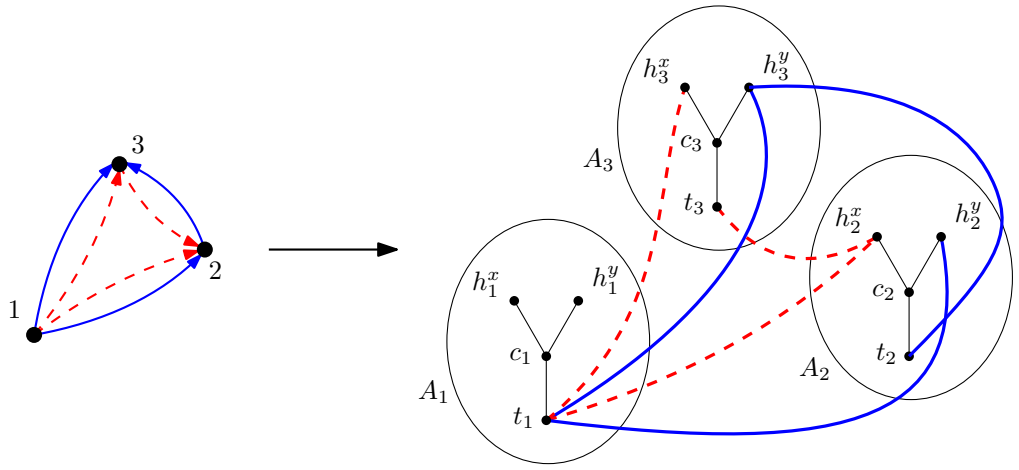

Figure 3.3: Encoding permutation 132 (left) into a 4-labeled graph G (right). We draw the edges (p, q) such that $p.y < q.y$ using blue solid curves while red dashed curves represent the edges (p, q) such that $p.x < q.x$.

$\ell \in L$. At the end, we simply interpret the labels $\{1\} \times L$ as the set L. It is easy to see that the obtained $(4 \cdot k)$-expression defines exactly the 4-labeled graph G.

Let φ be an MSO sentence in TOTO. We describe construction of an MSO sentence ψ in TOG_4 such that $\pi \models \varphi$ if and only if $G \models \psi$. Every element variable x is replaced with a set variable Z_x that is intended to describe the set A_x in G. To that end, we define a predicate $\mathit{single}(X)$ that tests whether a given set variable X is actually equal to A_p for some point p as

$$\begin{aligned}\mathit{single}(X) = \exists x_1, x_2, x_3, x_4 \in X \Big[&\mathit{head}_x(x_1) \wedge \mathit{head}_y(x_2) \wedge \mathit{center}(x_3) \wedge \mathit{tail}(x_4) \\ &\wedge E(x_1, x_3) \wedge E(x_2, x_3) \wedge E(x_3, x_4) \\ &\wedge \forall x \in X \ (x = x_1 \vee x = x_2 \vee x = x_3 \vee x = x_4)\Big].\end{aligned}$$

We replace every occurrence of $\exists x \ \rho$ in φ with $\exists Z_x(\mathit{single}(Z_x) \wedge \rho)$, and similarly every $\forall x \ \rho$ is replaced with $\forall Z_x(\mathit{single}(Z_x) \to \rho)$. Any occurrence of $x \in X$ is replaced simply with $Z_x \subseteq X$. We additionally enforce that we only consider sets obtained as unions of the sets A_p by replacing $\exists X \ \rho$ and $\forall X \ \rho$, respectively, with

$$\begin{aligned}\exists X \ \Big[\forall x \in X \ \exists Z_x(\mathit{single}(Z_x) \wedge Z_x \subseteq X \wedge x \in Z_x)\Big] \wedge \rho, \\ \forall X \ \Big[\forall x \in X \ \exists Z_x(\mathit{single}(Z_x) \wedge Z_x \subseteq X \wedge x \in Z_x)\Big] \to \rho.\end{aligned}$$

Finally, we translate predicates $x <_x y$ and $x <_y y$ with predicates $x\text{-}\mathit{arc}(Z_x, Z_y)$ and $y\text{-}\mathit{arc}(Z_x, Z_y)$, respectively, defined as follows

$$\begin{aligned}x\text{-}\mathit{arc}(Z_x, Z_y) &= \exists x' \in Z_x \ \exists y' \in Z_y \, (\mathit{tail}(x') \wedge \mathit{head}_x(y') \wedge E(x', y')) \,, \\ y\text{-}\mathit{arc}(Z_x, Z_y) &= \exists x' \in Z_x \ \exists y' \in Z_y \, (\mathit{tail}(x') \wedge \mathit{head}_y(y') \wedge E(x', y')) \,.\end{aligned}$$

It is straightforward to check that indeed $G \models \psi$ if and only if $\pi \models \varphi$.

To conclude the proof, we describe the whole algorithm. Given the permutation π and MSO sentence φ, we construct the 4-labeled graph G in time $O(n)$ and the MSO sentence ψ in time $O(|\varphi|)$. Observe that the tree-width of π is

at most $8 \cdot \mathrm{cw}(\pi)$ by combination of Corollaries 2.12 and 2.15. Therefore, the approximation algorithm of Theorem 2.3 provides us with a tree decomposition of π of width at most $16 \cdot \mathrm{cw}(\pi) + 1$ in time $f_1(\mathrm{cw}(\pi)) \cdot n$. We continue by translating this tree decomposition into a grid tree with grid-width at most $16 \cdot \mathrm{cw}(\pi) + 3$ by Proposition 2.10 and subsequently, to a $(2 \cdot (16 \cdot \mathrm{cw}(\pi) + 3)^2)$-expression of π by Proposition 2.13. As we described, we can translate this expression into a $(8 \cdot (16 \cdot \mathrm{cw}(\pi) + 3)^2)$-expression of G in linear time. Finally, it remains to invoke Theorem 3.16 and check whether $G \models \psi$ in time $f_2(|\psi|, \mathrm{cw}(G)) \cdot n$ for some computable function f_2. □

It has been pointed to us that, alternatively, Theorem 3.17 can be proved by replacing permutations with their incidence graphs considered as directed, edge-labeled graphs. It suffices to show that the expressive power of MSO in TOTO is equivalent to the expressive power of MSO over incidence graphs of permutations when we allow variables to represent not only vertices but also edges. A variant of Theorem 3.17 parameterized by tree-width then follows directly from the celebrated Courcelle's theorem [55].

We wrap up this section with an accompanying negative result. We have already remarked in Chapter 2 that all known classes of unbounded tree-width (and thus, also unbounded clique-width) share the long path property. We show that with the mild additional assumption of computability, deciding MSO sentences in any permutation class with the long path property is as hard as deciding MSO sentences on all graphs.

Modified $\mathcal{F}$-assembly. We construct a family of tiles using pairs of points called *atomic pairs* that form the pattern 12. We shall always consider both points of a single atomic pair at once which justifies their name. When creating a permutation from tiles consisting of atomic pairs, we slightly change each atomic pair as to force a specific relative order of two atomic pairs in neighboring tiles that share the same coordinates. Suppose we are given a monotone gridding matrix $\mathcal{M}$ whose cell graph is a tree. Assume that the tree is rooted and oriented all the edges consistently outwards from the root. For a vertex v of the tree, its parent is the only in-neighbor of v. Suppose we are given a consistent orientation $\mathcal{F}$ of $G_{\mathcal{M}}$ and a family of tiles $\mathcal{Q}$ that contain atomic pairs. We define a *modified $\mathcal{F}$-assembly* of $\mathcal{Q}$ as follows.

Let v be a vertex in $G_{\mathcal{M}}$ with parent w. Let $X = \{p, q\}$ be an atomic pair in Q_v such that p lies to the left and below of q. If w and v share the same row, then we move p up by a tiny distance (increasing its y-coordinate) and we move q down by a tiny distance (decreasing its y-coordinate) without changing relative position of any points in tiles T_v and T_w. On the other hand, if w and v share the same column, we do the same modification with the x-coordinate, i.e., we increase the x-coordinate of p and decrease the x-coordinate of q. Then we perform the usual $\mathcal{F}$-assembly on this modified family of tiles.

We say that a pair (a, b) of numbers *sandwiches* the pair (c, d) if $a < c < d < b$. Notice that by this modification, if there was an atomic pair X in Q_v and Y in Q_w then after the modification the x-coordinates of Y sandwich the x-coordinates of X if v and w share the same column. Otherwise the same holds with respect to the y-coordinates of X and Y. In both cases, we say that the atomic pair X

sandwiches the atomic pair Y. In this context, a *track* is a sequence of atomic pairs $X_1, \ldots, X_m$ such that X_i sandwiches X_{i+1}.

Theorem 3.18. *Let $\mathcal{C}$ be a permutation class with the poly-time computable long path property. There is a polynomial time algorithm that given a graph G on n vertices and an MSO sentence φ in TOG, computes a permutation $\pi \in \mathcal{C}$ of length $O(n^2)$ and an MSO sentence ψ in TOTO of length $O(|\varphi|)$ such that $G \models \varphi$ if and only if $\pi \models \psi$.*

Proof. We assume that the vertex set of G is precisely the set $[n]$. The basic idea is that we can represent the adjacency matrix of G by a permutation $\pi \in \mathcal{C}$ similarly to the proof of Theorem 3.6. However, this time we split the adjacency matrix into individual rows and we represent each row using a single cell along a path in the cell graph.

Construction of π. We first describe the construction of the permutation π as it is more straightforward. We start by obtaining in polynomial time a monotone gridding matrix $\mathcal{M}$ such that (i) $\mathrm{Grid}(\mathcal{M})$ is a subclass of $\mathcal{C}$, (ii) the cell graph $G_{\mathcal{M}}$ is a path on $n+3$ vertices and moreover, (iii) one endpoint of this path is the single non-empty cell in the leftmost column of $\mathcal{M}$. The properties (i) and (ii) are directly implied by the poly-time computable long path property. In order to guarantee (iii), we can generate a monotone gridding matrix $\mathcal{M}'$ that satisfies (i) and whose cell graph is a path on $2n+6$ vertices. We can then split the path in $G_{\mathcal{M}'}$ by removing either of the (at most two) non-empty cells in the leftmost column. One choice leads to a path of length at least $n+3$ with an endpoint in its leftmost column.

We orient the path in the cell graph outwards from the leftmost cell in $\mathcal{M}$ and denote the vertices in the order on the path as $v_1, \ldots, v_{n+3}$. We now describe a family of $(6n+18)$-tiles $\mathcal{P}$ such that every non-empty tile contains only points from its exact diagonal, i.e. the x- and y-coordinate of any point are equal. The tile P_{v_1} contains only a single atomic pair called the *anchor* defined as

$$S = \{(3,3),\ (6n+16, 6n+16)\}.$$

All remaining tiles share most of the same points. To start with, they contain 2 points below and to the left of everything else and 2 points above and to the right of everything else, called *barricades*, and defined as

$$(1,1), (2,2), (6n+17, 6n+17), (6n+18, 6n+18).$$

Other than these extremal points, the tile contains an atomic pair C_1 in its lower left corner and two atomic pairs C_2 and C_3 in its upper right corner. Formally,

$$\begin{aligned} C_1 &= \{(4,4),(5,5)\},\\ C_2 &= \{(6n+12, 6n+12), (6n+13, 6n+13)\},\\ C_3 &= \{(6n+14, 6n+14), (6n+15, 6n+15)\}. \end{aligned}$$

The atomic pairs C_1 and C_3 mark the ends of each tile when we omit the barricades while the pair C_2 marks the "inside" of each tile.

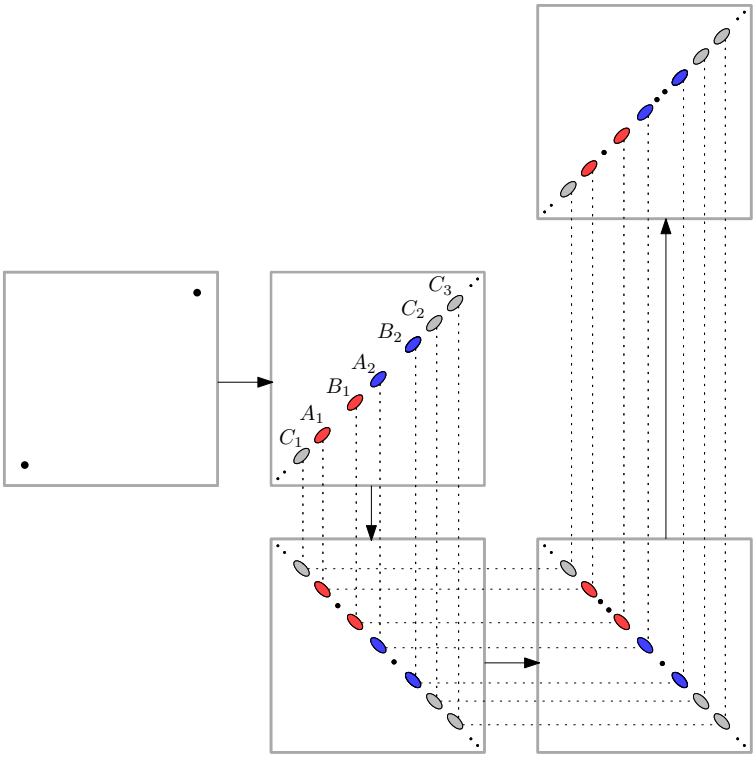

Figure 3.4: Encoding the complete graph on 2 vertices into a path consisting of 5 monotone tiles in the proof of Theorem 3.18. Atomic pairs are displayed as small ellipses that are connected into tracks via dotted lines. The pairs A_i and B_i are colored with the same color (red and blue, respectively), while the pairs C_1, C_2 and C_3 are gray.

For each vertex $i \in [n]$ of G, we add two atomic pairs A_i and B_i defined as

$$A_i = \{(6i, 6i), (6i+1, 6i+1)\}$$
$$B_i = \{(6i+4, 6i+4), (6i+5, 6i+5)\}.$$

The rest of the points differ between the tiles $P_{v_2}, \ldots, P_{v_{n+3}}$. The tile P_{v_2} does not contain any other points. The tile P_{v_3} contains additionally the point $(6i+3, 6i+3)$ for every $i \in [n]$. Each remaining tile represents one row of the adjacency matrix. For $i \in [n]$, the tile $P_{v_{i+3}}$ contains a pair of points $(6i+3, 6i+3)$ and $(6i+4, 6i+4)$. Moreover, it contains the point $(6j+3, 6j+3)$ for any $j \in [n]$ such that ij is an edge in G. See Figure 3.4.

The matrix $\mathcal{M}$ has a consistent orientation $\mathcal{F}$ by Lemma 1.2. Moreover, we can choose $\mathcal{F}$ such that the row containing the cells v_1 and v_2 does not get complemented. We set π as the modified $\mathcal{F}$-assembly of $\mathcal{P}$.

Observe that π contains $2n+3$ disjoint tracks starting in the cell v_2 and ending in v_{n+3}. Three of them are formed by the atomic pairs C_1, C_2 and C_3, respectively. The rest is formed by the atomic pairs A_i and B_i for each $i \in [n]$. We call the track formed by an atomic pair X the *X-track*. So, for instance, we have the C_1-track, A_2-track, B_3-track etc.

Construction of ψ. Now we describe how to translate the MSO sentence φ in TOG to the MSO sentence ψ in TOTO. Note that we shall define ψ using an

expanded language that is however easily translated to MSO sentences. For two points p, q in π and $\alpha \in \{x, y\}$, we denote by $(p, q)_\alpha$ all the points in π whose α-coordinates lie in the interval $(p.\alpha, q.\alpha)$. Similarly, we denote by $[p, q]_\alpha$ all the points in π whose α-coordinates lie in the interval $[p.\alpha, q.\alpha]$. It is easy to see that predicates like '$(p, q)_x = \emptyset$' or '$|(p, q)_x| = 2$' are easily expressed via MSO sentences.

First let us describe how we can test whether a set variable S in ψ is equal to a track. If we are given an atomic pair $D = (p_1, p_2)$ in the tile P_{v_2} then we claim that its corresponding track can be inductively generated by the following rules

(i) p_1, p_2 belong to T,
(ii) for every $q_1, q_2 \in T$ such that $(q_1, q_2)_x = \emptyset$ and $(q_1, q_2)_y = \{r_1, r_2\}$ for pairwise different r_1, r_2, the points r_1, r_2 also belong to T, and
(iii) for every $q_1, q_2 \in T$ such that $(q_1, q_2)_y = \emptyset$ and $(q_1, q_2)_x = \{r_1, r_2\}$ for pairwise different r_1, r_2, the points r_1, r_2 also belong to T.

Claim 3.19. *Let $D = \{p_1, p_2\}$ be an atomic pair in the tile P_{v_2}. A subset T of π is a D-track if and only if (i), (ii), (iii) hold for T and moreover, T is minimal such set with respect to inclusion.*

Suppose that T is a D-track. The condition (i) holds vacuously and thus, the only way T could violate these conditions is if there was a pair of points $q_1, q_2 \in T$ from different tiles for which the condition (ii) or (iii) fails. In that case, both sets $(q_1, q_2)_x$ and $(q_1, q_2)_y$ would contain at most 2 points. That is, however, not possible for q_1 and q_2 from different tiles since at least one of these sets must contain four points of the barricades. For any proper subset T' of T, let $i \geq 2$ be the index such that the tile P_{v_i} is the first where T' and T differ. If $i = 2$ then T' violates (i), otherwise T' violates (ii) or (iii) for the atomic pair in the tile $P_{v_{i-1}}$.

For the other direction, assume that T is an inclusion-wise minimal set satisfying the conditions (i), (ii) and (iii). The atomic pair D in the tile P_{v_2} belongs to T by condition (i). It then suffices to alternate using conditions (ii) and (iii) to show that T must contain the whole track D-track. Since the D-track itself satisfies (i), (ii) and (iii), T cannot contain any other points as it would not be inclusion-wise minimal.

The conditions (i), (ii) and (iii) can easily be encoded as an MSO predicate $\mathit{suptrack}(p_1, p_2, T)$. The power of MSO allows us to further enforce that T is a minimal set satisfying these conditions (and thus, equal to the desired track by Claim 3.19) by an MSO predicate $\mathit{track}(p_1, p_2, T)$ defined as

$$\mathit{track}(p_1, p_2, T) = \mathit{suptrack}(p_1, p_2, T) \wedge \forall S \left[(S \subsetneq T) \rightarrow \neg\, \mathit{suptrack}(p_1, p_2, S) \right].$$

At the very beginning of ψ, we can fix two variables a_1 and a_2 to the anchors of π since they are the two leftmost points of π as follows

$$\exists a_1 a_2 \left[a_1 <_y a_2 \wedge \forall x \; (x = a_1 \vee x = a_2 \vee a_2 <_x x) \right].$$

We additionally introduce three set variables T_{C_1}, T_{C_2} and T_{C_3}, set to the C_1-track, C_2-track and C_3-track, respectively. For $\alpha \in \{x, y\}$, let $p \lessdot_\alpha q$ be the FO predicate that evaluates true if and only if $p <_\alpha q$ and there is no point r such that $p <_\alpha r <_\alpha q$. It is sufficient to find the atomic pairs C_1, C_2 and C_3 inside P_{v_2} and then apply the predicate *track*. We can find the atomic pairs since C_1

contains the first two points just above the anchor a_1 while C_2 and C_3 consist of the last four points just below the anchor a_2. Here we use the fact that the row containing the cells v_1 and v_2 is not complemented when applying the $\mathcal{F}$-assembly. Formally, we write

$$\exists\, T_{C_1}, T_{C_2}, T_{C_3}\ [\exists p_1, q_1, p_2, q_2, p_3, q_3\ (a_1 \lessdot_y p_1 \lessdot_y q_1\ \wedge\ p_2 \lessdot_y q_2 \lessdot_y p_3 \lessdot_y q_3 \lessdot_y a_2 \\ \wedge\, track(p_1, q_1, T_{C_1}) \wedge track(p_2, q_2, T_{C_2}) \wedge track(p_3, q_3, T_{C_3}))].$$

We replace every vertex variable x in φ with two set variables T_{A_x} and T_{B_x} intended to describe the A_x-track and B_x-track. With that in mind, we define predicate $vertex(T_{A_x}, T_{B_x})$ that tests this requirement. It is sufficient to find two consecutive atomic pairs $\{p_1, q_1\}$ and $\{p_2, q_2\}$ inside P_{v_2} such that moreover, there are two points in the sets $(p_1, q_1)_x$, $(p_2, q_2)_x$ (the atomic pairs A_x and B_x inside P_{v_3}) and only one in the set $(q_1, p_2)_x$ (the additional point separating A_x and B_x inside P_{v_3}). Formally, we have

$$vertex(T_{A_x}, T_{B_x}) = \exists p_1, q_1, p_2, q_2\ (a_1 <_y p_1 \wedge q_2 <_y a_2 \wedge p_1 \lessdot_y q_1 \lessdot_y p_2 \lessdot_y q_2 \\ \wedge\, track(p_1, q_1, T_{A_x}) \wedge track(p_2, q_2, T_{B_x})).$$

We replace every occurrence of $\exists x\ \rho$ in φ with $\exists T_{A_x}, T_{B_x}(vertex(T_{A_x}, T_{B_x}) \wedge \rho)$, and similarly every $\forall x\ \rho$ is replaced with $\forall T_{A_x}, T_{B_x}(vertex(T_{A_x}, T_{B_x}) \rightarrow \rho)$. Furthermore, any occurrence of $x \in X$ is replaced simply with $T_{A_x} \subseteq X \wedge T_{B_x} \subseteq X$ and we again change quantifications over sets to capture only sets that are formed as union of tracks T_{A_x}, T_{B_x} for some set of vertices. Formally, we replace $\exists X\ \rho$ and $\forall X\ \rho$, respectively, with

$$\exists X\ \Big[\forall x \in X\ \exists T_A, T_B\ (vertex(T_A, T_B) \wedge T_A \cup T_B \subseteq X \wedge x \in T_A \cup T_B)\Big] \wedge \rho,$$
$$\forall X\ \Big[\forall x \in X\ \exists T_A, T_B\ (vertex(T_A, T_B) \wedge T_A \cup T_B \subseteq X \wedge x \in T_A \cup T_B)\Big] \rightarrow \rho.$$

Finally, we need to replace every predicate $E(x, y)$ inside φ. In order to do that, we first show that it is possible to define a predicate identifying a single tile inside π. More specifically, we define a predicate $tile^{(r,c)}(S)$ for every $r, c \in \{-1, 1\}$ that evaluates true if and only if S is the point set of a tile with row orientation r and column orientation c in $\mathcal{F}$. Instead of writing the full technical definition, we list individual properties that are sufficient and each of them is easily seen to be expressible by an MSO sentence. Moreover, we only list the properties defining $tile^{(1,1)}(S)$ as the other three possibilities are symmetric. The following must hold for a set S satisfying $tile^{(1,1)}(S)$:

(i) S is equal to $[p, q]_x \cap [r, s]_y$ for some points p, q, r and s,
(ii) S is an increasing point set,
(iii) $|S \cap T_{C_1}| = |S \cap T_{C_2}| = |S \cap T_{C_3}| = 2$, and
(iv) the bottommost two points of S belong to T_{C_1} while the topmost two points belong to T_{C_3}.

When we have a set variable S representing a single tile, we can test whether there is an edge between vertices x and y (represented by variables T_{A_x}, T_{B_x}, T_{A_y} and T_{B_y}) represented inside the tile S by verifying that

(i) there are exactly two points lying between $T_{A_x} \cap S$ and $T_{B_x} \cap S$ both horizontally and vertically, and
(ii) there is exactly one point lying between $T_{A_y} \cap S$ and $T_{B_y} \cap S$ both horizontally and vertically.

We can clearly encode this as an MSO predicate $\mathit{tile\_edge}(T_{A_x}, T_{B_x}, T_{A_y}, T_{B_y}, S)$. Finally, we replace every predicate $E(x, y)$ inside φ with the predicate $\mathit{edge}(T_{A_x}, T_{B_x}, T_{A_y}, T_{B_y})$ defined as

$$\mathit{edge}(T_{A_x}, T_{B_x}, T_{A_y}, T_{B_y}) = \exists S \bigvee_{r,c \in \{-1,1\}} \mathit{tile}^{(r,c)}(S) \wedge \mathit{tile\_edge}(T_{A_x}, T_{B_x}, T_{A_y}, T_{B_y}, S).$$

To wrap up, observe that π is a permutation of length $O(n^2)$ and the length of ψ is $O(|\varphi|)$ as promised. Moreover, it is clear from the construction of π and ψ that $G \models \varphi$ if and only if $\pi \models \psi$. □

4. Permutation pattern matching

In this chapter, we explore how hard is it to find a given pattern π in a given permutation τ. This problem, called Permutation Pattern Matching (PPM), is the most natural decision problem that involves permutations and as such it has received plenty attention in the literature [6, 23, 39, 49, 48, 78, 77, 86].

First in Section 4.1, we look at the problem through the lens of classical complexity theory. We start by introducing standard notions and known results. Additionally, we include our own original proof of the fact that PPM is NP-complete.

In Section 4.2, we employ the framework of parameterized complexity. We impose conditional lower bounds on the running time of algorithms counting patterns or finding vincular patterns. Moreover, we rule out fpt-algorithms for PPM parameterized by any reasonable parameter of the pattern π under standard complexity-theory assumptions. On the positive side, we devise efficient algorithms for counting bivincular and partially ordered patterns of small tree-width.

Finally in Section 4.3, we investigate the behavior of PPM with the additional restriction of the pattern π to a fixed class $\mathcal{C}$. Assuming that $\mathcal{C}$ has the computable long path property, we can reproduce most of the hardness results that hold for general patterns. The only caveat is that the obtained conditional lower bounds are weaker approximately by order of square root in the exponent. As an example, we rule out algorithm counting patterns from $\mathcal{C}$ in time $f(k)\cdot n^{o(\sqrt{k})}$ whereas general patterns cannot be counted in time $f(k) \cdot n^{o(k/\log k)}$ (both under the so-called Exponential Time Hypothesis). However, assuming that $\mathcal{C}$ has the computable deep tree property, we derive stronger conditional lower bounds that differ from the general case only by a logarithmic term in the exponent, e.g., we rule out algorithm counting patterns from $\mathcal{C}$ in time $f(k) \cdot n^{o(k/\log^2 k)}$.

4.1 Classical complexity

Let us first briefly recall the basic notions of classical complexity theory. For a detailed introduction, we refer, e.g., to the monograph by Arora and Barak [14]. Formally, a *decision problem* is a language over a finite alphabet Σ which encodes the positive instances. The two most fundamental classes of problems are P and NP. The class P contains all problems solvable in polynomial time (in the size of the input) by a deterministic Turing machine, while NP consists of all problems solvable in polynomial time by a non-deterministic Turing machine.

A *polynomial time reduction* from a problem A to a problem B is a mapping $f : \Sigma^* \to \Sigma^*$ computable in polynomial time by a deterministic Turing machine such that for every $x \in \Sigma^*$, x is a positive instance of A if and only if $f(x)$ is a positive instance of B. For a class of problems C, we say that a problem A is *C-hard* if every problem in C can be reduced to A by a polynomial time reduction. Moreover, we say that a problem A is *C-complete* if A is C-hard and simultaneously, A is contained in C. In particular, the NP-complete problems can be seen as the hardest problems in NP. The most prominent example of such problem is surely SAT, or its more restricted variant 3-SAT.

3-SAT	
Input:	A 3-CNF formula φ with n variables and m clauses.
Output:	Does φ have a satisfying assignment?

Showing that a decision problem is NP-hard implies that it cannot be solved deterministically in polynomial time under the (widely believed) assumption $\mathsf{P} \neq \mathsf{NP}$. In many cases, we are able to obtain more fine-grained lower bounds under stronger assumptions. Namely, we base some of our result on the famous *Exponential Time Hypothesis (ETH)* by Impagliazzo, Paturi and Zane [84, 85].

Hypothesis 1 (Exponential Time Hypothesis [84, 85])**.** *There exists $\delta > 0$ such that 3-SAT cannot be solved in time $O(2^{\delta n})$ where n is the number of variables.*

However, it is more suitable for our purposes to use an equivalent variant of ETH that bounds the running time in the number of clauses rather than in the number of variables. The equivalence follows from the Sparsification lemma of Impagliazzo, Paturi and Zane [85].

Hypothesis 2 (ETH, alternative version [85])**.** *There exists $\delta' > 0$ such that 3-SAT cannot be solved in time $O(2^{\delta' m})$, where m is the number of clauses. In particular, no $2^{o(m)}$ algorithm exists.*

Let us formally introduce Permutation Pattern Matching (PPM) as a decision problem. Additionally, we define its counting variant known as #PPM.

Permutation Pattern Matching (PPM)	
Input:	Permutations π of length k and τ of length n.
Output:	Is π contained in τ?

#Permutation Pattern Matching (#PPM)	
Input:	Permutations π of length k and τ of length n.
Output:	The number of occurrences of π in τ.

Permutation Pattern Matching was shown to be NP-complete by Bose, Buss and Lubiw [39]. Moreover, if we inspect their proof properly, we see that their reduction from 3-SAT implies that PPM cannot be solved in time $2^{o(n)}$ unless ETH fails. We include an alternative proof of this fact. The reason for this is twofold, we wanted to include a proof of this seminal result for the sake of completeness and simultaneously, it introduces ideas that will be utilized in later reductions.

Theorem 4.1. *PPM is NP-complete and moreover, it cannot be solved in time $2^{o(|\tau|)}$ where τ is the text permutation unless ETH fails.*

Before we start proving Theorem 4.1, let us introduce a handy way of enforcing satisfying assignments via the means of embedding words. This particular observation will prove itself useful in several of the upcoming reductions.

Observation 4.2. *Let $\Sigma = \bigcup_{i=1}^{3}\{T_i, F_i\}$ and suppose $w = x_1x_2x_3$ is a word over the alphabet Σ such that $x_i \in \{T_i, F_i\}$ for each $i \in [3]$. Then w is contained as a subsequence in the word $T_1F_2F_1T_2F_3F_2T_3$ if and only if $w \neq F_1F_2F_3$.*

Proof of Theorem 4.1. We describe an efficient reduction from 3-SAT. Let φ be a given 3-CNF formula. Suppose that φ has n variables $x_1, \ldots, x_n$ and m clauses $K_1, \ldots, K_m$. We will assume, without loss of generality, that each clause contains exactly three literals, and no variable appears in a single clause more than once. We will construct permutations π and τ such that φ is satisfiable if and only if τ contains π. However, we shall first reduce to a slightly more restricted problem called Left PPM.

Left-aligned Permutation Pattern Matching (Left PPM)	
Input:	Permutations π of length k and τ of length n.
Output:	Is there an embedding of π into τ that maps the leftmost point of π to the leftmost point of τ?

Any embedding of π into τ that maps the leftmost point of π to the leftmost point of τ is a *left-aligned embedding (of π into τ).* Observe that it is equivalent to only require that the leftmost point of τ is used by the embedding. It follows automatically that the only point that could map to the leftmost point in τ is the leftmost point in π.

Before delving into the formal construction, let us describe the global structure of π and τ. The permutations π and τ shall both have a matrix-like structure. In particular, the pattern π will represent a matrix with m rows (one per each clause) and n columns (one per each variable of φ) while the text τ will represent a matrix also with m rows but with $2n$ columns (two per each variable, representing the two possible assignments).

Constructing the pattern π. Let us first define the pattern as a point set P not necessarily in general position. We start by defining two points, called *anchors*, in π as

$$a_1^P = (1, 2n+2), \qquad a_2^P = (2m+2, 1).$$

For each $i \in [n]$, we define a pair of points X_i associated to the variable x_i as

$$X_i = \{(2m+3i, 2i),\ (2m+3i+2, 2i+1)\}.$$

For each $t \in [m]$, we associate to the clause K_t of φ a pair of points A_t defined as

$$A_t = \{(2t, 2n+5t-2),\ (2t+1, 2n+5t+2)\}.$$

Observe that the pairs $X_1, \ldots, X_n$ lie horizontally between a_1^P and a_2^P while the pairs $A_1, \ldots, A_m$ lie vertically between a_1^P and a_2^P. Moreover, the pairs $X_1, \ldots, X_n$ form an increasing permutation of length $2n$ while the pairs $A_1, \ldots, A_m$ form an increasing permutation of length $2m$. The pairs naturally define a grid-like structure. We call the vertical strip bounded by a pair X_i the *X_i-column* and similarly, we call the horizontal strip bounded by a pair A_t the *A_t-row*. Moreover, the *(X_i, A_t)-cell* is the intersection of the X_i-column and the A_t-row.

Suppose the clause K_t contains variables x_i, x_j and x_k, for some $i < j < k$. We add to the A_t-row three points, lying in the (X_i, A_t)-cell, (X_j, A_t)-cell and (X_k, A_t)-cell respectively. Moreover, these three points form the increasing pattern 123. Formally, these points are defined as

$$(2m+3i+1, 2m+5t-1), \quad (2m+3j+1, 2m+5t), \quad (2m+3k+1, 2m+5t+1).$$

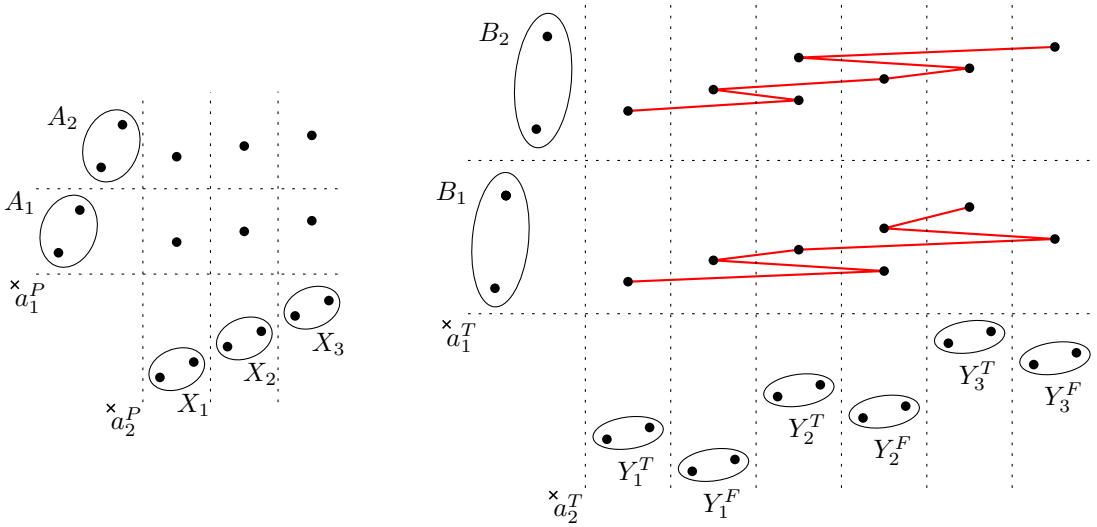

Figure 4.1: Permutations π (left) and τ (right) produced by the proof of Theorem 4.1 for the formula $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$. The red lines help visualize the vertical order of points inside each row. The B_1-row does not contain a pattern 123 in the columns Y_1^F, Y_2^F, Y_3^F and the same holds for the B_2-row and columns Y_1^F, Y_2^T, Y_3^T.

Finally, we rotate P slightly to guarantee that it is in general position and take the permutation π as its reduction. See the left part of Figure 4.1. Observe that the length of π is bounded by $O(n+m)$.

Constructing the text τ. As before, we start with defining the text as a point set T not necessarily in general position. We again first define two anchors as

$$a_1^T = (1, 4n+2), \qquad a_2^T = (2m+2, 1).$$

For each $i \in [n]$, we define two increasing pairs of points Y_i^T and Y_i^F associated to the variable x_i as

$$Y_i^T = \{(2m+6i-1, 4i-2),\ (2m+6i+1, 4i-1)\},$$
$$Y_i^F = \{(2m+6i+2, 4i),\ (2m+6i+4, 4i+1)\}.$$

For each $t \in [m]$, we associate to the clause K_t of φ a pair of points B_t defined as

$$B_t = \{(2t, 2n+9t-7),\ (2t+1, 2n+9t+1)\}.$$

This completes the definition of the 'boundaries' of the matrix. Observe that all pairs Y_i^T and Y_i^F lie horizontally between a_1^T and a_2^T while every pair B_t lies vertically between a_1^P and a_2^P. The pairs B_t form again an increasing permutation of length $2m$. However, the pairs Y_i^T and Y_i^F for a fixed $i \in [n]$ form the pattern 3412 and together they are arranged into an increasing sequence of n such blocks. We define the notions of Y_i^α-columns, B_t-rows and (Y_i^α, B_t)-cells where $\alpha \in \{T, F\}$ exactly as in the case of the pattern.

Suppose K_t is of the form $L_i \vee L_j \vee L_k$, with $L_i \in \{x_i, \neg x_i\}$, $L_j \in \{x_j, \neg x_j\}$ and $L_k \in \{x_k, \neg x_k\}$, for some $i < j < k$. For every $\alpha \in \{i, j, k\}$, we define two values t_α and f_α such that $t_\alpha = 2m + 6\alpha$ and $f_\alpha = 2m + 6\alpha + 3$ if $L_\alpha = x_\alpha$,

otherwise we swap the values, i.e., we set $t_\alpha = 2m + 6\alpha + 3$ and $f_\alpha = 2m + 6\alpha$. Observe that if $L_\alpha = x_\alpha$ then any point p such that $p.x = t_\alpha$ lies in the Y_α^T-column and any point p such that $p.x = f_\alpha$ lies in the Y_α^F-column. The opposite holds when $L_\alpha = \neg x_\alpha$. We add the following 7 points to the B_t-row

$$(t_i, 2m + 9t - 6), (f_j, 2m + 9t - 5), (f_i, 2m + 9t - 4), (t_j, 2m + 9t - 3),$$
$$(f_k, 2m + 9t - 2), (f_j, 2m + 9t - 1), (t_k, 2m + 9t).$$

Observe that the y-coordinates of these points impose on them a linear order, while the x-coordinates are arranged to form the word appearing in Observation 4.2.

We obtain the permutation τ by slightly rotating T to guarantee general position and subsequently, taking its reduction. See the right part of Figure 4.1. The length of τ is easily seen to be bounded by $O(n + m)$.

Claim 4.3. *The formula φ is satisfiable if and only if there is a left-aligned embedding of π into τ.*

Correctness ("only if"). Suppose that φ is satisfiable. Fix any satisfying assignment represented by a function $\rho : [n] \to \{T, F\}$, where $\rho(i) = T$ if and only if x_i is set to true. We describe how to find the desired left-aligned embedding of π into τ.

We start by mapping the anchors of π to the anchors of τ. Notice that by this we guarantee the 'left-alignment' of the embedding. Furthermore, we map the pair A_t to the pair B_t for every $t \in [m]$ and we map the pair X_i to the pair $Y_i^{\rho(i)}$ for every $i \in [n]$. It remains to map for each $t \in [m]$ the three points in the A_t-row. This is where Observation 4.2 comes to play.

Suppose K_t is of the form $L_i \vee L_j \vee L_k$, with $L_i \in \{x_i, \neg x_i\}$, $L_j \in \{x_j, \neg x_j\}$ and $L_k \in \{x_k, \neg x_k\}$, for some $i < j < k$. For $\alpha \in \{i, j, k\}$, let $T_\alpha \in \{T, F\}$ be the assignment of the variable x_i such that the literal L_α evaluates to true and let F_α be the opposite assignment. If we order the points in the B_t-row in τ according to their y-coordinates, they appear in the intersections with the following columns

$$Y_i^{T_i}, Y_j^{F_j}, Y_i^{F_i}, Y_j^{F_j}, Y_k^{F_k}, Y_j^{F_j}, Y_k^{T_k}.$$

We consider this sequence as a word w_t over the alphabet $\bigcup_{\alpha \in \{i,j,k\}} \{Y_\alpha^T, Y_\alpha^F\}$. Since ρ is a satisfying assignment of φ, at least one of the the literals L_i, L_j, L_k must evaluate to true. But then we can find the word $Y_i^{\rho(i)} Y_j^{\rho(j)} Y_k^{\rho(k)}$ as a subsequence in w_t by Observation 4.2 and we map the three points in π to the corresponding three points in τ.

It is trivial to see that the anchors and the pairs $A_1, \ldots, A_m$ and $X_1, \ldots, X_n$ retain the correct relative positions under the defined mapping. For the remaining points, the same is guaranteed by Observation 4.2.

Correctness ("if"). Suppose there is a left-aligned embedding ψ of π into τ. By definition, the anchor a_1^P in π is mapped to the anchor a_1^T in τ. We claim that moreover, the anchor a_2^P must be mapped to the anchor a_2^T. This holds since the points of π below a_1^P form an increasing sequence of length exactly $2n + 1$ starting with the point a_2^P while any longest increasing sequence in τ below a_1^T is also of length exactly $2n + 1$ and starts with the point a_2^T.

Since the pairs $A_1, \ldots, A_m$ are the only points enclosed in the vertical strip between anchors in π and the same holds for the pairs $B_1, \ldots, B_m$ in τ, the pair A_t must be mapped to the pair B_t for each $t \in [m]$. Similarly, the only points enclosed in the horizontal strip between anchors in π are pairs $X_1, \ldots, X_n$ and the only points occupying the same region in τ are the pairs $Y_1^T, \ldots, Y_n^T$ and $Y_1^F, \ldots, Y_n^F$. Recall that the pairs $X_1, \ldots, X_n$ form together an increasing sequence of length $2n$ while the pairs $Y_1^T, \ldots, Y_n^T$ and $Y_1^F, \ldots, Y_n^F$ together form the permutation $\oplus^n 3412$. It follows that for each $i \in [n]$, the pair X_i must be mapped either to Y_i^T or Y_i^F. We use this fact to define an assignment $\rho : [n] \to \{T, F\}$ such that X_i maps precisely to $Y_i^{\rho(i)}$.

In order to show that ρ represents a satisfying assignment of φ, it suffices to reuse the observations present in the proof of the other implication. Fix $t \in [m]$. Recall that the word w_t is obtained by listing the columns occupied by the points in the B_t-row in τ in the order of their y-coordinates. Since the three points in the A_t-row in π are mapped by ψ to the B_t-row in τ, the word $Y_i^{\rho(i)} Y_j^{\rho(j)} Y_k^{\rho(k)}$ can be found as a subsequence in w_t. Therefore, we cannot have $\rho(\alpha) = F_\alpha$ for every $\alpha \in \{i, j, k\}$ due to Observation 4.2 and ρ satisfies the clause K_t.

Inflating the leftmost points. Finally, we need to remove the assumption that we only care about left-aligned embeddings of π into τ. This is done via a trick that we shall see over and over again. Namely, let π' be the permutation obtained from π by inflating its leftmost point a_1^P with the decreasing sequence of length $|\tau|$ and let τ' be the permutation obtained from τ by inflating its leftmost point a_1^T with the decreasing sequence of length $|\tau|$. Note that we still have $|\tau'| \in O(n + m)$.

Claim 4.4. *There is a left-aligned embedding of π into τ if and only if there is an embedding of π' into τ'.*

It is clear that any left-aligned mapping of π into τ can easily be remade into an embedding of π' into τ'. For the other direction, assume there is a mapping ψ of π' into τ'. Observe that the inflated decreasing sequence in π' contains $|\tau|$ points while τ' contains only $|\tau| - 1$ points outside of its inflated decresing sequence. Hence, there must be at least one point of the inflated sequence in π' that maps to the inflated sequence in τ' by a simple counting argument.

Moreover, we claim that all points of π' outside of its inflated part cannot be mapped to the inflated part of τ'. The only points that could even map to the inflated part due to their relative positions with respect to a_1^P are the pairs $X_1, \ldots, X_n$ and the point a_2^P. If any point of X_i maps to the inflated anchor a_1^T then we have no place where we can map a_2^P. Conversely, if the point a_2^P maps to the inflated a_1^T then we cannot map the pairs $X_1, \ldots, X_n$. Therefore, if we restrict ψ to the points outside the inflated sequences, we get the description of a left-aligned embedding of π into τ minus the leftmost points. □

4.2 Parameterized complexity

In this section, we investigate PPM through the paradigm of parameterized complexity. We have already met some examples of parameterized algorithms (see, e.g., Theorems 2.3, 3.4, 3.17). Now, we formally (albeit briefly) introduce

the paradigm of parameterized algorithms and complexity. For more details, we refer an interested reader, e.g., to the monograph by Cygan et al. [58].

Formally, a *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is the input alphabet. We say that a parameterized problem L is *fixed-parameter tractable (FPT)* if there is an algorithm $\mathcal{A}$ and a computable function $f\colon \mathbb{N} \to \mathbb{N}$ such that, given $(x, k) \in \Sigma^* \times \mathbb{N}$, the algorithm $\mathcal{A}$ correctly decides whether $(x, k) \in L$ in time $f(k) \cdot |(x, k)|^{O(1)}$. The algorithm $\mathcal{A}$ is called an *fpt-algorithm*.

We can also define a corresponding notion of reduction. An *fpt-reduction* from a parameterized problem A to a parameterized problem B is an algorithm that, given an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, outputs an instance $(x', k') \in \Sigma^* \times \mathbb{N}$ such that

(i) $(x, k) \in A$ if and only if $(x', k') \in B$,
(ii) there exists a computable function g such that $k' \leq g(k)$, and
(iii) the algorithm runs in time $f(k) \cdot |x|^{O(1)}$ for some computable function f.

We define the class FPT to consist of all the fixed-parameter tractable problems. It is not hard to see that whenever we have a problem B in FPT and an fpt-reduction from a parameterized problem A to B, then A is also in FPT. As before, we can define the notion of hard and complete problems for a given class of parameterized problems. For a class C of parameterized problems, we say that a parameterized problem A is C-hard if every problem in C can be reduced to A by an fpt-reduction. Moreover, A is C-complete if A is C-hard and simultaneously A is contained in C.

The other important classes in the study of parameterized complexity are the classes $\mathsf{W}[1] \subseteq \mathsf{W}[2] \cdots$ forming the so-called W-hierarchy. We refrain from introducing the whole hierarchy as only the class W[1] is relevant for us because it is conjectured that $\mathsf{W}[1] \neq \mathsf{FPT}$. Therefore, showing that a parameterized problem is W[1]-hard can be seen as evidence that it is not fixed-parameter tractable. We omit the original definition of W[1] and instead we define it as the class of all parameterized problems that are fpt-reducible to the natural and well-known problem Clique.

Clique	
Input:	Graph $G = (V, E)$ and $k \in \mathbb{N}$.
Parameter:	k
Output:	Is there a clique of size k in G?

We have already alluded to the fact that PPM is in FPT when parameterized by the length of the pattern. It follows for instance from Corollary 3.5 since we can define the property of containing a pattern π as a FO sentence with length $O(|\pi|)$. However, the original algorithm of Guillemot and Marx [77] was tailored specifically for the PPM problem and thus, it achieves better running time dependence on the parameter k. Moreover, its running time was later slightly improved via a purely combinatorial result by Fox [68].

Theorem 4.5 ([77, 68])**.** *PPM can be solved in time $2^{O(k^2)} \cdot n$ where k is the length of the pattern and n is the length of the text.*

Notice that the previous theorem speaks only about the decision version of the problem. That is no coincidence since Berendsohn et al. [23] showed that an

fpt-algorithm for the counting version #PPM cannot exist unless ETH fails. We later show how this theorem follows from our own reductions.

Theorem 4.6 ([23]). *#PPM cannot be solved in time $f(k) \cdot n^{o(k/\log k)}$ where k is the length of the pattern and n is the length of the text, unless ETH fails.*

4.2.1 Left-aligned patterns

As a basis of all further reductions, let us show that similar conditional lower bound under ETH holds also for Left PPM. Notice that while the difference between PPM and Left PPM seems minuscule, they exhibit very different behavior with respect to parameterized complexity.

Proposition 4.7. *Left PPM is W[1]-complete with respect to k and unless ETH fails, it cannot be solved in time $f(k) \cdot n^{o(k/\log k)}$ for any function f, where k is the length of the pattern and n is the length of the text.*

Before the actual reduction, let us define the problem that we will eventually reduce to Left PPM. In particular, we will reduce from the well-known problem Partitioned Subgraph Isomorphism (PSI). Informally, we aim to find a given graph G as a subgraph of another graph H, but it is prescribed in advance where each vertex of G can be mapped.

Partitioned Subgraph Isomorphism (PSI)	
Input:	Graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$ together with a coloring $\chi\colon V_H \to V_G$ of vertices of H, using the vertices of G as colors.
Output:	Is there a mapping $\phi\colon V_G \to V_H$ such that whenever $\{u, v\} \in E_G$ then also $\{\phi(u), \phi(v)\} \in E_H$ and moreover, $\chi(\phi(v)) = v$ for every $v \in V_G$?

The problem PSI is W[1]-complete with respect to $k = |E_G|$ and moreover, Marx [99] showed that it cannot be solved in time $f(k) \cdot n^{o(k/\log k)}$ unless ETH fails. Bringmann et al. [45] later observed that this holds even if we require G to have the same number of vertices and edges.

Theorem 4.8 ([99, 45]). *PSI is W[1]-complete and unless ETH fails, PSI cannot be solved in time $f(k) \cdot n^{o(k/\log k)}$ for any function f, where $n = |V_H|$ and $k = |E_G|$. This is true even when we require G to have exactly as many vertices as edges.*

Proof of Proposition 4.7. The basic idea of the reduction is very similar both to a reduction from PSI to a different variant of PPM by Berendsohn et al. [23], and to the representation of graphs by permutations used in the proof of Theorem 3.6.

Let (G, H, χ) be an instance of PSI and set $n = |V_H|, k = |V_G|$. We assume that the vertex set V_G is in fact equal to $[k]$ and we define for each $i \in [k]$ the set $V_i \subseteq V_H$ as the set of vertices of H colored by i, i.e., $V_i = \chi^{-1}(i)$. Notice that $V_1, \ldots, V_k$ form a partition of the set V_H.

We shall construct two point sets P and T not necessarily in general position such that P will represent the adjacency matrix of G while T will represent the adjacency matrix of H.

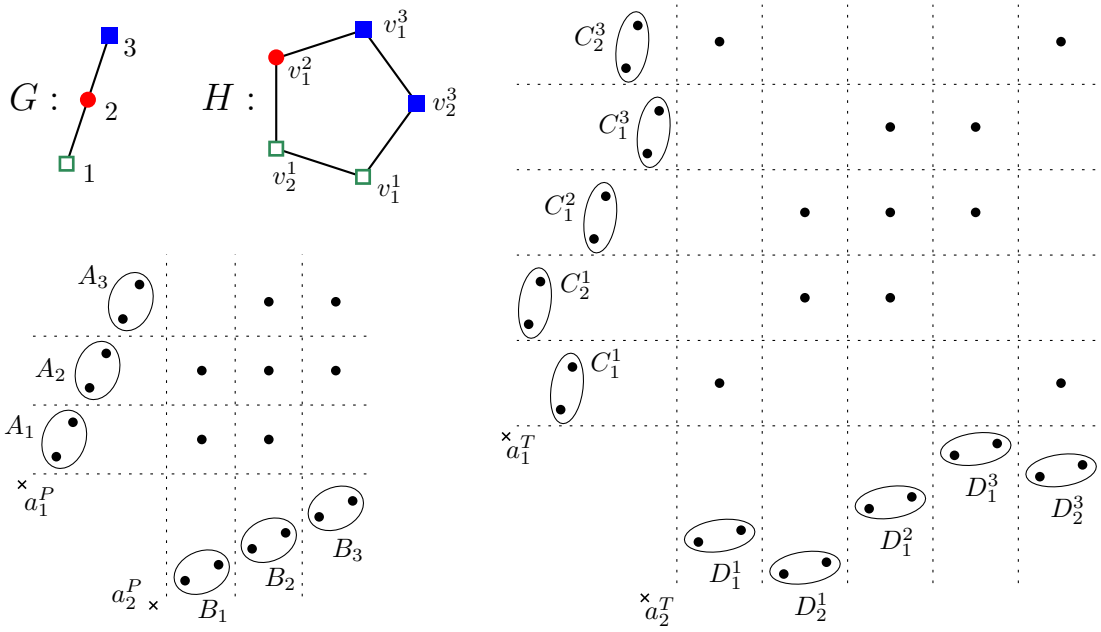

Figure 4.2: Permutations π (bottom left) and τ (right) produced by the proof of Proposition 4.7 for an instance (G, H, χ) of PSI (top left).

Constructing the pattern π. We start describing first the set P. It will be almost identical to the representation of adjacency matrix in the proof of Theorem 3.6. It contains two anchors a_1^P and a_2^P defined as

$$a_1^P = (1, 2k+2), \qquad a_2^P = (2k+2, 1).$$

For each $i \in [k]$, we associate two pairs of points to the vertex i defined as

$$A_i = \{(2i, 3i+2k), (2i+1, 3i+2k+2)\},$$
$$B_i = \{(3i+2k, 2i+1), (3i+2k+2, 2i)\}.$$

As before, every pair A_i lies horizontally between the anchors and every pair B_i lies vertically between the anchors. However, this time the pairs $A_1, \ldots, A_k$ form together an increasing permutation of length $2k$ and the same holds for the pairs $B_1, \ldots, B_k$. The pairs naturally impose a grid-like structure. We define the *A_i-row* as the horizontal strip bounded by the pair A_i, the *B_j-column* as the vertical strip bounded by B_j and the *(A_i, B_j)-cell* as their intersection.

For each edge $\{i, j\} \in E_G$, we simply add points to the (B_j, A_i)-cell and the (B_i, A_j)-cell, i.e., we add to P the points

$$(3i+2k+1, 3j+2k+1), \quad (3j+2k+1, 3i+2k+1).$$

Additionally, we also add a point to each cell on the diagonal, i.e., we add the point $(3i+2k+1, 3i+2k+1)$ for every i.

That wraps up the definition of P. We rotate P slightly to guarantee that it is in general position and take the permutation π as its reduction. See the bottom left part of Figure 4.2. The length of π is $O(|V_G| + |E_G|)$ which is bounded by $O(k)$ since we assumed that $|V_G| = |E_G|$.

Constructing the text τ. Now, we shift our attention to the point set T. It contains two anchors a_1^T and a_2^T defined as

$$a_1^T = (1, 2n+2), \qquad a_2^T = (2n+2, 1).$$

For each $i \in [k]$, set $n_i = |V_i|$ and we choose an arbitrary order of vertices in V_i denoting them v_j^i for $j \in [n_i]$. To every vertex v_j^i, we associate two values – the *rank of* v_j^i denoted by α_j^i and the *reverse rank of* v_j^i denoted by β_j^i where

$$\alpha_j^i = \sum_{i'<i} n_{i'} + j - 1 \quad \text{and} \quad \beta_j^i = \sum_{i'<i} n_{i'} + n_i - j. \tag{4.1}$$

Observe that the rank corresponds to the lexicographic order of v_j^i by (i, j) and the reverse rank corresponds to the lexicographic order by $(i, n_i - j)$.

For every $i \in [k]$ and $j \in [n_i]$, we add to T two pairs of points associated to the vertex v_j^i

$$C_j^i = \{(2\beta_j^i, 3\alpha_j^i + 2k), (2\beta_j^i + 1, 3\alpha_j^i + 2k + 2)\},$$
$$D_j^i = \{(3\alpha_j^i + 2k, 2\beta_j^i + 1), (3\alpha_j^i + 2k + 2, 2\beta_j^i)\}.$$

Every pair C_j^i again lies horizontally between the anchors while every pair D_j^i lies vertically between the anchors. For a fixed i, the pairs $C_1^i, \ldots, C_{n_i}^i$ form a co-layered permutation with each layer consisting of a single pair, and the same holds for the pairs $D_1^i, \ldots, D_{n_i}^i$. Moreover for different $i < j$, the pairs $C_1^i, \ldots, C_{n_i}^i$ lie all to the left and below the pairs $C_1^j, \ldots, C_{n_j}^j$. The same holds for pairs D_j^i.

Finally, we add to T a point to the $(D_j^i, C_{j'}^{i'})$-cell for $i, i' \in [k]$ and $j \in [n_i], j' \in [n_{i'}]$ whenever either $i = i'$ and $j = j'$, or $i \neq i'$ and $\{v_j^i, v_{j'}^{i'}\} \in E_H$. Formally, such a point is defined as $(3\alpha_j^i + 2k + 1, 3\alpha_{j'}^{i'} + 2k + 1)$. In other words, every non-empty cell either lies on the diagonal or corresponds to an edge between two vertices that do not share the same color.

As usual, we rotate T to guarantee general position and take τ as its reduction. See the right part of Figure 4.2. The length of τ is $O(|V_H| + |E_H|)$ which is clearly bounded by $O(n^2)$.

Correctness ("only if"). Suppose that (G, H, χ) is a positive instance of PSI. There is a witnessing mapping $\phi\colon [k] \to \mathbb{N}$ such that $\phi(i) \in [n_i]$ for every $i \in [k]$ and $\{v_{\phi(i)}^i, v_{\phi(j)}^j\} \in E_H$ for every different $i, j \in [k]$.

We define a left-aligned embedding ψ of π into τ in a straightforward manner. First, we take care of the left-aligned property by mapping the anchors in π to the anchors in τ. We map the pair A_i to $C_{\phi(i)}^i$ and B_i to $D_{\phi(i)}^i$ for each $i \in [k]$. It is sufficient to argue that every nonempty (B_j, A_i)-cell in π maps to a non-empty cell in τ. This follows immediately for the cells in π on the diagonal. Otherwise if $i \neq j$, we have $\{i, j\} \in E_G$ so there must be an edge $\{v_{\phi(i)}^i, v_{\phi(j)}^j\}$ in H and thus, the $(D_{\phi(j)}^j, C_{\phi(i)}^i)$-cell is non-empty.

Correctness ("if"). Suppose there exists a left-aligned embedding of π into τ. As in the proof of Theorem 4.1, we claim that any embedding of π into τ that maps the anchor a_1^P to the anchor a_1^T must also map a_2^P to the anchor a_2^T.. This holds since the points of π below a_1^P form an increasing sequence of length exactly

$2k+1$ starting with the point a_2^P while any longest increasing sequence in τ below a_1^T is also of length exactly $2k+1$ and starts with the point a_2^T.

The pairs $A_1, \ldots, A_k$ form an increasing sequence of length $2k$ sandwiched horizontally between the anchors of π. Therefore, they must all be mapped to the union of all the pairs C_j^i since these are the only points in the horizontal strip between the anchors in τ. However, the only increasing subsequences of length $2k$ in this strip are of the form $C_{i_1}^1, C_{i_2}^2, \ldots, C_{i_k}^k$ and in particular, the pair A_i is mapped to the pair C_j^i for some j. Let $\phi\colon [k] \to \mathbb{N}$ be the mapping such that A_i is mapped precisely to $C_{\phi(i)}^i$ for every $i \in [k]$.

The same argument can be applied to the pairs $B_1, \ldots, B_k$ and we define a mapping $\phi'\colon [k] \to \mathbb{N}$ such that B_i is mapped precisely to $D_{\phi'(i)}^i$ for every $i \in [k]$.

Recall that there is a point in π in the (B_i, A_i)-cell for every $i \in [k]$. The only non-empty cells in τ between vertices of the same color in H are on the diagonal and thus, we have that $\phi(i) = \phi'(i)$ for every $i \in [k]$. It remains to verify that $\{v_{\phi(i)}^i, v_{\phi(j)}^j\} \in E_H$ whenever $\{i, j\}$ is an edge in G. This is enforced since the non-empty cells in τ outside of the diagonal correspond to edges of H.

W[1]-completeness. The reduction shows that Left PPM is W[1]-hard with respect to k. It remains to show that Left PPM belongs to W[1]. We show how to encode an instance of Left PPM as a model checking problem of an existential first-order formula over a suitably chosen signature. The W[1]-membership of Left PPM follows since Flum and Grohe [67] showed that the following problem is W[1]-complete.

Existential first-order model checking	
Input:	A structure $\mathcal{A}$ and an existential FO formula φ.
Parameter:	$\lvert\varphi\rvert$
Output:	Does $\mathcal{A} \models \varphi$?

Let (π, τ) be an Left PPM instance. We compute a structure $\mathcal{A} = (S_\tau, \prec_x, \prec_y, L)$ where $(S_\tau, \prec_x, \prec_y)$ is the usual representation of τ as a model of TOTO (see Chapter 3) and L is interpreted as a unary relation containing only the leftmost point in τ. The existential formula φ then tests whether $(S_\tau, \prec_x, \prec_y)$ contains π and additionally, it enforces that the leftmost point in τ is used by the embedding. We omit further details. □

4.2.2 Pattern parameters

Using the hardness of Left PPM, we can rule out fpt-algorithms for PPM parameterized by almost any imaginable parameter of the pattern other than its length. In order to make this statement formal, we say that a permutation parameter w is *invariant under monotone inflations* if (i) $\mathrm{w}(\pi)$ is at most $|\pi|$ for every permutation π, and (ii) whenever π' is obtained by inflating a single element of π with a monotone permutation, we have $\mathrm{w}(\pi') \in O(\mathrm{w}(\pi))$. Observe that every parameter defined in Chapter 2 is, in fact, invariant under monotone inflations.

Corollary 4.9. *Suppose that* w *is a permutation parameter invariant under monotone inflations. PPM is W[1]-hard with respect to* $\mathrm{w}(\pi)$ *and unless ETH fails, it cannot be solved in time* $f(\mathrm{w}(\pi)) \cdot n^{o(\mathrm{w}(\pi)/\log \mathrm{w}(\pi))}$ *for any function* f*, where* π *is the pattern and* n *is the length of the text.*

Proof. The proof goes by reduction from Left PPM. Let (π, τ) be an instance of Left PPM produced by the proof of Proposition 4.7.

We define permutations π' and τ' by inflating the leftmost points with decreasing permutations of length $|\tau|$. Arguing as in the proof of Theorem 4.1, we see that there is a left-aligned embedding of π into τ if and only if there is an (ordinary) embedding of π' into τ'. Moreover, we have $\mathrm{w}(\pi') \in O(\mathrm{w}(\pi)) \subseteq O(k)$ since w is invariant under monotone inflations and therefore, an algorithm solving PPM in time $f(\mathrm{w}(\pi')) \cdot n^{o(\mathrm{w}(\pi')/\log \mathrm{w}(\pi'))}$ would refute ETH through Proposition 4.7. □

4.2.3 Counting patterns

Next, we show that Proposition 4.7 provides us with an alternative easy proof of Theorem 4.6.

Alternative proof of Theorem 4.6. We show that an algorithm solving #PPM in time $f(k) \cdot n^{o(k/\log k)}$ could be used to design an algorithm solving the counting version of Left PPM in time $g(k) \cdot n^{o(k/\log k)}$. And since counting the number of left-aligned embeddings is at least as hard as deciding if there is one, we conclude that algorithm with such running time would refute ETH via Proposition 4.7.

Suppose (π, τ) is an instance of Left PPM. Let $occ(\pi, \tau)$ denote the number of occurrences of π in τ and let τ' be the permutation obtained from τ by deleting its leftmost point. Every occurrence of π is either left-aligned or not and therefore, the number of left-aligned occurrences of π can be computed as

$$occ(\pi, \tau) - occ(\pi, \tau').$$

In other words, we can compute the number of left-aligned occurrences by invoking twice the algorithm for #PPM. Therefore, we would obtain an algorithm solving Left PPM in time $2 \cdot f(k) \cdot n^{o(k/\log k)}$. □

4.2.4 Generalized patterns

It is natural to ask how the complexity changes if we start searching for generalized patterns. With that in mind, we define the problems Vincular PPM, Covincular PPM, Bivincular PPM, Mesh PPM and POP PPM similarly to PPM, only replacing the classical pattern π with vincular, covincular, bivincular, mesh and partially ordered patterns, respectively. We keep the convention of using k to denote the length of the pattern and n to denote the length of the text. Additionally, we also define the counting version of all these problems, e.g., Bivincular #PPM is the problem of counting the number of occurrences of a given bivincular pattern.

First, observe that avoidance of partially ordered patterns is a hereditary property, i.e., whenever τ avoids a partially ordered pattern p then any subpermutation of τ also avoids p. In other words, the permutations avoiding a fixed partially ordered pattern form a permutation class. Therefore, we can obtain an fpt-algorithm for POP PPM using the algorithm for FO model checking.

Proposition 4.10. *POP PPM can be solved in time $f(k) \cdot n$ for some computable function f, where k is the length of the pattern and n is the length of the text.*

Proof. Let (p, τ) be an instance of POP PPM. We can construct a FO sentence φ of length $O(k)$ in TOTO that expresses whether a permutation contains the partially ordered pattern p. Since the permutations avoiding p form a permutation class, we can test whether τ avoids p in the desired time by Corollary 3.5. □

On the other hand, we can straightforwardly use Proposition 4.7 to get a conditional lower bound for Vincular PPM. Moreover, the lower bound for Vincular PPM automatically applies for Covincular PPM by symmetry, and also for both Bivincular PPM and Mesh PPM since any vincular pattern can be expressed as a bivincular or mesh pattern. We remark that these problems were already shown to be W[1]-complete with respect to the pattern length by Bruner and Lackner [49].

Corollary 4.11. *Vincular PPM is W[1]-complete with respect to k and unless ETH fails, it cannot be solved in time $f(k) \cdot n^{o(k/\log k)}$ for any function f, where k is the length of the pattern and n is the length of the text.*

Proof. Let (π, τ) be an instance of Left PPM. By taking $((\pi, \{0\}), \tau)$ as the instance of Vincular PPM, we are looking precisely for the left-aligned occurrences of π in τ. □

4.2.5 Patterns as CSPs

Berendsohn et al. [23] noticed that PPM can be phrased as the so-called constraint satisfaction problem (CSP). As a consequence, they showed that permutation patterns can be counted in polynomial time whenever their tree-width is bounded using well-known algorithms for CSP.

Theorem 4.12 ([23])**.** *#PPM can be solved in time $O(n^{\mathrm{tw}(\pi)+1})$ where π is the pattern and n is the length of the text.*

We will extend this approach to bivincular and partially ordered patterns. But first, we need to formally introduce CSPs. A *binary CSP instance* is a triplet (V, D, C) consisting of the set of variables V, the set of admissible values (also called *domain*) D and a set of constraints C where each constraint is a tuple $((x, y), R)$ such that $x, y \in V$ is a pair of variables and $R \subseteq D^2$ is an arbitrary binary relation over the domain D.

A *solution* of the binary CSP instance (V, D, C) is a function $f\colon V \to D$ such that for each constraint $((x, y), R) \in C$, we have $(f(x), f(y)) \in R$. Informally, a solution is an assignment of values from the domain to the variables that satisfies every constraint.

Finally, the *constraint graph* of the binary CSP instance (V, D, C) is the graph whose vertex set is the set of variables V with edges between any two variables x, y that share a common constraint. It is well-known that the number of solutions to a binary CSP instance can be computed efficiently whenever its constraint graph has small tree-width. We remark that even though the following algorithm of Freuder [70] was originally stated only in the decision form, it can be straightforwardly modified to count the number of solutions and it is regularly referenced as such in the literature.

Theorem 4.13 ([70])**.** *A number of solutions of a binary CSP instance (V, D, C) can be computed in time $O(|D|^{t+1})$ where t is the tree-width of the constraint graph.*

Before we can apply CSPs to solve Bivincular #PPM and POP #PPM, we first generalize the notion of incidence graphs to their setting. We define the incidence graph of a bivincular pattern (π, C, R) simply as the incidence graph G_π of the underlying classical pattern π.

The situation is a bit more complicated in the case of partially ordered patterns. We say that an element j *covers* an element i in a partial order $<_P$ if $i <_P j$ and there is no element i' such that $i <_P i' <_P j$. Note that an element j covers an element i in $<_P$ exactly when we would draw an edge between i and j in the Hasse diagram of $<_P$. Given a partially ordered pattern $p = (<_P, k)$, we define its *incidence graph* G_p as the graph on the vertex set $[k]$ with edges (i) between i and $i + 1$ for every $i \in [k]$, and (ii) between every i and j such that j covers i in $<_P$. In other words, the incidence graph is the union of the Hasse diagram of $<_P$ with the Hamiltonian path connecting $[k]$ in the increasing order.

Theorem 4.14. *#PPM, Bivincular #PPM and POP #PPM can all be solved in time $O(n^{\mathrm{tw}(G)+1})$ where n is the length of the text and G is the incidence graph of the pattern.*

Proof. We start by solving #PPM as shown by Berendsohn et al. [23]. The basic idea is to enforce that each solution of a certain CSP instance corresponds to a unique embedding. We could, in general, enforce that every pair of points in π maps to an isomorphic pair of points in τ. However, it is sufficient to guarantee this for the neighboring pairs.

Given permutations π and τ, we define a binary CSP instance (V, D, C) where

- the variable set V is defined as $\{x_p \mid p \in S_\pi\}$,
- the domain is the permutation diagram S_τ, and
- for each $\alpha \in \{R, L, U, D\}$ and points $p, N^\alpha(p) \in S_\pi$, we add a constraint $((x_p, x_{N^\alpha(p)}), R)$ where R consists of all the pairs $(a, b) \in S_\pi^2$ that are in the same relative position as p and $N^\alpha(p)$.

Solutions of this CSP instance are in one-to-one correspondence with embeddings of π into τ by Observation 1.1. Therefore, we can employ Theorem 4.13 to count the occurrences of π in τ. The desired running time follows since the constraint graph of the obtained CSP instance is exactly the incidence graph of π.

The generalization to bivincular patterns is straightforward. Given a permutation τ and a bivincular pattern (π, C, R), we construct a binary CSP instance (V, D, C) where V and D are defined exactly as before and the only difference in the constraints is that whenever $p, N^\alpha(p) \in S_\pi$ should be mapped to a neighboring pair in τ, we simply restrict the relation R to contain only neighboring pairs in the correct relative position. The constraint graph remains the same and thus, we can again use Theorem 4.13 to count the occurrences of (π, C, R) in τ.

Finally, let us handle partially ordered patterns. Given a permutation τ and a partially ordered pattern $(<_P, k)$, we define a binary CSP instance (V, D, C) where

- the variable set V is the set $[k]$,
- the domain is the permutation diagram S_τ, and

- the set C contains a constraint
  - $((i, i+1), R)$ for each $i \in [k-1]$ where R contains all the pairs $(p, q) \in S_\tau^2$ such that $p.x < q.x$, and
  - $((i, j), R)$ for each $i, j \in [k]$ such that j covers i in $<_P$ where R contains all the pairs $(p, q) \in S_\tau^2$ such that $p.y < q.y$.

Similar to Observation 1.1, any solution of this CSP instance encodes a particular embedding of $(<_P, k)$ into τ. Moreover, the constraint graph is again the incidence graph of the pattern and we invoke Theorem 4.13 to count the occurrences of $(<_P, k)$ in τ. □

Note that we could easily generalize this further for doubly partially ordered patterns. The incidence graph of these patterns is defined in the natural way as the union of Hasse diagrams of the two partial orders. Going even further, we could require certain pairs in the doubly partially ordered patterns to be formed by consecutive points.

4.3 Pattern from a given permutation class

In this section, we focus our attention to restricting the pattern to a specific permutation class. We will be able to recover results similar to the general case assuming the classes have the long path or the deep tree properties. To that end, we formally introduce the permutation pattern matching with patterns restricted to a fixed permutation class.

$\mathcal{C}$-Pattern Permutation Pattern Matching ($\mathcal{C}$-Pattern PPM)	
Input:	Permutations $\pi \in \mathcal{C}$ of length k and τ of length n.
Output:	Is π contained in τ?

Similarly, we can define the $\mathcal{C}$-Pattern variant of virtually any other PPM-flavored problem. So for instance, we consider the $\mathcal{C}$-Pattern #PPM, $\mathcal{C}$-Pattern Bivincular PPM , etc.

4.3.1 Classical complexity

Jelínek and Kynčl [86] showed that Av(321)-Pattern PPM and $\mathcal{S}$-Pattern PPM, where $\mathcal{S}$ is the clockwise spiral, are both NP-complete. Remarkably, their reduction relies only on the long path property and we show how to modify it for any class $\mathcal{C}$ with the poly-time computable long path property. Note that in [88], we used the same ideas to show that Grid($\mathcal{M}$)-Pattern PPM is NP-complete for any cyclic monotone gridding matrix $\mathcal{M}$.

Theorem 4.15. *Let $\mathcal{C}$ be a permutation class with the poly-time computable long path property. Then $\mathcal{C}$-Pattern PPM is NP-complete and unless ETH fails, it cannot be solved in time $2^{o(\sqrt{|\tau|})}$ where τ is the text.*

Proof. We use a similar top level idea as in the proof of Theorem 3.18. We want to take the reduction of Theorem 4.1 with its matrix-like structure and then map individual rows of the matrix to cells along a path in a gridded permutation.

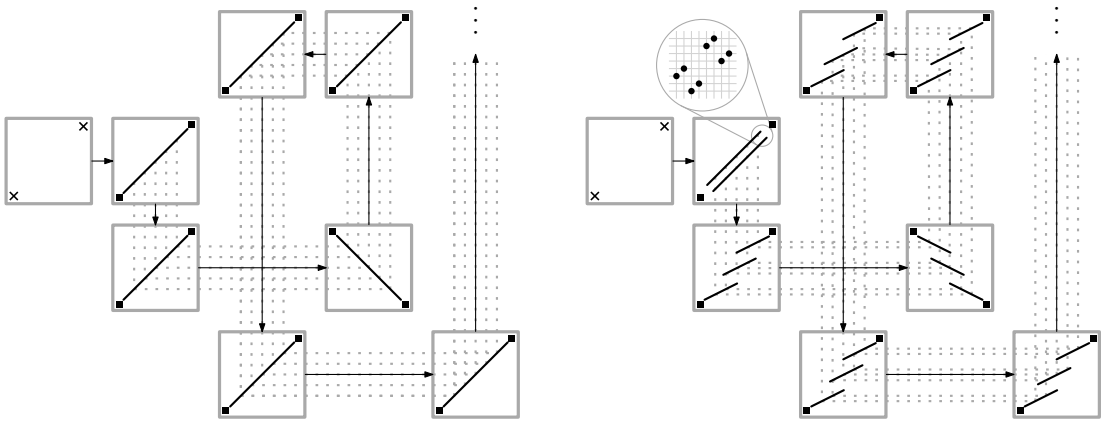

Figure 4.3: An overview of the general structure of permutations π (left) and τ (right) produced by the proof of Theorem 4.15. Anchors are represented as usual by crosses, guards by squares.

We reduce from 3-SAT. Let φ be a given 3-CNF formula with n variables $x_1, \ldots, x_n$ and m clauses $K_1, \ldots, K_m$. We will assume, as before, that each clause contains exactly three literals, and no variable appears in a single clause more than once. Our goal is to construct permutations $\pi \in \mathcal{C}$ and τ such that τ contains π if and only if φ is satisfiable.

Before describing the actual construction of π and τ, we obtain in a polynomial time a monotone gridding matrix $\mathcal{M}$ such that

(i) Grid($\mathcal{M}$) is a subclass of $\mathcal{C}$,
(ii) the cell graph of $\mathcal{M}$ is a proper-turning path on $2m + 2$ vertices, and
(iii) the first two cells on this path share a common row.

We denote the vertices in the order along the path as $v_1, \ldots, v_{2m+2}$. Let $\mathcal{F}$ be a consistent orientation of $\mathcal{M}$ guaranteed by Lemma 1.2. We shall construct both π and τ via the modified $\mathcal{F}$-assembly that was introduced before Theorem 3.18. Refer to Figure 4.3.

Constructing the pattern π. The pattern π will be constructed from a family of $(2n + 4)$-tiles $\mathcal{P}$. The tile P_{v_1} contains only a single atomic pair of points, forming the anchor, defined as

$$(1, 1),\ (2n + 4, 2n + 4).$$

Every other tile P_{v_j} contains two special atomic pairs, called *guards*, defined as

$$G_1^P = \{(1, 1),\ (2, 2)\},$$
$$G_2^P = \{(2n + 3, 2n + 3),\ (2n + 4, 2n + 4)\}.$$

Furthermore, it contains n additional atomic pairs, one per each variable. For each $i \in [n]$, there is an atomic pair X_i defined as

$$X_i = \{(2i + 1, 2i + 1),\ (2i + 2, 2i + 2)\}.$$

Let π be the modified $\mathcal{F}$-assembly of $\mathcal{P}$. Notice that apart from the anchors, π consists of $n + 2$ disjoint tracks starting in the cell v_2 and ending in v_{2m+2}. Two

of them are formed by the guards and we call them the G_1^P-track and G_2^P-track respectively. Every other track is formed by the atomic pairs X_i for some $i \in [n]$ and we call such a track the X_i-track.

Constructing the text τ. We construct the text τ from a family of $(12n+4)$-tiles $\mathcal{T}$. The first tile T_{v_1} again contains only a single atomic pair of anchors defined as

$$(1,1),\ (12n+4, 12n+4).$$

Every other tile on the path contains two guards defined as

$$G_1^T = \{(1,1),\ (2,2)\},$$
$$G_2^T = \{(12n+3, 12n+3),\ (12n+4, 12n+4)\}.$$

The tile T_{v_2} is meant to simulate the assignment of variables in φ. We add for each $i \in [n]$, two atomic pairs Y_i^T and Y_i^F defined as

$$Y_i^T = \{(12i-9, 12i-3),\ (12i-4, 12i+2)\},$$
$$Y_i^F = \{(12i-3, 12i-9),\ (12i+2, 12i-4)\}.$$

Observe that the pairs Y_i^T and Y_i^F for a fixed i form together the pattern 3412 and the whole tile T_{v_2} is a direct sum of n copies of 3412, one for each $i \in [n]$. We view these atomic pairs as the beginnings of $2n$ disjoint tracks.

For odd $j > 2$, the tile T_{v_j} is obtained by taking a specific subset of $6n$ atomic pairs, a pair $Z_{\ell,a}^{\alpha}$ for each $\ell \in [n], a \in [3]$, and a choice of $\alpha \in \{T, F\}$. The pairs $Z_{i,1}^T, Z_{i,2}^T$ and $Z_{i,3}^T$ are defined as

$$Z_{i,1}^T = \{(12i-5, 4i-1),\ (12i-4, 4i)\},$$
$$Z_{i,2}^T = \{(12i-7, 4n+4i-1),\ (12i-6, 4n+4i)\},$$
$$Z_{i,3}^T = \{(12i-9, 8n+4i-1),\ (12i-8, 8n+4i)\}.$$

Whereas the pairs $Z_{i,1}^F, Z_{i,2}^F$ and $Z_{i,3}^F$ are defined as

$$Z_{i,1}^F = \{(12i+1, 4i+1),\ (12i+2, 4i+2)\},$$
$$Z_{i,2}^F = \{(12i-1, 4n+4i+1),\ (12i, 4n+4i+2)\},$$
$$Z_{i,3}^F = \{(12i-3, 8n+4i+1),\ (12i-2, 8n+4i+2)\}.$$

The pairs $Z_{i,1}^T, Z_{i,2}^T$ and $Z_{i,3}^T$ form together the co-layered pattern 563412 and the same holds for the pairs $Z_{i,1}^F, Z_{i,2}^F$ and $Z_{i,3}^F$. Moreover, we can split the pairs into three horizontal layers – each containing an increasing sequence formed by the pairs $Z_{\ell,a}^{\alpha}$ for a fixed $a \in [3]$ and arbitrary $\alpha \in \{T, F\}$ and $\ell \in [n]$. See the left part of Figure 4.4.

For even $j > 2$, the tile T_{v_j} is also obtained by taking a specific subset of different $6n$ atomic pairs, a pair $\widetilde{Z}_{\ell,a}^{\alpha}$ for each $\ell \in [n], a \in [3]$, and a choice of $\alpha \in \{T, F\}$. The pairs $\widetilde{Z}_{i,1}^T, \widetilde{Z}_{i,2}^T$ and $\widetilde{Z}_{i,3}^T$ are defined as

$$\widetilde{Z}_{i,1}^T = \{(12i-9, 4i-1),\ (12i-4, 4i)\},$$
$$\widetilde{Z}_{i,2}^T = \{(12i-9, 4n+4i-1),\ (12i-4, 4n+4i)\},$$
$$\widetilde{Z}_{i,3}^T = \{(12i-9, 8n+4i-1),\ (12i-4, 8n+4i)\}.$$

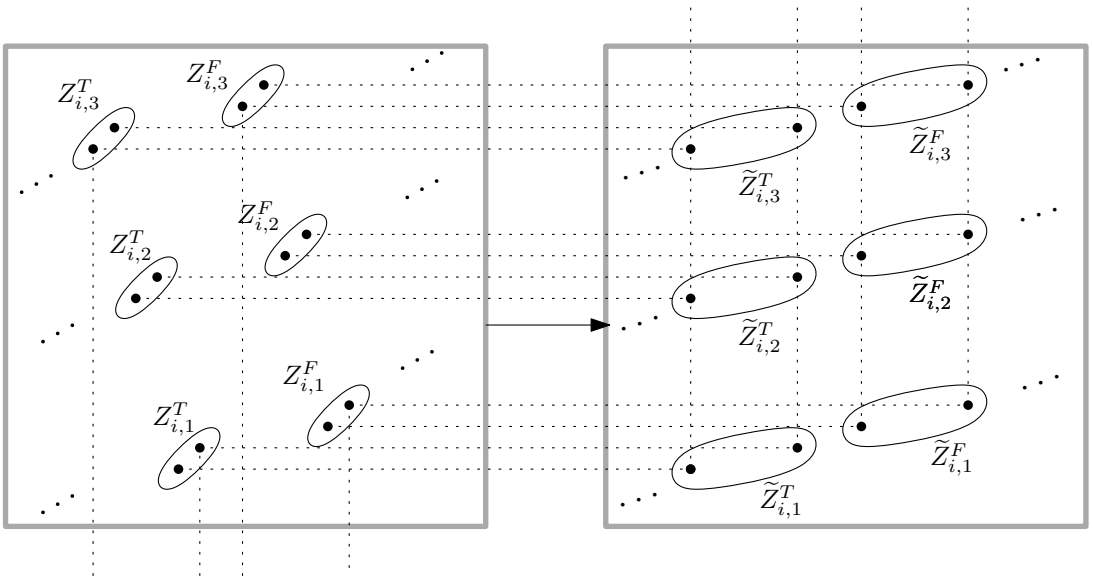

Figure 4.4: The relative position of atomic pairs $Z^\alpha_{i,a}, \widetilde{Z}^\alpha_{i,a}$ for a fixed i and all $\alpha \in \{T, F\}$, $a \in [3]$ inside two consecutive tiles in the proof of Theorem 4.15.

Whereas the pairs $\widetilde{Z}^F_{i,1}, \widetilde{Z}^F_{i,2}$ and $\widetilde{Z}^F_{i,3}$ are defined as

$$\begin{aligned}
\widetilde{Z}^F_{i,1} &= \{(12i-3, 4i+1),\ (12i+2, 4i+2)\},\\
\widetilde{Z}^F_{i,2} &= \{(12i-3, 4n+4i+1),\ (12i+2, 4n+4i+2)\},\\
\widetilde{Z}^F_{i,3} &= \{(12i-3, 8n+4i+1),\ (12i+2, 8n+4i+2)\}.
\end{aligned}$$

For a fixed $i \in [n]$, the pairs $\widetilde{Z}^T_{i,1}, \widetilde{Z}^T_{i,2}$ and $\widetilde{Z}^T_{i,3}$ all share the same x-coordinates, and the same holds for the pairs $\widetilde{Z}^F_{i,1}, \widetilde{Z}^F_{i,2}$ and $\widetilde{Z}^F_{i,3}$. And as before, we can split the pairs into three horizontal layers – each containing an increasing sequence formed by the pairs $\widetilde{Z}^\alpha_{\ell,a}$ for a fixed $a \in [3]$ and arbitrary $\alpha \in \{T, F\}$ and $\ell \in [n]$. See the right part of Figure 4.4.

Assume for now that every tile T_{v_j} for $j > 2$ contains all of the defined atomic pairs. For every $i \in [n]$ and $\alpha \in \{T, F\}$, the pair Y^α_i in the tile T_{v_2} sandwiches precisely the pairs $Z^\alpha_{i,1}, Z^\alpha_{i,2}$ and $Z^\alpha_{i,3}$ in T_{v_3}. For every $\ell \in [3]$, the pair $Z^\alpha_{i,\ell}$ from T_{v_3} then sandwiches only the pair $\widetilde{Z}^\alpha_{i,\ell}$ in T_{v_4}. Going further, the pair $\widetilde{Z}^\alpha_{i,\ell}$ in T_{v_4} sandwiches all the pairs $Z^\alpha_{i,1}, Z^\alpha_{i,2}$ and $Z^\alpha_{i,3}$ in T_{v_5} and so on. On more conceptual level, the text (apart from the guards) is formed by $2n$ tracks that fork in the tile $T_{v_{2t+1}}$ to several different atomic pairs, each of which sandwiches precisely one atomic pair in the tile $T_{v_{2t+2}}$ and they again merge to a single track going into the tile $T_{v_{2t+3}}$. However in reality, every tile T_{v_j} will contain possibly different subset of these pairs encoding some clause of φ as we now proceed to define.

Fix $t \in [m]$ and suppose the clause K_t is of the form $L_i \vee L_j \vee L_k$, with $L_i \in \{x_i, \neg x_i\}$, $L_j \in \{x_j, \neg x_j\}$ and $L_k \in \{x_k, \neg x_k\}$, for some $i < j < k$. The tile $T_{v_{2t+1}}$ consists of

- the pairs $Z^T_{\ell,1}, Z^F_{\ell,1}$ for $\ell < i$,
- the pairs $Z^T_{\ell,1}, Z^F_{\ell,1}, Z^T_{\ell,2}, Z^F_{\ell,2}$ for $i < \ell < j$,
- the pairs $Z^T_{\ell,2}, Z^F_{\ell,2}, Z^T_{\ell,3}, Z^F_{\ell,3}$ for $j < \ell < k$,
- the pairs $Z^T_{\ell,3}, Z^F_{\ell,3}$ for $\ell > k$, and

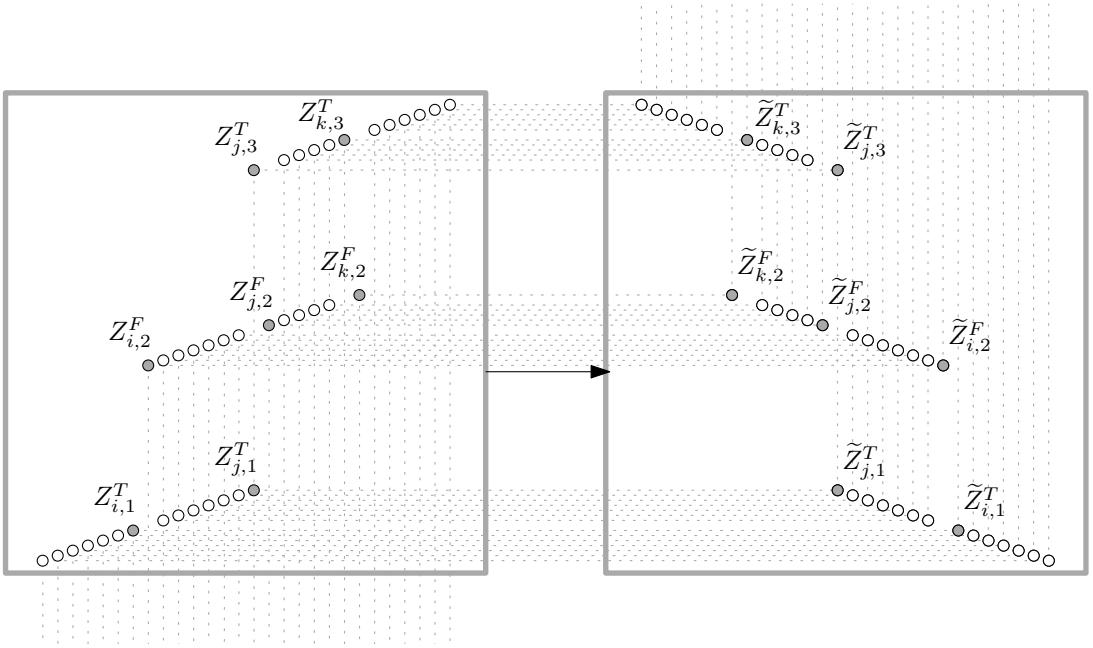

Figure 4.5: Enforcing satisfiability in the proof of Theorem 4.15. The permutation τ zoomed in to cells v_{2t+1} (left) and v_{2t+2} (right) for the clause $K_t = (x_i \vee \neg x_j \vee x_k)$. Atomic pairs are represented by circles. Observe that there is no choice of $a, b, c \in [3]$ such that pairs $Z^F_{i,a}, Z^T_{j,b}, Z^F_{k,c}$ form an increasing sequence in the cell v_{2t+1}.

- the pairs $Z^+_{i,1}, Z^-_{j,1}, Z^-_{i,2}, Z^+_{j,2}, Z^-_{k,2}, Z^-_{j,3}, Z^+_{k,3}$ where $Z^+_{\ell,a} = Z^T_{\ell,a}$ if L_ℓ is the positive literal x_ℓ, otherwise $Z^+_{\ell,a} = Z^F_{\ell,a}$, and $Z^-_{\ell,a} = \{Z^T_{\ell,a}, Z^F_{\ell,a}\} \setminus Z^+_{\ell,a}$.

Notice that the included pairs associated to the variables x_i, x_j and x_k are listed in increasing order of their y-coordinates and they are ordered exactly as in Observation 4.2. In particular, there is no increasing sequence formed by pairs $Z^-_{i,a}, Z^-_{j,b}, Z^-_{k,c}$ for any choice of $a, b, c \in [3]$. Finally, we set the tile $T_{v_{2t+2}}$ to contain an atomic pair $\widetilde{Z}^\alpha_{\ell,a}$ if and only if $T_{v_{2t+1}}$ contains the pair $Z^\alpha_{\ell,a}$. See Figure 4.5.

Let τ be the modified $\mathcal{F}$-assembly of $\mathcal{T}$. Observe that the length of τ is bounded by $O(nm)$. We say that an embedding of π into τ is *anchored*, if it maps the anchors in π to the anchors in τ.

Claim 4.16. *The formula φ is satisfiable if and only if there is an anchored embedding of π into τ.*

Correctness ("only if"). Suppose that φ is satisfiable. Fix any satisfying assignment represented by a function $\rho : [n] \to \{T, F\}$, where $\rho(i) = T$ if and only if x_i is set to true. We describe how to construct an anchored embedding of π into τ.

We map the anchors in π to the anchors in τ and moreover, we map the guards in the tile P_{v_j} for $j \geq 2$ to the guards in the tile T_{v_j}. For each $i \in [n]$, we map the atomic pair X_i to the pair $Y^{\rho(i)}_i$ in the tile T_{v_2}.

Fix $t \in [m]$ and let us describe the mapping inside the tiles $T_{v_{2t+1}}$ and $T_{v_{2t+2}}$. We assume again that the clause K_t is of the form $L_i \vee L_j \vee L_k$ for some $i < j < k$.

We map every atomic pair in $P_{v_{2t+1}}$ except for X_i, X_j and X_k to the following atomic pairs in $T_{v_{2t+1}}$:

- for $\ell < i$, we map the pair X_ℓ to the pair $Z_{\ell,1}^{\rho(\ell)}$,
- for $i < \ell < j$, we map the pair X_ℓ to the pair $Z_{\ell,1}^{\rho(\ell)}$ if the literal L_i is satisfied, otherwise we map it to $Z_{\ell,2}^{\rho(\ell)}$,
- for $j < \ell < k$, we map the pair X_ℓ to the pair $Z_{\ell,3}^{\rho(\ell)}$ if the literal L_k is satisfied, otherwise we map it to $Z_{\ell,2}^{\rho(\ell)}$, and
- for $\ell > k$, we map the pair X_ℓ to the pair $Z_{\ell,3}^{\rho(\ell)}$.

For $\ell \in \{i, k\}$, we map X_ℓ to the only pair of the form $Z_{\ell,a}^{\rho(\ell)}$ that is contained in $T_{v_{2t+1}}$. We carefully defined the mapping of pairs X_ℓ for $\ell \in \{i, i+1, \ldots, k\} \setminus \{j\}$ in order to guarantee that their images form an increasing sequence. If at least one of the literals L_i, L_k is satisfied then we can find a pair of the form $Z_{j,p}^{\rho(j)}$ that fits into this increasing sequence regardless of the value $\rho(j)$. Otherwise, the literal L_j must be satisfied and we can map X_j to the pair $Z_{j,2}^{+}$.

Finally, we define the mapping in the tile $T_{v_{2t+2}}$ in the exact same way. More precisely, if the pair X_ℓ in $P_{v_{2t+1}}$ is mapped to the pair $Z_{\ell,a}^{\rho(\ell)}$ in $T_{v_{2t+1}}$, we map the pair X_ℓ in $P_{v_{2t+2}}$ to the pair $\widetilde{Z}_{\ell,a}^{\rho(\ell)}$ in $T_{v_{2t+2}}$. It is easy to check that we have indeed produced a valid anchored embedding of π into τ.

Correctness ("if"). Suppose there is an anchored embedding ψ of π into τ. We immediately see that the tile P_{v_2} must be mapped to the tile T_{v_2}. Observe that P_{v_2} is an increasing sequence of length $2n+4$ and every increasing subsequence of T_{v_2} consists of the pairs G_1^T, G_2^T and Y_i^α for every $i \in [n]$ and some $\alpha \in \{T, F\}$. Therefore, the guards G_1^P, G_2^P are mapped to the guards G_1^T, G_2^T and every pair X_i is mapped to a pair Y_i^α for some α. We define an assignment $\rho\colon [n] \to \{T, F\}$ by setting $\rho(i)$ such that ψ maps the pair X_i to the pair $Y_i^{\rho(i)}$.

Since the guards in the tile P_{v_2} are mapped to the guards in T_{v_2}, the tracks enforce that the guards in any tile P_{v_j} for $j > 2$ are mapped to the guards in the tile T_{v_j}. As a consequence, the whole tile P_{v_j} must be mapped to the tile T_{v_j}. The structure of τ also enforces that for every $t \in [m]$, the pair X_i in the tile $P_{v_{2t+1}}$ maps to the pair $Z_{i,p}^{\rho(i)}$ in $T_{v_{2t+1}}$ and the pair X_i in the tile $P_{v_{2t+2}}$ maps to the pair $\widetilde{Z}_{i,p}^{\rho(i)}$ in $T_{v_{2t+2}}$ for some $p \in [3]$.

For $t \in [m]$, let us show that ρ satisfies the clause K_t. The pairs X_i, X_j and X_k in the tile $P_{v_{2t+1}}$ are mapped to pairs $Z_{i,a}^{\rho(i)}, Z_{j,b}^{\rho(j)}$ and $Z_{k,c}^{\rho(k)}$ for some $a, b, c \in [3]$ in $T_{v_{2t+1}}$. We included the pairs associated to variables x_i, x_j and x_k in $T_{v_{2t+1}}$ such that there is no increasing subsequence formed by pairs $Z_{i,a}^{-}, Z_{j,b}^{-}$ and $Z_{k,c}^{-}$ for any choice of a, b, c. In other words, there must be at least one $\ell \in \{i, j, k\}$ such that X_ℓ is mapped to a pair $Z_{\ell,a}^{+}$ for some $a \in [3]$ and thus, the literal L_ℓ (and by extension the whole clause K_t) is satisfied by ρ.

Inflating the anchors. Finally, we transfer from anchored embeddings to regular embeddings using inflations. Observe that we can inflate the anchors in π with either increasing or decreasing sequences (depending on $\mathcal{F}$) while keeping the pattern in the class $\mathcal{C}$. Therefore, let π' be the permutation obtained from π

by inflating both anchors with monotone sequences of length $|\tau|$ and let τ' be the permutation obtained from τ by inflating its both anchors with the decreasing sequence of length $|\tau|$. Note that we still have $|\tau'| \in O(nm)$.

Claim 4.17. *There is an anchored embedding of π into τ if and only if there is an embedding of π' into τ'.*

It is clear that any anchored mapping of π into τ can easily be remade into a mapping of π' into τ'. For the other direction, assume there is a mapping ψ of π' into τ'. Observe that the inflated anchors in π' contain $2 \cdot |\tau|$ points while τ' contains only $|\tau| - 2$ points outside of its inflated anchors. Hence, there must be at least $|\tau| + 2$ points of the anchors in π' that map to the anchors in τ' by simple counting argument. In particular each inflated anchor in π' contains a point that maps to the respective inflated anchor in τ'. It follows from the structure of π and τ that the image of the tile P_{v_2} maps to the image of the tile T_{v_2} and the guards then enforce that the image of every P_{v_i} in fact maps to the image of T_{v_i}. Therefore, if we restrict ψ to the points outside the anchors, we get the description of an anchored mapping of π into τ minus the anchors.

It follows that $\mathcal{C}$-Pattern PPM is NP-complete and it remains to show the conditional lower bound. Assume that there is an algorithm solving $\mathcal{C}$-Pattern PPM in time $2^{o(\sqrt{|\tau|})}$. By plugging in the instance produced by our reduction, we obtain an algorithm solving 3-SAT in time $2^{o(\sqrt{nm})}$ where n is the number of variables and m is the number of clauses. Such an algorithm would, however, refute ETH in its clause form (Hypothesis 2) since $n \leq 3m$. □

Assuming the deep tree property, the reduction of Theorem 4.15 can be made more efficient implying significantly better conditional lower bound under ETH.

Theorem 4.18. *Let $\mathcal{C}$ be a permutation class with the poly-time computable deep tree property. Then $\mathcal{C}$-Pattern PPM is NP-complete and unless ETH fails, it cannot be solved in time $2^{o(|\tau|/\log|\tau|)}$ where τ is the text.*

Proof. The reduction of Theorem 4.15 used the same idea as the reduction of Theorem 4.1 by mapping the rows of the matrix-like structure along the path of cells. We show that the deep tree property allows us to deal with each row (each clause of the formula) in one leaf of the 'deep tree' which leads to a more efficient reduction.

We again reduce from 3-SAT and let φ be a 3-CNF formula with n variables $x_1, \ldots, x_n$ and m clauses $K_1, \ldots, K_m$. We again assume that each clause contains exactly three literals and no variable appears more than once in a single clause. We start by obtaining in polynomial time a monotone gridding matrix $\mathcal{M}$ with the following properties:

(i) Grid($\mathcal{M}$) is a subclass of $\mathcal{C}$,
(ii) the cell graph of $\mathcal{M}$ is a rooted c-subdivided tree with m leaves,
(iii) the root r of the tree has a single child r', and
(iv) every leaf shares a common column with its parent.

The first two properties are directly guaranteed by the poly-time computable deep tree property, the second two are easily achieved by starting with a slightly larger

tree and subsequently cutting off some of its branches. Let $\mathcal{F}$ be a consistent orientation of $\mathcal{M}$ guaranteed by Lemma 1.2.

We now want to associate each leaf with a single clause. In order to do that, we shall take plenty of inspiration from the proof of Lemma 2.38. We orient the edges of $G_{\mathcal{M}}$ consistently away from the root r and we arbitrarily order the leaves as $v_1, \ldots, v_m$. We say that the *descendants of a vertex* v, denoted by $D(v)$, are the out-neighbors of v.

We inductively assign a set $A_w \subseteq [n]$ to each vertex w of the tree:

$$A_w = \begin{cases} \{i,j,k\} & \text{if } w = v_t \text{ for } t \in [m] \text{ and } K_t = L_i \vee L_j \vee L_k, \\ \bigcup_{v \in D(w)} A_v & \text{otherwise.} \end{cases}$$

Observe that $\sum_v |A_v| \in O(m \log m)$ since φ contains exactly $3m$ literals and each contributes at most $O(\log m)$ due to the path connecting it to the root r.

We use as a starting point the reduction of Theorem 4.15 with its two families of tiles $\mathcal{P}, \mathcal{T}$ and we define two families of tiles $\mathcal{P}', \mathcal{T}'$ using the same atomic pairs but distributing them more sparsely.

The root r plays the role of the anchors by setting $P'_r = P_{v_1}$ and $T'_r = T_{v_1}$. Its child r' simulates the assignment, i.e. we set $P'_{r'} = P_{v_2}$ and $T'_{r'} = T_{v_2}$. Let v be any other vertex of the tree. We let the tile P'_v contain the guards G^P_1, G^P_2 and then all the atomic pairs X_i for $i \in A_v$. This completes the definition of the tile family $\mathcal{P}'$ and we take the pattern π as its modified $\mathcal{F}$-assembly.

The tiles T'_v for any non-leaf vertex v will be formed by guards and a subset of atomic pairs $\widetilde{Y}^\alpha_i$ for $i \in [n]$ and $\alpha \in \{T, F\}$ defined as

$$\widetilde{Y}^T_i = \{(12i-9, 12i-9),\ (12i-4, 12i-4)\},$$
$$\widetilde{Y}^F_i = \{(12i-3, 12i-3),\ (12i+2, 12i+2)\}.$$

Observe that all these pairs together form an increasing sequence in the order $\widetilde{Y}^T_1$, $\widetilde{Y}^F_1, \widetilde{Y}^T_2, \widetilde{Y}^F_2, \ldots, \widetilde{Y}^T_n, \widetilde{Y}^F_n$. We set the tile T'_v to contain only the guards G^T_1 and G^T_2 and the pairs $\widetilde{Y}^T_i$ and $\widetilde{Y}^F_i$ for every $i \in A_v$.

Finally, we need to define the tile T'_{v_t} for each leaf v_t. We again assume that the clause K_t is of the form $L_i \vee L_j \vee L_k$ for some $i < j < k$. The tile T'_{v_t} contains the guards G^T_1 and G^T_2 and the pairs $Z^+_{i,1}, Z^-_{j,1}, Z^-_{i,2}, Z^+_{j,2}, Z^-_{k,2}, Z^-_{j,3}, Z^+_{k,3}$. Recall that we defined $Z^+_{\ell,a} = Z^T_{\ell,a}$ if L_ℓ is the positive literal x_ℓ, otherwise $Z^+_{\ell,a} = Z^F_{\ell,a}$, and $Z^-_{\ell,a} = \{Z^T_{\ell,a}, Z^F_{\ell,a}\} \setminus Z^+_{\ell,a}$.

We take τ as the modified $\mathcal{F}$-assembly of $\mathcal{T}'$. Observe that we have $|T'_v| \in O(|A_v|)$ for each vertex v of the tree. Subsequently, the length of τ is bounded by $O(m \log m)$.

Claim 4.19. *The formula φ is satisfiable if and only if there is an anchored embedding of π into τ.*

Correctness ("only if"). Let $\rho : [n] \to \{T, F\}$ represent a satisfying assignment for φ. We define an anchored mapping of π into τ. We map the anchors in π to the anchors in τ and the guards in every tile P'_v to the guards in the tile T'_v. For each $i \in [n]$, the pair X_i in the tile $P'_{r'}$ is mapped to the pair $Y^{\rho(i)}_i$, simulating the assignment ρ. Moreover for every non-leaf vertex v, the pair X_i in P'_v is mapped to the pair $\widetilde{Y}^{\rho(i)}_i$ in the tile T'_v.

Finally for every leaf v_t, we define the mapping as for the vertex v_{2t+1} in the proof of Theorem 4.15. The only difference is that P'_{v_t} and T'_{v_t} contain only the atomic pairs associated to the three variables present in the clause K_t. To be more precise, it follows using the same arguments that we can find an increasing sequence $Z^{\rho(i)}_{i,a}, Z^{\rho(j)}_{j,b}, Z^{\rho(k)}_{k,c}$ inside T'_{v_t} for some $a, b, c \in [3]$. We map the pairs X_i, X_j, X_k in P'_v precisely to this sequence. The correctness of the anchored embedding follows almost exactly as before.

Correctness ("if"). Suppose there is an anchored embedding ψ of π into τ. It is basically a mechanical replay of the arguments in the proof of Theorem 4.15 to show that φ is satisfiable.

We again define $\rho : [n] \to \{T, F\}$ such that X_i of the tile $P'_{r'}$ maps to $Y_i^{\rho(i)}$ in the tile $T'_{r'}$. The only purpose of the other non-leaf vertices in the tree is to transport information about the assignment ρ all the way to the leaves. In particular for a non-leaf vertex v, the pair X_i in P'_v must be mapped to the pair $\widetilde{Y}_i^{\rho(i)}$. For every $t \in [m]$, the leaf v_t then enforces that the clause K_t is satisfiable exactly in the same way as the vertex v_{2t+1} in the proof of Theorem 4.15.

Finally, we construct permutations π' and τ' by inflating the anchors in π and τ with a monotone sequences of length $|\tau|$. By now, we feel comfortable with omitting the details of this argument. Note that we still have $|\tau'| \in O(m \log m)$ and therefore, we rule out algorithm solving $\mathcal{C}$-Pattern PPM in time $2^{o(n/\log n)}$ under ETH where n is the length of τ. Otherwise, we could solve 3-SAT in time $2^{o(m)}$ and thus, refute ETH (Hypothesis 2). □

4.3.2 Parameterized complexity

As our next step, we focus on the hardness of $\mathcal{C}$-Pattern PPM through the lens of parameterized complexity. It turns out that we can replicate most results from Section 4.2 as long as we assume that $\mathcal{C}$ has either the computable long path or the computable deep tree property.

2-left-aligned patterns

Instead of using a pattern-restricted version of Left PPM, we need to strengthen our requirements and consider embeddings that map the two leftmost points in the pattern to the two leftmost points in the text.

2-left-aligned Pattern Matching (2-Left PPM)	
Input:	Permutations π of length k and τ of length n.
Output:	Is there an embedding of π into τ that maps the two leftmost points of π to the two leftmost points of τ?

Any embedding of π into τ with the desired property is a *2-left-aligned embedding (of π into τ).* Note that 2-Left PPM is W[1]-complete and the same lower bounds as for Left PPM apply by a straightforward reduction. In an instance (π, τ) of Left PPM, it is sufficient to inflate the leftmost points in both π and τ with the pattern 12. The W[1]-completeness of 2-Left PPM then follows since we can encode it as a model checking problem of an existential

first-order formula similarly to Left PPM. The problem $\mathcal{C}$-Pattern 2-Left PPM is defined naturally by restricting the pattern π to the class $\mathcal{C}$.

Our main technical contribution is the following pattern-restricted counterpart to Proposition 4.7 from which we deduce all subsequent hardness results.

Proposition 4.20. *Unless ETH fails, $\mathcal{C}$-Pattern 2-Left PPM cannot be solved for any function f*

- *in time $f(k) \cdot n^{o(\sqrt{k})}$ if $\mathcal{C}$ has the computable long path property, and*
- *in time $f(k) \cdot n^{o(k/\log^2 k)}$ if $\mathcal{C}$ has the computable deep tree property,*

where k is the length of the pattern and n is the length of the text. Moreover, $\mathcal{C}$-Pattern 2-Left PPM is W[1]-complete with respect to k in both these cases.

We shall prove each claim via a different reduction stated as a standalone lemma. In the big picture, we use once again the same idea of modifying the reduction of Proposition 4.7 with its matrix-like structure. For a class $\mathcal{C}$ with the computable long path property, we simulate each row of the matrix inside constantly many consecutive cells along a path. Whereas for a class $\mathcal{C}$ with the computable deep tree property, we simulate each row in a cell occupying a unique leaf of the subdivided binary tree.

Before diving into either of the two reductions, we set up some tools used by both of them. In particular, we define several types of tiles constructed from atomic pairs.

Fix an instance (G, H, χ) of PSI and set $n = |V_H|$, $k = |V_G|$. We mostly stick to the same notation as in the proof of Proposition 4.7. Namely, we identify the vertices of V_G with the set $[k]$ and we let $V_a \subseteq V_H$ be the set of all vertices colored by $a \in [k]$. Moreover, we set $n_a = |V_a|$ and we choose an arbitrary order of vertices in V_a denoting them v_i^a for $i \in [n_a]$. The (reverse) rank of a vertex v_i^a is defined exactly as in (4.1).

Set $m = 3n$. Let E_a be the set of all pairs (v_i^a, v_j^b) such that either the edge $\{a, b\} \notin E_G$ or $\{v_i^a, v_j^b\} \in E_H$. We shall build our reductions from only a handful different types of tiles that we now proceed to describe.

Except for the anchor tile, all atomic pairs inside the tiles are associated either to a vertex of the graph G or to a vertex of H. In particular, every atomic pair in a pattern tile is associated to a vertex a in G and atomic pairs of all the other tiles are associated to vertices of H. We will guarantee, that upon a modified $\mathcal{F}$-assembly of the constructed tile families, any embedding of the pattern into the text must map an atomic pair associated to a vertex $a \in V(G)$ to an atomic pair associated to a vertex $v_i^a \in V(H)$ for some $i \in [n_a]$.

Anchor tile. An *anchor tile*, denoted by Anchor, contains a single atomic pair

$$T = \{(0, 0),\ ((n+1) \cdot m, (n+1) \cdot m)\}.$$

Pattern tile. For every $W \subseteq [k]$, a *pattern tile of W*, denoted by Pattern(W), contains for each $a \in W$ an atomic pair

$$X_a = \{(2a+1, 2a+1),\ (2a+2, 2a+2)\}$$

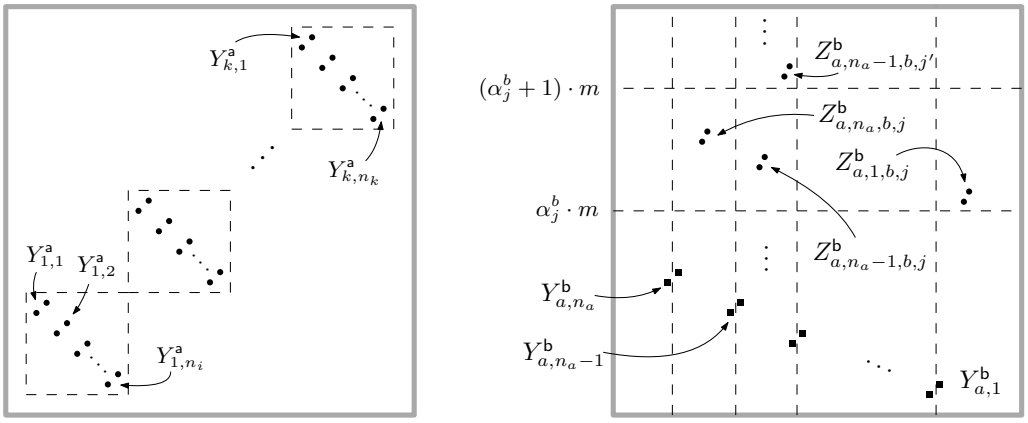

Figure 4.6: An assignment tile Assign on the left and a branch tile $\mathsf{Branch}(a, W)$ on the right. Note that the branch tile is zoomed to the vertical strip between $\alpha^a_i \cdot m$ and $(\alpha^a_{n_a}+1)\cdot m$ as the rest of the matrix is empty.

The pair X_a is associated to the vertex a in G. Observe that any pattern tile is simply an increasing sequence of length $2|W|$.

The pattern defined by our reductions will consist of a single anchor tile followed by a sequence of pattern tiles. Observe that upon a modified $\mathcal{F}$-assembly, the atomic pair X_a in any tile sandwiches the atomic pair X_a in the following tile.

Assignment tile. An *assignment tile*, denoted by Assign, contains for each $a \in [k]$ and $i \in [n_a]$ an atomic pair $Y^{\mathsf{a}}_{a,i}$ consisting of points

$$Y^{\mathsf{a}}_{a,i} = \{(\alpha^a_i \cdot m + 3,\ \beta^a_i \cdot m + 3)\,,\ ((\alpha^a_i+1)\cdot m + 2,\ (\beta^a_i+1)\cdot m + 2)\}\,.$$

Observe that each pair $Y^{\mathsf{a}}_{a,i}$ forms an occurrence of 12 and the atomic pairs corresponding to the vertices of V_a form a skew sum of n_a copies of 12. On the other hand for any $a < b$, the atomic pairs of V_a lie all in a block to the left and below of all the atomic pairs of V_b. See the left part of Figure 4.6. Observe that the assignment tile contains exactly $2 \cdot |V_H|$ points.

We associate the atomic pair $Y^{\mathsf{a}}_{a,i}$ to the vertex v^a_i in H. Observe that in any mapping of the tile $\mathsf{Pattern}([k])$ into the tile Assign, each atomic pair X_a in the pattern tile must be mapped to an atomic pair $Y^{\mathsf{a}}_{a,i}$ for some $i \in [n_a]$.

Identity tile. For a subset $W \subseteq V_G = [k]$, an identity tile of W, denoted by $\mathsf{Id}(W)$, contains for every $a \in W$ and $j \in [n_a]$ an atomic pair

$$Y^{\mathsf{id}}_{a,i} = \{(\alpha^a_i \cdot m + 3,\ \alpha^a_i \cdot m + 3)\,,\ ((\alpha^a_i+1)\cdot m + 2,\ (\alpha^a_i+1)\cdot m + 2)\}\,.$$

Once again, we associate the pair $Y^{\mathsf{id}}_{a,i}$ to the vertex v^a_i in H. Observe that $\mathsf{Id}(W)$ forms an increasing sequence of length $2 \cdot \sum_{a\in W} |V_a|$.

If we place the tile $\mathsf{Id}([k])$ in the same column as the tile Assign with an edge between them oriented towards the assign tile, then upon a modified $\mathcal{F}$-assembly, the atomic pair $Y^{\mathsf{a}}_{a,i}$ sandwiches the atomic pair $Y^{\mathsf{id}}_{a,i}$ for every $a \in [k]$ and $i \in [n_a]$.

Branch tile. Let $a \in [k]$ and $W \subseteq [k]$ such that $a < b$ for every $b \in W$. A branch tile of a and W, denoted by $\mathsf{Branch}(a, W)$, contains for every $i \in [n_a]$ an

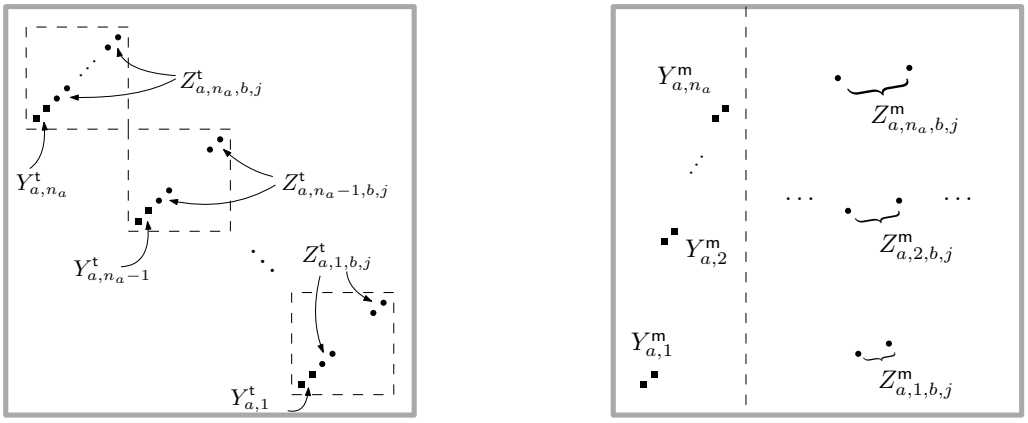

Figure 4.7: A test tile $\mathsf{Test}(a, W)$ on the left and a merge tile $\mathsf{Merge}(a, W)$ on the right. Note that the test tile is zoomed to the box between $\alpha^a_i \cdot m$ and $(\alpha^a_{n_a} + 1) \cdot m$ in both axes as the rest of the matrix is empty. In the merge tile we focus just on the vertical strip between between $\alpha^a_i \cdot m$ and $(\alpha^a_{n_a} + 1) \cdot m$ and on the strip between $\alpha^b_j \cdot m - n$ and $(\alpha^b_j + 1) \cdot m + n$.

atomic pair

$$Y^{\mathsf{b}}_{a,i} = \{(\beta^a_i \cdot m + 3,\ \alpha^a_i \cdot m + 3)\,, \quad (\beta^a_i \cdot m + 4,\ \alpha^a_i \cdot m + 4)\}\,.$$

Moreover it contains for every $b \in W$ and $(v^a_i, v^b_j) \in E_a$ an atomic pair

$$Z^{\mathsf{b}}_{a,i,b,j} = \left\{ \begin{array}{c} \left(\beta^a_i \cdot m + 2\alpha^b_j + 5,\ \alpha^b_j \cdot m + 2\alpha^a_i + 3\right), \\ \left(\beta^a_i \cdot m + 2\alpha^b_j + 6,\ \alpha^b_j \cdot m + 2\alpha^a_i + 4\right) \end{array} \right\}.$$

We associate the pair $Y^{\mathsf{b}}_{a,i}$ to the vertex v^a_i in H whereas all the pairs $Z^{\mathsf{b}}_{a,i,b,j}$ are associated to the vertex v^b_j. Observe that $Z^{\mathsf{b}}_{a,i,b,j}$ lies in the horizontal strip between $y = \alpha^b_j \cdot m + 3$ and $y = (\alpha^b_j + 1) \cdot m + 2$. If we look at all the atomic pairs in this strip, they are all of the form $Z^{\mathsf{b}}_{a,i',b,j}$ where $(v^a_{i'}, v^b_j) \in E_a$ and they form a skew sum of several copies of 12. Vertically, $Z^{\mathsf{b}}_{a,i,b,j}$ lies in the strip between $x = \beta^a_i \cdot m + 5$ and $x = (\beta^a_i + 1) \cdot m + 2$. And if we look at all the atomic pairs in this strip, they are all of the form $Z^{\mathsf{b}}_{a,i,b',j'}$ where $(v^a_i, v^{b'}_{j'}) \in E_a$ for some $b' > a$. Moreover, all the atomic pairs in this strip form an increasing sequence. See the right part of Figure 4.6.

In our construction of the text permutation, a branch tile will always follow an identity tile and moreover, they will share the same row. Observe that then upon a modified $\mathcal{F}$-assembly, the atomic pair $Y^{\mathsf{id}}_{a,i}$ sandwiches the atomic pair $Y^{\mathsf{b}}_{a,i}$ and the atomic pair $Y^{\mathsf{id}}_{b,j}$ sandwiches every existing pair of the form $Z^{\mathsf{b}}_{a,i,b,j}$.

Test tile. A test tile of $a \in [k]$ and $W \subseteq [k]$ such that $a < b$ for every $b \in W$, denoted by $\mathsf{Test}(a, W)$, contains for every $i \in [n_a]$ an atomic pair

$$Y^{\mathsf{t}}_{a,i} = \{(\beta^a_i \cdot m + 3,\ \alpha^a_i \cdot m + 3)\,, \quad (\beta^a_i \cdot m + 4,\ \alpha^a_i \cdot m + 4)\}\,.$$

Furthermore for every $b \in W$ and $(v^a_i, v^b_j) \in E_a$, it contains an atomic pair

$$Z^{\mathsf{t}}_{a,i,b,j} = \left\{ \begin{array}{c} \left(\beta^a_i \cdot m + 2\alpha^b_j + 5,\ \alpha^a_i \cdot m + 2\alpha^b_j + 5\right), \\ \left(\beta^a_i \cdot m + 2\alpha^b_j + 6,\ \alpha^a_i \cdot m + 2\alpha^b_j + 6\right) \end{array} \right\}.$$

Similar to the branch tile, we associate the pair $Y^{\mathsf{t}}_{a,i}$ to the vertex v^a_i in H whereas all the pairs $Z^{\mathsf{t}}_{a,i,b,j}$ are associated to the vertex v^b_j. The test tile can be obtained as a skew sum of increasing blocks. For every $i \in [n_a]$, there is a block B_i that contains the pair $Y^{\mathsf{t}}_{a,i}$ in its bottom left corner followed by pairs $Z^{\mathsf{t}}_{a,i,b,j}$ ordered lexicographically by (b, j). See the left part of Figure 4.7.

In our construction, a test tile will always follow an identity tile and moreover, they will share the same column. Observe that then upon a modified $\mathcal{F}$-assembly, the atomic pair $Y^{\mathsf{b}}_{a,i}$ sandwiches the atomic pair $Y^{\mathsf{t}}_{a,i}$ and every existing atomic pair $Z^{\mathsf{b}}_{a,i,b,j}$ in the branch tile sandwiches the pair $Z^{\mathsf{t}}_{a,i,b,j}$ in the test tile.

Merge tile. Finally, we define a merge tile of a and $W \subseteq [k]$ such that $a < b$ for every $b \in W$, denoted by $\mathsf{Merge}(a, W)$. For every $i \in [n_a]$, it contains an atomic pair

$$Y^{\mathsf{m}}_{a,i} = \{(\alpha^a_i \cdot m + 3,\ \alpha^a_i \cdot m + 3)\,,\ ((\alpha^a_i + 1) \cdot m + 2,\ \alpha^a_i \cdot m + 4)\}\,.$$

And for every $b \in W$ and $(v^a_i, v^b_j) \in E_a$, it contains an atomic pair

$$Z^{\mathsf{m}}_{a,i,b,j} = \left\{ \begin{array}{c} \left(\alpha^b_j \cdot m - \alpha^a_i + 3,\ \alpha^a_i \cdot m + 2\alpha^b_j + 5\right), \\ \left((\alpha^b_j + 1) \cdot m + \alpha^a_i + 2,\ \alpha^a_i \cdot m + 2\alpha^b_j + 6\right) \end{array} \right\}.$$

As before, we associate the pair $Y^{\mathsf{m}}_{a,i}$ to the vertex v^a_i in H and all the pairs $Z^{\mathsf{m}}_{a,i,b,j}$ to the vertex v^b_j. The merge tile is split into n vertical strips, the first n_a of them contains the atomic pairs $Y^{\mathsf{m}}_{a,i}$ and every following strip contains the pairs $Z^{\mathsf{m}}_{a,i,b,j}$ for fixed b, j and all possible pairs of a and i. See the right part of Figure 4.7.

In our construction, a merge tile will always follow an identity tile and moreover, they will share the same row. Observe that then upon a modified $\mathcal{F}$-assembly, the atomic pair $Y^{\mathsf{t}}_{a,i}$ sandwiches the atomic pair $Y^{\mathsf{m}}_{a,i}$ and every existing atomic pair $Z^{\mathsf{t}}_{a,i,b,j}$ in the test tile sandwiches the pair $Z^{\mathsf{m}}_{a,i,b,j}$ in the merge tile.

Moreover in our construction, a merge tile will always be followed only by an identity tile and they shall occupy the same column. Observe that then upon a modified $\mathcal{F}$-assembly, the atomic pair $Y^{\mathsf{m}}_{a,i}$ sandwiches the atomic pair $Y^{\mathsf{id}}_{a,i}$ (if it exists in the identity tile) and every existing atomic pair $Z^{\mathsf{m}}_{a,i,b,j}$ in the merge tile sandwiches the pair $Y^{\mathsf{id}}_{b,j}$ in the merge tile.

Every tile other than Anchor contains additionally two other atomic pairs called *guards* defined as

$$G_1 = \{(1,1),(2,2)\}\,, \quad \text{and} \quad G_2 = \{(nm+3, nm+3),(nm+4, nm+4)\}\,.$$

Observe that in any type of tile, G_1 lies to the left and below everything else while G_2 lies to the right and above everything else.

We are now almost ready to describe the reductions. There is only one additional detail to take care of. In order to obtain tighter lower bounds for the classes with the long path property, we actually do not reduce from PSI but rather from its specific variant where we fix G to be a complete graph.

Partitioned Clique	
Input:	A positive integer k and a graph $H = (V_H, E_H)$ together with a coloring $\chi\colon V_H \to [k]$ of its vertices.
Output:	Is there a mapping $\phi\colon [k] \to V_H$ such that $\phi([k])$ induces a clique in H and moreover, $\chi(\phi(i)) = i$ for every $i \in [k]$?

By now it is a folklore result that Partitioned Clique is W[1]-complete and ETH implies an asymptotically tight lower bound on the running time of any algorithm parameterized by k.

Theorem 4.21 ([58])**.** *Partitioned Clique is W[1]-complete with respect to k and unless ETH fails, it cannot be solved in time $f(k) \cdot n^{o(k)}$ for any function f, where $n = |V_H|$ and k is the size of the desired clique.*

Finally, we can present both reductions using the tools we have built up.

Lemma 4.22. *Let $\mathcal{C}$ be a class with the computable long path property. An instance (k, H, χ) of Partitioned Clique can be reduced to an instance (π, τ) of $\mathcal{C}$-Pattern 2-Left PPM where $|\pi| \in O(k^2)$ and $|\tau| \in O(|V_H|^2)$ in time $f(k) \cdot |V_H|^{O(1)}$ for some function f. Moreover, $\mathrm{tw}(\pi) \in O(k)$.*

Proof. As usual, we let $n = |V_H|$. Due to the computable long path property, we can obtain in time $f(k)$ for some computable function f a monotone gridding matrix $\mathcal{M}$ such that

(i) $\mathrm{Grid}(\mathcal{M})$ is a subclass of $\mathcal{C}$,
(ii) the cell graph $G_{\mathcal{M}}$ is a proper-turning path $v_1, \ldots, v_{4k-2}$, and
(iii) the cell v_1 is the single non-empty cell in the leftmost column of $\mathcal{M}$.

At this point, we think it is not necessary to argue how an existence of such $\mathcal{M}$ follows from the computable long path property. By Lemma 1.2, there is a consistent orientation $\mathcal{F}$ of $\mathcal{M}$. We define two families of tiles $\mathcal{P}$ and $\mathcal{T}$ which are then used for constructing π and τ via a modified $\mathcal{F}$-assembly.

Given the input instance (k, H, χ) of Partitioned Clique, we define the tiles using the equivalent PSI instance (G, H, χ) where G is the clique on the vertex set $[k]$. We set P_{v_1} and T_{v_1} to be the anchor tiles. The tile P_{v_2} is taken to be $\mathsf{Pattern}([k])$ and T_{v_2} is taken to be Assign. The rest of the tiles are defined in k consecutive groups. Let $I_a = \{a, \ldots, k\}$. For $a \in [k-1]$, we set $P_{v_i} = \mathsf{Pattern}(I_a)$ for every $4a-1 \leq i \leq 4a+2$. On the text side, we set $T_{v_{4a-1}} = \mathsf{Id}(I_a)$, $T_{v_{4a}} = \mathsf{Branch}(a, I_{a+1})$, $T_{v_{4a+1}} = \mathsf{Test}(a, I_{a+1})$ and $T_{v_{4a+2}} = \mathsf{Merge}(a, I_{a+1})$.

Then we take π as the modified $\mathcal{F}$-assembly of $\mathcal{P}$, τ as the modified $\mathcal{F}$-assembly of $\mathcal{T}$. Clearly, $\pi \in \mathcal{C}$ and thus, we constructed a valid instance (π, τ) of $\mathcal{C}$-Pattern 2-Left PPM in time $f(k) \cdot |V_H|^{O(1)}$. Now we check that we achieved the desired upper bounds on size. The total size of the pattern tiles is clearly $O(k^2)$. The tiles $T_{v_{4a-1}}, \ldots, T_{v_{4a+2}}$ each contain at most $2n_a \cdot n$ points and therefore, the size of the text tiles sums to $O(n^2)$.

Correctness ("only if"). Suppose that (k, H, χ) is a positive instance of Partitioned Clique and thus, (G, H, χ) is a positive instance of PSI. There is a witnessing mapping $\phi\colon [k] \to \mathbb{N}$ such that $\phi(a) \in [n_a]$ for every a and $\{v^a_{\phi(a)}, v^b_{\phi(b)}\} \in E_H$ for every different $a, b \in [k]$.

Let us define a 2-left-aligned embedding ψ of π into τ . The embedding shall be *grid-preserving*, meaning that the image of P_{v_i} is mapped to the image of T_{v_i}. That way, we automatically guarantee that ψ is 2-left-aligned since the leftmost two points in τ are formed by the image of T_{v_1} while the leftmost two points in π are formed by the image of P_{v_1}. First, we map the guards in P_{v_i} to the guards of T_{v_i} for every $i \geq 2$. Then we define the mapping of points in P_{v_2} to mimic the mapping ϕ. For each $a \in [k]$, we set the image of the atomic pair X_a in P_{v_2} to be the atomic pair $Y^{\mathsf{a}}_{a,\phi(a)}$ in T_{v_2}. If $P_{v_i} = \mathsf{Pattern}(W)$ and $T_{v_i} = \mathsf{Id}(W)$, we map the atomic pair X_a in P_{v_i} to the atomic pair $Y^{\mathsf{id}}_{a,\phi(a)}$ in T_{v_i}. If $P_{v_i} = \mathsf{Pattern}(W)$ and T_{v_i} is one of $\mathsf{Branch}(a, W)$, $\mathsf{Test}(a, W)$ or $\mathsf{Merge}(a, W)$ then we map X_a in P_{v_i} to the atomic pair $Y^{\delta}_{a,\phi(a)}$ in T_{v_i} and for $b \in W$, we map X_b to $Z^{\delta}_{a,\phi(a),b,\phi(b)}$ for the appropriate choice of $\delta \in \{\mathsf{b}, \mathsf{t}, \mathsf{m}\}$.

First, we need to make sure that the atomic pair $Z^{\delta}_{a,\phi(a),b,\phi(b)}$ for the appropriate choice of $\delta \in \{\mathsf{b}, \mathsf{t}, \mathsf{m}\}$ is well-defined in all of $\mathsf{Branch}(a, W)$, $\mathsf{Test}(a, W)$ and $\mathsf{Merge}(a, W)$. Equivalently, we need to verify that $(v^a_{\phi(a)}, v^b_{\phi(b)}) \in E_a$ which is guaranteed by ϕ.

In order to check the validity of the embedding, it is sufficient to verify that the image of P_{v_i} is an increasing sequence in T_{v_i} and that furthermore, the images of P_{v_i} and $P_{v_{i+1}}$ have the right relative order according to their x-coordinates if P_{v_i} and $P_{v_{i+1}}$ share a common column (i is even), and according to their y-coordinates if they share a common row (i is odd).

The increasing property is checked straightforwardly. Observe that for every $\delta \in \{\mathsf{id}, \mathsf{a}, \mathsf{b}, \mathsf{t}, \mathsf{m}\}$ and $a, b \in [k]$ such that $a < b$, whenever the atomic pairs $Y^{\delta}_{a,i}$ and $Y^{\delta}_{b,j}$ in a single tile T_{v_i} are defined, then $Y^{\delta}_{a,i}$ lies to the left and below of $Y^{\delta}_{b,j}$ regardless of the choices of $i \in [n_a]$ and $j \in [n_b]$. Moreover for a fixed a and $b < b'$, the atomic pair $Z^{\delta}_{a,\phi(a),b,i}$ lies to the left and below $Z^{\delta}_{a,\phi(a),b',j}$ again regardless of the choices of i and j. Finally, every defined atomic pair $Z^{\delta}_{a,\phi(a),b,j}$ lies to the right and above the pair $Y^{\delta}_{a,\phi(a)}$.

In regards to the relative positions of points in different tiles, it is sufficient to verify that the image of X_a in the tile P_{v_i} under ψ sandwiches the image of X_a in the tile $P_{v_{i+1}}$. It is a mechanical task to check that from the definition of ψ and the definitions of the respective tile types.

Correctness ("if"). Suppose there is a 2-left-aligned embedding ψ of π into τ. In particular, ψ maps the image of P_{v_1} to the image of T_{v_1}. The horizontal strip in τ between the anchors in T_{v_1} contains only the tile T_{v_2} and the horizontal strip between the anchors in P_{v_1} in π contains only the tile P_{v_2}. This implies that the guards in P_{v_2} must map to the guards in T_{v_2}. And inductively, the guards in P_{v_i} sandwich the guards in $P_{v_{i+1}}$ and thus, the guards in $P_{v_{i+1}}$ map to the only atomic pairs sandwiched in $T_{v_{i+1}}$ by the guards in T_{v_i} which are the guards in $T_{v_{i+1}}$. These guards then force that the whole embedding is necessarily grid-preserving.

Moreover, the atomic pair X_a in P_{v_2} for every $a \in [k]$ is mapped by ψ to the atomic pair $Y^{\mathsf{a}}_{a,i}$ for some i. Let $\phi\colon [k] \to \mathbb{N}$ be the mapping such that X_a is mapped precisely to $Y^{\mathsf{a}}_{a,\phi(a)}$ for every $a \in [k]$. Clearly, ϕ can be used to define a map that satisfies the coloring property – $\chi(v^a_{\phi(a)}) = a$ for every $a \in [k]$. It remains to show that $\{v^a_{\phi(a)}, v^b_{\phi(b)}\} \in E_H$ for every pair of distinct $a, b \in [k]$.

Claim 4.23. *Let $w = v_i$ for $i \geq 2$ and let $a \in [k]$. The atomic pair X_a in P_w (if defined) is mapped by ψ to the atomic pair $Y^{\delta}_{a,\phi(a)}$ for appropriate δ if T_w is one of $\mathsf{Id}(U)$, $\mathsf{Branch}(a, W)$, $\mathsf{Test}(a, W)$ or $\mathsf{Merge}(a, W)$ where $a \in U$ and $a \notin W$. For $b < a$, X_a in P_w is mapped to the pair $Z^{\delta}_{b,\phi(b),a,\phi(a)}$ if T_w is one of $\mathsf{Branch}(b, U)$, $\mathsf{Test}(b, U)$ or $\mathsf{Merge}(b, U)$.*

Proof. For $w = v_2$ the claim holds by the definition of ϕ. We prove it inductively by the lexicographic order on (a, i).

Suppose that $T_{v_i} = \mathsf{Id}(W)$ and recall that in fact $W = I_b$ for some $b \in [k]$. In such case the tile $T_{v_{i-1}}$ is either the tile Assign or $\mathsf{Merge}(b-1, I_b)$. In the first case, we already know that X_a in $P_{v_{i-1}}$ is mapped to $Y^{\mathsf{a}}_{a,\phi(a)}$ in $T_{v_{i-1}}$. Given the structure of these tiles, the atomic pair X_a in P_{v_i} is forced to map to the atomic pair $Y^{\mathsf{id}}_{a,\phi(a)}$ in T_{v_i} as it is the only pair contained in the horizontal or vertical strip bounded by $Y^{\mathsf{a}}_{a,\phi(a)}$. In the second case, the same holds since $Y^{\mathsf{id}}_{a,\phi(a)}$ is the only pair in T_{v_i} sandwiched by $Z^{\mathsf{m}}_{b,\phi(b),a,\phi(a)}$ from $T_{v_{i-1}}$.

Suppose that T_{v_i} is one of $\mathsf{Branch}(a, W)$, $\mathsf{Test}(a, W)$ or $\mathsf{Merge}(a, W)$ where $a \notin W$. In these cases, we know that X_a in $P_{v_{i-1}}$ maps to $Y^{\delta}_{a,\phi(a)}$ in $T_{v_{i-1}}$ for appropriate δ. And again, this leaves only a single option where X_a from P_{v_i} can be mapped to – the pair $Y^{\eta}_{a,\phi(a)}$ in T_{v_i} for appropriate η.

Suppose that $T_{v_i} = \mathsf{Branch}(b, W)$ for $b < a$ and notice that then $T_{v_{i+1}} = \mathsf{Test}(b, W)$. Using induction on X_a in $P_{v_{i-1}}$ and the structure of $\mathsf{Branch}(b, W)$, we see that X_a in P_{v_i} can be mapped to $Z^{\mathsf{b}}_{b,j,a,\phi(a)}$ for any $j \in [n_b]$. However, this also forces the mapping of X_a in $P_{v_{i+1}}$ to $Z^{\mathsf{t}}_{b,j,a,\phi(a)}$ in $T_{v_{i+1}}$. Since $b < a$, we already know that X_b in $P_{v_{i+1}}$ is mapped to $Y^{\mathsf{t}}_{b,\phi(b)}$ in $T_{v_{i+1}}$ by the induction. And for $j \neq \phi(b)$, the pairs $Y^{\mathsf{t}}_{b,\phi(b)}$ and $Z^{\mathsf{t}}_{b,j,a,\phi(a)}$ form an occurrence of 3412 in $T_{v_{i+1}}$ which cannot happen as $P_{v_{i+1}}$ itself is an increasing sequence. Therefore, X_a is mapped to $Z^{\delta}_{b,\phi(b),a,\phi(a)}$ in both T_{v_i} and $T_{v_{i+1}}$ for appropriate δ. In fact, this forces also the pair X_a in $P_{v_{i+2}}$ to match to the pair $Z^{\mathsf{m}}_{b,\phi(b),a,\phi(a)}$ in $P_{v_{i+2}} = \mathsf{Merge}(a, W)$ which concludes the proof of the claim. □

Now let $a, b \in [k]$ such that $a < b$. As we showed above, the atomic pair X_b is mapped to the atomic pair $Z^{\mathsf{t}}_{a,\phi(a),b,\phi(b)}$ in the tile $\mathsf{Test}(a, I_a)$. That means that $Z^{\mathsf{t}}_{a,\phi(a),b,\phi(b)}$ is well-defined and thus, we have $\{v^a_{\phi(a)}, v^b_{\phi(b)}\} \in E_a$. Recall that we defined G to be the clique on vertex set $[k]$. Therefore $\{a, b\} \in E_G$, and this necessarily implies that $\{v^a_{\phi(a)}, v^b_{\phi(b)}\} \in E_H$. Therefore, we see that (G, H, χ) is a positive instance of PSI as witnessed by the map $\rho\colon V_G \to V_H$ obtained by setting $\rho(a) = v^a_{\phi(a)}$ and thus also (H, χ) is a positive instance of Partitioned Clique.

Tree-width of π. Finally, let us show that $\mathrm{tw}(\pi) \in O(k)$. Let us say that an edge of G_π is *exceptional* if its endpoints share neither the same row nor the same column of the gridding. Only the lowest and highest point of each tile can participate in an exceptional edge. Therefore, there are at most $4k-2$ exceptional edges. Let G' be the graph obtained from G_π by removing all the exceptional edges. It is sufficient to show that $\mathrm{tw}(G') \in O(k)$ as adding the $4k-2$ edges back increases the tree-width at most by $4k-2$.

We define a tree decomposition (T, γ) such that T is a path on $4k-1$ vertices $p_1, \ldots, p_{4k-1}$ and $\gamma(p_i)$ is the image of tiles P_{v_i} and $P_{v_{i+1}}$. Clearly, every point of π lies in a set of bags that induces a connected subtree. Moreover, every edge runs either inside a single cell or between points in two neighboring cells on the path

since any other pair of points would occupy different row and different column. Therefore, we defined a valid tree decomposition of width $O(k)$ and $\mathrm{tw}(G') \in O(k)$. This completes the proof of Lemma 4.22. □

Lemma 4.24. *Let $\mathcal{C}$ be a class with the computable deep tree property. An instance (G, H, χ) of PSI can be reduced to an instance (π, τ) of $\mathcal{C}$-Pattern 2-Left PPM where $|\pi| \in O(|E_G| \cdot \log |E_G|)$ and $|\tau| \in O(|E_H| + |V_H| \cdot |E_G|)$ in time $f(|E_G|) \cdot |V_H|^{O(1)}$ for some function f.*

Proof. We shall modify the previous reduction of Lemma 4.22 similarly to the way we modified the reduction of Theorem 4.15 to prove Theorem 4.18. Once again, we borrow some ideas from the proof of Lemma 2.38.

As before, we set $n = |V_H|$ and $k = |V_G|$. We can furthermore assume that $|E_G| = |V_G| = k$ by Theorem 4.8. Due to the computable deep tree property, we can compute in time $f(k)$ for some computable function f a monotone gridding matrix $\mathcal{M}$ such that

(i) $\mathrm{Grid}(\mathcal{M}) \subseteq \mathcal{C}$,
(ii) the cell graph $G_{\mathcal{M}}$ is a rooted tree T with k leaves,
(iii) the distance of any two vertices in T is $O(\log k)$,
(iv) the root r of the tree T is the single non-empty cell in the leftmost column of $\mathcal{M}$ and it has a single child r',
(v) every non-root vertex with a single child is a corner (its two neighbors in $G_{\mathcal{M}}$ occupy both different rows and different columns),
(vi) every leaf shares a common column with its neighbor, and
(vii) every leaf is at distance at least two from the nearest vertex with degree larger than 1.

The properties (i)–(iii) are guaranteed directly by the computable deep tree property. For (iii)–(vii), it suffices to start with a larger tree and then cut off some branches to achieve the desired shape.

Following along the proof of Lemma 2.38, we orient the edges of $G_{\mathcal{M}}$ consistently away from r and for any vertex v, the descendants of v, denoted by $D(v)$, are all the out-neighbors of v. We arbitrarily order the edges $E_G = \{e_1, \ldots, e_k\}$ and also the k leaves of $G_{\mathcal{M}}$ as $v_1, \ldots, v_k$, and we define the sets A_w exactly as in (2.1) on page 49. We additionally assume that $A_r = [k]$ which corresponds to G having no isolated vertices. We have $\sum_v A_v \in O(k \log k)$.

We call the only neighbor of the leaf v_i a *petiole*, denoted u_i, and we call the set of all vertices that are neither one of r, r' nor the leaves or petioles the *stem*. We proceed to define two families of tiles $\mathcal{P}$ and $\mathcal{T}$. Both P_r and T_r are set to the anchor tile Anchor. The tile $T_{r'}$ is set to be the assignment tile Assign. We set the tile P_v for every vertex v of the tree other than r to the tile $\mathsf{Pattern}(A_v)$. For every v that is part of the stem, we set T_v to be the identity tile $\mathsf{Id}(A_v)$. For $i \in [k]$ such that $e_i = \{a, b\}$ with $a < b$, we set T_{u_i} to be the branch tile $\mathsf{Branch}(a, \{b\})$ and we set T_{v_i} to be the test tile $\mathsf{Test}(a, \{b\})$.

By Lemma 1.2, there is a consistent orientation $\mathcal{F}$ of $\mathcal{M}$. We set π to be the modified $\mathcal{F}$-assembly of $\mathcal{P}$ and τ to be the modified $\mathcal{F}$-assembly of $\mathcal{T}$. The total size of $\mathcal{P}$ is clearly $O(k \log k)$. The size of all tiles in $\mathcal{T}$ corresponding to leaves and petioles is $O(|E_H|)$ and each one of the remaining $O(k)$ tiles contains at most $O(n)$ points which gives the desired bounds on the sizes of π and τ.

Correctness ("only if"). The correctness essentially follows the same arguments as in the case of the long path property. Suppose that there is a mapping $\phi\colon [k] \to \mathbb{N}$ witnessing that (G, H, χ) is a positive instance of PSI. We define a grid-preserving mapping ψ of π into τ. In the Assign tile, we map X_a to $Y^{\mathsf{a}}_{a,\phi(a)}$ and in each identity tile $\mathsf{Id}(W)$ where $a \in W$, we map X_a to $Y^{\mathsf{id}}_{a,\phi(a)}$. In the petiole u_i and the leaf v_i such that $e_i = \{a, b\}$ with $a < b$, the mapping ψ sends X_a to $Y^{\delta}_{a,\phi(a)}$ and X_b to $Z^{\delta}_{a,\phi(a)b,\phi(b)}$ for suitable δ. Observe that $Z^{\delta}_{a,\phi(a)b,\phi(b)}$ is well-defined since $\{v^a_{\phi(a)}, v^b_{\phi(b)}\} \in E_H$. The same arguments as before show that ψ is indeed a 2-left-aligned embedding of π into τ.

Correctness ("if"). Suppose there is a 2-left-aligned embedding ψ of π into τ. Again, ψ maps the tile P_{v_1} to the tile T_{v_1}. Following the guards from the root, we see that ψ must be grid-preserving and as before, we define the mapping $\phi\colon [k] \to \mathbb{N}$ such that the pair X_a in $P_{r'}$ maps to $Y^{\mathsf{a}}_{a,\phi(a)}$ in $T_{r'}$. It inductively follows that in any vertex of the stem, X_a must be mapped to $Y^{\mathsf{id}}_{a,\phi(a)}$ (assuming $a \in W$). This holds in particular for the parent of each petiole u_i. Consequently, the petiole u_i and leaf v_i can be seen as applying the reduction from Lemma 4.22 to check the subgraph property for the single edge $\{a, b\}$ in the subgraph of H induced by $V_a \cup V_b$. Thus, the same arguments show that $\{v^a_{\phi(a)}, v^b_{\phi(b)}\} \in E_H$ and (G, H, χ) is a positive instance of PSI. □

Proof of Proposition 4.20. Lemmas 4.22 and 4.24 imply the $\mathsf{W}[1]$-hardness and conditional lower bounds. The $\mathsf{W}[1]$-membership of $\mathcal{C}$-Pattern 2-Left PPM follows trivially from the $\mathsf{W}[1]$-membership of 2-Left PPM. □

Pattern parameters

As a first consequence of our results involving $\mathcal{C}$-Pattern 2-Left PPM, we can show conditional lower bounds for $\mathcal{C}$-Pattern PPM with respect to various parameters of the pattern. In particular, we can exhibit lower bounds with respect to any parameter that is invariant under monotone inflations.

Corollary 4.25. *Suppose that* w *is a permutation parameter invariant under monotone inflations. Unless ETH fails, $\mathcal{C}$-Pattern PPM cannot be solved for any function f*

- *in time $f(\mathrm{w}(\pi)) \cdot n^{o\left(\sqrt{\mathrm{w}(\pi)}\right)}$ if $\mathcal{C}$ has the computable long path property, and*
- *in time $f(\mathrm{w}(\pi)) \cdot n^{o\left(\mathrm{w}(\pi)/\log^2 \mathrm{w}(\pi)\right)}$ if $\mathcal{C}$ has the computable deep tree property,*

where π is the pattern and n is the length of the text. Moreover, $\mathcal{C}$-Pattern PPM is $\mathsf{W}[1]$-hard with respect to $\mathrm{w}(\pi)$ in both these cases.

Proof. Let (π, τ) be the instance of C-Pattern 2-Left PPM produced by Lemma 4.22 or 4.24. We define π' as the permutation obtained from π by inflating both of its anchors (the two leftmost points) with either two increasing or decreasing sequences of length $|\tau|$ such that π' is still contained in $\mathcal{C}$. Recall that our reductions guarantee that one of these inflations is always possible. And similarly, we let τ' be the permutation obtained from τ by inflating both of its anchors (the two leftmost points) with the same type of monotone sequences of length $|\tau|$ as in π'.

At this point, we have used this 'inflation trick' many times in our reductions. Therefore, we consider it obvious that π' is contained in τ' if and only if there is a 2-left-aligned embedding of π into τ. It remains to observe that $\mathrm{w}(\pi') \in O(|\pi|)$ since w is invariant under monotone inflations. The respective lower bounds follow. □

However, Lemma 4.22 produces instances with patterns whose tree-width is bounded by $O(k)$ instead of the trivial $O(k^2)$ guaranteed for any parameter bounded by the length of permutation. Therefore, it follows using the very same argument that $\mathcal{C}$-Pattern PPM cannot be solved in time $f(\mathrm{tw}(\pi)) \cdot n^{o(\mathrm{tw}(\pi))}$ for any class $\mathcal{C}$ with the computable long path property. Consequently, the algorithm of Theorem 4.12 is asymptotically the best possible, even when we restrict the patterns to a permutation class with the computable long path property.

Theorem 4.26. *If $\mathcal{C}$ has the computable long path property then $\mathcal{C}$-Pattern PPM cannot be solved in time $f(\mathrm{tw}(\pi)) \cdot n^{o(\mathrm{tw}(\pi))}$ where π is the pattern and n is the length of the text for any function f, unless ETH fails.*

Counting patterns

Having the conditional lower bounds for $\mathcal{C}$-Pattern 2-Left PPM, we can obtain lower bounds for $\mathcal{C}$-Pattern #PPM simply by mimicking our proof showing the hardness of #PPM in general.

Theorem 4.27. *Unless ETH fails, $\mathcal{C}$-Pattern #PPM cannot be solved for any function f*

- *in time $f(k) \cdot n^{o\left(\sqrt{k}\right)}$ if $\mathcal{C}$ has the computable long path property, and*
- *in time $f(k) \cdot n^{o\left(k/\log^2 k\right)}$ if $\mathcal{C}$ has the computable deep tree property,*

where k is the length of the pattern and n is the length of the text.

Proof. Similar to the proof of Theorem 4.6, we show that an algorithm solving $\mathcal{C}$-Pattern #PPM in time $f(k) \cdot n^{h(k)}$ for some functions f and h could be used to design an algorithm solving the counting version of $\mathcal{C}$-Pattern 2-Left PPM in time $g(k) \cdot n^{h(k)}$. The desired conditional lower bounds follow from Proposition 4.7.

Suppose (π, τ) is an instance of Left PPM. Recall that $occ(\pi, \tau)$ denotes the number of occurrences of π in τ. In order to count the number of 2-left-aligned embeddings, it is sufficient to count the number of embeddings of π into τ that hit the two leftmost points in τ. We can do this easily by the inclusion-exclusion principle. For a set $S \subseteq [n]$, let τ^{-S} be the permutation obtained from τ by removing every point (i, τ_i) for $i \in S$. The total number of 2-left-aligned embeddings can be expressed as

$$occ(\pi, \tau) - occ(\pi, \tau^{-\{1\}}) - occ(\pi, \tau^{-\{2\}}) + occ(\pi, \tau^{-\{1,2\}}).$$

In other words, we can compute the number of 2-left-aligned occurrences by invoking the algorithm for $\mathcal{C}$-Pattern #PPM four times. Therefore, we would obtain an algorithm solving $\mathcal{C}$-Pattern 2-Left PPM in time $4 \cdot f(k) \cdot n^{h(k)}$. □

Generalized patterns

As our last application, we can lift the hardness of $\mathcal{C}$-Pattern 2-Left PPM to show that similar conditional lower bounds hold when we replace 2-left-aligned patterns with vincular, covincular, bivincular or mesh patterns. Note that it is sufficient to prove the hardness of $\mathcal{C}$-Pattern Vincular PPM. The equivalent result for $\mathcal{C}$-Pattern Covincular PPM follows since the long path and deep tree properties are preserved under rotations by 90°, and $\mathcal{C}$-Pattern Bivincular PPM and $\mathcal{C}$-Pattern Mesh PPM can be seen as mere generalizations of $\mathcal{C}$-Pattern Vincular PPM.

Corollary 4.28. *Unless ETH fails, $\mathcal{C}$-Pattern Vincular PPM cannot be solved for any function f*

- *in time $f(k) \cdot n^{o(\sqrt{k})}$ if $\mathcal{C}$ has the computable long path property, and*
- *in time $f(k) \cdot n^{o(k/\log^2 k)}$ if $\mathcal{C}$ has the computable deep tree property,*

where k is the length of the pattern and n is the length of the text. Moreover, in both cases these problems are W[1]-complete with respect to k.

Proof. We reduce a $\mathcal{C}$-Pattern 2-Left PPM instance (π, τ) to an equivalent instance $((\pi, \{0, 1\}), \tau)$ of $\mathcal{C}$-Pattern Vincular PPM. The rest follows from Proposition 4.20. □

5. Pattern matching inside a fixed permutation class

In this chapter, we examine how the hardness of PPM changes when we restrict both pattern and text to a fixed permutation class. For a class $\mathcal{C}$, we formally introduce the problem $\mathcal{C}$-Permutation Pattern Matching.

$\mathcal{C}$-Permutation Pattern Matching ($\mathcal{C}$-PPM)	
Input:	Permutations $\pi \in \mathcal{C}$ of length k and $\tau \in \mathcal{C}$ of length n.
Output:	Is π contained in τ?

After summarizing the previous knowledge in Section 5.1, we show that $\mathcal{C}$-PPM is polynomial-time solvable for any subclass of a monotone grid class in Section 5.2. Then we provide an accompanying intractability results in Section 5.3. First, we show that $\mathcal{C}$-PPM is NP-complete under a suitable technical assumption on $\mathcal{C}$ and subsequently, we prove that most principal classes satisfy this assumption and thus, the corresponding Av(σ)-PPM problem is NP-complete.

5.1 Current state of the art

We now summarize all the previous knowledge on the complexity of $\mathcal{C}$-PPM. Unsurprisingly, most previous results concern Av(σ)-PPM for various principal classes Av(σ). First, we focus on the cases that were previously known to be polynomial-time solvable.

Proposition 5.1 ([10, 101])**.** Av(σ)*-PPM is polynomial-time solvable for any* σ *of length at most 3.*

For σ symmetric to 21 or 132, it follows from the fact that Av(σ) is a subclass of separable permutations and thus, even Av(σ)-Pattern PPM is polynomial-time solvable by Theorem 4.12. Moreover, Neou et al. [101] designed a more efficient algorithm deciding $\mathcal{S}$-PPM where $\mathcal{S}$ is the class of separable permutations in time $O(n^2k)$. The only remaining case (up to symmetry) is when σ is equal to 321. The first polynomial algorithm deciding Av(321)-PPM was designed by Guillemot and Vialette [78]. Later, Albert et al. [10] showed that Av(321)-PPM can be solved in time $O(nk)$ and the same holds for the class of skew-merged permutations. Notably, this case highlights the difference between $\mathcal{C}$-Pattern PPM and $\mathcal{C}$-PPM since Av(321)-Pattern PPM is NP-complete due to Theorem 4.15.

On the other hand, Jelínek and Kynčl [86] proved that Av(4321)-PPM is NP-complete and the same is true for the class of permutations obtained as a union of one decreasing and two increasing sequences.

Theorem 5.2 ([86])**.** Av(4321)*-PPM and (*Av(12) $\odot$ Av(321)*)-PPM are both NP-complete.*

Combining Theorem 5.2 with the Erdős–Szekeres theorem [65], it follows that Av(σ)-PPM is NP-complete for any σ of length at least 10. However, this still leaves a sizable gap in our knowledge of the polynomial cases of Av(σ)-PPM. Our main contribution is that in Section 5.3, we narrow this gap to only 5 unresolved cases up to symmetry.

5.2 Monotone-griddable classes

A vital role in the rest of this chapter is played by the concept of monotone griddability. We say that a class $\mathcal{C}$ is *monotone-griddable* if there exists a monotone gridding matrix $\mathcal{M}$ such that $\mathcal{C}$ is contained in $\mathrm{Grid}(\mathcal{M})$. Huczynska and Vatter [83] provided a neat and useful characterization of monotone-griddable classes.

We define the *sum completion* of a permutation π to be the permutation class

$$\oplus\pi = \{\sigma_1 \oplus \sigma_2 \oplus \cdots \oplus \sigma_k \mid \sigma_i \preceq \pi \text{ for all } i \leq k \in \mathbb{N}\}.$$

Analogously, we define the *skew completion* $\ominus\pi$ of π. The class $\oplus 21$ is known as the *Fibonacci class*.

Theorem 5.3 ([83])**.** *A permutation class $\mathcal{C}$ is monotone-griddable if and only if it contains neither the Fibonacci class $\oplus 21$ nor its symmetry $\ominus 12$.*

We say that a permutation π is *t-monotone* if there is a partition $\Pi = (S_1, \ldots, S_t)$ of S_π such that S_i is a monotone point set for each $i \in [t]$. The partition Π is called a *t-monotone partition*.

Given a t-monotone partition $\Pi = (S_1, \ldots, S_t)$ of a permutation π and a t-monotone partition $\Sigma = (S'_1, \ldots, S'_t)$ of τ, an embedding ϕ of π into τ is a (Π, Σ)-*embedding* if $\phi(S_i) \subseteq S'_i$ for every $i \in [t]$. Less formally, it is an embedding that respects the partitions Π and Σ. Guillemot and Marx [77] showed that if we fix t-monotone partitions of both π and τ, the problem of finding a (Π, Σ)-embedding is polynomial-time solvable.

Proposition 5.4 ([77])**.** *Given a permutation π of length m with a t-monotone partition Π and a permutation τ of length n with a t-monotone partition Σ, we can decide if there is a (Π, Σ)-embedding of π into τ in time $O(m^2n^2)$.*

We can combine this result with the fact that there is only a bounded number of ways how to grid a permutation, and obtain a polynomial algorithm for $\mathcal{C}$-PPM.

Theorem 5.5. *$\mathcal{C}$-PPM is polynomial-time solvable for any monotone-griddable class $\mathcal{C}$.*

Proof. Let $\mathcal{M}$ be a $k \times \ell$ monotone gridding matrix such that $\mathrm{Grid}(\mathcal{M})$ contains the class $\mathcal{C}$. We have to decide whether π is contained in τ for two given permutations π of length m and τ of length n, both belonging to the class $\mathcal{C}$.

First, we find an $\mathcal{M}$-gridding of τ. We enumerate all possible $k \times \ell$ griddings and for each, we test if it is a valid $\mathcal{M}$-gridding. Observe that there are in total $O(n^{k+\ell-2})$ such griddings since they are determined by two sequences of values from the set $[n]$, one of length $k-1$ and the other of length $\ell - 1$. Moreover, it is straightforward to test in time $O(n^2)$ whether a given $k \times \ell$ gridding is in fact an $\mathcal{M}$-gridding. Note that we are guaranteed to find an $\mathcal{M}$-gridding as τ belongs to $\mathcal{C} \subseteq \mathrm{Grid}(\mathcal{M})$. We set Σ to be the $(k \cdot \ell)$-monotone partition of τ into the monotone sequences given by the individual cells of the gridding.

In the second step, we enumerate all possible $\mathcal{M}$-griddings of π. As with τ, we enumerate all possible $O(m^{k+\ell-2})$ $k \times \ell$ griddings of π and check for each gridding whether it is actually an $\mathcal{M}$-gridding in time $O(m^2)$. For each $\mathcal{M}$-gridding found,

we let Π be the $(k \cdot \ell)$-monotone partition of π given by the gridding, and we apply Proposition 5.4 to test whether there is a (Π, Σ)-embedding in time $O(m^2 n^2)$.

If there is an embedding ϕ of π into τ, there is a $(k \cdot \ell)$-monotone partition Σ' of π such that ϕ is a (Π, Σ')-embedding. Therefore, the algorithm correctly solves $\mathcal{C}$-PPM in time $O(n^{k+\ell} + m^{k+\ell} n^2)$ — polynomial in n, m. □

Notice that if $\mathcal{M}$ is a gridding matrix whose every entry is monotone-griddable, or equivalently no entry contains the Fibonacci class or its reverse as a subclass, then the class Grid($\mathcal{M}$) is monotone-griddable as well. It follows that for such $\mathcal{M}$, the Grid($\mathcal{M}$)-PPM problem is in P. In Chapter 6 (Theorem 6.8), we will see that if a gridding matrix $\mathcal{M}$ has an acyclic cell graph, and if every nonempty cell is either monotone or symmetric to a Fibonacci class, then Grid($\mathcal{M}$)-Pattern PPM, and therefore also Grid($\mathcal{M}$)-PPM, is polynomial-time solvable as well. These two tractability results contrast with Proposition 5.14, which implies that for any gridding matrix $\mathcal{M}$ whose cell graph is a cycle, and whose nonempty cells are all monotone except for one Fibonacci cell, Grid($\mathcal{M}$)-PPM is already NP-hard.

5.3 Hardness results

In this section, we present the main technical hardness result and then derive its several corollaries. Similar to the case of the pattern-restricted variant of PPM, we first exhibit in Subsection 5.3.2 the hardness of $\mathcal{C}$-PPM assuming that $\mathcal{C}$ satisfies a certain technical property and then we show in Subsection 5.3.3 that this property, in fact, holds for all principal classes Av(σ) for σ of length at least 4 and not symmetric to any of $3412, 3142, 4213, 4123$ or 41352.

We say that a permutation class $\mathcal{C}$ has the *$\mathcal{D}$-rich path property* for a class $\mathcal{D}$ if there is a positive constant ϵ such that for every k, the class $\mathcal{C}$ contains a grid subclass whose cell graph is a proper-turning path of length k with at least $\epsilon \cdot k$ entries equal to $\mathcal{D}$. Moreover, we say that $\mathcal{C}$ has the *computable $\mathcal{D}$-rich path property*, if $\mathcal{C}$ has the $\mathcal{D}$-rich path property and there is an algorithm that, for a given k, outputs a witnessing proper-turning path of length k with at least $\epsilon \cdot k$ copies of $\mathcal{D}$ in time polynomial in k.

Theorem 5.6. *Let $\mathcal{C}$ be a permutation class with the computable $\mathcal{D}$-rich path property for a non-monotone-griddable class $\mathcal{D}$. Then $\mathcal{C}$-PPM is NP-complete, and unless ETH fails, there can be no algorithm that solves $\mathcal{C}$-PPM*

- *in time $2^{o(n/\log n)}$ if $\mathcal{D}$ moreover contains any monotone juxtaposition,*
- *in time $2^{o(\sqrt{n})}$ otherwise.*

Let us remark that the two lower bounds we obtained under ETH are close to optimal. It is clear that the bound of $2^{o(n/\log n)}$ matches, up to the $\log n$ term in the exponent, the trivial $2^{O(n)}$ brute-force algorithm for PPM. Moreover, the lower bound of $2^{o(\sqrt{n})}$ for $\mathcal{C}$-PPM also cannot be substantially improved without adding assumptions about the class $\mathcal{C}$. Consider for instance the class $\mathcal{C} = \text{Grid}\begin{pmatrix} \diagdown & \diagup\!\diagdown \\ \oplus 21 & \diagdown \end{pmatrix}$. As we shall see in Proposition 5.14, this class has the computable $\oplus 21$-rich path property, and therefore the $2^{o(\sqrt{n})}$ conditional lower bound applies to it. However, we can show that $\text{tw}_{\mathcal{C}}(n) \in O(\sqrt{n})$ using similar ideas as in the proof of

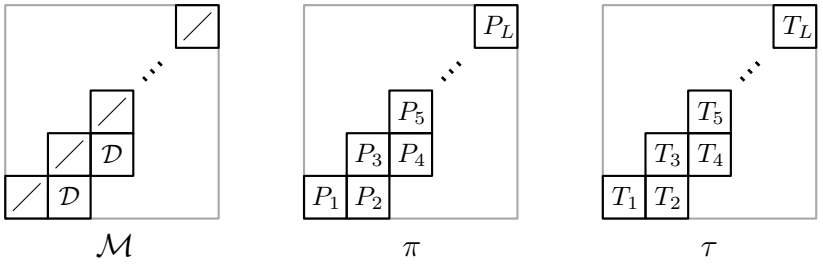

Figure 5.1: The gridding matrix $\mathcal{M}$, the gridded permutation π (the pattern) and the gridded permutation τ (the text), used in the simplified overview of the proof of Theorem 5.6.

Proposition 6.5 and thus, we can solve $\mathcal{C}$-PPM (even $\mathcal{C}$-Pattern PPM) in time $n^{O(\sqrt{n})}$ using the algorithm of Theorem 4.12. We omit the details of the argument here.

5.3.1 Overview of the proof of Theorem 5.6

The proof of Theorem 5.6 is based on a reduction from 3-SAT. The individual steps of the construction are rather technical, and we therefore begin with a general overview of the reduction. In Subsection 5.3.2, we then present the reduction in full detail, together with the proof of correctness.

Suppose that $\mathcal{C}$ is a class with the computable $\mathcal{D}$-rich path property, where $\mathcal{D}$ is not monotone-griddable. Theorem 5.3 implies that $\mathcal{D}$ contains the Fibonacci class $\oplus 21$ or its reversal $\ominus 12$ as subclass. Suppose then, without loss of generality, that $\mathcal{D}$ contains $\oplus 21$.

To reduce 3-SAT to $\mathcal{C}$-PPM, consider a 3-CNF formula φ, with n variables $x_1, \ldots, x_n$ and m clauses $K_1, \ldots, K_m$. Once again we assume, without loss of generality, that each clause contains exactly three literals, and no variable appears in a single clause more than once.

Let $L = L(m, n)$ be an integer whose value will be specified later. By the $\mathcal{D}$-rich path property, $\mathcal{C}$ contains a grid subclass Grid($\mathcal{M}$) where the cell graph of $\mathcal{M}$ is a proper-turning path of length L, in which a constant fraction of cells is equal to $\mathcal{D}$.

To simplify our notation in this high-level overview, we will assume that the cell graph of $\mathcal{M}$ corresponds to an increasing staircase. More precisely, the cells of $\mathcal{M}$ representing infinite classes can be arranged into a sequence $C_1, C_2, \ldots, C_L$, where C_1 is the bottom-left cell $\mathcal{M}_{1,1}$ of $\mathcal{M}$, each odd-numbered cell C_{2i-1} corresponds to the diagonal cell $\mathcal{M}_{i,i}$, and each even numbered cell C_{2i} corresponds to $\mathcal{M}_{i+1,i}$. All the remaining cells of $\mathcal{M}$ are empty. To simplify the exposition even further, we will assume that each odd-numbered cell of the path is equal to $\boxslash$ and each even-numbered cell is equal to $\mathcal{D}$. See Figure 5.1.

With the gridding matrix $\mathcal{M}$ specified above, we will construct two $\mathcal{M}$-gridded permutations, the pattern π and the text τ both equipped with a pair of anchors, such that there is an anchored embedding o π into τ if and only if the formula φ is satisfiable. Afterwards, we remove the assumption of anchored embeddings via the usual trick of inflating the anchors with sufficiently long monotone sequences. We will describe π and τ geometrically, as permutation diagrams, which are partitioned

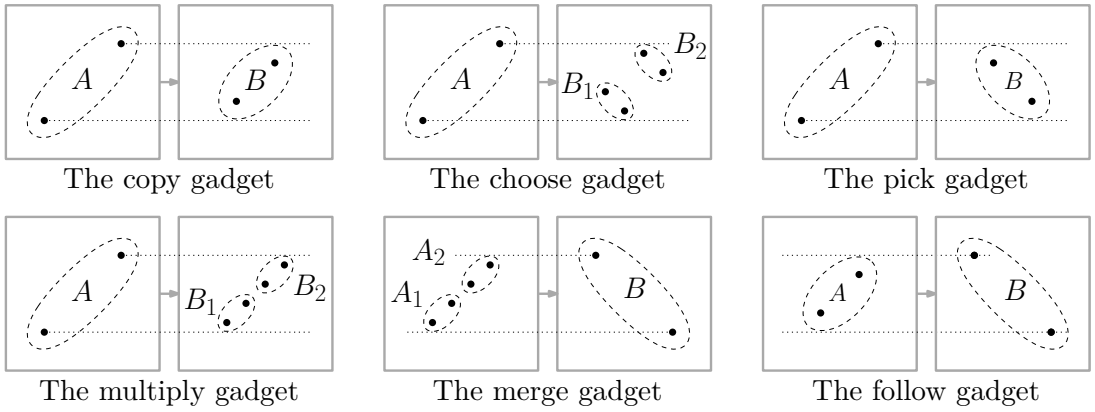

Figure 5.2: The constructions of simple gadgets. The tile Q_i is always on the left and the tile Q_{i+1} is on the right. The dotted lines show the relative vertical order of points.

into blocks by the $\mathcal{M}$-gridding. We let P_i denote the part of π corresponding to the cell C_i of $\mathcal{M}$, and similarly we let T_i be the part of τ corresponding to C_i.

To get an intuitive understanding of the reduction, it is convenient to first restrict our attention to *grid-preserving embeddings* of π into τ, that is, to embeddings which map the elements of P_i to elements of T_i for each i. Note that this assumption can later be easily handled by adding two tracks of guarding atomic pairs to both pattern and text as in the proof of Theorem 4.15.

The basic building blocks in the description of π and τ are again atomic pairs. It is a feature of the construction that in any grid-preserving embedding of π into τ, an atomic pair inside a pattern block P_i is mapped to an atomic pair inside the corresponding text block T_i. Moreover, each atomic pair in π or τ is associated with one of the variables $x_1, \ldots, x_n$ of φ, and any grid-preserving embedding will maintain the association, that is, atomic pairs associated to a variable x_j inside π will map to atomic pairs associated to x_j in τ.

To describe π and τ, we need to specify the relative positions of the atomic pairs in two adjacent blocks P_i and P_{i+1} (or T_i and T_{i+1}). These relative positions are given by several typical configurations, which we call *gadgets*. Several examples of gadgets are depicted in Figure 5.2. In the figure, the pairs of points enclosed by an ellipse are atomic pairs. The choose, multiply and merge gadgets are used in the construction of τ, while the pick and follow gadgets are used in π. The copy gadget will be used in both. We also need more complicated gadgets, namely the *flip gadgets* of Figure 5.3, which span more than two consecutive blocks. In all cases, the atomic pairs participating in a single gadget are all associated to the same variable of φ.

The sequence of pattern blocks $P_1, P_2, \ldots, P_L$, as well as their corresponding text blocks $T_1, \ldots, T_L$, is divided into several contiguous parts, which we call *phases*. We now describe the individual phases in the order in which they appear.

The initial phase and the assignment phase. The initial phase involves two pattern blocks P_1, P_2 and the corresponding text blocks T_1, T_2. Both P_1 and T_1 consist of single atomic pair forming the anchors. The blocks P_2 and T_2 consist of an increasing sequence of $2n$ points, sandwiched by the anchor of the respective first block and divided into n consecutive atomic pairs $X_1^0, X_2^0, \ldots, X_n^0 \subseteq P_1$ and

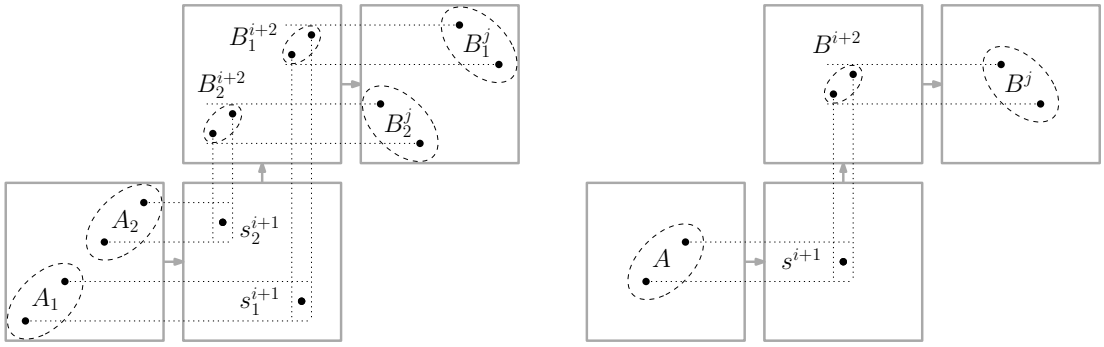

Figure 5.3: A flip text gadget on the left and a flip pattern gadget on the right. The first tile pictured is Q_i and the last tile is Q_j where $j = i + 3$. As before, the dotted lines show the relative order of points.

$Y_1^0, Y_2^0, \ldots, Y_n^0 \subseteq T_1$, numbered in increasing order. The pairs X_j^0 and Y_j^0 are both associated to the variable x_j. Clearly, any embedding of P_1 into T_1 will map the pair X_j^0 to the pair Y_j^0, for each $j \in [n]$.

The initial phase is followed by the assignment phase, which also involves only one pattern block P_3 and the corresponding text block T_3. P_3 will consist of an increasing sequence of n atomic pairs $X_1^1, X_2^1, \ldots, X_n^1$, where each X_j^1 is a decreasing pair, i.e., a copy of 21. Moreover, $X_j^0 \cup X_j^1$ forms the pick gadget, so the first two pattern blocks can be viewed as a sequence of n pick gadgets stacked on top of each other.

The block T_3 then consists of $2n$ atomic pairs $\{Y_j^1, Z_j^1;\ j \in [n]\}$, positioned in such a way that $Y_j^0 \cup Y_j^1 \cup Z_j^1$ is a choose gadget. Thus, $T_2 \cup T_3$ is a sequence of n choose gadgets stacked on top of each other, each associated with one of the variables of φ.

In a grid-preserving embedding of π into τ, each pick gadget $X_j^0 \cup X_j^1$ must be mapped to the corresponding choose gadget $Y_j^0 \cup Y_j^1 \cup Z_j^1$, with X_j^0 mapped to Y_j^0, and X_j^1 mapped either to Y_j^1 or to Z_j^1. There are thus 2^n grid-preserving embeddings of $P_2 \cup P_3$ into $T_2 \cup T_3$, and these embeddings encode in a natural way the 2^n assignments of truth values to the variables of φ. Specifically, if X_j^1 is mapped to Y_j^1, we will say that x_j is false, while if X_j^1 maps to Z_j^1, we say that x_j is true. The aim is to ensure that an embedding of $P_2 \cup P_3$ into $T_2 \cup T_3$ can be extended to an embedding of π into τ if and only if the assignment encoded by the embedding satisfies φ.

Each atomic pair that appears in one of the text blocks $T_3, T_4, \ldots, T_L$ is not only associated with a variable of φ, but also with its truth value; that is, there are 'true' and 'false' atomic pairs associated with each variable x_j. The construction of π and τ ensures that in an embedding of π into τ in which X_j^1 is mapped to Y_j^1 (corresponding to setting x_j to false), all the atomic pairs associated to x_j in the subsequent stages of π will map to false atomic pairs associated to x_j in τ, and conversely, if X_j^1 is mapped to Z_j^1, then the atomic pairs of π associated to x_j will only map to the true atomic pairs associated to x_j in τ.

The multiplication phase. The purpose of the multiplication phase is to 'duplicate' the information encoded in the assignment phase. Without delving into the technical details, we describe the end result of the multiplication phase

and its intended behaviour with respect to embeddings. Let d_j be the number of occurrences (positive or negative) of the variable x_j in φ. Note that $d_1 + d_2 + \cdots + d_n = 3m$, since φ has m clauses, each of them with three literals. Let P_k and T_k be the final pattern block and text block of the multiplication phase. Then P_k is an increasing sequence of $3m$ increasing atomic pairs, among which there are d_j atomic pairs associated to x_j. Moreover, the pairs are ordered in such a way that the d_1 pairs associated to x_1 are at the bottom, followed by the d_2 pairs associated to x_2 and so on. The structure of T_k is similar to P_k, except that T_k has $6m$ atomic pairs. In fact, we may obtain T_k from P_k by replacing each atomic pair $X_i^k \subseteq P_k$ associated to a variable x_j by two adjacent atomic pairs Y_i^k, Z_i^k, associated to the same variable, where Y_i^k is false and Z_i^k is true.

It is useful to identify each pair $X_i^k \subseteq P_k$ as well as the corresponding two pairs $Y_i^k, Z_i^k \subseteq T_k$ with a specific occurrence of x_j in φ. Thus, each literal in φ is represented by one atomic pair in P_k and two adjacent atomic pairs of opposite truth values in T_k.

The blocks $P_4, \ldots, P_k$ and $T_4, \ldots, T_k$ are constructed in such a way that any embedding of π into τ that encodes an assignment in which x_j is false has the property that all the atomic pairs in P_k associated to x_j are mapped to the false atomic pairs of T_k associated to x_j, and similarly, when x_j is encoded as true in the assignment phase, the pairs of P_k associated to x_j are only mapped to the true atomic pairs of T_k. Thus, the mapping of any atomic pair of P_k encodes the information on the truth assignment of the associated variable.

The multiplication phase is implemented by a combination of multiply gadgets and flip text gadgets in τ, and copy gadgets and flip pattern gadgets in π. It requires no more than $O(\log m)$ blocks in π and τ, i.e., $k = O(\log m)$.

The sorting phase. The purpose of the sorting phase is to rearrange the relative positions of the atomic pairs. While at the end of the multiplication phase, the pairs representing occurrences of the same variable appear consecutively, after the sorting phase, the pairs representing literals belonging to the same clause will appear consecutively. More precisely, letting P_ℓ and T_ℓ denote the last pattern block and the last text block of the sorting phase, P_ℓ has the same number of atomic pairs associated to a given variable x_j as P_k, and similarly for T_ℓ and T_k. For each clause K_j, P_ℓ contains three consecutive atomic pairs corresponding to the three literals in K_j, and T_ℓ contains the corresponding six atomic pairs, again appearing consecutively. Similarly as in P_k and T_k, each atomic pair in P_ℓ must map to an atomic pair in T_ℓ representing the same literal and having the correct truth value encoded in the assignment phase.

To prove Theorem 5.6, we need two different ways of implementing the sorting phase, depending on whether the class $\mathcal{D}$ contains a monotone juxtaposition or not. The first construction, which we call *sorting by gadgets*, does not put any extra assumptions on $\mathcal{D}$. However, it may require up to $\Theta(m)$ blocks to perform the sorting, that is $\ell = \Theta(m)$.

The other implementation of the sorting phase, which we call *sorting by juxtapositions* is only applicable when $\mathcal{D}$ contains a monotone juxtaposition, and it can be performed with only $O(\log m)$ blocks. The difference between the lengths of the two versions of sorting is the reason for the two different lower bounds in Theorem 5.6.

The evaluation phase. The final phase of the construction is the evaluation phase. The purpose of this phase is to ensure that for any embedding of π into τ, the truth assignment encoded by the embedding satisfies all the clauses of φ. For each clause K_j, we attach suitable gadgets to the atomic pairs in P_ℓ and T_ℓ representing the literals of K_j. These gadgets force that K_j is satisfied via Observation 4.2. Using the fact that the atomic pairs representing the literals of a given clause are consecutive in P_j and T_j, this can be done for all the clauses simultaneously, with only $O(1)$ blocks in π and τ. This completes an overview of the hardness reduction proving Theorem 5.6.

When the reduction is performed with sorting by gadgets, it produces permutations π and τ of size $O(m^2)$, since we have $L = O(m)$ blocks and each block has size $O(m)$. When sorting is done by juxtapositions, the number of blocks drops to $L = O(\log m)$, hence π and τ have size $O(m \log m)$. The lower bounds from Theorem 5.6 follow by the clause form of ETH (Hypothesis 2).

The details of the reduction, as well as the full correctness proof, are presented in the following subsections.

5.3.2 Details of the hardness reduction

Our job is to construct a pair of permutations π and τ, both having a gridding corresponding to a $\mathcal{D}$-rich path, with the property that the anchored embeddings of π into τ will simulate satisfying assignments of a given 3-SAT formula. Once again, we shall construct both π and τ via an $\mathcal{F}$-assembly of tile families.

We describe a reduction from 3-SAT to $\mathcal{C}$-PPM. Let φ be a given 3-CNF formula with n variables $x_1, x_2, \ldots, x_n$ and m clauses $K_1, K_2, \ldots, K_m$. As already mentioned, we again assume, without loss of generality, that each clause contains exactly three literals, and no variable appears in a single clause more than once. We will construct permutations $\pi, \tau \in \mathcal{C}$ each containing an anchoring pair of points such that φ is satisfiable if and only if there is an anchored embedding of π into τ.

Let $\mathcal{M}$ be a $g \times h$ gridding matrix such that $\mathrm{Grid}(\mathcal{M})$ is a subclass of $\mathcal{C}$, the cell graph $G_{\mathcal{M}}$ is a proper-turning path of length $L = L(m, n)$ to be determined later, a constant fraction of its entries is equal to $\mathcal{D}$, and the remaining entries of the path are monotone. We aim to construct π and τ such that they both belong to $\mathrm{Grid}(\mathcal{M})$.

First, we label the vertices of the path as $v_1, v_2, \ldots, v_L$ choosing the direction such that at least half of the $\mathcal{D}$-entries share a row with their predecessor. We replace all other $\mathcal{D}$-entries with their monotone subclasses. Note that at least half of the $\mathcal{D}$-entries survived this modification and thus, the path still contains a constant fraction of $\mathcal{D}$-entries. We claim that there is a $g \times h$ orientation $\mathcal{F}$ such that the class $\mathcal{F}(\mathcal{M})_{i,j}$ is equal to $\boxslash$ for every monotone entry $\mathcal{M}_{i,j}$ and the class $\mathcal{F}(\mathcal{M})_{i,j}$ contains $\oplus 21$ for every $\mathcal{D}$-entry $\mathcal{M}_{i,j}$. To see this, consider a gridding matrix $\mathcal{M}'$ obtained by replacing every $\mathcal{D}$-entry in $\mathcal{M}$ with $\boxslash$ if $\mathcal{D}$ contains $\oplus 21$ or with $\boxbslash$ if $\mathcal{D}$ contains $\ominus 12$. It then suffices to apply Lemma 1.2 on $\mathcal{M}'$.

Our plan is to simultaneously construct two $g \times h$ families of $(2n + 2)$-tiles $\mathcal{P}$ and $\mathcal{T}$ and then set π and τ to be the $\mathcal{F}$-assemblies of $\mathcal{P}$ and $\mathcal{T}$, respectively. We abuse the notation and for any $g \times h$ family of tiles $\mathcal{Q}$ (in particular for $\mathcal{P}$

and $\mathcal{T}$), we use Q_i instead of Q_{v_i} to denote the tile corresponding to the i-th cell of the path.

For now, we will only consider restricted embeddings that respect the partition into tiles and deal with arbitrary embeddings later. Let π be an $\mathcal{F}$-assembly of $\mathcal{P}$ and τ an $\mathcal{F}$-assembly of $\mathcal{T}$; we then say that an embedding of π into τ is *grid-preserving* if the image of tile $P_{i,j}$ is mapped to the image of $T_{i,j}$ for every i and j. We slightly abuse the terminology in the case of grid-preserving embeddings and say that a point q in the tile $P_{i,j}$ is mapped to a point r in the tile $T_{i,j}$ instead of saying that the image of q under the $\mathcal{F}$-assembly is mapped to the image of r under the $\mathcal{F}$-assembly.

Many of the definitions to follow are stated for a general $g \times h$ family of tiles $\mathcal{Q}$ as we later apply them on both $\mathcal{P}$ and $\mathcal{T}$. Recall that a pair of points r, q in the tile Q_i sandwiches a set of points A in the tile Q_{i+1} if for every point $t \in A$

- $r.y < t.y < q.y$ in case p_i and p_{i+1} occupy a common row, or
- $r.x < t.x < q.x$ in case p_i and p_{i+1} occupy a common column.

Moreover, we say that a pair of points r, q *strictly sandwiches* a set of points A if the pair r, q sandwiches A and there exists no other point $t \in (Q_{i+1} \setminus A)$ sandwiched by r, q.

Simple gadgets

Before constructing the actual tile families, we describe several simple gadgets that we will utilize later. All these gadgets include either one atomic pair $A = (r, q)$ or two atomic pairs $A_\alpha = (r_\alpha, q_\alpha)$ for $\alpha \in \{1, 2\}$ in the tile Q_i and one atomic pair $B = (s, t)$ or two atomic pairs $B_\alpha = (s_\alpha, t_\alpha)$ for $\alpha \in \{1, 2\}$ in the tile Q_{i+1}. Moreover, the coordinates of the points in Q_{i+1} are fully determined from the coordinates of the points in Q_i. We assume that each tile is formed as direct sum of the corresponding pieces of individual gadgets; in other words, if $A, B \subseteq Q_i$ are point sets of two different gadgets, then either A lies entirely to the right and above B or vice versa.

We describe the gadgets in the case when p_i and p_{i+1} share a common row and p_i is left from p_{i+1}, as the other cases are symmetric. Moreover, we assume that $r.y < q.y$ in the case of a single atomic pair in Q_i, and that $r_1.y < q_1.y < r_2.y < q_2.y$ in the case of two atomic pairs in Q_i. See Figure 5.2.

The copy gadget. The copy gadget consists of one atomic pair A in Q_i and one atomic pair B in Q_{i+1} defined as

$$s = (r.y, r.y + \epsilon) \qquad t = (q.y, q.y - \epsilon)$$

where ϵ is small positive value such that s, t form an occurrence of 12. We say that the copy gadget connects A to B. Since A sandwiches B, we can use the copy gadget to extend the construction to additional tiles along the path while preserving the embedding properties. Notice that many of our previous reductions consisted of sequences of copy gadgets that we called tracks.

Observation 5.7. *Suppose there is a copy gadget in $\mathcal{T}$ that connects an atomic pair A^T in the tile T_i to an atomic pair B^T in the tile T_{i+1}, and a copy gadget*

in $\mathcal{P}$ that connects an atomic pair A^P in the tile P_i to an atomic pair B^P in the tile P_{i+1}. In any grid-preserving embedding of π into τ, if A^P is mapped to A^T, then B^P is mapped to B^T.

The multiply gadget. The multiply gadget is only slightly more involved than the previous one, and it consists of one atomic pair A in Q_i and two atomic pairs B_1, B_2 in the tile Q_{i+1} defined as

$$s_1 = (r.y, r.y + \epsilon) \qquad s_2 = \left(\frac{r.y + 2 \cdot q.y}{3}, \frac{r.y + 2 \cdot q.y}{3}\right)$$
$$t_1 = \left(\frac{2 \cdot r.y + q.y}{3}, \frac{2 \cdot r.y + q.y}{3}\right) \qquad t_2 = (q.y, q.y - \epsilon)$$

where ϵ is small positive value such that s_1, t_1, s_2, t_2 form an occurrence of 1234. We say that the multiply gadget multiplies A to B_1 and B_2. A property analogous to Observation 5.7 holds when both text and pattern contain a multiply gadget.

The choose gadget. The choose gadget consists of one atomic pair A in Q_i and two atomic pairs B_1, B_2 in the tile Q_{i+1} defined as

$$s_1 = \left(r.y, \frac{2 \cdot r.y + q.y}{3}\right) \qquad s_2 = \left(\frac{r.y + 2 \cdot q.y}{3}, q.y - \epsilon\right)$$
$$t_1 = \left(\frac{2 \cdot r.y + q.y}{3}, r.y + \epsilon\right) \qquad t_2 = \left(q.y, \frac{r.y + 2 \cdot q.y}{3}\right).$$

where ϵ is small positive value such that s_1, t_1, s_2, t_2 form an occurrence of 2143. We say that the choose gadget branches A to B_1 and B_2.

The pick gadget. The pick gadget is essentially identical to the copy gadget except that the pair B forms an occurrence of 21 instead of 12. Formally, the pick gadget consists of one atomic pair A in Q_i and one atomic pair B in Q_{i+1} defined as

$$s = (r.y, q.y - \epsilon) \qquad t = (q.y, r.y + \epsilon)$$

where ϵ is chosen such that s, t form an occurrence of 21. We say that the pick gadget connects A to B.

The name of the choose and pick gadgets becomes clear with the following observation that follows from the fact that 2143 admits only two possible embeddings of the pattern 21.

Observation 5.8. *Suppose there is a choose gadget in $\mathcal{T}$ that branches an atomic pair A^T in the tile T_i to two atomic pairs B_1^T and B_2^T in the tile T_{i+1}, and a pick gadget in $\mathcal{P}$ that connects an atomic pair A^P in the tile P_i to an atomic pair B^P in the tile P_{i+1}. In any grid-preserving embedding of π into τ, if A^P is mapped to A^T then B^P is mapped either to B_1^T or to B_2^T.*

The merge gadget. The merge gadget consists of two atomic pairs A_1, A_2 in Q_i and one atomic pair B in Q_{i+1} defined as

$$s = (r_1.y, q_2.y + \epsilon) \qquad t = (q_2.y, r_1.y - \epsilon)$$

where ϵ is chosen small such that the relative order of the points of merge gadget and the points outside is not changed. Notice that now B sandwiches the set $A_1 \cup A_2$. We say that the merge gadget merges A_1 and A_2 into B.

The follow gadget. This gadget is almost identical to the pick gadget except that here B sandwiches A. Formally, the follow gadget contains one atomic pair A in Q_i and one atomic pair B in Q_{i+1} defined as

$$s = (r.y, q.y + \epsilon) \qquad t = (q.y, r.y - \epsilon)$$

where ϵ is chosen small such that the relative order of follow gadget with points outside is not changed. We say that the follow gadget connects A to B.

We can observe that merge and follow gadgets act in a way as an inverse to choose and pick gadgets.

Observation 5.9. *Suppose there is a merge gadget in $\mathcal{T}$ that merges atomic pairs A_1^T and A_2^T in the tile T_i into an atomic pair B^T in the tile T_{i+1}, and a follow gadget in $\mathcal{P}$ that connects an atomic pair A^P in the tile P_i to an atomic pair B^P in the tile P_{i+1}. In any grid-preserving embedding of π into τ, if A^P is mapped to A_α^T for some $\alpha \in \{1, 2\}$ then B^P is mapped to B^T.*

To see this, notice that B^P forms an occurrence of 21 and the only occurrence of 21 in T_{i+1} that sandwiches A_1^T or A_2^T is B^T. Here it is important that T_{i+1} is formed as a direct sum of the individual gadgets and in particular, every other occurrence of 21 in T_{i+1} lies either above or below B^T.

The flip gadget

We proceed to define two gadgets – a flip text gadget and a flip pattern gadget. The construction of this final pair of gadgets is a bit more involved. It is insufficient to consider just two neighboring tiles as we need two $\mathcal{D}$-entries for the construction. To that end, let i and j be indices such that both v_{i+1} and v_j are $\mathcal{D}$-entries and every entry v_k for k between $i+1$ and j is a monotone entry. Recall that every $\mathcal{D}$-entry shares a row with its predecessor. In particular, v_i occupies the same row as v_{i+1} and v_{j-1} occupies the same row as v_j.

As before, suppose that $A_1 = (r_1, q_1)$ and $A_2 = (r_2, q_2)$ are two atomic pairs in Q_i such that $r_1.y < q_1.y < r_2.y < q_2.y$. The *flip text gadget* attached to the atomic pairs A_1 and A_2 consists of two points s_1^{i+1}, s_2^{i+1} in the tile Q_{i+1} and two atomic pairs $B_1^k = (s_1^k, t_1^k), B_2^k = (s_2^k, t_2^k)$ in each tile Q_k for every $k \in [i+2, j] = \{i+2, i+3, \ldots, j\}$. The points s_1^{i+1}, s_2^{i+1} are defined as

$$s_1^{i+1} = \left(\frac{r_1.y + q_1.y}{2}, \frac{r_2.y + q_2.y}{2}\right) \quad s_2^{i+1} = \left(\frac{r_2.y + q_2.y}{2}, \frac{r_1.y + q_1.y}{2}\right).$$

Observe that s_1^{i+1}, s_2^{i+1} form an occurrence of 21 such that s_α^{i+1} is sandwiched by the pair A_α for each $\alpha \in \{1, 2\}$. The points $s_1^k, t_1^k, s_2^k, t_2^k$ for $k \in [i+2, j-1]$ are defined as

$$\begin{aligned} s_1^k &= (r_2.y + \epsilon, r_2.y) & s_2^k &= (r_1.y + \epsilon, r_1.y) \\ t_1^k &= (q_2.y - \epsilon, q_2.y) & t_2^k &= (q_1.y - \epsilon, q_1.y)\,. \end{aligned}$$

if v_k and v_{k+1} share the same column, otherwise we just apply the adjustments by ϵ in the y-coordinates. The positive constant ϵ is chosen such that the points $s_2^k, t_2^k, s_1^k, t_1^k$ (in this precise left-to-right order) form an occurrence of 1234 for every k between $i+1$ and j.

Finally, the points $s_1^j, t_1^j, s_2^j, t_2^j$ are defined as

$$s_1^j = (r_2.y, q_2.y) \qquad s_2^j = (r_1.y, q_1.y)$$
$$t_1^j = (q_2.y, r_2.y) \qquad t_2^j = (q_1.y, r_1.y)\,.$$

Observe that they form an occurrence of 2143. We say that the flip text gadget flips the pairs A_1, A_2 in Q_i to the pairs B_2^j, B_1^j in Q_j. See the left part of Figure 5.3.

We defined the points such that for every k between $i+1$ and j and $\alpha \in \{1,2\}$, the pair B_α^{k+1} sandwiches the pair B_α^k. Alternatively, this can be seen as B_α^{k+1} and B_α^k forming a copy gadget alas in the opposite direction. Moreover, the pair B_α^{i+2} sandwiches the point s_α^{i+1}.

Observe that independently of the actual orientation $\mathcal{F}$, the images of all points s_1^k, t_1^k for all k and the images of all points s_2^k, t_2^k for all k under the $\mathcal{F}$-assembly will be isomorphic. We define the flip pattern gadget as a set of points isomorphic to this particular set of points.

Given an atomic pair $A = (r, q)$ in Q_i, the *flip pattern gadget* attached to the atomic pair A consists of a single point s^{i+1} in the tile Q_{i+1} and an atomic pair $B^k = (s^k, t^k)$ in the tile Q_k for every $k \in [i+2, j]$, where

$$s^{i+1} = \left(\frac{r.y + q.y}{2}, \frac{r.y + q.y}{2}\right),$$

the atomic pair B^k is defined for every $k \in [i+2, j-1]$ as

$$s^k = (r.y + \epsilon, q.y) \qquad t^k = (q.y - \epsilon, r.y)$$

if v_k and v_{k+1} share a common column, otherwise we just apply the adjustments by ϵ in the y-coordinate, and finally, the atomic pair B^j is defined as

$$s^j = (r.y, q.y) \qquad t^j = (q.y, r.y)\,.$$

We say that the flip pattern gadget connects the pair A to the pair B^j. See the right part of Figure 5.3.

Lemma 5.10. *Suppose there is a flip pattern gadget in $\mathcal{P}$ that connects an atomic pair $\overline{A}$ in P_i with an atomic pair $\overline{B}$ in P_j. Furthermore, suppose that there is a flip text gadget in $\mathcal{T}$ that flips atomic pairs A_1 and A_2 in T_i to atomic pairs B_2 and B_1 in T_j. In any grid-preserving embedding of π into τ, if $\overline{A}$ is mapped to A_α for some $\alpha \in \{1,2\}$ then $\overline{B}$ is mapped to B_α.*

Proof. We denote the points of both gadgets as in their respective definitions. Additionally, we use overlined letters to denote points of the flip pattern gadget in $\mathcal{P}$ to distinguish them from the points of the flip text gadget in $\mathcal{T}$.

Observe that $\overline{s}^{i+1}$ must be mapped to s_α^{i+1}. This implies that the point $\overline{s}^{i+2}$ must be mapped to the point s_α^{i+2} or below and the point $\overline{t}^{i+2}$ must be mapped to the point t_α^{i+2} or above. By repeating this argument, we see that $\overline{s}^k$ must be mapped to the point s_α^k or below and the point $\overline{t}^k$ must be mapped to the point t_α^k or above for every $k \in [i+2, j]$. But the only occurrence of 21 in T_j with this property is precisely the pair B_α which concludes the proof. We remark that here we again use the property that T_j can be expressed as a direct sum of the individual gadgets. Otherwise, we could find a suitable occurrence of 21 in T_j as part of a different gadget. □

Flip gadgets will see two slightly different applications. First, as the name suggests, a flip gadget allows us to shuffle the order of atomic pairs. Second and perhaps more cunning use of the flip gadget is that it allows us to test if only one of its initial atomic pairs is used in the embedding.

Lemma 5.11. *Suppose that there are two flip pattern gadgets in $\mathcal{P}$ each connecting an atomic pair A^P_α in P_i to an atomic pair B^P_α in P_j for $\alpha \in \{1, 2\}$. Suppose that there is a flip text gadget in $\mathcal{T}$ that flips atomic pairs A^T_1 and A^T_2 in T_i to atomic pairs B^T_2 and B^T_1 in T_j. There cannot exist a grid-preserving embedding ϕ of π into τ that maps A^P_α to A^T_α for each $\alpha \in \{1, 2\}$.*

Proof. Using Lemma 5.10, we see that ϕ would map also B^P_α to B^T_α for each $\alpha \in \{1, 2\}$. But that is a contradiction since B^T_1 lies to right and above B^T_2 in T_j while B^P_1 lies to the left and below B^P_2 in P_j. □

We conclude the introduction of gadgets with one more definition. All the gadgets except for the copy and multiply ones need the entry v_{i+1} to be non-monotone, more precisely the image of $\mathcal{M}_{v_{i+1}}$ under $\mathcal{F}$ has to contain the Fibonacci class $\oplus 21$. Suppose there is an atomic pair A in the tile Q_i and that j is the smallest index larger than i such that p_j is a $\mathcal{D}$-entry. By *attaching a gadget* other than copy or multiply to the pair A, we mean the following procedure. We add an atomic pair A^k to each tile Q_k for $k \in [i + 1, j - 1]$ such that A^k and A^{k+1} form a copy gadget for each $k \in [i, j - 2]$ when we additionally define $A^i = A$. Finally, we attach the desired gadget to the atomic pair A^{j-1}. Similarly, when attaching a gadget that contains two atomic pairs A_1 and A_2 in its first tile, we just copy both of these pairs all the way to the tile Q_{j-1} and then attach the desired gadget. It follows from Observation 5.7 that the embedding properties are preserved via this construction.

Constructing the $\mathcal{C}$-PPM instance

We define the initial tiles P_1 and T_1 to both contain a single atomic pair, the *anchor*, defined as

$$(1, 1), \quad (2n + 2, 2n + 2).$$

The tile P_2 consists of the atomic pairs $X^0_k = (q_k, r_k)$ for $k \in [n]$ and the tile T_2 consists of the atomic pairs $Y^0_k = (s_k, t_k)$ for $k \in [n]$ where

$$q_k = s_k = (2k, 2k) \qquad r_k = t_k = (2k + 1, 2k + 1).$$

Any grid-preserving embedding of π into τ must obviously map the anchors to each other and X^0_k to Y^0_k for every $k \in [n]$.

Assignment phase. In the first phase, we simulate the assignment of truth values to the variables. To that end, we attach to each pair Y^0_k for $k \in [n]$ a choose gadget that branches Y^0_k to two atomic pairs $Y^1_{k,1}$ and $Z^1_{k,1}$. On the pattern side, we attach to each pair X^0_k for $k \in [n]$ a pick gadget that connects X^0_k to an atomic pair $X^1_{k,1}$. The properties of choose and pick gadgets (Observation 5.8) imply that in any grid-preserving embedding, $X^1_{k,1}$ is either mapped to $Y^1_{k,1}$ or to $Z^1_{k,1}$.

Multiplication phase. Our next goal is to multiply the atomic pairs corresponding to a single variable into as many pairs as there are occurrences of this variable in the clauses. We describe the gadgets dealing with each variable individually.

Fix $k \in [n]$ and let d_k for $k \in [n]$ denote the total number of occurrences of x_k and $\neg x_k$ in φ. We are going to describe the construction inductively in $\ell_k = \lceil \log d_k \rceil$ steps. In i-th step, we define atomic pairs $X^{i+1}_{k,j}, Y^{i+1}_{k,j}, Z^{i+1}_{k,j}$ for $j \in [2^{i+1}]$ such that in any grid-preserving embedding, $X^{i+1}_{k,j}$ maps either to $Y^{i+1}_{k,j}$ or to $Z^{i+1}_{k,j}$. Moreover, the order of the atomic pairs in the pattern tile is $X^{i+1}_{k,1}, X^{i+1}_{k,2}, \ldots, X^{i+1}_{k,2^{i+1}}$ and the order of the atomic pairs in the text tile is

$$Y^{i+1}_{k,1}, Z^{i+1}_{k,1}, Y^{i+1}_{k,2}, Z^{i+1}_{k,2}, \ldots, Y^{i+1}_{k,2^{i+1}}, Z^{i+1}_{k,2^{i+1}}.$$

First, notice that the properties hold for $i = 0$ at the end of the assignment phase. Fix $i \geq 1$. We add for each $j \in [2^i]$ three multiply gadgets, one that multiplies the atomic pair $X^i_{k,j}$ to atomic pairs $\widetilde{X}^{i+1}_{k,2j-1}$ and $\widetilde{X}^{i+1}_{k,2j}$, one that multiplies the pair $Y^i_{k,j}$ to $\widetilde{Y}^{i+1}_{k,2j-1}$ and $\widetilde{Y}^{i+1}_{k,2j}$, and finally one that multiplies $Z^i_{k,j}$ to $\widetilde{Z}^{i+1}_{k,2j-1}$ and $\widetilde{Z}^{i+1}_{k,2j}$. Observe that the properties of gadgets together with induction imply that for arbitrary $j \in [2^{i+1}]$, $\widetilde{X}^{i+1}_{k,j}$ maps either to $\widetilde{Y}^{i+1}_{k,j}$ or to $\widetilde{Z}^{i+1}_{k,j}$. Moreover, the atomic pairs $\widetilde{X}^{i+1}_{k,j}$ are already ordered in the pattern by j as desired. However, the order of the atomic pairs in text is incorrect as for each $j \in [2^i]$ we have the quadruple

$$\widetilde{Y}^{i+1}_{k,2j-1}, \widetilde{Y}^{i+1}_{k,2j}, \widetilde{Z}^{i+1}_{k,2j-1}, \widetilde{Z}^{i+1}_{k,2j}$$

in this specific order.

To solve this, we add for each $j \in [2^i]$ a flip text gadget that flips $\widetilde{Y}^{i+1}_{k,2j}$, $\widetilde{Z}^{i+1}_{k,2j-1}$ to atomic pairs $Z^{i+1}_{k,2j-1}, Y^{i+1}_{k,2j}$. Furthermore, we attach a flip pattern gadget to every other atomic pair in both pattern and text. In particular, we add one that connects the pair $\widetilde{Y}^{i+1}_{k,2j-1}$ to a pair $Y^{i+1}_{k,2j-1}$, one that connects $\widetilde{Z}^{i+1}_{k,2j}$ to a pair $Z^{i+1}_{k,2j}$ and finally two that connect $\widetilde{X}^{i+1}_{k,\alpha}$ to $X^{i+1}_{k,\alpha}$ for $\alpha \in \{2j-1, 2j\}$. The properties of flip gadgets guarantee that for every $j \in [2^{i+1}]$, the pair $X^{i+1}_{k,j}$ is mapped either to $Y^{i+1}_{k,j}$ or to $Z^{i+1}_{k,j}$. Moreover, the order of atomic pairs in the text now alternates between Y and Z as desired.

We described the gadget constructions independently for each variable. The unfortunate effect is that we might have used up a different total number of tiles for each variable. We describe a way to fix this. Let k be such that d_k is the largest value and let j be the largest index such that the tiles T_j and P_j have been used in the multiplication phase for the k-th variable. For every other variable, we simply attach a chain of copy gadgets connecting every pair at the end of its multiplication phase all the way to the tiles P_j and T_j. Observe that we need in total $O(\log m)$ entries equal to $\mathcal{D}$ for the multiplication phase as d_k is at most $3m$.

Sorting phase via gadgets. The multiplication phase ended with atomic pairs $X^i_{k,j}$ in the pattern and $Y^i_{k,j}$ and $Z^i_{k,j}$ in the text for some i, every $k \in [n]$ and $j \in [d_k]$.[1] These pairs are ordered lexicographically by (k, j), i.e., they are bundled in blocks by the variables. The goal of the sorting phase, as the name suggests, is

[1] In fact for every $j \in [2^{\lceil \log d_k \rceil}]$ but we simply ignore the pairs for $j > d_k$.

to rearrange them such that they become bundled by clauses while retaining the embedding properties.

We show how to use gadgets to swap two neighboring atomic pairs in the pattern. Suppose that X_1 and X_2 are two atomic pairs in some tile P_i such that X_1 is to the left and below X_2 and all the remaining atomic pairs are either to the right and top of both X_1, X_2 or to the left and below both X_1, X_2. Suppose that Y_1, Y_2, Z_1, Z_2 are atomic pairs in the tile T_i such that they are ordered in T_i as Y_1, Z_1, Y_2, Z_2, and every other atomic pair lies either to the right and top or to the left and below of all of them. Furthermore, suppose that in any grid-preserving embedding ϕ, the pair X_α is always mapped either to Y_α or to Z_α for each $\alpha \in \{1, 2\}$. Notice that this is precisely the case at the end of the multiplication phase.

We attach choose gadgets to each of the pairs Y_1, Z_1, Y_2, Z_2 and a pick gadget to both X_1, X_2. In particular, we add for every $\alpha \in \{1, 2\}$

- a choose gadget that branches Y_α to pairs $\overline{Y}_\alpha$ and $\widetilde{Y}_\alpha$,
- a choose gadget that branches Z_α to pairs $\overline{Z}_\alpha$ and $\widetilde{Z}_\alpha$, and
- a pick gadget that connects X_α to $\overline{X}_\alpha$.

We will now abuse our notation slightly and use the same letters to denote atomic pairs in different tiles so that the names are carried through with the flip gadgets. In other words, a flip text gadget shall flip atomic pairs A_1, A_2 to pairs A_2, A_1 and a flip pattern gadget connects atomic pair A to an atomic pair A. Using three layers of flip gadgets in $\mathcal{T}$, we change the order of the pairs in the following way. Note that the blue color is used to highlight pairs to which $\overline{X}_1$ can be mapped, and red color for the pairs to which $\overline{X}_2$ can be mapped and arrows show which pairs are flipped in each step.

$$\overset{\curvearrowright}{\overline{Y}_1}\overset{\curvearrowright}{\widetilde{Y}_1}\overline{Z}_1\overset{\curvearrowright}{\widetilde{Z}_1}\overline{Y}_2\overset{\curvearrowright}{\widetilde{Y}_2}\overline{Z}_2\widetilde{Z}_2 \to \overline{Y}_1\widetilde{Y}_1\overset{\curvearrowright}{\overline{Z}_1}\overline{Y}_2\overset{\curvearrowright}{\widetilde{Z}_1}\overline{Z}_2\widetilde{Y}_2\widetilde{Z}_2 \to \overline{Y}_1\overset{\curvearrowright}{\widetilde{Y}_1}\overline{Y}_2\overset{\curvearrowright}{\overline{Z}_1}\overline{Z}_2\overset{\curvearrowright}{\widetilde{Z}_1}\widetilde{Y}_2\widetilde{Z}_2$$
$$\to \overline{Y}_1\overline{Y}_2\widetilde{Y}_1\overline{Z}_2\overline{Z}_1\widetilde{Y}_2\widetilde{Z}_1\widetilde{Z}_2$$

On the pattern side, we attach three consecutive copies of flip pattern gadget to both $\overline{X}_1$ and $\overline{X}_2$.

Now we perform the actual swap by attaching a flip text gadget that flips $\overline{X}_1$ and $\overline{X}_2$ and attaching flip text gadget between each of the four neighboring pairs in the text, i.e.

$$\overset{\curvearrowright}{\overline{Y}_1}\overline{Y}_2\overset{\curvearrowright}{\widetilde{Y}_1}\overline{Z}_2\overset{\curvearrowright}{\overline{Z}_1}\widetilde{Y}_2\overset{\curvearrowright}{\widetilde{Z}_1}\widetilde{Z}_2 \to \overline{Y}_2\overline{Y}_1\overline{Z}_2\widetilde{Y}_1\widetilde{Y}_2\overline{Z}_1\widetilde{Z}_2\widetilde{Z}_1.$$

As a next step, we unshuffle the pairs to which $\overline{X}_1$ and $\overline{X}_2$ can be mapped, again by three layers of flip gadgets

$$\overline{Y}_2\overline{Y}_1\overline{Z}_2\widetilde{Y}_1\widetilde{Y}_2\overline{Z}_1\widetilde{Z}_2\widetilde{Z}_1 \to \overline{Y}_2\overline{Z}_2\overline{Y}_1\widetilde{Y}_2\widetilde{Y}_1\widetilde{Z}_2\overline{Z}_1\widetilde{Z}_1 \to \overline{Y}_2\overline{Z}_2\widetilde{Y}_2\overline{Y}_1\widetilde{Z}_2\widetilde{Y}_1\overline{Z}_1\widetilde{Z}_1$$
$$\to \overline{Y}_2\widetilde{Y}_2\overline{Z}_2\widetilde{Z}_2\overline{Y}_1\widetilde{Y}_1\overline{Z}_1\widetilde{Z}_1$$

As before, we attach three consecutive copies of flip pattern gadget to both $\overline{X}_1$ and $\overline{X}_2$.

Finally, we add for every $\alpha \in \{1, 2\}$

- a merge gadget that merges $\overline{Y}_\alpha$ and $\widetilde{Y}_\alpha$ to a pair Y'_α,
- a merge gadget that merges $\overline{Z}_\alpha$ and $\widetilde{Z}_\alpha$ to a pair Z'_α, and
- a follow gadget that connects $\overline{X}_\alpha$ to X'_α.

It follows from the properties of the individual gadgets that in any grid-preserving embedding, the pair X'_α is mapped either to Y'_α or Z'_α for each $\alpha \in \{1, 2\}$. On the other hand, any grid-preserving embedding of the first i tiles can be extended to the points added in the construction. The crucial observation is that in the step when the order of $\overline{X}_1$ and $\overline{X}_2$ is reversed, four neighboring pairs of atomic pairs are flipped in $\mathcal{T}$ which form all possible combinations of Y_1, Z_1 and Y_2, Z_2. Therefore, we can always choose where to map the pairs $\overline{X}_1$ and $\overline{X}_2$ at the first step to arrive at one of these pairs that get flipped.

Notice that we can use the described construction to swap arbitrary subset of disjoint neighboring atomic pairs in the pattern in parallel. In general, any sequence of length ℓ can be sorted using at most ℓ rounds of such parallel swaps. One possible way to do so is to use the so-called odd-even sort introduced by Habermann [79]. Since the total number of atomic pairs in the pattern after the multiplication phase is exactly $3m$, the number of rounds needed is at most $3m$. Individually, each of the $3m$ steps uses only a constant number of layers of gadgets and thus only a constant amount of $\mathcal{D}$-entries. Therefore, the sorting phase takes in total $O(m)$ $\mathcal{D}$-entries.

Sorting phase via juxtapositions. We claim that the sorting phase can be done using significantly fewer $\mathcal{D}$-entries if the class $\mathcal{D}$ contains a monotone juxtaposition.

Suppose that at the beginning of the sorting phase there are atomic pairs $X_1, X_2, \ldots, X_{3m}$ in the pattern and atomic pairs $Y_1, Z_1, Y_2, Z_2, \ldots, Y_{3m}, Z_{3m}$ in the text (both in this precise order) such that in any grid-preserving mapping, X_i is mapped either to Y_i or to Z_i. Our goal is to rearrange the pairs so that there are atomic pairs $\overline{X}_{\sigma_1}, \overline{X}_{\sigma_2}, \ldots, \overline{X}_{\sigma_{3m}}$ in the pattern and atomic pairs $\overline{Y}_{\sigma_1}, \overline{Z}_{\sigma_1}, \overline{Y}_{\sigma_2}, \overline{Z}_{\sigma_2}, \ldots, \overline{Y}_{\sigma_{3m}}, \overline{Z}_{\sigma_{3m}}$ in the text in this order given by permutation σ of length $3m$. Moreover, in any grid-preserving embedding $\overline{X}_i$ is mapped to $\overline{Y}_i$ if X_i is mapped to Y_i, and it is mapped to $\overline{Z}_i$ if X_i is mapped to Z_i.

Suppose that the proper-turning path $v_1, v_2, \ldots$ contains D entries equal to $\mathcal{D}$. Since there are in total only 4 possible images of $\mathcal{D}$ given by the orientation $\mathcal{F}$, there exists a monotone juxtaposition $\mathcal{B}$ and at least $D/4$ indices j such that $\mathcal{M}_{v_j} = \mathcal{D}$ and the class $\mathcal{F}(\mathcal{M})_{v_j}$ contains $\mathcal{B}$. We are going to use only these entries for the sorting phase.

First, suppose that $\mathcal{B} = \mathrm{Grid}(\boxslash\boxslash)$. Let v_i be an entry such that $\mathcal{F}(\mathcal{M})_{c_i}$ contains $\mathcal{B}$ and recall that we consider only those non-monotone entries that share a common row with their predecessor, i.e., v_i shares a common row with v_{i-1}. We can construct a tile Q_i from two tiles Q_i^1 and Q_i^2 where both Q_i^1 and Q_i^2 contain an increasing point set and Q_i^1 and Q_i^2 are placed next to each other. In particular, we can then attach to any atomic pair A in Q_{i-1} a copy gadget connecting A to an atomic pair B and choose arbitrarily whether B lies in Q_i^1 or Q_i^2.

Let J_1 and J_2 be a partition of the set $[3m]$. We attach a copy gadget ending in Q_i^α to each X_j, Y_j and Z_j with $j \in J_\alpha$ for each $\alpha \in \{1, 2\}$. In this way, we rearranged the atomic pairs in $\mathcal{P}$ such that first we have all pairs X_j such that $j \in J_1$ (sorted by the indices) followed by all pairs X_j for $j \in J_2$ (again themselves

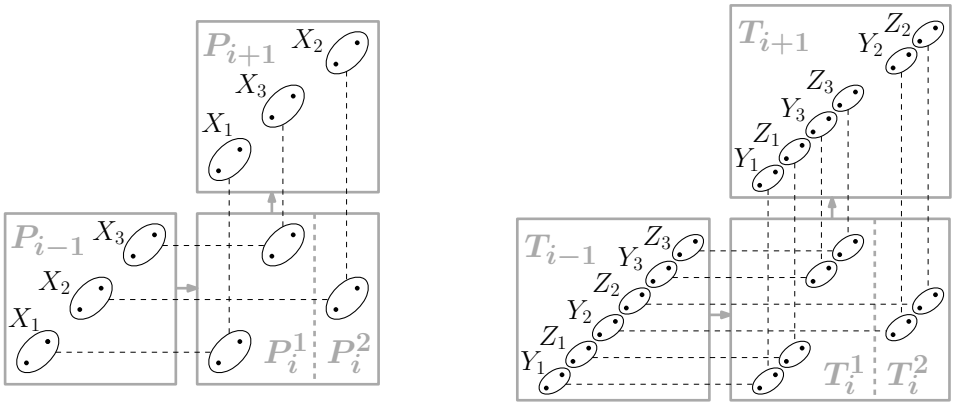

Figure 5.4: Example of one sorting step using the juxtaposition $\mathcal{B} = \mathrm{Grid}(\boxslash\boxslash)$ and partition of the set $\{1, 2, 3\}$ into $J_1 = \{1, 3\}$ and $J_2 = \{2\}$. The pattern tiles are on the left, the text tiles are on the right.

sorted by the indices). Similarly in $\mathcal{T}$, we first have Y_j, Z_j for $j \in J_1$ followed by Y_j, Z_j for $j \in J_2$. See Figure 5.4.

Notice that the described operation simulates a stable bucket sort with two buckets. Therefore, we can simulate radix sort and rearrange the atomic pairs into arbitrary order given by σ by iterating this operation $O(\log m)$ times. In this way, the whole sorting phase uses only $O(\log m)$ entries equal to $\mathcal{D}$.

Now suppose that $\mathcal{B} = \mathrm{Grid}(\mathcal{A}_1\ \mathcal{A}_2)$ where $\mathcal{A}_1$, $\mathcal{A}_2$ are two arbitrary monotone classes. We can use the same construction as before. However, we need to be careful that some of Q_i^1 and Q_i^2 should actually contain a decreasing sequence instead of increasing. Following the procedure as before, we still have in $\mathcal{P}$ first all the pairs X_j for $j \in J_1$ followed by the pairs X_j for $j \in J_2$. However, the order of pairs X_j for $j \in J_\alpha$ is now reversed if $\mathcal{A}_\alpha = \boxslash$. This can be fixed by using one extra entry whose image under $\mathcal{F}$ is $\mathcal{B}$. We partition $[3m]$ into $J_1' = J_1$ and $J_2' = J_2$ and attach the same construction once again. The property that every X_j for $j \in J_1$ precedes every X_j for $j \in J_2$ is preserved and moreover, any of the two parts that were reversed in the first step is now ordered again in the correct increasing order by the indices. Thus, we again implemented a stable bucket sort with two buckets and we can rearrange the atomic pairs into arbitrary order using $O(\log m)$ such steps.

Now, suppose that $\mathcal{B} = \mathrm{Grid}\binom{\mathcal{A}_2}{\mathcal{A}_1}$ for two monotone classes $\mathcal{A}_1$, $\mathcal{A}_2$. In this case, we can again construct the tile Q_i from two tiles Q_i^1 and Q_i^2 where both Q_i^1 and Q_i^2 contain a monotone point set determined by the classes $\mathcal{A}_1$, $\mathcal{A}_2$ but Q_i^1 and Q_i^2 are this time placed on top of each other. First, suppose that $\mathcal{A}_1 = \mathcal{A}_2 = \boxslash$.

This time we can choose how to split the set $[3m]$ into two sets of consecutive numbers J_1 and J_2 such that every element of J_1 is smaller than every element of J_2 and again connect a copy gadget ending in Q_i^α to each X_i, Y_j, and Z_j with $j \in J_\alpha$ for each $\alpha \in \{1, 2\}$. However, this time, we also choose the relative order of gadgets between Q_i^1 and Q_i^2, which enables us to arbitrarily interleave the sequences of atomic pairs indexed by J_1 and J_2, respectively. Effectively, we implemented an inverse operation to the stable bucket sort with two buckets – we split the sequence of atomic pairs into two (uneven) halves and then interleave them arbitrarily while keeping each of the two parts in the original order. Therefore, we can again rearrange the atomic pairs into an arbitrary order iterating this operation $O(\log m)$

times and thus using only $O(\log m)$ entries equal to $\mathcal{D}$.

Finally, it remains to deal with the case when $\mathcal{B} = \mathrm{Grid}\binom{\mathcal{A}_2}{\mathcal{A}_1}$ and $\mathcal{A}_1$, $\mathcal{A}_2$ are arbitrary monotone classes. Observe that the construction described in the previous paragraph results in interleaving the two sequences and simultaneously reversing the order of pairs in J_α if $\mathcal{A}_\alpha = \boxslash$. We can easily resolve this by prepending an extra step of the same construction. We first split $[3m]$ into the same halves $J'_1 = J_1$ and $J'_2 = J_2$ and use the described construction. However, we do not interleave the two sets and keep them in the same order. Therefore, we have only reversed the order of pairs in the half J_α if $\mathcal{A}_\alpha = \boxslash$. If we then perform the actual sorting step, each of the sequences will end up in the correct original order.

Evaluation phase. After the sorting phase, the atomic pairs in the pattern are bundled into consecutive triples determined by clauses of Φ. We show how to test whether a clause $K_j = (L_a \vee L_b \vee L_c)$ is satisfied where $L_\alpha \in \{x_\alpha, \neg x_\alpha\}$ for $\alpha \in \{a, b, c\}$ and $a < b < c$.

Suppose X_a, X_b and X_c are the three neighboring atomic pairs in $\mathcal{P}$ that correspond to the three literals in K_j. In $\mathcal{T}$, there are six neighboring atomic pairs Y_a, Z_a, Y_b, Z_b, Y_c, Z_c in this precise order such that in any grid-preserving embedding, the pair X_α is mapped to either Y_α or Z_α for every $\alpha \in \{a, b, c\}$.

First, we claim that we can without loss of generality assume that in fact $K_j = (x_a \vee x_b \vee x_c)$. If that was not the case, we could use one layer of flip gadgets to reverse the order of Y_α, Z_α for each α such that $L_\alpha = \neg x_\alpha$.

As in the sorting phase, we slightly abuse the notation and use the same letters to denote atomic pairs in different tiles so that any gadget with the same number of input and output atomic pairs carries the names through. We add one layer of gadgets, in particular

- a choose gadget that branches Z_b to $\overline{Z}_b$ and $\widetilde{Z}_b$, and
- pick gadgets to $Y_a, Z_a, Y_b, Y_c, Z_c, X_a, X_b$ and X_c.

We continue with adding two layers of flip gadgets, modifying the order of atomic pairs in the text as follows

$$Y_a Z_a \overset{\curvearrowright}{Y_b \overline{Z}_b} \widetilde{Z}_b \overset{\curvearrowright}{Y_c Z_c} \to Y_a \overset{\curvearrowright}{Z_a \overline{Z}_b} Y_b \overset{\curvearrowright}{\widetilde{Z}_b Z_c} Y_c \to Y_a \overline{Z}_b Z_a Y_b Z_c \widetilde{Z}_b Y_c,$$

and two consecutive copies of flip pattern gadget to all X_a, X_b and X_c. Notice that the order of the atomic pairs in the text has exactly the same structure as the word in Observation 4.2. This construction is done for each clause in parallel. Therefore, it uses only constantly many layers of gadgets and in particular, it uses only $O(1)$ $\mathcal{D}$-entries of the path.

That concludes the construction of $\mathcal{P}$ and $\mathcal{T}$. Observe that each tile in both $\mathcal{P}$ and $\mathcal{T}$ contains $O(m)$ points. If the construction uses D entries equal to $\mathcal{D}$, then we need to start with a proper-turning path of length $L = 8/\epsilon \cdot D$ where ϵ is the constant given by the $\mathcal{D}$-rich property of $\mathcal{C}$. The constant 8 appears since we first throw away at most half $\mathcal{D}$-entries that do not share the same row with their predecessor, and subsequently, we use only the most frequent symmetry (out of 4) of $\mathcal{D}$ under $\mathcal{F}$ when sorting by juxtapositions.

The total amount of $\mathcal{D}$-entries used by the construction depends on how many steps are needed for the sorting phase. If $\mathcal{D}$ contains a monotone juxtaposition, then the total amount of $\mathcal{D}$-entries used is $O(\log m)$ and thus, the length of both π and τ is bounded by $O(m \log m)$. Otherwise, if we sort using only gadgets, the total amount of $\mathcal{D}$-entries used by the reduction is $O(m)$ and thus the length of both π and τ is bounded by $O(m^2)$. This gives rise to the two different lower bounds for the runtime of an algorithm solving $\mathcal{C}$-PPM under ETH.

Beyond grid-preserving embeddings. Now, we modify both π and τ so that any anchored embedding must already be grid-preserving. To that end, take $\mathcal{P}'$ to be the family of tiles obtained from $\mathcal{P}$ by adding atomic pairs A_1, A_2 to the initial tile P_2 so that A_1 is to the left and below everything else in P_2 and A_2 is to the right and above everything else in P_2. However, both A_1 and A_2 are sandwiched by the anchor in P_1. We then attach to both A_1, A_2 a chain of copy gadgets spreading all the way to the last tile used by $\mathcal{P}$. We obtain $\mathcal{T}'$ from $\mathcal{T}$ in the same way by adding atomic pairs B_1, B_2 sandwiched by the anchor to T_1 and a chain of copy gadgets attached to each. We let π' and τ' be the $\mathcal{F}$-assemblies of $\mathcal{P}'$ and $\mathcal{T}'$.

Observe that in any anchored embedding of π' into τ', the image of A_α is mapped to the image of B_α for each $\alpha \in \{1, 2\}$. The chain of copy gadgets attached to A_α then must map to the chain of gadgets attached to B_α and these chains force that the image of P'_i maps to the image of T'_i for every i. Observe that $|\pi'| = O(|\pi|)$ and $|\tau'| = O(|\tau|)$.

Claim 5.12. *The formula φ is satisfiable if and only if there is an anchored embedding of π' into τ'.*

The "only if" part Let φ be a satisfiable formula and fix arbitrary satisfying assignment represented by a function $\rho\colon [n] \to \{T, F\}$, where $\rho(k) = T$ if and only if the variable x_k is set to true in the chosen assignment.

We map the image of $P'_1 \cup P'_2$ to the image of $T'_1 \cup T'_2$. In the assignment phase, we map the pair $X^1_{k,1}$ to $Y^1_{k,1}$ if $\rho(k) = T$, otherwise we map it to $Z^1_{k,1}$. The embedding of the multiplication phase is then uniquely determined by the properties of the gadgets. In particular at the end of the multiplication phase, the pair $X^\ell_{k,j}$ for every j is mapped to $Y^\ell_{k,j}$ if $\rho(k) = T$ and to $Z^\ell_{k,j}$ otherwise.

The mapping is then straightforwardly extended through the sorting phase. We just have to be careful when swapping two neighboring pairs to pick correctly between $\overline{Y}_\alpha$ and $\widetilde{Y}_\alpha$ (or $\overline{Z}_\alpha$ and $\widetilde{Z}_\alpha$) such that the swap itself is possible.

Recall that at the end of the sorting phase, we have for each clause $K_j = (L_a \vee L_b \vee L_c)$ a consecutive block of three pairs X_a, X_b, X_c in a pattern tile and a consecutive block of six pairs Y_a, Z_a, Y_b, Z_b, Y_c, Z_c in a text tile such that X_α is mapped to Y_α if $\rho(\alpha) = T$ for every $\alpha = \{a, b, c\}$. The fact that we can extend the embedding through the evaluation phase is a direct consequence of Observation 4.2 since the clause K_j is satisfied by ρ.

The "if" part Let ϕ be an anchored embedding of π' into τ'. As we argued, this implies that there is a grid-preserving embedding of π into τ. Using this grid-preserving embedding, we define a satisfying assignment $\rho\colon [n] \to \{T, F\}$.

We set $\rho(k) = T$ if the pair $X^1_{k,1}$ is mapped to $Y^1_{k,1}$ and we set $\rho(k) = F$ if it is mapped to $Z^1_{k,1}$. This property is clearly maintained throughout the multiplication and sorting phases due to the properties of gadgets.

At the beginning of the evaluation phase, we thus have for each clause $K_j = (L_a \vee L_b \vee L_c)$ a consecutive block of three pairs X_a, X_b, X_c in a pattern tile and a consecutive block of six pairs Y_a, Z_a, Y_b, Z_b, Y_c, Z_c in a text tile such that X_α is mapped to Y_α if and only if $\rho(\alpha) = T$ for every $\alpha = \{a, b, c\}$. As before, we assume that $K_j = (x_a \vee x_b \vee x_c)$. It remains to argue that it cannot happen that the pair X_α is mapped to Z_α for every $\alpha \in \{a, b, c\}$. But that is not possible due to Observation 4.2 and thus, every clause of φ is satisfied with the assignment given by ρ.

Finally, we drop the assumption about anchored embeddings via the usual trick of inflating the anchors in π' and τ' with monotone sequences of length $|\tau'|$. This concludes the proof of Theorem 5.6.

5.3.3 Principal classes

We aim to use the properties of principal classes derived in Section 2.3. As a result, we will be able to show that $\mathrm{Av}(\sigma)$-PPM is NP-complete for every principal class with the bicycle property.

First of all, Lemma 2.54 immediately shows that $\mathrm{Av}(1 \ominus \pi)$ has the computable $\mathrm{Av}(\pi)$-rich path property. In particular, it proves that $\mathrm{Av}(\sigma)$-PPM is NP-complete for σ equal to $4321, 4312, 4231$ or any of their symmetries. Moreover, the stronger conditional lower bound on the runtime of any algorithm holds due to the following observation.

Observation 5.13. *The class* $\mathrm{Av}(\pi)$ *for any* π *of length 3 is non-monotone-griddable and moreover, it contains a monotone juxtaposition as a subclass.*

The bicycle property of all the other principal classes followed from Lemma 2.55. Let us show that the same lemma implies that each of these classes has the computable $\mathrm{Av}(\sigma)$-rich path property for some σ of length 3. The proof of the following proposition follows closely the proof of Proposition 2.34.

Proposition 5.14. *Let* $\mathcal{D}$ *be a non-monotone-griddable class that is sum-closed or skew-closed. If* $\mathcal{M}$ *is a gridding matrix whose cell graph* $G_{\mathcal{M}}$ *contains a proper-turning cycle with at least one entry equal to* $\mathcal{D}$*, then* $\mathrm{Grid}(\mathcal{M})$ *has the computable* $\mathcal{D}$*-rich path property.*

Proof. Before the actual proof, we need to extend the notion of consistent orientations to gridding matrices whose every entry is either sum-closed or skew-closed. We say that $\mathcal{F} = (f_c, f_r)$ is a consistent orientation of such a gridding matrix $\mathcal{M}$ if the entry $\mathcal{M}_{i,j}$ is sum-closed if $f_c(i) \cdot f_r(j) = 1$ and otherwise, $\mathcal{M}_{i,j}$ is skew-closed.

We claim that it is possible to obtain in polynomial time a gridding matrix $\mathcal{M}$ with the following properties:

(i) $\mathrm{Grid}(\mathcal{M}) \subseteq \mathcal{C}$,
(ii) the cell graph of $\mathcal{M}$ consists of a single proper-turning cycle,

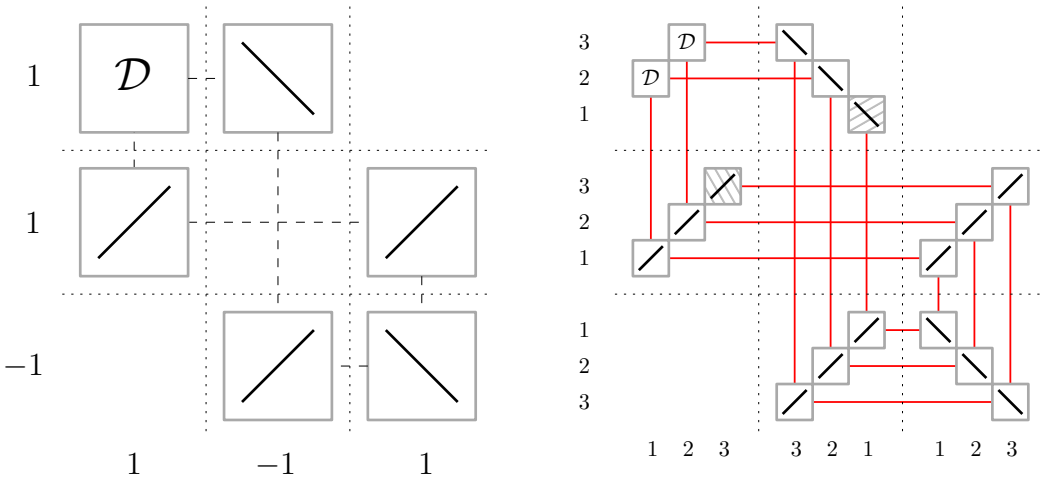

Figure 5.5: Left: a gridding matrix $\mathcal{M}$ whose cell graph is a cycle with a single $\mathcal{D}$-entry together with a consistent orientation $\mathcal{F}$ given by the numbers along the edges. Right: a gridding matrix $\mathcal{N}$ whose grid class is contained in $\mathrm{Grid}(\mathcal{M})$ and whose cell graph forms one long path with constant fraction of $\mathcal{D}$-entries. The numbers along the edges are the coordinates of the original cells under matrix $\mathcal{F}$-assembly and we highlighted the endpoints of the path.

(iii) $\mathcal{M}$ contains a unique entry equal to $\mathcal{D}$ and all other non-empty entries are equal to $\boxslash$ or to $\boxbslash$, and
(iv) $\mathcal{M}$ has a consistent orientation $\mathcal{F}$.

Establishing (i), (ii) and (iii) is straightforward because each infinite permutation class contains either $\boxslash$ or $\boxbslash$ as a subclass, and replacing an entry of $\mathcal{M}$ by its subclass can only change $\mathrm{Grid}(\mathcal{M})$ into its subclass. Let $\mathcal{M}'$ be a gridding matrix that satisfies (i), (ii) and (iii). In order to guarantee (iv), we use the usual trick of replacing each entry of $\mathcal{M}'$ by a suitable 2×2 matrix. Let $\mathcal{M}'^{\times 2}$ be a gridding matrix obtained from $\mathcal{M}$ by replacing every entry $\mathcal{M}_{i,j}$ with the matrix $\begin{pmatrix} \cdot & \mathcal{M}_{i,j} \\ \mathcal{M}_{i,j} & \cdot \end{pmatrix}$ if $\mathcal{M}_{i,j}$ is sum-closed and with the matrix $\begin{pmatrix} \mathcal{M}_{i,j} & \cdot \\ \cdot & \mathcal{M}_{i,j} \end{pmatrix}$ otherwise. Clearly, $\mathrm{Grid}(\mathcal{M}'^{\times 2})$ is a subclass of $\mathrm{Grid}(\mathcal{M}')$ and a consistent orientation $\mathcal{F}$ of $\mathcal{M}'^{\times 2}$ can be obtained using exactly the same argument as for the monotone matrices in Lemma 2.35. We take as $\mathcal{M}$ any connected component of $G_{\mathcal{M}'^{\times 2}}$.

Let us assume for the rest of the proof that $\mathcal{D}$ is sum-closed, since the skew-closed case is symmetric and let $\mathcal{M}_{i^\star,j^\star}$ be the $\mathcal{D}$-entry in $\mathcal{M}$. Without loss of generality, we assume that $f_c(i^\star) = f_r(j^\star) = 1$ or in other words, the orientation $\mathcal{F}$ does not transform the $(i^\star, j^\star)$-cell at all. This is possible since we assume that $\mathcal{D}$ is sum-closed.

For a natural number k, we show how to define a family of matrices Λ that will create upon its $\mathcal{F}$-assembly a gridding matrix whose cell graph is a path of length at least k. We define $\mathcal{L}^{i^\star,j^\star}$ as the $k \times k$ matrix S with non-empty entries of the form $S_{i,i+1} = \mathcal{D}$ for $i \in [k-1]$. And for every other vertex v of $G_{\mathcal{M}}$, we simply take as $\mathcal{L}^v$ the increasing matrix with k non-empty entries on its diagonal.

Let $\mathcal{N}$ be the $\mathcal{F}$-assembly of Λ. See Figure 5.5. It is easy to see that $\mathrm{Grid}(\mathcal{N})$ is a subclass of $\mathrm{Grid}(\mathcal{M})$ since $\mathcal{F}$ is a consistent orientation. Moreover, $\mathcal{N}$ contains a constant fraction of $\mathcal{D}$-entries. Therefore, it remains only to show that $G_{\mathcal{N}}$ is a path. This fact follows using an identical argument as in the proof of Proposition 2.34

and we omit further details here. □

Corollary 5.15. *If σ is a permutation of length at least 4 that is not symmetric to any of* $3412, 3142, 4213, 4123$ *or* 41352*, then* $\mathrm{Av}(\sigma)$*-PPM is* NP*-complete, and unless ETH fails, it cannot be solved in time* $2^{o(n/\log n)}$*.*

Proof. We showed in Section 2.3 that all the classes of interest are covered by Lemmas 2.54 and 2.55. It follows immediately that every class covered by Lemma 2.54 has the computable $\mathrm{Av}(\pi)$-property for some π of length 3. For the remaining classes covered by Lemma 2.55, the computable $\mathrm{Av}(\pi)$-rich path property follows from Proposition 5.14. Finally, it suffices to apply Observation 5.13 and Theorem 5.6. □

This result leaves the hardness of $\mathrm{Av}(\sigma)$-PPM open only for 5 choices of σ up to symmetry. We collect all the remaining unresolved cases as an open problem.

Open problem 5.16. *What is the complexity of* $\mathrm{Av}(\sigma)$*-PPM, when σ is a permutation from the set* $\{3412, 3142, 4213, 4123, 41352\}$*?*

6. Grid classes

In this chapter, we focus our attention solely on grid classes. On one hand, we see how the results obtained in previous chapters apply in this specific case. However, we also derive new results tailored specifically for grid classes.

The organization of this chapter is as follows. In Section 6.1, we consider only monotone grid classes. We show that there exists a sharp trichotomy for the tree-width growth and hardness of Grid($\mathcal{M}$)-Pattern PPM determined by simple properties of the cell graph. Namely, there are three distinct regimes – (i) the cell graph is acyclic, (ii) it contains at most one cycle in each connected component, or (iii) it contains two connected cycles.

In Section 6.2, we extend this characterization to general grid classes. We fully characterize the grid classes of unbounded tree-width and moreover, we show that they coincide with the classes for which the problem Grid($\mathcal{M}$)-Pattern PPM is NP-complete.

6.1 Monotone grid classes

Depending on the properties of the cell graph $G_{\mathcal{M}}$, we distinguish between three main types of monotone gridding matrices. We say that a monotone gridding matrix $\mathcal{M}$ is

- *acyclic* if $G_{\mathcal{M}}$ is acyclic,
- *unicyclic* if $G_{\mathcal{M}}$ is not acyclic but it contains at most one cycle in each connected component, and
- *polycyclic* if $G_{\mathcal{M}}$ contains two connected cycles.

It has been previously observed that there is a sharp increase in structural complexity when going from acyclic to unicyclic classes. This is mostly captured by the fact that acyclic grid classes can be expressed as geometric grid classes (Proposition 1.4) which are well-behaved in many ways. For instance, geometric grid classes (and thus also acyclic grid classes) are well-quasi-ordered while unicyclic (and polycyclic) grid classes are not [7].

In this section, we confirm this behavior and moreover, we show that a second jump in complexity occurs when going from unicyclic to polycyclic classes. In particular, we exhibit a complete classification of the tree-width growth function as well as the complexity of pattern-restricted PPM for monotone gridding matrices. See Table 6.1 for summary.

6.1.1 Tree-width

Theorem 6.1. *For a monotone gridding matrix $\mathcal{M}$ one of the following holds:*

- *Either $\mathcal{M}$ is acyclic and* $\mathrm{pw}_{\mathrm{Grid}(\mathcal{M})} \in \Theta(1)$*, or*
- *$\mathcal{M}$ is unicyclic and* $\mathrm{tw}_{\mathrm{Grid}(\mathcal{M})}(n) \in \Theta(\sqrt{n})$*, or*
- *$\mathcal{M}$ is polycyclic and* $\mathrm{tw}_{\mathrm{Grid}(\mathcal{M})}(n) \in \Theta(n)$*.*

Property	acyclic	unicyclic	polycyclic
$\mathrm{tw}_{\mathrm{Grid}(\mathcal{M})}(n)$	$\Theta(1)$	$\Theta(\sqrt{n})$	$\Theta(n)$
Grid($\mathcal{M}$)-Pattern PPM	in P	NP-complete	NP-complete
– best known algorithm	$n^{O(1)}$	$n^{O(\sqrt{n})}$	2^n
– lower bound under ETH	—	$2^{o(\sqrt{n})}$	$2^{o(n)}$
Grid($\mathcal{M}$)-PPM	in P	in P	in P

Table 6.1: Tree-width growths and complexity of PPM for monotone grid classes.

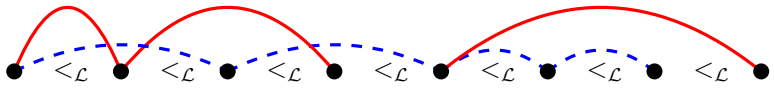

Figure 6.1: Two non-overlapping sets of edges (drawn by red solid and blue dashed lines, respectively) with respect to a linear order $<_\mathcal{L}$ of the vertices.

Recall that the lower bounds of all three cases were proved in Section 2.2. Moreover, the upper bound $\mathrm{tw}_{\mathrm{Grid}(\mathcal{M})}(n) \in O(n)$ for a polycyclic gridding matrix $\mathcal{M}$ is trivial. Therefore, it remains to prove the upper bounds for acyclic and unicyclic classes.

Geometric classes

First, we focus our attention on geometric grid classes. This will allow us on one hand to infer the constant upper bound for any acyclic monotone grid class. On the other hand, we will obtain along the way tools necessary for proving the upper bound on the tree-width of unicyclic grid classes later.

As the main standalone result regarding geometric grid classes, we show that any geometric grid class has bounded path-width. Note that the constant bound for acyclic monotone grid classes follows immediately from Proposition 1.4.

Proposition 6.2. *If $\mathcal{M}$ is a $k \times \ell$ monotone gridding matrix then*

$$\mathrm{pw}_{\mathrm{Geom}(\mathcal{M})}(n) \leq 4k + 4\ell - 4.$$

Before proving Proposition 6.2, we need to first expand our set of tools. Let $G = (V, E)$ be a graph together with a linear ordering $<_\mathcal{L}$ of its vertices. We say that a set of edges $F \subseteq E$ is *non-overlapping with respect to* $<_\mathcal{L}$ if for every two edges $\{u_1, u_2\}, \{v_1, v_2\} \in F$ it holds that either $u_\alpha \leq_\mathcal{L} v_\beta$ for all possible choices of $\alpha, \beta \in \{1, 2\}$ or $v_\beta \leq_\mathcal{L} u_\alpha$ for all possible choices of $\alpha, \beta \in \{1, 2\}$. In other words, we forbid two edges whose endpoints are ordered as $u_1 \leq_\mathcal{L} v_1 <_\mathcal{L} u_2 \leq_\mathcal{L} v_2$ or $u_1 \leq_\mathcal{L} v_1 <_\mathcal{L} v_2 \leq_\mathcal{L} u_2$. See Figure 6.1.

Let $\mathcal{M}$ be a $k \times \ell$ gridding matrix and let π be an $\mathcal{M}$-gridded permutation. An edge $\{(i, \pi_i), (j, \pi_j)\}$ of the incidence graph G_π is *horizontal* if $|i - j| = 1$, and it is *vertical* if $|\pi_i - \pi_j| = 1$. Thus, the horizontal edges form a left-to-right path, and the vertical ones form a bottom-to-top path. A horizontal edge is said to be *exceptional* (with respect to the given gridding) if its vertices belong to different columns of the gridding, and a vertical edge is *exceptional* if its vertices belong to different rows. There are therefore $k - 1$ exceptional horizontal edges and $\ell - 1$ exceptional vertical edges, hence at most $k + \ell - 2$ exceptional edges overall.

Assuming that a monotone gridding matrix $\mathcal{M}$ is consistently oriented, we show that there is a vertex ordering of any permutation $\pi \in \mathrm{Geom}(\mathcal{M})$ such that the edges of the incidence graph G_π can be partitioned into at most $O(1)$ non-overlapping sets. Later, this ordering is used for drawing incidence graphs on a surface with a small genus.

Lemma 6.3. *Let $\mathcal{M}$ be a monotone $k\times\ell$ gridding matrix equipped with a consistent $k \times \ell$ orientation $\mathcal{F}$, and let π be a permutation from* $\mathrm{Geom}(\mathcal{M})$ *equipped with an $\mathcal{M}$-gridding. There exists a linear order $<_{\mathcal{L}}$ on the points of π such that*

(a) the points of the i-th column of the $\mathcal{M}$-gridding induce a suborder given by the increasing order of their x-coordinates if $f_c(i) = 1$ and the decreasing order otherwise,
(b) the points of the j-th row of the $\mathcal{M}$-gridding induce a suborder given by the increasing order of their y-coordinates if $f_r(j) = 1$ and the decreasing order otherwise, and
(c) the edges of the graph G_π can be partitioned into at most $2k + 2\ell - 2$ non-overlapping sets with respect to $<_{\mathcal{L}}$.

Proof. We partition the exceptional edges into at most $k + \ell - 2$ singleton sets which are trivially non-overlapping with respect to arbitrary linear order.

Let S be the subset of the standard figure $\Lambda_{\mathcal{M}}$ isomorphic to S_π and let $g\colon S_\pi \to S$ be the corresponding bijection. For a point $p \in S_\pi$ such that p belongs to the (i, j)-cell of the $\mathcal{M}$-gridding, we define the rank of p as the distance of the point $g(p)$ from the point

$$\left(i - \frac{1}{2} - \frac{f_c(i)}{2},\ j - \frac{1}{2} - \frac{f_r(j)}{2}\right).$$

Less formally, we define the rank to be the distance to a specific corner of the rectangle $[i-1, i]\times[j-1, j]$, depending on the orientation of the (i, j)-cell. Observe that we can always choose S in a way such that the ranks are pairwise different. That allows us to define $<_{\mathcal{L}}$ as the linear order given by the ranks.

The conditions (a) and (b) follow straightforwardly from the consistency of the orientation $\mathcal{F}$ and our choices of suitable corners used for computing the ranks. Moreover, all the non-exceptional horizontal edges of G_π can be partitioned into k sets, each connecting points in a single column. And since we already know that points of a single column are ordered accordingly in $\mathcal{L}$, each such set is non-overlapping. Using the same argument, we split the vertical non-exceptional edges of G_π into ℓ non-overlapping sets, one for each row of the $\mathcal{M}$-gridding. Together with the exceptional edges, we obtain the partition of all edges into at most $2k + 2\ell - 2$ non-overlapping sets as promised. □

Now we show how Lemma 6.3 (c) implies that every geometric grid class has bounded path-width. In fact, we show that every geometric grid class has bounded a different parameter, called cut-width, and we use the fact that path-width of any graph is upper-bounded by its cut-width.

Informally, small cut-width captures that the vertices of a graph can be ordered in a way such that every cut between a prefix and suffix with respect to this order

contains only small number of edges. Given a graph G and a linear order $<_{\mathcal{L}}$ of its vertices, the *cut-width of G with respect to $<_{\mathcal{L}}$* is

$$\mathrm{cutw}_{<_{\mathcal{L}}}(G) = \max_{u \in V(G)} |\{\{v, w\} \in E(G) \mid v \leq_{\mathcal{L}} u <_{\mathcal{L}} w\}|.$$

The *cut-width* of a graph G, denoted by $\mathrm{cutw}(G)$, is defined as the minimum $\mathrm{cutw}_{<_{\mathcal{L}}}(G)$ over all possible vertex orderings $<_{\mathcal{L}}$. The following connection of cut-width to path-width is due to Korach and Solel [94] and based on an earlier proof of Bodlaender [27].

Lemma 6.4 ([27, 94])**.** *For any graph G, we have* $\mathrm{pw}(G) \leq \mathrm{cutw}(G)$.

Proof of Proposition 6.2. Observe that the refinement $\mathcal{M}^{\times 2}$ defines the same geometric grid class as $\mathcal{M}$. Therefore, we can assume without loss of generality that $\mathcal{M}$ possesses a consistent orientation $\mathcal{F}$ due to Lemma 2.35. The only caveat is that any bound we obtain assuming an existence of a consistent orientation must be adjusted by a factor of two due to the size increase occurring when we replace $\mathcal{M}$ with $\mathcal{M}^{\times 2}$.

Let π be a permutation from the class $\mathrm{Geom}(\mathcal{M})$ with the corresponding $\mathcal{M}$-gridding. We can apply Lemma 6.3 to obtain a linear order $<_{\mathcal{L}}$ on the points of π. We claim that G_π has small cut-width with respect to the order $<_{\mathcal{L}}$. To see that, observe that any cut respecting the order $<_{\mathcal{L}}$ contains at most one edge from each non-overlapping set and therefore, $\mathrm{cutw}_{<_{\mathcal{L}}}(G_\pi)$ is at most $2k + 2\ell - 2$. The desired upper bound on the path-width of π follows from Lemma 6.4. □

Unicyclic grid classes

We state and prove the upper bound on the tree-width growth of unicyclic grid classes as a standalone proposition.

Proposition 6.5. *If $\mathcal{M}$ is a unicyclic monotone gridding matrix then*

$$\mathrm{tw}_{\mathrm{Grid}(\mathcal{M})}(n) \in O(\sqrt{n}).$$

The last piece missing before before we can prove Proposition 6.5 is the following lemma about graphs drawn on surfaces with few crossings, which is proved using standard methods [59].

Lemma 6.6. *If G is a graph on n vertices that can be drawn on a surface with Euler genus g with $O(n)$ crossings, then* $\mathrm{tw}(G) \in O(\sqrt{gn})$.

Proof. Let $G = (V, E)$ be a graph on n vertices together with a drawing on a surface with Euler genus g. We assume that no three edges cross in a single point. We define a graph G' in the following way. We replace each crossing with a new vertex of degree 4, and we split each $e \in E$ into its consecutive segments between endpoints and crossings. It follows from the assumptions that G' has $O(n)$ vertices and moreover, it can be drawn on the surface with Euler genus g without any crossings. It follows from the work of Gilbert, Hutchinson and Tarjan [74], that $\mathrm{tw}(G') \in O(\sqrt{gn})$.

Using a tree decomposition (T, β) of G' with width t, we define a tree decomposition (T, β') of G with width $4t + 3$ by replacing every vertex corresponding to

a crossing with the four endpoints of the two edges participating in this particular crossing. Clearly, every vertex belongs to some bag $\beta(v)$, the same holds for every edge, and the size of each bag is at most $4t + 4$. It is also not hard to check that each vertex induces a connected subtree in T. □

Proof of Proposition 6.5. Let π be a permutation of Grid($\mathcal{M}$) with a given $\mathcal{M}$-gridding. We start by removing all exceptional edges. As we observed, there are at most $k+\ell-2$ exceptional edges. Let $G' = (V', E')$ be the graph obtained from G_π by removing them from G_π. It is sufficient to show that $\mathrm{tw}(G') \in O(\sqrt{n})$, as adding back the exceptional edges increases the tree-width at most by a constant.

We aim to show that G' can be drawn on a surface of Euler genus 1 with $O(n)$ total crossings. Suppose that $c_1, c_2, \ldots, c_m$ are the entries of $\mathcal{M}$ that lie on its only cycle in this order. We choose the starting entry c_1 to be a corner of the cycle, i.e., it shares a common row with its one neighbor on the cycle and a common column with the other. Let a_i, b_i be the coordinates of the entry c_i, i.e., c_i lies in the a_i-th column and b_i-th row. The cell graph $G_{\mathcal{M}}$ consists of the cycle and trees that are attached to it. If we remove all the edges that participate in the cycle, we end up with m trees $T_1, \ldots, T_m$ called *tendrils* such that the tree T_i contains the entry c_i.

We now define two functions $f_c^\star, f_r^\star \colon [m] \to \{-1, 1\}$ that will capture an almost consistent orientation of the entries $c_1, \ldots, c_m$. Let $\mathcal{M}'$ be the gridding matrix obtained from $\mathcal{M}$ by removing everything except for the entries on the cycle and then additionally removing c_1. It follows from our assumption about c_1 that $\mathcal{M}'$ is acyclic and thus, it has a consistent orientation $\mathcal{F}' = (f_c', f_r')$. For every $i \geq 2$, we set $f_c^\star(i), f_r^\star(i)$ to be the values $f_c'(a_i), f_r'(b_i)$. Additionally, we define $f_c^\star(1) = f_c'(a_1)$ and then we set $f_r^\star(1)$ such that $f_c^\star(1) \cdot f_r^\star(1) = 1$ if and only if c_1 is an increasing entry. In this way, it holds for every $i \in [m]$ that $f_c^\star(i) \cdot f_r^\star(i) = 1$ if and only if c_i is increasing. Moreover, if c_i and c_j share a common column then $f_c^\star(i) = f_c^\star(j)$. And finally, if c_i and c_j share a common row then also $f_r^\star(i) = f_r^\star(j)$ as long as i and j are different from 1.

Now for $i \in [m]$, let $\mathcal{M}_i$ be the gridding matrix obtained from $\mathcal{M}$ by removing everything except the tendril T_i and let π^i be the subpermutation of π induced by the non-empty entries of $\mathcal{M}_i$. Since $\mathcal{M}_i$ is acyclic, it has a consistent orientation $\mathcal{F}^i = (f_c^i, f_r^i)$. Moreover, we can assume that $f_c^i(a_i) = f_c^\star(i)$ and $f_r^i(b_i) = f_r^\star(i)$ as otherwise we could just flip all the signs in $\mathcal{F}^i$. By Proposition 1.4, the induced $\mathcal{M}_i$-gridding of π^i shows that π^i belongs to the geometric grid class Geom($\mathcal{M}_i$) since $\mathcal{M}_i$ is acyclic. Applying Lemma 6.3 on $\mathcal{M}_i$, $\mathcal{F}^i$ and π^i, we obtain a linear order $<_{\mathcal{L}_i}$ on the points of π^i.

We are ready to describe the drawing of G'. In order to simplify our arguments, we draw G' on an infinite cylinder $[0, m-1] \times \mathbb{R}$ where we implicitly work with the x-coordinates in arithmetic modulo m. Such a drawing can then easily be transformed into a drawing on the plane with the same number of crossings via a suitable projection. We call the vertical line $x = i$ the *i-th meridian*. First, let us describe the position of the vertices. For every $i \in [m]$, the vertices corresponding to points of π^i are placed on i-th meridian in the order $<_{\mathcal{L}_i}$ from bottom to top. The actual distances in between them do not matter.

We split the edges of E' into two groups. We call any edge that connects two points of π^i an *inner* edge, and every edge that connects a point in π^i with a point in π^j for $i \neq j$ an *outer* edge. We start by drawing the inner edges. We first draw every inner edge as a circular arc connecting its two endpoints and, afterwards,

we horizontally shrink all these edges so that they occupy only a narrow band around the i-th meridian.

Let us count the number of crossings between two inner edges. Due to the shrinking step, two inner edges can intersect only if they both connect two points of π^i for some i. We fix $i \in [m]$. Notice that the edges of E' induced by π^i form a subset of the edges of G_{π^i}. By part (c) of Lemma 6.3, these edges can be partitioned into at most $2k + 2\ell - 2$ sets $E^i_1, E^i_2, \ldots, E^i_{2k+2\ell-2}$ that are all non-overlapping with respect to $<_{\mathcal{L}_i}$. The non-overlapping property implies that any edge $e \in E^i_j$ cannot intersect any other edge from E^i_j, and it can intersect at most two edges from every $E^i_{j'}$ for $j' \neq j$. Therefore, each edge participates in at most $4k + 4\ell - 6 = O(1)$ crossings. Summing over all choices of i, there are at most $O(n)$ crossings between two inner edges.

Now, we describe the drawing of the outer edges. Observe that the outer edges can be partitioned into at most m sets depending on the column or row that is shared by both their endpoints. If an outer edge connects points of π^i and π^j such that $|i - j| = 1$, then we draw it as the straight-line segment between the two vertices that does not cross any meridian. We need to be more careful with the remaining edges. We draw an edge that connects points of π^i and π^j for $|i - j| > 1$ as a polyline consisting of straight-line segments between the h-th and $(h + 1)$-th meridian for every $h \in \{i, \ldots, j - 1\}$. Suitably choosing the points on each meridian, we can guarantee that the segments between h-th and $(h + 1)$-th meridian do not intersect as long as $<_{\mathcal{L}_h}$ and $<_{\mathcal{L}_{h+1}}$ order the points in c_h and c_{h+1} consistently.

If the whole matrix $\mathcal{M}$ possesses a consistent orientation, then Lemma 6.3 parts (a) and (b) imply that no two outer edges intersect as the orders of cells in a given row or a column all agree. Otherwise, this is true for every column and every row except for the row b_1. Assuming without loss of generality that c_1 shares a common row with c_2, the order of points of π^1 and π^2 on the 1st and 2nd meridian are reversed. In such case, the opposite happens and every two segments of outer edges in the band between 1st and 2nd meridian intersect. This is, however, easily fixed by adding a cross-cap in between these two meridians such that every segment of an outer edge lying in this band crosses through it. Thus, we obtained a drawing of G' either on a plane or on a projective plane. See Figure 6.2.

It remains to count the number of intersections formed between pairs consisting of an inner and an outer edge. Fix an outer edge e and recall that e is a polyline formed by at most m segments. A segment between the i-th and $(i + 1)$-th meridian can intersect only with inner edges of π^i in the near vicinity of the i-th meridian and with inner edges of π^{i+1} in the near vicinity of the $(i+1)$-th meridian. Moreover, it can intersect at most one edge from any non-overlapping set with respect to $<_{\mathcal{L}_i}$ or $<_{\mathcal{L}_{i+1}}$. Thus, it follows from Lemma 6.3 (c) that each segment intersects at most $4k+4\ell-4$ edges and e intersects at most $m \cdot (4k+4\ell-4) = O(1)$ inner edges in total. Putting it all together, there are $O(n)$ crossings in total and by Lemma 6.6 we get that $\mathrm{tw}(G') \in O\left(\sqrt{n}\right)$. □

This concludes the proof of Theorem 6.1.

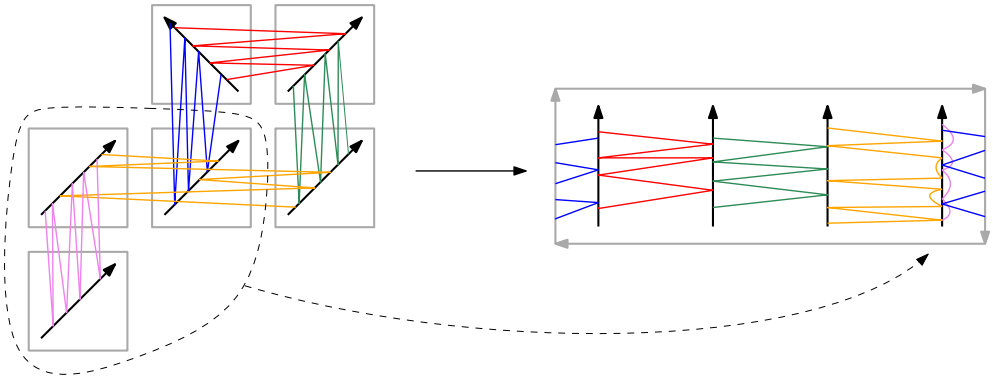

Figure 6.2: A schematic drawing of G_π for π from a unicyclic grid class on the projective plane. Instead of drawing the specific points of π, we place arrows to indicate the orientation of each cell. Different color is used for each set of edges that share a single row or column, and the exceptional edges are omitted. Note that edges connecting points in the same row (column) can be both inner and outer as is the case of the yellow edges.

6.1.2 Pattern matching

We complement Theorem 6.1 by showing that a similar classification exists for the complexity of Grid($\mathcal{M}$)-Pattern PPM. Note that previously not much was known about complexity of this problem. Neou, Rizzi and Vialette [101] designed a polynomial-time algorithm solving Grid$\left(\begin{smallmatrix}\diagdown & \\ & \diagdown\end{smallmatrix}\right)$-Pattern PPM for the class of the so-called wedge permutations. All other cases were wide open and Neou [100] asked about the hardness of Grid($\mathcal{M}$)-Pattern PPM for a monotone gridding matrix $\mathcal{M}$ in his thesis. We completely resolve this question.

Theorem 6.7. *For a monotone gridding matrix $\mathcal{M}$ one of the following holds:*

- *Either $\mathcal{M}$ is acyclic and* Grid($\mathcal{M}$)-Pattern *PPM is in* P*, or*
- *$\mathcal{M}$ is unicyclic,* Grid($\mathcal{M}$)-Pattern *PPM is* NP*-complete and it cannot be solved in $2^{o(\sqrt{n})}$ under ETH, or*
- *$\mathcal{M}$ is polycyclic,* Grid($\mathcal{M}$)-Pattern *PPM is* NP*-complete and it cannot be solved in $2^{o(n/\log n)}$ under ETH.*

Proof. The first case follows using the $O(n^{\mathrm{tw}(\pi)+1})$ algorithm of Theorem 4.12 since the tree-width of any acyclic monotone grid class is bounded by constant due to Theorem 6.1. As for the second case, any unicyclic monotone grid class has the poly-time computable long path property by Proposition 2.34 and thus, we can apply the hardness reduction of Theorem 4.15. Similarly, any polycyclic monotone grid class has the poly-time computable deep tree property by Proposition 2.41 and we can apply the hardness reduction of Theorem 4.18. □

We conclude this section with a remark that the situation is different and much easier in the case of the text-restricted variant of PPM. It follows directly from Theorem 5.5 that the problem Grid($\mathcal{M}$)-PPM is polynomial-time solvable for any monotone gridding matrix $\mathcal{M}$.

6.2 General grid classes

In this section, we generalize the results of Theorems 6.1 and 6.7 to any gridding matrix whose every entry has bounded tree-width. We shall see that in addition to cycles, there is only one other obstacle to bounded tree-width.

Let $\mathcal{M}$ be a gridding matrix whose every entry has bounded tree-width. We say that an ordered pair (p, q) of vertices in $G_{\mathcal{M}}$ is a *bumper* if either $\mathcal{M}_q$ has unbounded horizontal grid-width and shares the same column with $\mathcal{M}_p$, or if $\mathcal{M}_q$ has unbounded vertical grid-width and shares the same row with $\mathcal{M}_p$. A *bumper-ended path* is a path $P = p_1, \ldots, p_k$ in $G_{\mathcal{M}}$ such that both (p_2, p_1) and (p_{k-1}, p_k) are bumpers. An example is a path with endpoints $\mathrm{Av}(\sigma_1)$ and $\mathrm{Av}(\sigma_2)$ where σ_1 and σ_2 are arbitrary permutations of length at least 3.

We show that an existence of a bumper-ended path implies in a fairly straightforward way that $\mathrm{Grid}(\mathcal{M})$ has the cycle property and thus, unbounded tree-width. It is perhaps more surprising (and certainly harder to prove) that conversely, any acyclic class with no bumper-ended path has tree-width bounded by constant. Again, the same dividing line holds for the hardness of $\mathrm{Grid}(\mathcal{M})$-Pattern PPM.

Theorem 6.8. *Let $\mathcal{M}$ be a gridding matrix such that every entry of $\mathcal{M}$ has bounded tree-width. Then one of the following holds:*

- *Either $G_{\mathcal{M}}$ is a forest that avoids a bumper-ended path, $\mathrm{tw}_{\mathrm{Grid}(\mathcal{M})} \in \Theta(1)$ and $\mathrm{Grid}(\mathcal{M})$-Pattern PPM can be decided in polynomial time, or*
- *$G_{\mathcal{M}}$ contains a bumper-ended path or a cycle, $\mathrm{tw}_{\mathrm{Grid}(\mathcal{M})} \in \Omega(\sqrt{n})$ and $\mathrm{Grid}(\mathcal{M})$-Pattern PPM is NP-complete.*

Unlike monotone grid classes, the gridding matrices of general grid classes may contain finite non-empty classes. However, we can ignore these entries without affecting the properties we are interested in. To see this, let $\mathcal{M}'$ be the gridding matrix obtained by removing all finite entries from a gridding matrix $\mathcal{M}$. Note that the cell graph of $\mathcal{M}$ is equal to the cell graph of $\mathcal{M}'$. Moreover, the NP-completeness of $\mathrm{Grid}(\mathcal{M}')$-Pattern PPM trivially implies the NP-completeness of $\mathrm{Grid}(\mathcal{M})$-Pattern PPM. Finally, $\mathrm{Grid}(\mathcal{M}')$ has bounded tree-width if and only if $\mathrm{Grid}(\mathcal{M})$ has bounded tree-width since inserting a constant number of points into a permutation increases its tree-width at most by a constant. Thus, if $\mathcal{M}'$ satisfies one of the two options in Theorem 6.8, then $\mathcal{M}$ satisfies this option as well. From now on, we will assume that $\mathcal{M}$ contains only infinite (or empty) entries.

6.2.1 Classes with bumper-ended paths

We show that any grid class $\mathrm{Grid}(\mathcal{M})$ containing a bumper-ended path in its cell graph $G_{\mathcal{M}}$ has the cycle property. The second part of Theorem 6.8 then follows since $\mathrm{Grid}(\mathcal{M})$ has the poly-time computable long path property by Proposition 2.34 and we can apply the relevant results for classes with the long path property(Proposition 2.33 and Theorem 4.15).

Lemma 6.9. *Let $\mathcal{M}$ be a gridding matrix such that every entry of $\mathcal{M}$ has bounded tree-width. If $G_{\mathcal{M}}$ contains a bumper-ended path then $\mathrm{Grid}(\mathcal{M})$ has the cycle property.*

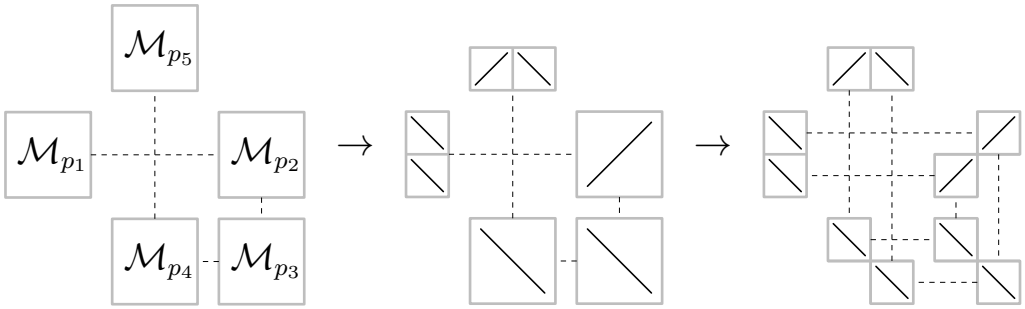

Figure 6.3: Transforming a bumper-ended path to cycle in the proof of Lemma 6.9.

Proof. Consider the bumper-ended path $p_1, p_2, \ldots, p_k$. Let us assume that p_1 shares a column with p_2 and p_k shares a column with p_{k-1}, and therefore both $\mathcal{M}_{p_1}$ and $\mathcal{M}_{p_k}$ have unbounded horizontal grid-width; the other cases can be proved analogously. Each of the infinite classes $\mathcal{M}_{p_i}$ contains a monotone subclass $\mathcal{C}_i$ due to the Erdős–Szekeres theorem [65]. Moreover, the classes $\mathcal{M}_{p_1}$ and $\mathcal{M}_{p_k}$ contain a horizontal monotone juxtaposition by Lemma 2.21. Let $\mathrm{Grid}(\mathcal{C}_1\ \mathcal{D}_1)$ be the juxtaposition contained in $\mathcal{M}_{p_1}$ and $\mathrm{Grid}(\mathcal{C}_k\ \mathcal{D}_k)$ the juxtaposition contained in $\mathcal{M}_{p_k}$. We define the monotone gridding matrix $\mathcal{M}'$ by replacing every entry of $\mathcal{M}$ with the following 2×2 matrix:

- entry $\mathcal{M}_{p_i}$ for i between 2 and $k-1$ is replaced with $\mathcal{C}_i^{\times 2}$
- entry $\mathcal{M}_{p_t}$ for $t \in \{1, k\}$ is replaced with $\left(\begin{smallmatrix} \emptyset & \emptyset \\ \mathcal{C}_t & \mathcal{D}_t \end{smallmatrix}\right)$, and
- every other entry is replaced with an empty 2×2 matrix.

See Figure 6.3.

Clearly, $\mathrm{Grid}(\mathcal{M}')$ is a subclass of $\mathrm{Grid}(\mathcal{M})$. The (i,j)-block of $\mathcal{M}'$ is the 2×2 submatrix obtained from the (i,j)-cell in $\mathcal{M}$. If we forget about the blocks of p_1 and p_k we are left with two disjoint copies of the original path. Adding back the blocks connects the endpoints of both paths together and creates a cycle since p_2 shares the same column with p_1 and p_{k-1} shares the same column with p_k. Thus, $\mathrm{Grid}(\mathcal{M}')$ is a monotone grid subclass of $\mathrm{Grid}(\mathcal{M})$ whose cell graph contains a cycle. □

6.2.2 Classes without bumper-ended paths

For the rest of this section, we choose to work with grid-width rather than tree-width as it is more convenient. This does not affect any of the results since grid-width and tree-width are equal up to a multiplicative constant by Corollary 2.12. We show that any acyclic grid class avoiding a bumper-ended path has bounded grid-width. The first part of Theorem 6.8 then follows since PPM can be solved in time $O(n^{\mathrm{tw}(\pi)+1})$ due to Theorem 4.12.

Recall that the intervalicity of a set $A \subset \mathbb{N}$, denoted $\mathrm{int}(A)$, is the size of the smallest interval family $\mathcal{I}$ such that $\bigcup \mathcal{I} = A$. For a point set $S \subset \mathbb{N}^2$, we let $\Pi_x(S)$ be the projection of S onto the x-axis and $\Pi_y(S)$ the projection of S onto the y-axis. The grid-complexity of S is then defined as the maximum of $\mathrm{int}(\Pi_x(S))$ and $\mathrm{int}(\Pi_y(S))$. In the rest of this section, we use $\mathrm{int}_x(S)$ and $\mathrm{int}_y(S)$ as shorthands for $\mathrm{int}(\Pi_x(S))$ and $\mathrm{int}(\Pi_y(S))$, respectively.

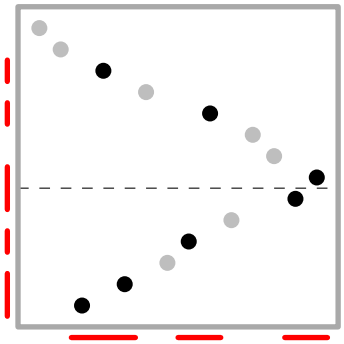

Figure 6.4: Illustration of Lemma 6.11. For any subset S of a permutation $\pi \in \mathrm{Grid}\left(\begin{smallmatrix} \diagdown & \\ & \diagdown \end{smallmatrix}\right)$, we have $\mathrm{int}_y(S) \leq 2 \cdot \mathrm{int}_x(S)$. The red segments along the borders of the diagram show the intervalicity of each projection.

Proposition 6.10. *Let $\mathcal{M}$ be a gridding matrix such that every entry of $\mathcal{M}$ has bounded grid-width. If $G_{\mathcal{M}}$ is a forest that avoids a bumper-ended path then* $\mathrm{Grid}(\mathcal{M})$ *has bounded grid-width.*

Due to the technical nature of the proof, we first briefly describe the main idea. Let $\mathcal{M}$ be an acyclic gridding matrix avoiding a bumper ended-path and let $\pi \in \mathrm{Grid}(\mathcal{M})$. First, we show that we can find a 'central' entry r in $\mathcal{M}$ such that no path starting in r ends with a bumper.

Let π_r be the subpermutation of π contained in the cell r. We take an optimal grid tree T_r of π_r with grid-width g. Using T_r as a guide, we construct a grid tree T of the whole permutation π. When reading T_r from the root, we can interpret every vertex as splitting a set of grid-complexity at most g into two sets of grid-complexity at most g. Suppose that r occupies the (i, j)-cell. Instead of cutting up only the points in the cell r, we split in T both the i-th column and j-th row into two sets of at most g strips such that on π_r, this splitting coincides with T_r.

If we can show that there is a function f such that this splitting has grid-complexity at most $f(g)$ inside any cell v in the i-th column, we can use the same procedure to split the row occupied by v into two unions of at most $f(g)$ horizontal strips (and symmetrically for the cells in the j-th row). Applying this idea recursively, we can 'spread' the partition of π_r into a partition of the whole permutation π whose grid-complexity remains bounded even though it quickly increases with the distance from r. Luckily, the grid-complexity remains bounded at each step of this process as a result of the fact that no path starting in r ends with a bumper.

The final missing piece before the proof of Proposition 6.10 is the following lemma. We prove that whenever we select k horizontal strips in a permutation π of bounded horizontal grid-width, the resulting subset of π has grid-complexity at most $\alpha \cdot k$ for some constant α. Naturally, a symmetric version holds for vertical grid-width. See Figure 6.4.

Lemma 6.11. *Let π be a permutation from a class $\mathcal{C}$ with bounded horizontal grid-width and let S be a subset of π such that* $\mathrm{int}_x(S) = k$. *Then* $\mathrm{int}_y(S) \leq \alpha \cdot k$ *where the constant α depends only on $\mathcal{C}$.*

Proof. By Lemma 2.21, there exists an l such that $\mathcal{C}$ does not contain any horizontal alternation of size l. Let $\mathcal{I}$ be the interval family of size k such that $\bigcup\mathcal{I} = \Pi_x(S)$ and let I be an interval of $\mathcal{I}$. Let S_I be the subset of S such that $\Pi_x(S_I) = I$ and let $\mathcal{J}$ be the smallest interval family such that $\Pi_y(S_I) = \bigcup\mathcal{J}$. We claim that $\mathcal{J}$ contains at most $2l-1$ intervals. For contradiction, suppose that the size of $\mathcal{J}$ is at least $2l$. Then each pair of consecutive intervals in $\mathcal{J}$ is separated by a value j such that π_j^{-1} lies outside the interval I. There is at least $2l-1$ gaps between intervals of $\mathcal{J}$ and therefore by the pigeon-hole principle either l of them contain a point to the right of I or at least l of the gaps contain a point to the left of I. Either way, we obtain a horizontal alternation of size l, which is a contradiction.

For each interval $I \in \mathcal{I}$, we showed that the intervalicity of $\Pi_y(S_I)$ is at most $2l-1$ and thus, the intervalicity of $\Pi_y(S)$ is at most $k \cdot (2l-1)$. □

Proof of Proposition 6.10. First, suppose that $G_{\mathcal{M}}$ contains more than one component. In that case, choose a component of $G_{\mathcal{M}}$, and let $\mathcal{M}_1$ be the submatrix of $\mathcal{M}$ spanned by the rows and columns containing the vertices of the chosen component, while $\mathcal{M}_2$ is the submatrix spanned by the remaining rows and colums of $\mathcal{M}$. An $\mathcal{M}$-gridded permutation π can be partitioned into two subpermutations π_1 and π_2 where π_i is the $\mathcal{M}_i$-gridded subpermutation of π consisting of the rows and columns of $\mathcal{M}_i$. Let T_i be the optimal grid tree of π_i. We define a grid tree T of π by taking a root vertex with children T_1 and T_2. The grid-width of any vertex in T_1 or T_2 has increased at most by $\max(k,\ell)$ where k and ℓ are the dimensions of $\mathcal{M}$. Therefore $\mathrm{gw}(\pi) \leq \max(\mathrm{gw}(\pi_1), \mathrm{gw}(\pi_2)) + \max(k,\ell)$. Applying this argument inductively shows that $\mathrm{Grid}(\mathcal{M})$ has bounded grid-width if and only if the grid-width of $\mathrm{Grid}(\mathcal{M}')$ is bounded for every submatrix $\mathcal{M}'$ of $\mathcal{M}$ spanned by a connected component $G_{\mathcal{M}}$. In the rest of the proof, we assume that $G_{\mathcal{M}}$ is a tree. The proof is based on a sequence of claims that will be stated and proven independently.

Claim 6.12. *The tree $G_{\mathcal{M}}$ contains a vertex r such that for any other vertex $q \neq r$, the path from r to q does not end in a bumper.*

Assume for contradiction that the claim fails. Let r be any vertex of $G_{\mathcal{M}}$. By assumption, there is a vertex $q \neq r$ such that the path from r to q ends in a bumper (p,q). Choose such a vertex q as far as possible from r. Applying our assumption for q in the role of r, there is a vertex $q' \neq q$ such that the path from q to q' ends in a bumper (p',q'). If the path from q to q' contains the vertex p, then it is a bumper-ended path, which is impossible. If the path from q to q' avoids p, it means that the path from r to q' ends in the bumper (p',q') and is strictly longer than the path from r to q, which contradicts the choice of q. This proves Claim 6.12.

Let r be the vertex of $G_{\mathcal{M}}$ whose existence is guaranteed by Claim 6.12. We now define a rooted tree $T_{\mathcal{M}}$ on the same vertex set as $G_{\mathcal{M}}$ as follows (see Figure 6.5). The vertex r is the root $T_{\mathcal{M}}$. For a vertex $v \neq r$ in $G_{\mathcal{M}}$, we set the parent of v in $T_{\mathcal{M}}$ to be the furthest vertex on the vr-path in $G_{\mathcal{M}}$ that shares the same row or column with v. Observe that whenever a vertex v shares the same column with its parent w in $T_{\mathcal{M}}$ then the entry $\mathcal{M}_v$ has bounded horizontal grid-width, and whenever v shares the same row with w, the entry $\mathcal{M}_v$ has bounded vertical grid-width. The *dominant cell* of a row, or a column, is the cell v such that all the other cells in the row, or column, are its children in $T_{\mathcal{M}}$.

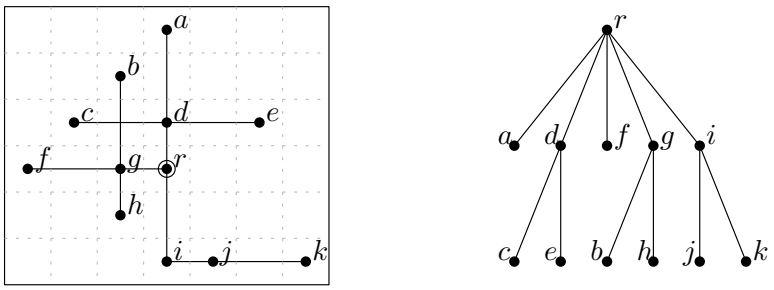

Figure 6.5: Example of a gridding matrix $\mathcal{M}$ with the tree $G_{\mathcal{M}}$ (left) and the corresponding rooted tree $T_{\mathcal{M}}$ (right).

Let π be an $\mathcal{M}$-gridded permutation, and let π_v denote the subset of points contained in a cell v. Assume that π_v is non-empty whenever $\mathcal{M}_v$ is a non-empty cell of $\mathcal{M}$. We define an auxiliary directed graph H on the points of π whose every connected component is a tree rooted in some point of π_r. Suppose that the vertex v of $T_{\mathcal{M}}$ shares the same column with its parent w. The parent of a point p in π_v is the nearest point in π_w to the right of p, and if there is no such point (p lies to the right of all the points of π_w) then the rightmost point in π_w. If v and w share the same row, then the parent of p is the nearest point in π_w above p, or if there is no such point (p lies above all the points of π_w) then the topmost point in π_w. Let P be a subset of the permutation diagram of π. The point set $\overline{P}$ contains P and every point that lies in H in a subtree of some point $p \in P$.

Let v be a non-empty cell such that π_v contains m points. Recall that $\mathrm{red}(\pi_v)$ is the reduction of π_v, i.e. the point set inside $[m] \times [m]$ that is isomorphic to the point set π_v. The construction of the graph H guarantees the following property and its symmetric version.

Observation 6.13. *Let v be the dominant cell of its row. Let S be a subset of π_v, let $\widetilde{S}$ be the corresponding subset of the reduction* $\mathrm{red}(\pi_v)$ *and let S' be the set containing S and all its children in H that lie in the same row. Then* $\mathrm{int}_y(\widetilde{S}) = \mathrm{int}_y(S')$. *Symmetrically, if v is dominant in its column, S and $\widetilde{S}$ are as above, and S' contains S and all its children in H in the same column, then* $\mathrm{int}_x(\widetilde{S}) = \mathrm{int}_x(S')$.

We inductively define a function h on the vertex set of $T_{\mathcal{M}}$, which will later serve us as an upper bound for the grid-width of any $\pi \in \mathrm{Grid}(\mathcal{M})$. For any leaf u of $T_{\mathcal{M}}$ we set $h(u) = 1$. For any other vertex v, we let W be the set of children of v and define $h(v) = 1 + \sum_{w \in W} \alpha_w h(w)$ where α_w is the constant obtained as follows. If v shares a column with w, then α_w is the constant from Lemma 6.11 applied on the class $\mathcal{M}_w$, otherwise it is the constant from the 'vertical' version of Lemma 6.11 applied on the class $\mathcal{M}_w$. We state only one of the symmetric versions of the following two claims. However, we are proving both of them simultaneously by induction.

Claim 6.14. *Let S be a subset of the i-th column of the $\mathcal{M}$-gridding of π such that* $\mathrm{int}_x(S) = 1$. *Let v be the dominant cell of the i-th column and let $S_v = S \cap \pi_v$. Then* $\mathrm{int}_x(\overline{(S \setminus S_v)} \cup S_v) \leq h(v)$ *and* $\mathrm{int}_y(\overline{S \setminus S_v}) \leq h(v) - 1$.

We remark that if v is not equal to the root r, then the set $\overline{(S \setminus S_v)} \cup S_v$ from the claim above is actually equal to $\overline{S}$.

To prove Claim 6.14, suppose first that v is the only non-empty cell in its column, and therefore v is a leaf of $T_{\mathcal{M}}$. Then $S = S_v$, and hence $\mathrm{int}_x(\overline{(S \setminus S_v)} \cup S_v) = \mathrm{int}_x(S) = 1 = h(v)$ and $\mathrm{int}_y(\overline{S \setminus S_v}) = \mathrm{int}_y(\emptyset) = 0 = h(v) - 1$, as claimed.

Now suppose that v is not the only non-empty cell in its column, and let C be the set of nonempty cells different from v in the same column as v. Note that each cell in C is a child of v in $T_{\mathcal{M}}$, and if $v \neq r$, then C is precisely the set of children of v. Observe that $\overline{S \setminus S_v}$ is a disjoint union of the sets $\overline{S_w}$ over all $w \in C$. For a cell $w \in C$, let S_w be the set $S \cap \pi_w$ and let $\widetilde{S_w}$ be the subset of $\mathrm{red}(\pi_w)$ that corresponds to S_w. From Lemma 6.11 we get $\mathrm{int}_y(\widetilde{S_w}) \leq \alpha_w$, and in particular, $\widetilde{S_w}$ can be partitioned into sets $\widetilde{S_w^1}, \widetilde{S_w^2}, \ldots, \widetilde{S_w^{\ell_w}}$ for some $\ell_w \leq \alpha_w$, where $\mathrm{int}_y(\widetilde{S_w^i}) = 1$ for each i. Let S_w^i be the subset of π_w that corresponds to $\widetilde{S_w^i}$ in the reduction $\mathrm{red}(\pi_w)$. Let R_w^i be the set S_w^i together with all its children in H that lie in the same row. By Observation 6.13, for each i we have $\mathrm{int}_y(R_w^i) = 1$. Using the symmetric version of Claim 6.14 with each R_w^i in the role of S shows that

$$\begin{aligned}
\mathrm{int}_y(\overline{S \setminus S_v}) &\leq \sum_{w \in C} \mathrm{int}_y(\overline{S_w}) \\
&\leq \sum_{w \in C} \sum_{i=1}^{\ell_w} \mathrm{int}_y(\overline{S_w^i}) \\
&= \sum_{w \in C} \sum_{i=1}^{\ell_w} \mathrm{int}_y\left(\overline{(R_w^i \setminus S_w^i)} \cup S_w^i\right) \\
&\leq \sum_{w \in C} \alpha_w h(w) \\
&\leq h(v) - 1,
\end{aligned}$$

and similarly,

$$\begin{aligned}
\mathrm{int}_x(\overline{(S \setminus S_v)} \cup S_v) &= \mathrm{int}_x\left(S \cup \bigcup_{w \in C} \overline{S_w}\right) \\
&= \mathrm{int}_x\left(S \cup \bigcup_{w \in C} \bigcup_{i=1}^{\ell_w} \overline{R_w^i \setminus S_w^i}\right) \\
&\leq 1 + \sum_{w \in C} \sum_{i=1}^{\ell_w} \mathrm{int}_x(\overline{R_w^i \setminus S_w^i}) \\
&\leq 1 + \sum_{w \in C} \alpha_w (h(w) - 1) \\
&\leq h(v),
\end{aligned}$$

proving Claim 6.14.

We will now define, for every $p \in \pi$, a grid tree T_p whose leaves are exactly the points in $\overline{\{p\}}$. The definition proceeds inductively on the size of $\overline{\{p\}}$. If p has no children in H, that is if $\overline{\{p\}} = \{p\}$, then T_p consists of the single vertex p. Suppose now that p has at least one child in H. Recall that each child of p belongs to a cell in the gridding of π which is in the same row or in the same column as the cell of p. Let C and R denote, respectively, the set of children of p in the same column and the set of children of p in the same row. Note that C and R are disjoint, and if p does not belong to the root cell π_r then one of C and R is empty.

Recall that a caterpillar is a binary tree whose every internal node has at least one leaf child. The leaves of a caterpillar can be ordered top to bottom by their distance from the root, where the order of the bottommost pair of leaves is irrelevant.

If C is nonempty, we construct a tree T_p^C in the following two steps:

- Construct a caterpillar whose leaves are the points from $C \cup \{p\}$, and the top-to-bottom order of the leaves in the caterpillar coincides with the left-to-right order of the points in π.
- In the caterpillar constructed above, for each $q \in C$ replace the leaf q with a copy of the tree T_q. Call the resulting tree T_p^C.

Symmetrically, if R is nonempty, construct a tree T_p^R by first taking the caterpillar whose leaves in top-to-bottom order are the points of $R \cup \{p\}$ in top-to-bottom order, and then for each $q \in R$ replace the leaf q with a copy of T_q.

If the set R is empty, we define $T_p = T_p^C$, and if C is empty, we define $T_p = T_p^R$. If both C and R are non-empty (which may only happen when p is in π_r), we let T_p be the tree obtained by replacing the leaf p in T_p^C by a copy of T_p^R. Note that in all the cases, the leaves of T_p form precisely the set $\overline{\{p\}}$.

Claim 6.15. *Let v be a nonempty cell of $\mathcal{M}$. If $v \neq r$, then for every $p \in \pi_v$, the tree T_p has grid-width at most $h(v)$. For every $p \in r$, the tree T_p has grid-width at most $2h(r)$.*

We prove the claim by induction on the size of T_p. The claim clearly holds when T_p is the single vertex p. Suppose now that T_p has more vertices, and that $v \neq r$. In such case T_p is equal to T_p^C or to T_p^R. Suppose that $T_p = T_p^C$, the other case being symmetric. Let C be again the set of children of p in H (necessarily, they are all in the same column of the gridding as the point p, since v is not the root vertex). Let u be a node of T_p, and let L_u be the set of leaves of the subtree of T_p rooted at u. Our goal is to show that the grid complexity of L_u is at most $h(v)$. If u is the leaf p or u is inside a copy of T_q for some $q \in C$, the claim follows by induction. Suppose that u is a node of the caterpillar from which T_p was constructed. Let S be the set of points in L_u that are in the same gridding column as p. By the construction of T_p, the set S satisfies $\mathrm{int}_x(S) = 1$, and L_u is equal to $\overline{S}$. Note that the set $S_v = S \cap \pi_v$ is either empty or contains the single point p. By Claim 6.14,

$$\mathrm{int}_x(L_u) = \mathrm{int}_x(\overline{S}) = \mathrm{int}_x(\overline{(S \setminus S_v)} \cup S_v) \leq h(v)$$

and

$$\begin{aligned}\mathrm{int}_y(L_u) \leq \mathrm{int}_y(S_v) + \mathrm{int}_y(\overline{S \setminus S_v}) &\leq \mathrm{int}_y(\{p\}) + \mathrm{int}_y(\overline{S \setminus S_v}) \\ &\leq 1 + (h(v) - 1) = h(v).\end{aligned}$$

This shows that L_u has grid complexity at most $h(v)$, and therefore T_p has grid-width at most $h(v)$.

It remains to deal with the case when p belongs to the root cell r. Using the same argument as in the first part of the proof, we again see that both T_p^C and T_p^R have grid-width at most $h(r)$. Moreover, for each node u of T_p the subtree of T_p rooted at u is either equal to a subtree of T_p^C, or it is equal to a subtree of T_p^R,

or it contains the entire tree T_p^R together with a subtree of T_p^C. In the former two cases, the set of leaves of the subtree has grid complexity at most $h(r)$, in the last case it has grid complexity at most $2h(r)$. This proves Claim 6.15.

We are ready to construct a grid tree T of the permutation π and provide a bound on its grid-width. By assumption, the entries of $\mathcal{M}$ have bounded grid-width, and we let g be the grid-width of the root entry $\mathcal{M}_r$. Let T_r be the optimum grid tree of $\mathrm{red}(\pi_r)$; in particular, T_r has grid-width at most g. A grid tree T of the whole permutation π is obtained by taking T_r and replacing the leaf corresponding to a point $p \in \pi_r$ with the tree T_p. We claim that T has grid-width at most $4gh(r)$. The tree T contains every point of π, and we showed in Claim 6.15 that the grid-width of any node contained in a copy of some T_p is at most $2h(r)$.

Let now u be a node of T that is not contained in any copy of the tree T_p, in other words, u is an internal node of T_r. Let $L^* \subseteq \mathrm{red}(\pi_r)$ be the set of leaves of T_r in the subtree rooted at u, and let L be the subset of π_r that is mapped to L^* by the bijection that maps π_r to $\mathrm{red}(\pi_r)$. Then the subset of π contained in the subtree of T rooted at u is precisely $\overline{L}$.

Applying Observation 6.13, we see that L together with its neighbors in H spans at most g consecutive intervals in the row and column of the r-cell. By applying Claim 6.14 individually on each of these $2g$ intervals, we get that the grid-complexity of $\overline{L}$ is at most $4gh(r)$. It follows that $\mathrm{Grid}(\mathcal{M})$ has bounded grid-width. □

7. Generalized coloring

In this chapter, we focus on a different algorithmic problem involving permutations. Namely, we study the algorithmic complexity of determining whether a permutation π can be obtained as a merge of two permutations $\sigma \in \mathcal{C}$ and $\tau \in \mathcal{D}$ for fixed permutation classes $\mathcal{C}$ and $\mathcal{D}$. Formally, we define it as a problem of recognizing permutations from the class $\mathcal{C} \odot \mathcal{D}$, which we view as a specific variant of the more general problem called $\mathcal{C}$-RECOGNITION.

$\mathcal{C}$-RECOGNITION	
Input:	A permutation π of length n.
Output:	Does π belong to $\mathcal{C}$?

The organization of this chapter is as follows. We start by summarizing previous knowledge of this problem in Section 7.1.

In Section 7.2, we develop several general frameworks of recognizing permutation classes. Our first approach, which we present in Subsection 7.2.1, is based on the concept of non-deterministically logspace on-line recognizable (or NLOL-recognizable) permutation classes. We will show that an arbitrary merge of NLOL-recognizable classes is polynomially recognizable. While this approach is conceptually quite simple, it generalizes all the previously known examples of tractable merges. As a second approach, we introduce a tree automata-based recognition scheme, called BD, in Subsection 7.2.2. Finally, we introduce in Subsection 7.2.3 the GT-recognizable classes, whose membership can be decided by evaluation of a tree automaton over grid trees.

In Section 7.3, we combine the GT- and BD-recognizability to prove that the merge of a BD-recognizable class with a GT-recognizable class of bounded grid-width can be recognized in polynomial time. This approach allows us to handle further cases of natural permutation classes that are not NLOL-recognizable, such as the class Av(213), or the class of separable permutations. For a summary of the known and new tractability results, see Table 7.1.

To complement our tractability results, we also provide, in Section 7.4, an NP-hardness result. In particular, we show that $\mathrm{Av}(\alpha) \odot \mathrm{Av}(\alpha)$-RECOGNITION is NP-complete for any simple pattern α of length at least 4.

7.1 Previous results

The notion of merge has been originally introduced as an approach for the enumeration of pattern-avoiding permutations [5, 4]. For instance, Claesson et al. [53] have shown that every 1324-avoiding permutation can be obtained by merging a 132-avoiding permutation with a 213-avoiding one, and this result, and its subsequent strengthenings by Bóna [30, 29] and Bevan et al. [26], are the basis of the best known upper bounds for the number of 1324-avoiding permutations.

Apart from enumeration questions, the research on permutation merges has also addressed structural issues, such as whether a given permutation class can be obtained by merging two of its proper subclasses [89, 87], or which classes can be obtained by merging a bounded number of permutations from a given

class [3, 90, 111]. This is, however, the first effort to address algorithmic aspects of permutation merges.

The decision problem of recognizing permutations from $\mathcal{C} \odot \mathcal{D}$ can be viewed as a permutation analogue of the generalized graph coloring problem from graph theory. For a fixed k-tuple $\mathcal{G}_1, \ldots, \mathcal{G}_k$ of graph classes, a generalized coloring of a graph is an assignment of colors $1, 2, \ldots, k$ to its vertices so that the vertices of color i induce a subgraph from $\mathcal{G}_i$. In particular, if all the $\mathcal{G}_i$ are equal to the class of edgeless graphs, this notion reduces to the classical notion of k-coloring. The research into the complexity of generalized graph coloring was initiated by Rutenburg [105], who considered graph properties defined by a finite set of forbidden subgraphs. Later, Farrugia [66] showed that if all the $\mathcal{G}_i$ are hereditary and additive (i.e., closed under taking induced subgraphs and forming disjoint unions) then the problem is NP-hard, except the trivially polynomial case when $k = 2$ and both $\mathcal{G}_1$ and $\mathcal{G}_2$ are equal to the class of edgeless graphs. Further results in this area were obtained, e.g., by Brown [47], Alexeev et al. [12], Achlioptas et al. [1], or Borowiecki [38].

As with generalized graph coloring, the recognition of permutation merges admits several cases which are trivially polynomial. For instance, let $\mathcal{I}_k$ be the class of permutations that can be merged from at most k increasing subsequences, or equivalently, of permutations that avoid the pattern $k+1, k, \ldots, 1$. Similarly, let $\mathcal{D}_k$ be the permutations merged from at most k decreasing subsequences, which are exactly the avoiders of $1, 2, \ldots, k+1$. One may easily see that $\mathcal{I}_k \odot \mathcal{I}_\ell = \mathcal{I}_{k+\ell}$ and $\mathcal{D}_k \odot \mathcal{D}_\ell = \mathcal{D}_{k+\ell}$, and in particular, these merges are trivially polynomially recognizable. Moreover, Kézdy et al. [90] have shown that for any $k, \ell \geq 1$, the class $\mathcal{I}_k \odot \mathcal{D}_\ell$ has only finitely many minimal excluded patterns, and therefore these classes are polynomially recognizable as well.

There are only two other non-trivial results concerning $\mathcal{C} \odot \mathcal{D}$-RECOGNITION that we are aware of. Ekim et al. [62] studied the complexity of generalized 2-colorings when the input graph is restricted to the class of the so-called permutation graphs. Their results, in our terminology, imply the polynomial recognition of $\mathcal{L} \odot \boxslash$ and of $\mathcal{L} \odot \overline{\mathcal{L}}$, where $\mathcal{L}$ and $\overline{\mathcal{L}}$ denote the classes of layered and co-layered permutations. Broersma et al. [46] studied the complexity of partitioning graphs so that each part is a union of disjoint cliques. Their results for permutation graphs translate into a polynomial-time algorithm for $\mathcal{L} \odot \mathcal{L}$-RECOGNITION and by symmetry also for $\overline{\mathcal{L}} \odot \overline{\mathcal{L}}$-RECOGNITION. See Table 7.1 for a summary of both previous results and our contributions.

7.2 Different recognition frameworks

7.2.1 NLOL-recognizable classes

Our first nontrivial example of classes whose merges can be efficiently recognized are the so-called NLOL-recognizable permutation classes. Informally speaking, a permutation class is NLOL-recognizable if its members can be recognized by a single-pass nondeterministic streaming algorithm with logarithmic memory.

More formally, we say that a permutation class $\mathcal{C}$ is *nondeterministically logspace on-line* recognizable, or NLOL-recognizable for short, if there is a nondeterministic algorithm A that recognizes $\mathcal{C}$ in the following setting: as the first

$\mathcal{D} \setminus \mathcal{C}$	Av(21)	Av$(k \cdots 1)$	layered	Av(213)	sep.	Av(3142)
Av(21)	Av(321)	Av$((k+1) \cdots 1)$	P [62]	$\mathbf{P}^\dagger$	$\mathbf{P}^\dagger$	?
Av(12)	Av $\binom{2143,}{3412}$	P [90]	$\mathbf{P}^\star$	$\mathbf{P}^\dagger$	$\mathbf{P}^\dagger$	?
Av$(k \cdots 1)$		Av$((2k-1) \cdots 1)$	$\mathbf{P}^\star$	$\mathbf{P}^\dagger$	$\mathbf{P}^\dagger$	?
Av$(1 \cdots k)$		P [90]	$\mathbf{P}^\star$	$\mathbf{P}^\dagger$	$\mathbf{P}^\dagger$	?
layered			P [46]	$\mathbf{P}^\dagger$	$\mathbf{P}^\dagger$	?
co-layered			P [62]	$\mathbf{P}^\dagger$	$\mathbf{P}^\dagger$	?
Av(213)				?	?	?
separable					?	?
Av(3142)						**NP-c**

Table 7.1: The table shows known and new complexity results for $\mathcal{C} \odot \mathcal{D}$-RECOGNITION. The boldface entries are the new results of this chapter, in particular entries $\mathbf{P}^\star$and $\mathbf{P}^\dagger$ follow from Corollary 7.2 and Corollary 7.8, respectively. Note that some columns are omitted due to symmetries.

part of the input, the algorithm A receives a number n, which is an upper bound on the length and also on the largest value in the input sequence. The algorithm is then given access to $O(\log n)$ bits of memory, and it receives a sequence of distinct values $\pi_1, \pi_2, \ldots, \pi_k$ from the set $[n]$, terminated by a special symbol `EOF`. Upon receiving the `EOF` symbol, A answers whether the input sequence is order-isomorphic to a permutation in $\mathcal{C}$. The algorithm can store arbitrary data of size $O(\log n)$ in its memory, but as soon as it reads the input value π_i, it can no longer access the previous values of the input. A is nondeterministically recognizing $\mathcal{C}$ in the sense that the input sequence is order-isomorphic to a permutation in $\mathcal{C}$ if and only if at least one computation of A accepts it. The algorithm A is then called an *NLOL-recognizer* of $\mathcal{C}$. Note that the input sequence is guaranteed to consist of distinct values, so the NLOL-recognizer itself does not need to verify this property. This also implies that the input sequence has length at most n.

We let NLOL denote the set of the NLOL-recognizable permutation classes. Clearly, for any permutation class $\mathcal{C} \in$ NLOL, the $\mathcal{C}$-RECOGNITION problem is tractable, since nondeterministic logspace computations can be simulated in polynomial time.

One may easily observe that NLOL contains any finite permutation class, as well as the classes Av(12) and Av(21). The key feature of NLOL is that it is closed under many important operations with permutation classes, including the merge operation.

Lemma 7.1. *If $\mathcal{C}$ and $\mathcal{D}$ are NLOL-recognizable classes, then the following classes are NLOL-recognizable as well:*

(a) The classes $\mathcal{C} \cap \mathcal{D}$ and $\mathcal{C} \cup \mathcal{D}$.
(b) The classes $\mathcal{C}^r$ and $\mathcal{C}^c$, which contain the reverses and the complements of the permutations of $\mathcal{C}$.
(c) The classes $\mathcal{C}^\oplus$ and $\mathcal{C}^\ominus$, i.e., the sum-closure and skew-closure of $\mathcal{C}$.
(d) The classes $\mathcal{C} \oplus \mathcal{D}$ and $\mathcal{C} \ominus \mathcal{D}$, and more generally, any grid class Grid$(\mathcal{M})$*, where $\mathcal{M}$ is a gridding matrix whose entries all belong to NLOL.*
(e) The class $\mathcal{C} \odot \mathcal{D}$.

Proof. Part (a) of the lemma is an easy observation which follows directly from the definition of NLOL.

In part (b), it is obvious that $\mathcal{C}^c$ is in NLOL. For $\mathcal{C}^r$, the argument is less trivial: suppose A is an NLOL-recognizer of $\mathcal{C}$. Let us describe an NLOL-recognizer A^r of $\mathcal{C}^r$. On an input π, A^r will simulate in reverse the computation of A on input sequence π^r as follows: once A^r knows the length n, it guesses nondeterministically a possible state s_0 of A, which should correspond to the final state of the algorithm A after A processed the whole input sequence π^r. Then, in step i of the computation of A^r, if the simulation of A has reached a state s_{i-1} and the input contains a value π_i, A^r will again guess a state s_i of A, and verify that A can reach the state s_{i-1} from s_i on input value π_i. Finally after receiving `EOF`, the algorithm checks that the last state s_i is actually the initial state of A. We easily see that A^r is an NLOL-recognizer of $\mathcal{C}^r$.

To see that $\mathcal{C}^\oplus$ is in NLOL, suppose again that A is an NLOL-recognizer for $\mathcal{C}$, and let us describe an NLOL-recognizer $A^\oplus$ for $\mathcal{C}^\oplus$. A permutation $\pi \in \mathcal{C}^\oplus$ can be expressed as a concatenation of blocks, where each block is isomorphic to a permutation of $\mathcal{C}$, and the values in the i-th block are larger than any value in the preceding blocks. Thus, $A^\oplus$ will guess the boundaries between blocks, verify that all the values appearing in the current block are larger than the largest value in the previous blocks, and simulate the computation of A on each block separately to verify that it is order-isomorphic to a permutation from $\mathcal{C}$. Note that $A^\oplus$ does not have enough memory to guess in advance all the boundaries between blocks. However, that is not a problem as $A^\oplus$ can simply non-deterministically decide to end the current block and start a new simulated computation of A at any point. The argument in case of $\mathcal{C}^\ominus$ is similar, the NLOL-recognizer $A^\ominus$ just makes sure that all values in the current block are smaller than the smallest value in the previous block. This concludes the proof of part (c).

To prove part (d), suppose that $\mathcal{C}$ is a grid class defined by a gridding matrix composed of NLOL-recognizable class. An NLOL-recognizer for $\mathcal{C}$ will guess the positions of grid lines for the input sequence, and then verify that each cell of the gridded input sequence is order-isomorphic to the member of the corresponding class.

Finally, to prove part (e), note that a recognizer for $\mathcal{C} \odot \mathcal{D}$ can guess a way to split the input sequence π into two disjoint subsequences π' and π'', and simulate a recognizer for $\mathcal{C}$ on π' and a recognizer for $\mathcal{D}$ on π''. Note that in order to comply with the memory restriction, the recognizer for $\mathcal{C} \odot \mathcal{D}$ guesses whether π_i belongs to π' or π'' only upon receiving the value π_i. □

Corollary 7.2. *For any sequence of classes $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k \in$ NLOL, the class $\mathcal{C}_1 \odot \mathcal{C}_2 \odot \cdots \odot \mathcal{C}_k$ is in NLOL, and therefore polynomially recognizable.*

Lemma 7.1 shows that NLOL contains many important permutation classes, including the classes of layered and co-layered permutations, as well as any class of the form $\mathrm{Av}(12\cdots k)$ or $\mathrm{Av}(k(k-1)\cdots 1)$.

On the negative side, it can be shown that NLOL does not contain some other important classes, such as the class $\mathrm{Av}(2413, 3142)$ of separable permutations, or its subclasses $\mathrm{Av}(231)$, $\mathrm{Av}(213)$, $\mathrm{Av}(312)$ and $\mathrm{Av}(132)$. These five classes share a common feature: their grid-width is at most 1, i.e. their elements have a simple recursive tree-like structure involving only direct sums and skew sums. Before focusing on these classes, we first introduce a restricted form of NLOL that will play an important part in conjunction with bounded grid-width.

7.2.2 BD-recognizable classes

From a different angle, one can view the crucial property of NLOL-recognizable classes in the following way. For an NLOL-recognizable class $\mathcal{C}$, a small amount of information about permutations π_1, π_2 is sufficient to decide whether concatenating them results in a permutation $\pi \in \mathcal{C}$. However, we have no such guarantee for the symmetrical operation of putting π_1 on top of π_2. We now proceed to introduce the so-called BD-recognizable classes that add this symmetry. This will be used later in conjunction with the concept of bounded grid-width to derive some further tractable cases of recognition.

Tree automata

First, let us define the notion of tree automata. We refer an interested reader to the book [54] for a more extensive introduction. A *signature* $\mathcal{F}$ is a set of symbols, with each symbol of $\mathcal{F}$ having a non-negative integer, called *arity*, associated to it. The set of symbols of arity p is denoted by $\mathcal{F}_p$. The set $T(\mathcal{F})$ of *ground terms over* $\mathcal{F}$ is the smallest set that contains the set of constant symbols $\mathcal{F}_0$, and if $p \geq 1$, $f \in \mathcal{F}_p$ and $t_1, \ldots, t_p \in T(\mathcal{F})$ then $f(t_1, \ldots, t_p) \in T(\mathcal{F})$. Ground terms can be straightforwardly visualized as trees.

A *tree automaton* is a quadruple $A = (\mathcal{F}, Q, \Delta, Q_f)$, where $\mathcal{F}$ is a signature, Q is a finite set of states, $Q_f \subseteq Q$ is a set of accepting states, and Δ is a set of transition rules. Every transition is of the form $f(q_1, \ldots, q_n) \to q$, where f is a symbol of arity n, and $q_1, \ldots q_n, q$ are states from Q. Note that the automata we consider here are non-deterministic, in the sense that Δ may contain several transition rules with the same left-hand side.

Tree automata over $\mathcal{F}$ run on ground terms over $\mathcal{F}$, represented as trees. An automaton starts at the leaves and moves upward, labeling inductively each vertex with a state from Q. If the children $u_1, \ldots, u_n$ of $v = f(u_1, \ldots, u_n)$ are labeled with states $q_1, \ldots, q_n$ then u will receive a state q such that Δ contains the rule $f(q_1, \ldots, q_n) \to q$. Notice that the automaton has no explicit initial states. However, a transition rule for a constant symbol $f \in \mathcal{F}_0$ reads as $f \to q$, which can be interpreted as assigning to the leaf an initial state q.

A ground term $t \in T(\mathcal{F})$ is accepted by a tree automaton $A = (\mathcal{F}, Q, \Delta, Q_f)$ if there exists a run of A on t which assigns to the root of t a state from the set Q_f.

BD-recognition

Let T be a binary tree with leaves precisely the set $[n] \times [n]$. We say that T is a *box decomposition tree of size* n if the leaves of any rooted subtree constitute a box, i.e. a product of two integer intervals. This implies that each inner node corresponds to splitting the parent box into two boxes via either horizontal or vertical cut. We can thus represent any subset P of $[n] \times [n]$ using T by simply labeling the leaves whose boxes contain a point from P. Therefore, we can actually represent P by a ground term over the signature $\mathcal{F}_{\mathsf{BD}} = \{\boxdot, \square, \boxplus, \boxminus\}$ where $\boxdot, \square$ are constant symbols and $\boxplus, \boxminus$ are binary symbols. We call this ground term a *box decomposition tree of* P, or BD-tree for short; see Figure 7.1.

We say that the tree automaton $A = (\mathcal{F}^{\mathsf{BD}}, Q, \Delta, Q_f)$ recognizes permutations from $\mathcal{C}$ of size up to n over BD-trees when the following holds. Whenever A is given

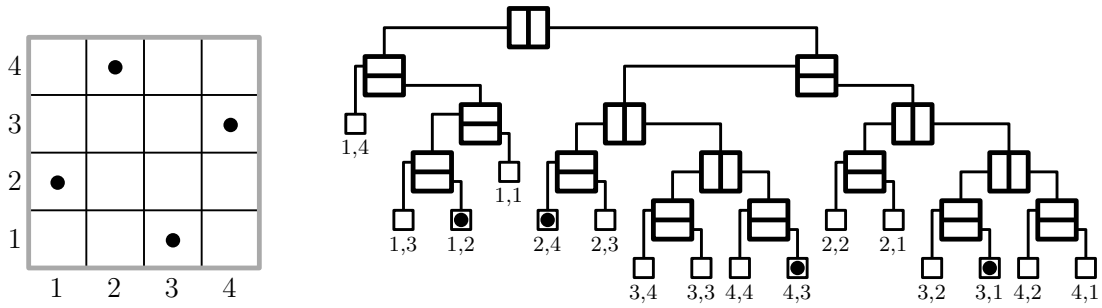

Figure 7.1: The diagram of the permutation 2413 (left) and a BD-tree representing it (right). For added clarity, the leaves are labeled with the coordinates of the corresponding boxes of the diagram. Note however, that these labels are not part of the BD-tree, which represents the permutation uniquely even without them.

a BD-tree T of a point set P in general position inside $[n] \times [n]$, it accepts if and only if the permutation isomorphic to P belongs to $\mathcal{C}$. Moreover, the set of states $Q' \subseteq Q$ that can be assigned by some run of A to the root of T is independent of the choice of BD-tree. This allows the following definition. Whenever a state $q \in Q$ can be assigned to the root of an arbitrary BD-tree of P we say that q is *reachable on* P.

Notice that we do not require A to reject point sets that are not in general position. In this paper, we will always have the guarantee that P is a subset of a permutation diagram, i.e. P satisfies this condition automatically. Moreover, we remark that P does not have to consist of n points, it can be arbitrarily small.

A permutation class $\mathcal{C}$ is *box decomposition* recognizable, or BD-recognizable for short, if there exists an algorithm that receives n and outputs in time polynomial in n a tree automaton $(\mathcal{F}^{\mathsf{BD}}, Q, \Delta, Q_f)$ that recognizes permutations from $\mathcal{C}$ of size up to n over BD-trees. Notice that this implies that the total number of possible states, i.e. the size of Q, is polynomial in n.

Similar to NLOL, it is easy to see that many simple permutation classes are BD-recognizable. The property is also preserved under many important operations as witnessed by the following lemma.

Lemma 7.3. *If $\mathcal{C}$ and $\mathcal{D}$ are BD-recognizable classes, then the following classes are BD-recognizable as well:*

(a) The classes $\mathcal{C} \cap \mathcal{D}$ and $\mathcal{C} \cup \mathcal{D}$.
(b) The classes $\mathcal{C}^{-1}$, $\mathcal{C}^r$ and $\mathcal{C}^c$.
(c) The classes $\mathcal{C} \oplus \mathcal{D}$ and $\mathcal{C} \ominus \mathcal{D}$, and more generally, the class $\mathrm{Grid}(\mathcal{M})$ where $\mathcal{M}$ is a gridding matrix whose entries all belong to BD.
(d) The classes $\mathcal{C}^{\oplus}$ and $\mathcal{C}^{\ominus}$, i.e., the sum-closure and skew-closure of $\mathcal{C}$.
(e) The class $\mathcal{C} \odot \mathcal{D}$.

Proof. For each of the individual claims, we will describe a tree automaton that recognizes the desired class. However, we will leave out the details of how to construct the automaton in polynomial time from the tree automata for $\mathcal{C}$ and $\mathcal{D}$ as they are fairly straightforward and uninteresting. Suppose that the tree automata for recognizing permutations of $\mathcal{C}$, respectively $\mathcal{D}$, up to size n are $(\mathcal{F}^{\mathsf{BD}}, Q_{\mathcal{C}}, \Delta_{\mathcal{C}}, R_{\mathcal{C}})$, respectively $(\mathcal{F}^{\mathsf{BD}}, Q_{\mathcal{D}}, \Delta_{\mathcal{D}}, R_{\mathcal{D}})$.

Part (a) follows easily by simulating both automatons simultaneously, i.e. we set the set of states to be the cartesian product $Q_{\mathcal{C}} \times Q_{\mathcal{D}}$ and the rules to be all possible combinations of rules of the individual automata, e.g. for every rule $\boxplus\,(p_1, p_2) \to p \in \Delta_{\mathcal{C}}$ and every rule $\boxplus\,(q_1, q_2) \to q \in \Delta_{\mathcal{D}}$ there is a rule $\boxplus\,((p_1, q_1), (p_2, q_2)) \to (p, q)$. The only difference is the set of accepting states, we set them to be $R_{\mathcal{C}} \times R_{\mathcal{D}}$ in the case of intersection and $Q_{\mathcal{C}} \times R_{\mathcal{D}} \cup R_{\mathcal{C}} \times Q_{\mathcal{D}}$ in the case of union.

In order to prove part (b), let us start with the observation that a BD-tree of permutation π^r can be obtained from a BD-tree of permutation π simply by changing the order of operands of $\boxplus$. Therefore, the automaton for $\mathcal{C}^r$ can be obtained by simply replacing every rule of the form $\boxplus\,(q_1, q_2) \to q$ with the rule $\boxplus\,(q_2, q_1) \to q$. The construction for $\mathcal{C}^c$ is identical, only with respect to $\boxminus$. And in order to obtain a BD-tree of permutation π^{-1} from the BD-tree of π, it suffices to swap $\boxminus$ with $\boxplus$ while also changing the order of their operands. Therefore, the automaton for $\mathcal{C}^{-1}$ can be obtained by simply replacing every rule of form $\boxplus\,(q_1, q_2) \to q$ with the rule $\boxminus\,(q_2, q_1) \to q$ and vice versa.

Before we tackle the remaining parts, let us make a general observation. Without loss of generality, we can assume that each state contains the coordinates of the box represented by the node. This assumption is possible since the number of possible coordinates is polynomial in n and we can simply guess the coordinates in the transition rules for leaves and then check consistency in the inner nodes and enter a failure state whenever the coordinates do not match. Notice that this also does not violate the independence of the reachable set of states on the shape of the particular BD-tree. Regardless of the tree shape, precisely any coordinates coherent with the size of box B will be reachable on the ground term corresponding to B.

We construct the tree automaton recognizing Grid($\mathcal{M}$) in the following way. The automaton simulates the individual automata of all the classes in M. Moreover, each state contains the positions of all the grid lines for the input permutation. This is again achieved via guessing the grid lines in the transition rules for leaves and ensuring consistency. In particular, the transition rules for the symbol $\boxdot$ simulate the $\square$ rules for all but the one automaton given by the position of the grid lines. This information together with the coordinates of each box is sufficient to simulate all the automata and accept only if all of them end in an accepting state. And again, note that this information does not depend on the particular BD-tree.

In showing part (d), we will only describe the automaton for $\mathcal{C}^{\oplus}$ as the case of $\mathcal{C}^{\ominus}$ follows from symmetry. Using the previous ideas, one might hope to simply guess the grid lines between individual $\oplus$-indecomposable parts and to simulate many copies of the automaton recognizing $\mathcal{C}$. Unfortunately, there can be too many of these parts which would cause the set of states to be superpolynomial. However, notice that each box can intersect at most two $\oplus$-indecomposable parts that are not fully covered by the box. It is therefore sufficient to simulate only these two incomplete parts together with their coordinates, and upon completion of a part, check whether its simulated automaton for $\mathcal{C}$ finished in an accepting state.

Finally, part (e) is similar to the NLOL case. As in part (a), the automaton for $\mathcal{C} \odot \mathcal{D}$ simulates the automata for $\mathcal{C}$ and $\mathcal{D}$ with set of accepting states being

$R_{\mathcal{C}} \times R_{\mathcal{D}}$. The only difference is in the transition rules for the symbol $\boxdot$, as we want to initialize the non-empty leaf for only one of the automata. Therefore, we add a rule $\boxdot\!\!\rightarrow (p, q)$ whenever $\boxdot\!\!\rightarrow p \in \Delta_{\mathcal{C}}$ and $\square\!\!\rightarrow q \in \Delta_{\mathcal{D}}$, or $\square\!\!\rightarrow p \in \Delta_{\mathcal{C}}$ and $\boxdot\!\!\rightarrow q \in \Delta_{\mathcal{D}}$. □

Let us conclude this section by remarking that $\mathcal{C}$-RECOGNITION can be solved in polynomial time whenever $\mathcal{C}$ is BD-recognizable. First, we generate the automaton A recognizing permutations from a BD-recognizable class $\mathcal{C}$ of size up to n. Since the set of reachable states does not depend on the shape of BD-tree, we choose an arbitrary BD-tree T of P (e.g. first merging each column individually bottom-up and then merging columns from left to right). Then we compute the set of reachable states for the root of T in time $O(n \cdot |Q|)$, and thus polynomial in n, using standard tree automata approach [54].

7.2.3 GT-recognizable classes

The recognition problem for a permutation class of bounded grid-width is not necessarily tractable. For instance, it is known that all the proper subclasses of Av(321) have bounded grid-width [86]; however, Av(321) has uncountably many subclasses, and therefore some of them have undecidable recognition problem.

Our next goal is therefore to introduce a type of permutation classes whose recognition problem is tractable on inputs of bounded grid-width. Informally speaking, a class $\mathcal{C}$ is GT-recognizable, if for any g, the $\mathcal{C}$-recognition problem on inputs of grid-width at most g can be solved by a tree automaton operating on a grid tree of grid-width at most g.

To work within the tree automata framework, it is not suitable to use the grid trees directly, because each leaf of the grid tree contains the 'global' coordinates of the corresponding element with respect to the entire permutation. For the purposes of automata computation, it is more convenient to distribute the information about the elements into all the vertices of the tree, in such a way that the ground term corresponding to the subtree rooted in a given vertex describes the relative position of the elements in represented by the subtree, without referring to the size of the entire permutation. To this end, we will introduce a modification of grid trees, so-called merge-labeled trees, which can be seen as ground terms over a signature whose size depends only on g. We start by introducing the symbols of the signature.

Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be two interval families. We say that an interval family $\mathcal{I} = \{I_1 < I_2 < \cdots < I_m\}$ is a *merge* of $\mathcal{I}_1$ and $\mathcal{I}_2$, if $\bigcup\mathcal{I}_1$ is disjoint from $\bigcup\mathcal{I}_2$ and $\mathcal{I}$ is the union of $\mathcal{I}_1$ and $\mathcal{I}_2$. For such $\mathcal{I}_1$, $\mathcal{I}_2$ and $\mathcal{I}$, a *merge description of* $\mathcal{I}_1$ *and* $\mathcal{I}_2$ *into* $\mathcal{I}$ is the tuple (m, f), where $f\colon [m] \to \{1, 2\}$ is a function which describes which interval belongs to $\mathcal{I}_1$ and which belongs to $\mathcal{I}_2$, i.e. for every $k \in [m]$ the interval I_k belongs to $\mathcal{I}_{f(k)}$.

Let $\mathsf{M}_{\leq g}$ be the set of all merge descriptions such that both $\mathcal{I}_1$ and $\mathcal{I}_2$ contain at most g intervals. Observe that the merge description, together with the interval families $\mathcal{I}_1$ and $\mathcal{I}_2$, uniquely determines the merged interval family $\mathcal{I}$. There exists less than $2g\,2^g$ such merge descriptions, i.e. the size of $\mathsf{M}_{\leq g}$ is bounded by a function of g.

When we merge two interval families $\mathcal{I}_1$ and $\mathcal{I}_2$ into $\mathcal{I}$ there might occur two consecutive intervals inside $\mathcal{I}$ such that their union is also an interval. Let

$\mathcal{I} = \{I_1 < I_2 < \cdots < I_m\}$ be an interval family such that $I_k \cup I_{k+1}$ constitutes an interval. The *k-th interval join of* $\mathcal{I}$ is the interval family $\mathcal{I}'$ obtained by replacing the intervals I_k and I_{k+1} with their union $I_k \cup I_{k+1}$.

Let T be a grid tree representing a permutation π of grid-width at most g. We will now describe how it can be transformed into a ground term over signature $\mathcal{F}_g^{\mathsf{GT}}$ that consist of

- a constant symbol Leaf,
- a set of unary symbols $\{\mathsf{Col}_1, \mathsf{Col}_2, \ldots, \mathsf{Col}_{2g-1}\}$,
- a set of unary symbols $\{\mathsf{Row}_1, \mathsf{Row}_2, \ldots, \mathsf{Row}_{2g-1}\}$, and
- a set of binary symbols $\mathsf{M}_{\leq g} \times \mathsf{M}_{\leq g}$.

Proposition 7.4. *We can transform a grid tree T representing a permutation π of grid-width at most g into a ground term t over signature $\mathcal{F}_g^{\mathsf{GT}}$ such that it is possible to reconstruct the original permutation π from t.*

Proof. As the first step, for each node v of T we choose the smallest interval family $\mathcal{I}_v$ such that $\bigcup \mathcal{I}_v = \Pi_x(\pi_v^T)$ and similarly the smallest interval family $\mathcal{J}_v$ such that $\bigcup \mathcal{J}_v = \Pi_y(\pi_v^T)$. Observe that each of the families has size at most g since their size is exactly equal to the intervalicity of the sets $\Pi_x(\pi_v^T)$, respectively $\Pi_y(\pi_v^T)$. As the second step of the transformation, we generate recursively for each vertex v of T a ground term t_v.

For a leaf v of the tree T we set $t_v = \mathsf{Leaf}$. For an internal node u with children v and w, we generate a sequence of nested terms that transforms the interval families $\mathcal{I}_v$, $\mathcal{I}_w$ into $\mathcal{I}_u$ and simultaneously $\mathcal{J}_v$,$\mathcal{J}_w$ into $\mathcal{J}_u$. Observe that there is a sequence of pairs of interval families $(\mathcal{I}_1, \mathcal{J}_1), \ldots, (\mathcal{I}_l, \mathcal{J}_l)$ such that

- $\mathcal{I}_1$ is the merge of $\mathcal{I}_v$ and $\mathcal{I}_w$ and $\mathcal{J}_1$ is the merge of $\mathcal{J}_v$ and $\mathcal{J}_w$,
- $\mathcal{I}_l = \mathcal{I}_u$ and $\mathcal{J}_l = \mathcal{J}_u$, and
- for every $i \in \{2, \ldots, l\}$ either $\mathcal{I}_i$ is the k-th interval join of $\mathcal{I}_{i-1}$ for some k and $\mathcal{J}_{i-1} = \mathcal{J}_i$, or the other way round.

In other words, the sequence changes gradually the result of merging the interval families of v and w into the interval families of u, concatenating a pair of neighboring intervals in each step. Using this sequence we generate a sequence of nested ground terms $t_1, \ldots, t_l$ in the following way.

First, we generate the term $t_1 = \mathcal{M}_{\mathcal{I}}, \mathcal{M}_{\mathcal{J}}(t_v, t_w)$, where $\mathcal{M}_{\mathcal{I}}$ is the merge description of $\mathcal{I}_v$ and $\mathcal{I}_w$, and similarly $\mathcal{M}_{\mathcal{J}}$ is the merge description of $\mathcal{J}_v$ and $\mathcal{J}_w$, and t_v, t_w, are the ground terms generated recursively for the children of u.

Now we describe the terms t_i for all $i \geq 2$. If $\mathcal{I}_i$ is a k-th interval join of $\mathcal{I}_{i-1}$, we set the term $t_i = \mathsf{Col}_k(t_{i-1})$. Otherwise, if $\mathcal{J}_i$ is a k-th interval join of $\mathcal{J}_{i-1}$, we set $t_i = \mathsf{Row}_k(t_{i-1})$. Observe that k is at most $2g - 1$ in each step since the interval families $\mathcal{I}_1$, $\mathcal{J}_1$ are obtained by merging two interval families of size at most g, and the process never increases the sizes of the interval families. Finally, we set $t_v = t_l$.

It follows from the transformation that the term t_v uniquely determines the interval families $\mathcal{I}_v$ and $\mathcal{J}_v$, and therefore also the grid tree T and the permutation π. □

The ground term generated in Proposition 7.4 for a permutation π is called a *merge-labeled tree of* π.

We say that a tree automaton $A = (\mathcal{F}_g^{\mathsf{GT}}, Q, \Delta, Q_f)$ *recognizes* a permutation class $\mathcal{C}$ on inputs of grid-width at most g, if for every merge-labeled tree T of grid-width at most g the automaton A accepts T if and only if T represents a permutation from $\mathcal{C}$. Note that we do not assume here that every permutation from $\mathcal{C}$ has grid-width bounded by g.

We say that a permutation class $\mathcal{C}$ is *grid tree recognizable*, or GT-recognizable for short, if there is an algorithm that receives the constant g as input and outputs a tree automaton A_g that recognizes $\mathcal{C}$ over merge-labeled trees of grid-width at most g.

Proposition 7.5. *If $\mathcal{C}$ is GT-recognizable class, then the $\mathcal{C}$-RECOGNITION problem can be solved in time $O(f(\mathrm{gw}(\pi)) \cdot n)$, where f is a computable function. In particular, if a permutation class $\mathcal{C}$ has bounded grid-width and is GT-recognizable, then $\mathcal{C}$-RECOGNITION is polynomial-time solvable.*

Proof. Let π be the input permutation of length n. Using the tools developed in Section 2.1, we can obtain a tree decomposition of G_π of width at most $2\,\mathrm{tw}(\pi)+1 \leq 16\,\mathrm{gw}(\pi)+1$ and transform it into a grid tree of grid-width at most $16\,\mathrm{gw}(\pi)+3$ all in time $2^{O(\mathrm{gw}(\pi))} \cdot n$. The grid tree can then be transformed to a merge-labeled tree t by Proposition 7.4. Finally, we construct the corresponding automaton A_g in time depending only on g, and simulate it on t in time $O(f(g) \cdot |t|) = O(f(g) \cdot n)$. □

Before we move towards our main result, let us make a few remarks about the expressive power of GT-recognizability. The notion of GT-recognizable classes is closely related to the previously well-studied notion of tree-automata on bounded tree-width decompositions of graphs and other relational structures. The key result in this area is Courcelle's theorem [55], which states that any property expressible in monadic second-order logic can be recognized by a tree automaton on decomposition trees of bounded width. One could hope that indeed every class defined by an MSO sentence in TOTO is GT-recognizable. Unfortunately, we were not able to prove this fact.

It can however be shown that every class determined by finitely many minimal forbidden patterns is GT-recognizable, and moreover, if $\mathcal{C}$ and $\mathcal{D}$ are both GT-recognizable classes, then so are $\mathcal{C}^r$, $\mathcal{C}^c$, $\mathcal{C}^{-1}$, $\mathcal{C} \oplus \mathcal{D}$, $\mathcal{C} \cup \mathcal{D}$, $\mathcal{C} \cap \mathcal{D}$ or $\mathcal{C} \odot \mathcal{D}$, among other similar examples.

Unfortunately, these facts do not yield any new cases of tractable $\mathcal{C} \odot \mathcal{D}$-RECOGNITION. The reason is that if both $\mathcal{C}$ and $\mathcal{D}$ are infinite classes, then by Erdős–Szekeres theorem [65], their merge $\mathcal{C} \odot \mathcal{D}$ contains either $\mathrm{Av}(321)$, $\mathrm{Av}(123)$ or $\mathrm{Av}(21) \odot \mathrm{Av}(12)$ as a subclass, and all of these classes have the long path property and thus, they contain permutations of arbitrarily large tree-width and grid-width (Proposition 2.33). We therefore omit here the proofs of most of the above-mentioned facts on GT-recognizability, and instead focus on the connection between GT-recognizable and BD-recognizable classes, which will lead us to our main result. The only fact needed for our application is that any principal class is GT-recognizable.

Proposition 7.6. *Any class of the form $\mathrm{Av}(\sigma)$ is GT-recognizable.*

Proof. Suppose that σ has length k. Observe that every merge-labeled tree over the signature $\mathcal{F}_g^{\mathsf{GT}}$ can be interpreted as a representation of a gridded permutation.

Formally, we associate to the ground term t a permutation π^t together with two interval families $\mathcal{I}^t = \{I_1^t, \ldots, I_g^t\}$ and $\mathcal{J}^t = \{J_1^t, \ldots, J_g^t\}$ such that $\mathcal{I}^t$ forms a partition of $\Pi_x(\pi^t)$ and $\mathcal{J}^t$ forms a partition of $\Pi_y(\pi^t)$. Note that whenever some of the interval families defined by t has size smaller than g we complete it to length g with empty intervals. We describe construction of a tree automaton $\mathcal{A}$ that keeps track of all the possible embeddings of π^t but only with respect to the families $\mathcal{I}^t$ and $\mathcal{J}^t$.

A *distribution (of σ into a $g \times g$ grid)* is a function $f\colon [k] \to [g] \times [g] \cup \{\epsilon\}$. For a distribution f of σ into a $g \times g$ grid, let $S_f \subseteq [k]$ denote the indices that do not map to ϵ. Then we say that such a distribution f is *realizable for a ground term t* over $\mathcal{F}_g^{\mathsf{GT}}$, if there is a partial mapping $g : S_f \to \pi^t$ such that moreover, f correctly prescribes in which intervals lie the images under g, i.e. for each $i \in S_f$ with $f(i) = (m, l)$ we have $g(i) \in I_m^t \times J_l^t$.

Clearly for a fixed σ and g there are k^{g^2+1} such distributions. Fix any ordering of them $f_1, f_2, \ldots$ that can be computed just from k and g. We define the set of states of $\mathcal{A}$ to be the set of all $\{0,1\}$-vectors of length k^{g^2+1}, where the i-th bit is set to 1 if and only if the distribution f_i is realizable for the given term. It is trivial to set the vector when $t = \mathsf{Leaf}$ since a distribution f is realizable if and only if there is at most one $i \in [k]$ such that $f(i) \neq \epsilon$.

When $t = \mathsf{Col}_i(t')$, the associated transition rules are also very simple. For every distribution f' that is realizable for the term t', there is exactly one distribution f such that f is realizable for the term t. This holds since the interval family $\mathcal{I}^t$ is obtained from $\mathcal{I}^{t'}$ by replacing a union of two consecutive intervals which determines exactly how to transform the distribution f' to f. The same holds symmetrically when $t = \mathsf{Row}_i(t')$.

Finally, suppose that $t = \mathcal{M}_\mathcal{I}, \mathcal{M}_\mathcal{J}(t_1, t_2)$. We claim that tor every pair of distributions f_1 and f_2 such that f_1 is realizable for t_1 and f_2 is realizable for t_2, we can decide whether they combine to a realizable distribution f for t. This follows since the mutual position of points $p \in \pi^{t_1}$ such that p belongs to the box $I_a^{t_1} \times J_b^{t_1}$ and $q \in \pi^{t_2}$ such that q belongs to the box $I_c^{t_2} \times J_d^{t_2}$ is uniquely determined by the indices a, b, c, d together with the merge descriptions $\mathcal{M}_\mathcal{I}$ and $\mathcal{M}_\mathcal{J}$. Therefore, we can compute the transition rules in this case as well. □

7.3 Combining BD- with GT-recognizability

In this section, we use a dynamic programming approach to recognize any merge of a BD-recognizable class with a GT-recognizable class of bounded grid-width.

Theorem 7.7. *If $\mathcal{C}$ is a BD-recognizable class and $\mathcal{D}$ is a GT-recognizable class such that moreover $\mathrm{gw}_\mathcal{D}(n) \leq g$ for some constant g, then $\mathcal{C} \odot \mathcal{D}$-RECOGNITION is polynomial-time solvable.*

Proof. For the proof of Theorem 7.7, let the input be a permutation π of length n. We generate first the tree automaton $A_\mathcal{C} = (\mathcal{F}^{\mathsf{BD}}, Q_\mathcal{C}, \Delta_C, R_\mathcal{C})$ recognizing permutations from $\mathcal{C}$ of size up to n over BD-trees, and the tree automaton $A_\mathcal{D} = (\mathcal{F}_g^{\mathsf{GT}}, Q_\mathcal{D}, \Delta_\mathcal{D}, R_\mathcal{D})$ recognizing $\mathcal{D}$ over merge-labeled trees of grid-width at most g. Note that we can obtain these automata in time polynomial in n. The general outline of our approach is that we want to efficiently simulate $A_\mathcal{D}$ on subpermutations of π of grid-width at most g, while at the same time simulating

$A_{\mathcal{C}}$ on the remaining elements. We know that the size of $Q_{\mathcal{C}}$ is at most n^c for a constant c, and the size of $Q_{\mathcal{D}}$ is at most $f(g)$ for a computable function f.

We shall define a polynomially bounded number of subproblems that can be efficiently solved by recursion. A *subproblem* is a tuple $p = (\mathcal{I}, \mathcal{J}, h, s)$, where

- $\mathcal{I}$ and $\mathcal{J}$ are interval families of integers in $[n]$, each of size at most g,
- $h\colon \mathcal{I} \times \mathcal{J} \to Q_{\mathcal{C}}$ assigns a state of $A_{\mathcal{C}}$ to each box in $\mathcal{I} \times \mathcal{J}$, and
- $s \in Q_{\mathcal{D}}$ is a state of the automaton $A_{\mathcal{D}}$.

There are at most n^{4g} choices for $\mathcal{I}$ and $\mathcal{J}$, at most n^{cg^2} choices for h, and at most $f(g)$ choices for s, which makes the total number of subproblems at most $f(g)n^{4g+4cg^2}$, i.e. polynomial in n for fixed g.

We say that a subproblem $(\mathcal{I}, \mathcal{J}, h, s)$ is *feasible* if the points of S_π that lie in $\bigcup(\mathcal{I} \times \mathcal{J})$ can be partitioned into two disjoint sets $P_{\mathcal{C}}$ and $P_{\mathcal{D}}$ with the following properties:

(a) For any two intervals $I \in \mathcal{I}$ and $J \in \mathcal{J}$ the state $h(I, J)$ is reachable on $P_{\mathcal{C}} \cap I \times J$ by the automaton $A_{\mathcal{C}}$.
(b) The permutation $\pi_{\mathcal{D}}$ corresponding to $P_{\mathcal{D}}$ can be represented by a merge-labeled tree t of width at most g with the property that the interval families $\mathcal{I}_r$ and $\mathcal{J}_r$ represented at the root node r of t are precisely the standardization of $\mathcal{I}$ and $\mathcal{J}$ with respect to $P_{\mathcal{D}}$, and there is a run of the automaton $A_{\mathcal{D}}$ over the tree t that assigns to r the state s.

Moreover, we say that a subproblem $(\mathcal{I}, \mathcal{J}, h, s)$ is *initial* if

(a) both $\mathcal{I}$ and $\mathcal{J}$ contain only single intervals $I = [n]$ and $J = [n]$ respectively,
(b) $h(I, J) \in R_{\mathcal{C}}$, i.e., it is an accepting state of the automaton $A_{\mathcal{C}}$, and
(c) $s \in R_{\mathcal{D}}$, i.e it is an accepting state of the automaton $A_{\mathcal{D}}$.

It follows from the definition that π belongs to $\mathcal{C} \odot \mathcal{D}$ if and only if at least one of the initial subproblems is feasible. Thus, if we can efficiently determine the feasibility of the initial subproblems, we can solve $\mathcal{C} \odot \mathcal{D}$-recognition.

We describe a recursive procedure Feasible($\mathcal{I}$, $\mathcal{J}$, h, s) that answers whether a given subproblem is feasible. Intuitively, we aim to mimic the way interval families are manipulated in merged-labeled trees.

First we would like to identify subproblems that can correspond to the leaves of a merge-labeled tree. We say that a subproblem $(\mathcal{I}, \mathcal{J}, h, s)$ is *leaf-feasible* if it is feasible and moreover

(a) both $\mathcal{I}$ and $\mathcal{J}$ contain only single intervals I and J respectively,
(b) the conditions of feasibility hold for a partition of $S_\pi \cap I \times J$ into $P_{\mathcal{C}}$ and $P_{\mathcal{D}}$ such that the set $P_{\mathcal{D}}$ contains only a single point, and
(c) there exists a rule $\mathsf{Leaf} \to s \in \Delta_{\mathcal{D}}$, i.e a leaf of a merge-labeled tree can obtain the state s during a run of the automaton $A_{\mathcal{D}}$.

It is possible to efficiently check whether a subproblem is leaf-feasible. First we check conditions (a) and (c). Since we require the set $P_{\mathcal{D}}$ to be a singleton, this guarantees the second condition of feasibility. In this case, we have at most n possibilities how to choose the singleton set $P_{\mathcal{D}}$ and for each such choice we can test in polynomial time whether the state $h(I, J)$ can be reached by a run of $A_{\mathcal{C}}$ over an arbitrarily chosen BD-tree of $P_{\mathcal{C}}$, thus verifying the final condition (b).

We continue with identifying subproblems that correspond to merging rows or columns. We say that subproblem $(\mathcal{I}', \mathcal{J}', h', s')$ is a *column split* of a subproblem $(\mathcal{I}, \mathcal{J}, h, s)$ if

(a) $\mathcal{J}' = \mathcal{J}$,
(b) there is a $k \in m$ such that $\mathcal{I}' = \{I'_1 < \cdots < I'_{m+1}\}$ is the k-th interval join of $\mathcal{I} = \{I_1 < \cdots < I_m\}$,
(c) the rule $\mathsf{Col}_k(s') \to s$ exists in $\Delta_{\mathcal{D}}$,
(d) $h'(I, J) = h(I, J)$ for every $I \in \mathcal{I} \setminus \{I_k\}$ and every $J \in \mathcal{J}$, and
(e) for every $J \in \mathcal{J}$ there exists a rule $\square\!\!\square\, (h'(I'_k, J), h'(I'_{k+1}, J)) \to h(I_k, J)$ in $\Delta_{\mathcal{C}}$.

In this case, all the conditions (a) up to (e) are straightforward to check algorithmically. Symmetrically, we define a *row split* of a subproblem just by swapping the roles of $\mathcal{I}$ and $\mathcal{J}$, replacing Col with Row in condition (c) and replacing the symbol $\square\!\!\square$ with $\boxminus$ in condition (d).

Finally, we need to take care of the 'merge' nodes of merge-labeled trees. We say that subproblems $(\mathcal{I}_1, \mathcal{J}_1, h_1, s_1)$ and $(\mathcal{I}_2, \mathcal{J}_2, h_2, s_2)$ *combine to the subproblem* $(\mathcal{I}, \mathcal{J}, h, s)$ if

(a) $\mathcal{I}_1$ and $\mathcal{I}_2$ merge to $\mathcal{I}$ as witnessed by the merge description $\mathcal{M}_{\mathcal{I}}$,
(b) $\mathcal{J}_1$ and $\mathcal{J}_2$ merge to $\mathcal{J}$ as witnessed by the merge description $\mathcal{M}_{\mathcal{J}}$,
(c) the rule $\mathcal{M}_{\mathcal{I}}, \mathcal{M}_{\mathcal{J}}(s_1, s_2) \to s$ exists in $\Delta_{\mathcal{D}}$,
(d) for every $i \in \{1, 2\}$ and every $I \in \mathcal{I}_i$ and $J \in \mathcal{J}_i$, $h(I, J) = h_i(I, J)$, and
(e) for every pair of intervals (I, J) from $(\mathcal{I}_1 \times \mathcal{J}_2) \cup (\mathcal{I}_2 \times \mathcal{J}_1)$, the state $h(I, J)$ is reachable by $A_{\mathcal{C}}$ on the point set $S_\pi \cap (I \times J)$.

As before, we can check all the conditions (a) up to (e) algorithmically in polynomial time.

In summary, Feasible returns a positive answer for a subproblem p if p is leaf-feasible, or if there is a subproblem p' that is either a row or a column split of p and Feasible gives a positive answer on p', or if there are two subproblems p_1 and p_2 that combine to p and Feasible returns positive answer for both of them.

We store all the computed results of Feasible in a table, effectively turning it into a dynamic programming approach. The algorithm for $\mathcal{C} \odot \mathcal{D}$-recognition declares that π is in $\mathcal{C} \odot \mathcal{D}$ if and only if Feasible succeeds for at least one initial problem. See Algorithm 1.

We claim that the total running time of Algorithm 1 is polynomial in n. We have seen that the total number of subproblems is polynomial in n and that each of the checks during the runtime of Feasible can also be performed in polynomial time. Thus, the claim follows. Note that we are not trying to optimize the runtime as this is a fairly generic algorithm and our goal is to identify which cases of $\mathcal{C} \odot \mathcal{D}$-recognition belong to P.

We are left with proving the correctness of the algorithm. First, we claim that whenever Feasible outputs a positive answer on a subproblem $(\mathcal{I}, \mathcal{J}, h, s)$ then it is indeed feasible. This is easy to see since it is possible to reconstruct from the recursive calls a partition of S_π into $P_{\mathcal{C}}$ and $P_{\mathcal{D}}$ together with a certificates in the form of box decomposition tree T with an accepting run of $A_{\mathcal{C}}$ over T, and a merge-labeled tree t with an accepting run of $A_{\mathcal{D}}$ over t.

Algorithm 1: Deciding whether π belongs to $\mathcal{C} \odot \mathcal{D}$

Function FEASIBLE$(\mathcal{I}, \mathcal{J}, h, s)$**:**
  The function FEASIBLE is memoized, i.e. it gets fully executed at most once for each subproblem. If the function is invoked again with the same parameters as in a previous call, it will return the cached result immediately.
  if $(\mathcal{I}, \mathcal{J}, h, s)$ *is leaf-feasible* **then**
    return true
  for *each subproblem* $(\mathcal{I}', \mathcal{J}'', h', s)$ *that is a row or a column split of* $(\mathcal{I}, \mathcal{J}, h, s)$ **do**
    if FEASIBLE$(\mathcal{I}', \mathcal{J}', h', s')$ **then**
      return true
  for *each pair of subproblems* $(\mathcal{I}_1, \mathcal{J}_1, h_1, s_1)$ *and* $(\mathcal{I}_2, \mathcal{J}_2, h_2, s_2)$ *that combine to* $(\mathcal{I}, \mathcal{J}, h, s)$ **do**
    if FEASIBLE$(\mathcal{I}_1, \mathcal{J}_1, h_1, s_1)$ *and* FEASIBLE$(\mathcal{I}_2, \mathcal{J}_2, h_2, s_2)$ **then**
      return true
  return false

for *each initial problem* $(\mathcal{I}, \mathcal{J}, h, s)$ **do**
  if FEASIBLE$(\mathcal{I}, \mathcal{J}, h, s)$ **then**
    return "yes"
return "no"

For the converse, we proceed to show by induction on the sizes of $\mathcal{I}$ and $\mathcal{J}$ that whenever a subproblem $(\mathcal{I}, \mathcal{J}, h, s)$ is feasible the function FEASIBLE returns a positive answer. Let $P_\mathcal{C}$ and $P_\mathcal{D}$ be the partition of the points of S_π that lie in $\bigcup \mathcal{I} \times \mathcal{J}$ that witnesses its feasibility. Furthemore, let t be the merge-labeled tree such that the interval families $\mathcal{I}_r$ and $\mathcal{J}_r$ represented at the root node r of t are precisely the standardization of $\mathcal{I}$ and $\mathcal{J}$ with respect to $P_\mathcal{D}$, together with a run of the automaton $A_\mathcal{D}$ over the tree t that assigns to r the state s.

The rest of the proof depends on the root symbol r of t. First, suppose that the root symbol is Leaf. This implies that the subproblem is actually leaf-feasible and FEASIBLE just checks the leaf-feasibility of a subproblem according to its definition.

Now suppose that the root symbol is Col_k or Row_k for some k. Let us assume without loss of generality that it is Col_k as the other case is symmetrical. Our goal is to show that there is a feasible subproblem $(\mathcal{I}', \mathcal{J}', h', s')$ that is a column-split of $(\mathcal{I}, \mathcal{J}, h, s)$. Observe that there is an interval family $\mathcal{I}'$ such that $\mathcal{I}$ is a k-th interval join of $\mathcal{I}'$ and the points of $P_\mathcal{D}$ are distributed in $\mathcal{J} \times \mathcal{I}'$ in accordance with the merge-labeled tree t. The run of automaton $A_\mathcal{D}$ over t gives us also the state s' assigned to the child of r. We only need to take care of the function h' on the newly created boxes in order to satisfy the condition (e) of the column-split definition. Recall that $I_k \in \mathcal{I}$ is the interval partitioned into the intervals I'_k and I'_{k+1} in $\mathcal{I}'$. We claim that for any $J \in \mathcal{J}$ there are two states $q_1, q_2 \in Q_\mathcal{C}$ such that the rule $\square\,(q_1, q_2) \to h(I_k, J)$ exists in $\Delta_\mathcal{C}$. That is true since the state $h(I_k, J)$ is reachable over arbitrary BD-tree of $P_\mathcal{C} \cap (I_k \times J)$ so it does not matter how we choose to initially split the box $I_k \times J$. In this way, we obtained a feasible

column split of the subproblem $(\mathcal{I}, \mathcal{J}, h, s)$. Thus it follows from induction that FEASIBLE gives positive answer for $(\mathcal{I}, \mathcal{J}, h, s)$.

Finally, suppose that the root symbol of the merge-labeled tree is $\mathcal{M}_\mathcal{I} \times \mathcal{M}_J$. Similarly to before, we can obtain interval families $\mathcal{I}_1$, $\mathcal{I}_2$ and $\mathcal{J}_1$, $\mathcal{J}_2$ such that the merge description $\mathcal{M}_\mathcal{I}$ describes the merge of $\mathcal{I}_1$ and $\mathcal{I}_2$ into $\mathcal{I}$, $\mathcal{M}_\mathcal{J}$ describes the merge of $\mathcal{J}_1$ and $\mathcal{J}_2$ into $\mathcal{J}$, and moreover the interval families agree with the distribution of the points $P_\mathcal{D}$ prescribed by t. The run of automaton $A_\mathcal{D}$ over t gives us also the states s_1 and s_2 assigned to the children of r. Moreover, the condition (d) in the definition of subproblems combination tells how to define the functions h_1 and h_2. In this way, we obtained two feasible subproblems that combine to $(\mathcal{I}, \mathcal{J}, h, s)$. The rest follows from induction. □

A typical example of a GT-recognizable permutation class of bounded grid-width is the class of separable permutations. In particular, Theorem 7.7 in combination with Proposition 7.6 yields the following consequence.

Corollary 7.8. *Let $\mathcal{D}$ be the class* Av(213)*, one of its symmetries, or the class of separable permutations. For any BD-recognizable class $\mathcal{C}$, the $\mathcal{C} \odot \mathcal{D}$-RECOGNITION is polynomial-time solvable.*

7.4 Hardness results

Let us focus on examples of merge classes whose recognition problem is NP-hard. Our goal is to show that $\mathrm{Av}(\alpha) \odot \mathrm{Av}(\alpha)$-RECOGNITION is NP-complete for any simple pattern α of length at least 4. Observe that this coincides with the results obtained in Proposition 3.10 where we show that the class $\mathrm{Av}(\alpha) \odot \mathrm{Av}(\alpha)$ for such α cannot be defined by an FO sentence in TOTO.

Due to the technical nature of the reduction, we first prove the case $\alpha = 2413$ and later we show how to modify the proof for arbitrary simple pattern α. Note that $\mathrm{Av}(\alpha) \odot \mathrm{Av}(\alpha)$-RECOGNITION is clearly in NP for arbitrary α and thus, it is sufficient to prove its NP-hardness.

Theorem 7.9. $\mathrm{Av}(3142) \odot \mathrm{Av}(3142)$-*RECOGNITION is NP-complete.*

Proof. Our NP-hardness argument is based on the reduction from the classical NP-complete problem known as MONOTONE NOT-ALL-EQUAL 3-SAT [106]. We say that a 3-CNF formula φ is *positive* if it contains no negative literal $\neg x_i$.

MONOTONE NOT-ALL-EQUAL 3-SAT (MONOTONE NAE-3-SAT)	
Input:	A set of variables V and a positive 3-CNF formula φ over V.
Output:	Is there a truth assignment $\phi\colon V \to \{T, F\}$ such that each clause in φ contains at least one true and at least one false variable?

If an assignment with the desired property exists, we say that φ is *NAE-satisfiable*. Note that MONOTONE NAE-3-SAT remains NP-hard even if we require that every clause in φ contains exactly three positive, pairwise different literals.

This proof shares a fair portion of notation and concepts with the proof of Proposition 3.10 and therefore, we start by recalling these. We state all of them in their general form for an arbitrary simple pattern α.

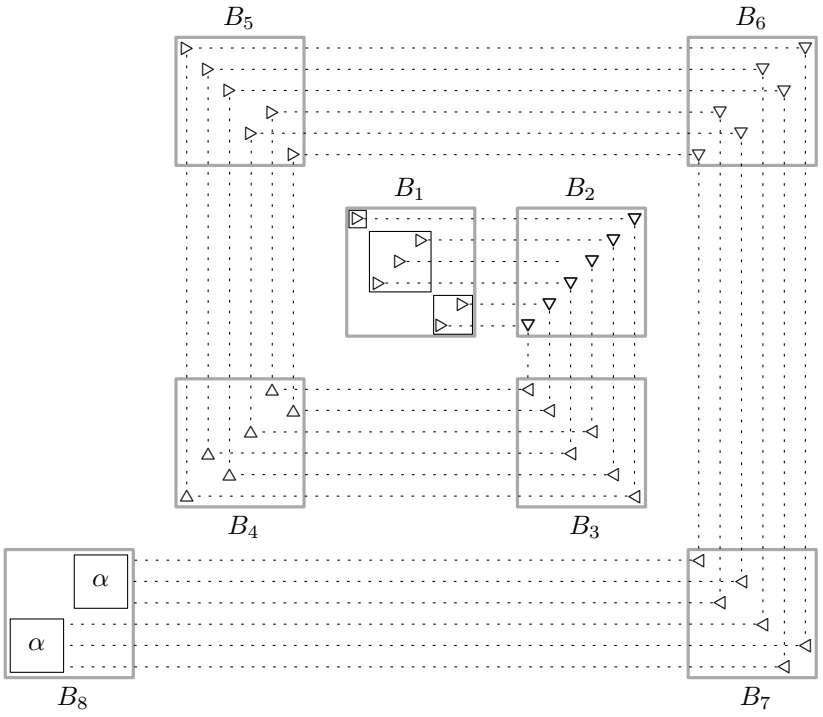

Figure 7.2: The global structure of the permutation π used in the proof of Theorems 7.9 and 7.11.

A 2-coloring $\psi\colon \pi \to \{\text{red}, \text{blue}\}$ is *admissible*, if π does not contain a monochromatic copy of α. Clearly, π belongs to $\mathrm{Av}(\alpha) \odot \mathrm{Av}(\alpha)$ if and only if it has an admissible coloring. The permutations $\alpha^{\triangleright}$, $\alpha^{\triangleleft}$, $\alpha^{\triangle}$ and $\alpha^{\triangledown}$ are obtained from α by removing the rightmost, leftmost, topmost and bottommost element, respectively. We say that a point set P forms a right arrow if it is isomorphic to $\alpha^{\triangleright}$. A point p is said to lie in the range of the right arrow P, if p lies to the right of P and the point set $P \cup \{p\}$ is isomorphic to α. Similarly, we define top, left and down arrows as point sets isomorphic to $\alpha^{\triangle}$, $\alpha^{\triangleleft}$ and $\alpha^{\triangledown}$, respectively. Their ranges are defined analogously.

Fix an admissible coloring ψ. If π contains a monochromatic arrow of any kind in ψ then all the points in its range must receive the opposite color. In particular, if we have a sequence of disjoint arrows $P_1, \ldots, P_\ell$ such that P_{i+1} lies in the range of P_i and the arrow P_1 is monochromatic, then in fact each P_i must be monochromatic and their colors alternate along the sequence.

Construction of π. Given an instance (V, φ) of Monotone NAE-3-SAT, we will construct a permutation $\pi \equiv \pi(V, \varphi)$ such that $\pi \in \mathrm{Av}(\alpha) \odot \mathrm{Av}(\alpha)$ if and only if φ is NAE-satisfiable. Suppose that $V = \{x_1, \ldots, x_n\}$ and φ consists of clauses $K_1, \ldots, K_m$.

Let us now describe the overall structure of the permutation π. Refer to Figure 7.2. The elements of π are partitioned into a sequence of blocks $B_1, B_2, \ldots, B_\ell$ forming a clockwise spiral starting with the innermost block B_1. Thus, for instance, if i is a multiple of four, then any element in the block B_i is to the left of all the elements in the block B_{i-1}, and it is to the left and below all the elements in the blocks $B_1, B_2, \ldots, B_{i-2}$. The total number ℓ of blocks is a multiple of four; we will specify the exact value of ℓ later.

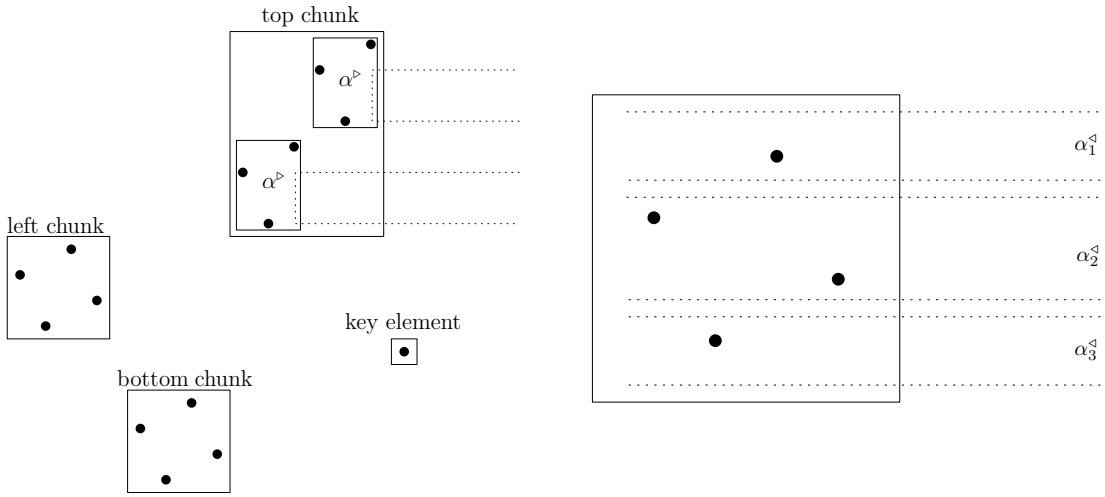

Figure 7.3: Left: the structure of a variable gadget $\sigma(x_i)$, for a variable x_i appearing in two clauses. The dotted lines are the boundaries of the ranges of the two copies of $\alpha^{\triangleright}$. Right: the structure of a clause gadget $\tau(K_j)$, for a clause K_j. The dotted lines are the boundaries of the ranges of $\alpha_1^{\triangleleft}$, $\alpha_2^{\triangleleft}$ and $\alpha_3^{\triangleleft}$ in $B_{\ell-1}$.

The first block B_1 is the *variable block*. It contains, for each variable $x_i \in V$, a copy of a *variable gadget* $\sigma(x_i)$, to be described later. The variable gadgets are arranged inside B_1 into a decreasing sequence, that is, the elements of B_1 are isomorphic to the permutation $\sigma(x_1) \ominus \sigma(x_2) \ominus \cdots \ominus \sigma(x_n)$.

The final block B_ℓ is the *clause block*. It contains, for each clause K_j in φ, a copy of a *clause gadget* $\tau(K_j)$, with the clause gadgets being arranged into an increasing sequence $\tau(K_1) \oplus \tau(K_2) \oplus \cdots \oplus \tau(K_m)$.

Let us now describe the variable gadget $\sigma(x_i)$ for a variable $x_i \in V$. Suppose that x_i appears in d_i distinct clauses. The gadget $\sigma(x_i)$ is obtained by inflating $\alpha = 3142$ as follows (see the left part of Figure 7.3):

- The leftmost and the bottommost element of α are both inflated into a copy of α; we will call the two copies of α obtained by this inflation the *left chunk* and the *bottom chunk* of $\sigma(x_i)$.
- The rightmost element remains a singleton (i.e., it is not inflated at all); we will call this element the *key element* of $\sigma(x_i)$.
- The topmost element is inflated into a direct sum of d_i copies of $\alpha^{\triangleright}$, and each of these d_i copies of $\alpha^{\triangleright}$ is associated with a distinct clause containing x_i. We call this group of elements the *top chunk* of $\sigma(x_i)$.

Observe that in any admissible two-coloring of $\sigma(x_i)$, all the elements in the top chunk must have a different color than the key element. This is because both the left chunk and the bottom chunk necessarily contain elements of both colors. Therefore, if the top chunk contained an element sharing a color with the key element, this would create a monochromatic copy of α and the coloring would not be admissible.

Let us now look at the intermediate blocks $B_2, B_3, \cdots, B_{\ell-1}$. Recall that for every clause K_j and every variable $x_i \in K_j$, the top chunk in the variable gadget $\sigma(x_i)$ contains a right arrow associated with the clause K_j. In particular, B_1 contains exactly $3m$ such right arrows, one for each pair (i, j) such that $x_i \in K_j$. The block B_2 will contain $3m$ down arrows, each down arrow placed fully inside

the range of one of the $3m$ right arrows from B_1. Likewise, the block B_3 will contain $3m$ left arrows, each one in the range of a distinct top arrow from B_2. In general, for any $i \in \{2, \ldots, \ell - 1\}$, the elements in a block B_i consist of $3m$ disjoint arrows, all oriented towards B_{i+1}, and each of them inside the range of a distinct arrow in B_{i-1}.

In particular, for each pair (i, j) such that the variable x_i is in the clause K_j, the permutation π contains a sequence of $\ell - 1$ arrows arranged into a clockwise spiral, with one arrow of the sequence in each of the blocks $B_1, \ldots, B_{\ell-1}$, and with the property that each arrow in the sequence except the first is in the range of the immediately preceding arrow. We call this sequence of arrows the *track* of the pair (i, j).

Let us now specify the relative position of the arrows inside each block B_i, and therefore the relative position of the $3m$ tracks in π. In the block B_2, the arrows form an increasing sequence, that is, the elements of π inside B_2 are order-isomorphic to the direct sum of $3m$ copies of $\alpha^{\triangledown}$.

For every subsequent block B_i with $3 \leq i \leq \ell - 1$, we distinguish two cases, depending on the parity of i. If i is even, then the $3m$ arrows in B_i are arranged into layered sequence with layers of size at most two; that is, the elements inside B_i are isomorphic to a permutation of length $3m$ from the Fibonacci class $\oplus 21$ with each element inflated by a copy of α^*, where α^* is either equal to $\alpha^{\triangledown}$ (if $i \equiv 2 \mod 4$), or to $\alpha^{\triangle}$ (if $i \equiv 0 \mod 4$). Symmetrically, for $i \geq 3$ odd, the $3m$ arrows inside B_i are arranged to form an inflation of a permutation of length $3m$ from the class $\ominus 12$.

The effect of this arrangement of arrows is to ensure that when the tracks pass through a block B_i, their relative position changes by swapping in parallel some pairs of neighboring tracks. Note that this is the same procedure that we used when 'sorting by gadgets' in the proof of Theorem 5.6.

Inside the block B_1, the arrows of the $3m$ tracks are ordered in such a way that the tracks involving the variable x_1 are the outermost, followed by those involving x_2 and so on, up to the tracks involving x_n which are nearest to the center of the spiral. Inside the block B_2, the relative position of the tracks will remain the same.

By suitably exchanging some set of pairs of adjacent tracks in each of the blocks $B_3, \ldots, B_{\ell-1}$, we eventually reach an ordering in which the three tracks involving the clause K_1 are the outermost, followed by those involving K_2, and so on, up to the three tracks involving K_m that are nearest to the center. The number ℓ of blocks is chosen to be large enough, so that the initial ordering of the tracks inside B_1 can be transformed into the desired final ordering by at most $\ell - 3$ parallel swaps. It is easy to see that we can choose $\ell \in O(m)$.

Let us now describe the clause block B_ℓ. Recall that this block contains an increasing sequence $\tau(K_1) \oplus \tau(K_2) \oplus \cdots \oplus \tau(K_m)$ of clause gadgets. Let K_j be a clause in φ. Suppose that $\alpha_1^{\triangleleft}$, $\alpha_2^{\triangleleft}$ and $\alpha_3^{\triangleleft}$ are the three left arrows in $B_{\ell-1}$ belonging to the three tracks involving the clause K_j, numbered top to bottom. The clause gadget $\tau(K_j)$ then consists of four elements forming a copy of the pattern $\alpha = 3142$, where the topmost element of $\tau(K_j)$ is in the range of $\alpha_1^{\triangleleft}$, the bottommost is in the range of $\alpha_3^{\triangleleft}$ and the remaining two are in the range of $\alpha_2^{\triangleleft}$. See the right part of Figure 7.3. This completes the description of π.

Claim 7.10. *The* Monotone NAE-3-SAT *instance* (V, φ) *has a NAE-satisfying assignment if and only if* π *has an admissible coloring.*

Correctness ("if"). Suppose first that π has an admissible coloring ψ. Define a truth assignment $\phi\colon V \to \{T, F\}$ by setting $\phi(x_i) = T$ if and only if the key element in the variable gadget $\sigma(x_i)$ is blue. We claim that ϕ is a NAE-satisfying assignment. To see this, choose a clause K_j in φ. If ϕ does not NAE-satisfy K_j, it means that the key elements of the three variables in K_j all have the same color in ψ, and therefore all the four elements in the clause gadget $\tau(K_j)$ have the same color, contradicting the admissibility of ψ.

Correctness ("only if"). Suppose that (V, φ) has a NAE-satisfying assignment ϕ, and let us define a red-blue coloring ψ of π as follows: the key element of the variable gadget $\sigma(x_i)$ is blue if and only if $\phi(x_i) = T$. The elements of the left chunk and of the bottom chunk of $\sigma(x_i)$ are colored arbitrarily in such a way that each of these chunks has at least one red and at least one blue element. Finally, the top chunk is colored by the color that differs from the color of the key element. Having colored all the variable gadgets, we extend the coloring to the entire π by following, inside each track, the rule that any element in the range of a monochromatic arrow must have a color that differs from the color of the arrow.

We claim that the coloring ψ is admissible. To see this, let π_R and π_B be the subpermutations of π formed by the red elements and the blue elements, respectively. We claim that both these permutations avoid α. Let us look at π_R, the case of π_B being analogous. To check that π_R avoids α, we will repeatedly apply Observation 3.12 which we now restate.

Observation 3.12. *Suppose that* γ *is a permutation which contains an interval* I*, and suppose that* I *has no copy of the simple pattern* α*. Let* γ^- *be a permutation obtained from* γ *by 'deflating' the interval* I*, i.e., by replacing* I *by a single element. Then* γ *contains* α *if and only if* γ^- *contains* α*.*

Our goal is to show that by repeatedly deflating α-avoiding intervals, π_R can be reduced to a permutation from the clockwise spiral which is a subclass of Av(3142) (see Subsection 2.3.1).

Consider first the red elements inside a variable gadget $\sigma(x_i)$. If the key element in this gadget is red, then there are no red elements in the top chunk of $\sigma(x_i)$, and the red elements in the left and the bottom chunk form an interval in π_R of size at most three. Thus, each chunk can be deflated in π_R to a single point, with the top chunk being empty. In particular, the deflation of chunks turns the red part of $\sigma(x_i)$ into an interval of size at most three, which can then be deflated to a single point.

Similarly, if the key point of $\sigma(v_i)$ is blue, the top chunk of the gadget is all red. Since all the elements in B_2 in the range of the arrows in the top chunk are blue, the entire top chunk is an interval in π_R, and can be deflated to a single point. We then easily see that the red part of $\sigma(x_i)$ can again be deflated to a point. Thus, the red part of B_1 is deflated to a single decreasing sequence.

Consider now a block B_i with $2 \leq i \leq \ell - 1$, and let us look at the $3m$ arrows in this block that belong to the $3m$ tracks. If such an arrow is red, all the elements in its range are blue, and therefore every such red arrow is an interval in π_R and

can be deflated to a point. These points will form a permutation either from the class $\oplus 21$ or from its symmetry $\ominus 12$. Suppose without loss of generality that i is even as the other case is symmetric. In this case, the points obtained by deflating the red arrows in B_i form a permutation from the class $\oplus 21$ and we focus on pairs of points in the 'wrong' position, i.e. forming a pattern 21. Any such pair of points is necessarily consecutive, and since the corresponding two tracks only contain blue arrows in B_{i-1} and B_{i+1}, the two red points again form an interval in π_R, and can be deflated to a single point, reducing the red part of B_i to an increasing sequence.

It remains to deal with the clause block B_ℓ. We will show that the red part of each clause gadget $\tau(K_j)$ can be deflated either to a single point or to a direct sum of two points, and therefore the red part of B_ℓ can be deflated to an increasing sequence of points. Recall that $\tau(K_j)$ is vertically partitioned into three parts, each part belonging to the range of a different arrow in $B_{\ell-1}$, with the top part and the bottom part of $\tau(K_j)$ consisting of one element each, and the middle part consisting of two elements. Since ϕ was a NAE-satisfying assignment, at most two of the three parts can be red. If the top and bottom parts are red, we do not need to deflate anything, as these two parts form a direct sum of two singletons. In any other case, we easily see that the red points of $\tau(K_j)$ form a single interval and can be deflated into a point.

We conclude that ψ is indeed admissible. This completes the proof. □

We remark that we have not put much effort into making the reduction more efficient and thus, we do not obtain reasonable conditional lower bounds under ETH. We continue by generalizing the reduction for arbitrary simple pattern α.

Theorem 7.11. *For any simple permutation α of length at least 4,* $\mathrm{Av}(\alpha) \odot \mathrm{Av}(\alpha)$-RECOGNITION *is* NP*-complete.*

Proof. Let us explain how to adapt the proof of Theorem 7.9 to the case of general α. Let $\alpha = \alpha_1, \alpha_2, \ldots, \alpha_k$ be a simple permutation of size $k \geq 4$. Note that α must contain 2413 or 3142, because otherwise it would be separable, but there are no separable simple permutations of size 3 or more. Suppose therefore that α contains 3142, the other case being symmetric.

The general structure of π remains the same as in the proof of Theorem 7.9 (refer to Figure 7.2). Moreover, the generalization of the variable gadget $\sigma(x_i)$ is straightforward: we take a copy of α, we inflate all its elements except the rightmost one and the topmost one by a copy of α, we inflate the topmost element by the d_i-fold direct sum $\alpha^\triangleright \oplus \alpha^\triangleright \oplus \cdots \oplus \alpha^\triangleright$, where d_i is the number of clauses containing the variable x_i.

The main difficulty lies in the construction of the clause gadget $\tau(K_j)$. Let α_b and α_t be the bottommost and the topmost element of α, respectively; in other words, we have $\alpha_b = 1$ and $\alpha_t = k$. Note that both b and t are greater than 1 and smaller then k, for otherwise α could be written as a direct sum or skew sum of smaller permutations, and it would not be simple.

Suppose now that $b < t$, that is, the bottommost element of α is to the left of the topmost one. In such case, we may directly generalize the construction of $\tau(K_j)$ of the previous proof: $\tau(K_j)$ will consist of a copy of α distributed into the ranges of three copies of $\alpha^\triangleleft$, with the topmost element of $\tau(K_j)$ in the range of the topmost $\alpha_1^\triangleleft$, the bottommost element of $\tau(K_j)$ in the range of the bottommost $\alpha_3^\triangleleft$,

and the remaining $k-2$ elements of $\tau(K_j)$ in the range of the middle $\alpha_2^{\triangleleft}$. With these gadgets, we can perform the reduction from MONOTONE NAE-3-SAT and prove its correctness exactly as in the proof of Theorem 7.11.

Now consider the case $b > t$. In this situation, we will look at the leftmost and rightmost element of α, that is, the elements α_1 and α_k. If $\alpha_1 > \alpha_k$, we replace the permutation α by the permutation $\alpha' = (\alpha^{-1})^r$, which corresponds to the counter-clockwise 90-degree rotation of α. We see that α' still contains 3142, and its topmost element is to the right of the bottommost one, therefore our situation is reduced to the previous case, and we may conclude that $\mathrm{Av}(\alpha') \odot \mathrm{Av}(\alpha')$-RECOGNITION is NP-complete. By symmetry, the problem of $\mathrm{Av}(\alpha) \odot \mathrm{Av}(\alpha)$-RECOGNITION is NP-complete as well.

Finally, suppose that $b > t$ and $\alpha_1 < \alpha_k$. Then the four elements α_1, α_b, α_t and α_k form a copy of 2413 in α. Therefore, the reverse α^r of α contains 3142, and its topmost element is to the right of the bottommost one, and we may again conclude that $\mathrm{Av}(\alpha^r) \odot \mathrm{Av}(\alpha^r)$-RECOGNITION, and therefore also $\mathrm{Av}(\alpha) \odot \mathrm{Av}(\alpha)$-RECOGNITION, is NP-complete. □

Despite our results, the complexity of many cases of $\mathcal{C} \odot \mathcal{D}$-RECOGNITION remains open. Perhaps the most natural question is to consider the merge of classes that have bounded grid-width but do not belong to NLOL. Classes of this type include many important examples, such as the class $\mathrm{Av}(2413, 3142)$ of separable permutations, or the class $\mathrm{Av}(213)$ and its symmetries.

Open problem 7.12. *What is the complexity of $\mathcal{C} \odot \mathcal{D}$-RECOGNITION when $\mathcal{C}$ and $\mathcal{D}$ are any two (possibly identical) classes from the set* $\{\mathrm{Av}(2413, 3142), \mathrm{Av}(213), \mathrm{Av}(231), \mathrm{Av}(132), \mathrm{Av}(312)\}$*?*

The complexity of recognizing permutations merged from two separable subpermutations, or more generally from k separable permutations, is also mentioned as an open problem by Hoàng and Le [81, Problem 1], who state it in an equivalent graph-theoretic setting as generalized coloring of permutation graphs by k colors, with each color inducing a P_4-free subgraph.

It is also natural to consider 'unbalanced' merges, when one of the two classes is very simple, e.g., the class $\mathrm{Av}(21)$ of increasing permutations. Our results imply that $\mathcal{C} \odot \mathrm{Av}(21)$-RECOGNITION is tractable when $\mathcal{C}$ is in NLOL or when $\mathcal{C}$ is a GT-recognizable class of bounded grid-width, but we know nothing about the remaining cases.

Open problem 7.13. *For which classes $\mathcal{C}$ is the $\mathcal{C} \odot \mathrm{Av}(21)$-RECOGNITION solvable in polynomial time?*

Conclusion

We collect all the open problems and unresolved questions that occurred throughout the thesis. In Chapter 2, we showed that any permutation class with the long path property has unbounded tree-width (Proposition 2.33). However, there is no known example of a class without the long path property having unbounded tree-width and we conjecture that these two properties are equivalent.

Conjecture 2.31. *A permutation class $\mathcal{C}$ has unbounded tree-width if and only if it has the long path property.*

Furthermore, we used our tools to prove tight asymptotic bounds on the tree-width growth of all principal classes with five exceptions up to symmetry. We restate our question asking how quickly the tree-width can grow in these classes.

Open problem 2.58. *What is the tree-width growth of* $\mathrm{Av}(\sigma)$*-PPM, when σ is a permutation from the set* $\{3412, 3142, 4213, 4123, 41352\}$*?*

We have not presented any explicit open problems in Chapter 3. One possible direction of future research might be to further unravel the boundary of FO- and MSO-definability. In other words, we ask to determine which permutation classes can be defined using first-order or monadic second-order sentences in TOTO. A less ambitious instance of this question is presented by merge classes. We showed that the class $\mathrm{Av}(\alpha) \odot \mathrm{Av}(\alpha)$ is not FO-definable for any simple pattern α (Proposition 3.10). How does the situation change when we look at non-simple α? It is unclear where the boundary is since, e.g., for arbitrarily large monotone α, the class $\mathrm{Av}(\alpha) \odot \mathrm{Av}(\alpha)$ is actually finitely based and thus, definable by an FO sentence.

Open problem 7.14. *Which classes of the form* $\mathrm{Av}(\alpha) \odot \mathrm{Av}(\alpha)$ *are definable by FO sentences in TOTO?*

In Chapter 4, the complexity of unrestricted PPM was thoroughly explored both from the classical and from the parameterized point of view. However, the pattern-restricted variant $\mathcal{C}$-Pattern PPM presents opportunities for future research. For instance assuming that $\mathcal{C}$ has the computable deep tree property, we showed that $\mathcal{C}$-Pattern #PPM cannot be solved in time $f(k) \cdot n^{o(k/\log^2 k)}$ where k is the length of the pattern and n is the length of the text, unless ETH fails (Theorem 4.27). However, the best lower bound we can impose on the tree-width growth of $\mathcal{C}$ is $\Omega(n/\log n)$ (Proposition 2.39) which implies that the tree-width based algorithm (Theorem 4.13) cannot solve $\mathcal{C}$-Pattern #PPM in time $f(k) \cdot n^{o(k/\log k)}$. Notice the single logarithmic factor in the exponent. Can it be that, in fact, $\mathcal{C}$-Pattern #PPM cannot be solved in time $f(k) \cdot n^{o(k/\log k)}$ for any class with the computable deep tree property?

Additionally, we have not been able to utilize the bicycle property in any of our reductions. Is there a way to leverage the bicycle property in order to obtain more efficient reductions and as a result, sharper conditional lower bounds under ETH?

In Chapter 5, we pursued the complexity of the text-restricted PPM variant $\mathcal{C}$-PPM. Even though we made significant progress, the complete picture remains elusive, unlike in the case of $\mathcal{C}$-Pattern PPM. The most striking open problem is to determine the complexity of Av(σ)-PPM for the five remaining principal classes up to symmetry.

Open problem 5.16. *What is the complexity of* Av(σ)*-PPM, when* σ *is a permutation from the set* $\{3412, 3142, 4213, 4123, 41352\}$*?*

In Chapter 6, we showed that a sharp dichotomy exists for the tree-width growth and complexity of Grid($\mathcal{M}$)-Pattern PPM over monotone grid classes (Theorems 6.1 and 6.7). We can also characterize exactly when a general grid class Grid($\mathcal{M}$) has bounded tree-width and when the corresponding Grid($\mathcal{M}$)-Pattern PPM problem is NP-hard (Theorem 6.8). However, the situation is widely open for Grid($\mathcal{M}$)-PPM.

Open problem 7.15. *Characterize the complexity of* Grid($\mathcal{M}$)*-PPM for non-monotone gridding matrices.*

Finally, Chapter 7 is definitely the most fruitful source of open problems since the complexity of $\mathcal{C} \odot \mathcal{D}$-recognition remains in most cases unresolved. We restate the two particular questions of interest posed at the end of the chapter. The first asks about merges involving subclasses of separable permutations. The second concerns the case of 'unbalanced' merges, when one of the two classes is very simple, e.g., the class Av(21) of increasing permutations.

Open problem 7.12. *What is the complexity of* $\mathcal{C} \odot \mathcal{D}$*-recognition when* $\mathcal{C}$ *and* $\mathcal{D}$ *are any two (possibly identical) classes from the set* $\{\mathrm{Av}(2413, 3142), \mathrm{Av}(213), \mathrm{Av}(231), \mathrm{Av}(132), \mathrm{Av}(312)\}$*?*

Open problem 7.13. *For which classes* $\mathcal{C}$ *is the* $\mathcal{C} \odot$ Av(21)*-recognition solvable in polynomial time?*

Permutations in higher dimensions. Finally, we propose a possible future direction of research. There have been lately some works considering the extension of permutations to higher dimensions [15, 43, 31]. A *d-dimensional permutation* can be defined easily either as a subset of $[n]^d$ in general position or equivalently as a relational structure consisting of d linear orders.

Luckily, we treated both dimensions of permutations as equals[1] and owing to that, many of our results can be generalized to higher dimensions in a straightforward way. For example, all the parameters defined in Chapter 2 can be extended to higher dimensions and moreover, it can be shown that similar relationships hold between them. Similarly, the algorithm for PPM based on CSPs (Subsection 4.2.5) carries through with minimal modifications. On the other hand, we know that higher dimensional permutations act differently at least in some aspects. For instance, Guillemot and Marx [77] proved that the 3-dimensional analogue of PPM is W[1]-hard with respect to the pattern length, which sharply contrasts with their fpt-algorithm for 2-dimensional PPM (Theorem 4.5). Therefore, we find it intriguing to explore the different behavior that occurs in higher dimensions.

[1] that is often not the case when one works with permutations in the one-line notation.

Bibliography

[1] Dimitris Achlioptas, Jason I. Brown, Derek G. Corneil, and Michael Molloy. The existence of uniquely $-G$ colourable graphs. *Discrete Math.*, 179(1-3):1–11, 1998. DOI: 10.1016/S0012-365X(97)00022-8.

[2] Shlomo Ahal and Yuri Rabinovich. On complexity of the subpattern problem. *SIAM Journal on Discrete Mathematics*, 22(2):629–649, 2008. DOI: 10.1137/S0895480104444776.

[3] Michael Albert and Vít Jelínek. Unsplittable classes of separable permutations. *Electron. J. Combin.*, 23(2):Paper 2.49, 20, 2016. DOI: 10.37236/6115.

[4] Michael Albert, Jay Pantone, and Vincent Vatter. On the growth of merges and staircases of permutation classes. *Rocky Mountain J. Math.*, 49(2):355–367, 2019. DOI: 10.1216/RMJ-2019-49-2-355.

[5] Michael H. Albert. On the length of the longest subsequence avoiding an arbitrary pattern in a random permutation. *Random Structures Algorithms*, 31(2):227–238, 2007. DOI: 10.1002/rsa.20140.

[6] Michael H. Albert, Robert E. L. Aldred, Mike D. Atkinson, and Derek A. Holton. Algorithms for pattern involvement in permutations. In *Algorithms and computation (Christchurch, 2001)*, volume 2223, pages 355–366, 2001. DOI: 10.1007/3-540-45678-3_31.

[7] Michael H. Albert, M. D. Atkinson, Mathilde Bouvel, Nik Ruškuc, and Vincent Vatter. Geometric grid classes of permutations. *Trans. Amer. Math. Soc.*, 365(11):5859–5881, 2013. DOI: 10.1090/S0002-9947-2013-05804-7.

[8] Michael H. Albert and Mike D. Atkinson. Simple permutations and pattern restricted permutations. *Discrete Math.*, 300(1-3):1–15, 2005. DOI: 10.1016/j.disc.2005.06.016.

[9] Michael H. Albert, Mathilde Bouvel, and Valentin Féray. Two first-order logics of permutations. *J. Combin. Theory Ser. A*, 171:105158, 46, 2020. DOI: 10.1016/j.jcta.2019.105158.

[10] Michael H. Albert, Marie-Louise Lackner, Martin Lackner, and Vincent Vatter. The complexity of pattern matching for 321-avoiding and skew-merged permutations. *Discrete Math. Theor. Comput. Sci.*, 18(2):Paper No. 11, 17, 2016. DOI: 10.46298/dmtcs.1308.

[11] Michael H. Albert, Steve Linton, and Nikola Ruskuc. The insertion encoding of permutations. *Electron. J. Combin.*, 12:Research Paper 47, 31, 2005. DOI: 10.37236/1944.

[12] Vladimir E. Alekseev, Alastair Farrugia, and Vadim V. Lozin. New results on generalized graph coloring. *Discrete Math. Theor. Comput. Sci.*, 6(2):215–221, 2004.

[13] Ragnar Pall Ardal, Tomas Ken Magnusson, Émile Nadeau, Bjarni Jens Kristinsson, Bjarki Agust Gudmundsson, Christian Bean, Henning Ulfarsson, Jon Steinn Eliasson, Murray Tannock, Alfur Birkir Bjarnason, Jay Pantone, Arnar Bjarni Arnarson, and Sigurjón Ingi Jónsson. Permutatriangle/permuta: version 2.1.0, version v2.1.0, 2021. DOI: 10.5281/zenodo.4945792.

[14] Sanjeev Arora and Boaz Barak. *Computational complexity*. Cambridge University Press, Cambridge, 2009, pages xxiv+579. DOI: 10.1017/CBO9780511804090. A modern approach.

[15] Andrei Asinowski and Toufik Mansour. Separable d-permutations and guillotine partitions. *Ann. Comb.*, 14(1):17–43, 2010. DOI: 10.1007/s00026-010-0043-8.

[16] Aistis Atminas, Robert Brignall, Vadim V. Lozin, and Juraj Stacho. Minimal classes of graphs of unbounded clique-width defined by finitely many forbidden induced subgraphs. *Discrete Applied Mathematics*, 295:57–69, 2021. DOI: 10.1016/j.dam.2021.02.007.

[17] Sergey Avgustinovich, Sergey Kitaev, and Alexandr Valyuzhenich. Avoidance of boxed mesh patterns on permutations. *Discrete Applied Mathematics*, 161(1-2):43–51, 2013. DOI: 10.1016/j.dam.2012.08.015.

[18] Eric Babson and Einar Steingrímsson. Generalized permutation patterns and a classification of the Mahonian statistics. *Sém. Lothar. Combin.*, 44:Art. B44b, 18, 2000.

[19] Jakub Balabán and Petr Hliněný. Twin-width is linear in the poset width. In *16th International Symposium on Parameterized and Exact Computation, IPEC 2021, September 8-10, 2021, Lisbon, Portugal*, volume 214, 6:1–6:13, 2021. DOI: 10.4230/LIPIcs.IPEC.2021.6.

[20] Christian Bean, Bjarki Gudmundsson, and Henning Ulfarsson. Automatic discovery of structural rules of permutation classes. *Math. Comp.*, 88(318):1967–1990, 2019. DOI: 10.1090/mcom/3386.

[21] Christian Bean, Henning Ulfarsson, and Anders Claesson. Enumerations of permutations simultaneously avoiding a vincular and a covincular pattern of length 3. *J. Integer Seq.*, 20(7):Art. 17.7.6, 25, 2017.

[22] Benjamin Aram Berendsohn. *Complexity of Permutation Pattern Matching*. Master's thesis, Freie Universität Berlin, 2019.

[23] Benjamin Aram Berendsohn, László Kozma, and Dániel Marx. Finding and counting permutations via CSPs. *Algorithmica*, 83(8):2552–2577, 2021. DOI: 10.1007/s00453-021-00812-z.

[24] Frank Bernhart and Paul C. Kainen. The book thickness of a graph. *J. Combin. Theory Ser. B*, 27(3):320–331, 1979. DOI: 10.1016/0095-8956(79)90021-2.

[25] David I. Bevan. Growth rates of geometric grid classes of permutations. *Electron. J. Combin.*, 21(4):Paper 4.51, 17, 2014. DOI: 10.37236/4834.

[26] David I. Bevan, Robert Brignall, Andrew Elvey Price, and Jay Pantone. Staircases, dominoes, and the growth rate of 1324-avoiders. *Electronic Notes in Discrete Mathematics*, 61:123–129, 2017. DOI: https://doi.org/10.1016/j.endm.2017.06.029. The European Conference on Combinatorics, Graph Theory and Applications (EUROCOMB'17).

[27] Hans L. Bodlaender. Some classes of graphs with bounded treewidth. *Bull. EATCS*, 36:116–125, 1988.

[28] Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. A c^k n 5-approximation algorithm for treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016. DOI: 10.1137/130947374.

[29] Miklós Bóna. A new record for 1324-avoiding permutations. *Eur. J. Math.*, 1(1):198–206, 2015. DOI: 10.1007/s40879-014-0020-6.

[30] Miklós Bóna. A new upper bound for 1324-avoiding permutations. *Combin. Probab. Comput.*, 23(5):717–724, 2014. DOI: 10.1017/S0963548314000091.

[31] Nicolas Bonichon and Pierre-Jean Morel. Baxter d-permutations and other pattern avoiding classes. arXiv:2202.12677, 2022.

[32] Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width II: small classes. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1977–1996, 2021. DOI: 10.1137/1.9781611976465.118.

[33] Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width III: max independent set, min dominating set, and coloring. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198, 35:1–35:20, 2021. DOI: 10.4230/LIPIcs.ICALP.2021.35.

[34] Édouard Bonnet, Eun Jung Kim, Amadeus Reinald, and Stéphan Thomassé. Twin-width VI: the lens of contraction sequences. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 1036–1056, 2022. DOI: 10.1137/1.9781611977073.45.

[35] Édouard Bonnet, Eun Jung Kim, Amadeus Reinald, Stéphan Thomassé, and Rémi Watrigant. Twin-width and polynomial kernels. In *16th International Symposium on Parameterized and Exact Computation, IPEC 2021, September 8-10, 2021, Lisbon, Portugal*, volume 214, 10:1–10:16, 2021. DOI: 10.4230/LIPIcs.IPEC.2021.10.

[36] Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width I: tractable FO model checking. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 601–612, 2020. DOI: 10.1109/FOCS46700.2020.00062.

[37] Édouard Bonnet, Jaroslav Nesetril, Patrice Ossona de Mendez, Sebastian Siebertz, and Stéphan Thomassé. Twin-width and permutations. arXiv:2102.06880, 2021.

[38] Piotr Borowiecki. Computational aspects of greedy partitioning of graphs. *J. Comb. Optim.*, 35(2):641–665, 2018. DOI: 10.1007/s10878-017-0185-2.

[39] Prosenjit Bose, Jonathan F. Buss, and Anna Lubiw. Pattern matching for permutations. *Inform. Process. Lett.*, 65(5):277–283, 1998. DOI: 10.1016/S0020-0190(97)00209-3.

[40] Jean Bourgain and Amir Yehudayoff. Expansion in $\mathrm{SL}_2(\mathbb{R})$ and monotone expanders. *Geom. Funct. Anal.*, 23(1):1–41, 2013. DOI: 10.1007/s00039-012-0200-9.

[41] Mireille Bousquet-Mélou, Anders Claesson, Mark Dukes, and Sergey Kitaev. $(2+2)$-free posets, ascent sequences and pattern avoiding permutations. *J. Combin. Theory Ser. A*, 117(7):884–909, 2010. DOI: 10.1016/j.jcta.2009.12.007.

[42] Petter Brändén and Anders Claesson. Mesh patterns and the expansion of permutation statistics as sums of permutation patterns. *Electron. J. Combin.*, 18(2):Paper 5, 14, 2011. DOI: 10.37236/2001.

[43] Samuel Braunfeld. The undecidability of joint embedding for 3-dimensional permutation classes. *Discrete Mathematics & Theoretical Computer Science*, vol. 22 no. 2, Permutation Patterns 2019, 2021. DOI: 10.46298/dmtcs.6165.

[44] Robert Brignall. Grid classes and partial well order. *J. Combin. Theory Ser. A*, 119(1):99–116, 2012. DOI: 10.1016/j.jcta.2011.08.005.

[45] Karl Bringmann, László Kozma, Shay Moran, and N. S. Narayanaswamy. Hitting set for hypergraphs of low VC-dimension. In *24th Annual European Symposium on Algorithms*, volume 57, Art. No. 23, 18, 2016. DOI: 10.4230/LIPIcs.ESA.2016.23.

[46] Hajo Broersma, Fedor V. Fomin, Jaroslav Nešetřil, and Gerhard J. Woeginger. More about subcolorings. *Computing*, 69(3):187–203, 2002. DOI: 10.1007/s00607-002-1461-1.

[47] Jason I. Brown. The complexity of generalized graph colorings. *Discrete Applied Mathematics*, 69(3):257–270, 1996. DOI: 10.1016/0166-218X(96)00096-0.

[48] Marie-Louise Bruner and Martin Lackner. A fast algorithm for permutation pattern matching based on alternating runs. *Algorithmica*, 75(1):84–117, 2016. DOI: 10.1007/s00453-015-0013-y.

[49] Marie-Louise Bruner and Martin Lackner. The computational landscape of permutation patterns. *Pure Math. Appl. (PU.M.A.)*, 24(2):83–101, 2013.

[50] J. Richard Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik Grundlagen Math.*, 6:66–92, 1960. DOI: 10.1002/malq.19600060105.

[51] Binh-Minh Bui Xuan, Michel Habib, and Christophe Paul. Revisiting T. Uno and M. Yagiura's algorithm (extended abstract). In *Algorithms and computation*, volume 3827, pages 146–155, 2005. DOI: 10.1007/11602613_16.

[52] Laurent Bulteau, Guillaume Fertin, Vincent Jugé, and Stéphane Vialette. Permutation Pattern Matching for Doubly Partially Ordered Patterns. In *33rd Annual Symposium on Combinatorial Pattern Matching*, 2022.

[53] Anders Claesson, Vít Jelínek, and Einar Steingrímsson. Upper bounds for the Stanley–Wilf limit of 1324 and other layered patterns. *J. Comb. Theory A*, 119:1680–1691, 8, 2012. DOI: 10.1016/j.jcta.2012.05.006.

[54] Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Christof Löding, Sophie Tison, and Marc Tommasi. *Tree Automata Techniques and Applications*. 2008, page 262.

[55] Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inform. and Comput.*, 85(1):12–75, 1990. DOI: 10.1016/0890-5401(90)90043-H.

[56] Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*, volume 138. Cambridge University Press, 2012.

[57] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000. DOI: 10.1007/s002249910009.

[58] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*. Springer, Cham, 2015, pages xviii+613. DOI: 10.1007/978-3-319-21275-3.

[59] Vida Dujmović, David Eppstein, and David R. Wood. Structure of graphs with locally restricted crossings. *SIAM Journal on Discrete Mathematics*, 31(2):805–824, 2017. DOI: 10.1137/16M1062879.

[60] Vida Dujmović, Anastasios Sidiropoulos, and David R. Wood. Layouts of expander graphs. *Chic. J. Theoret. Comput. Sci.*:Art. 1, 21, 2016. DOI: 10.4086/cjtcs.2016.001.

[61] Andrzej Ehrenfeucht. An application of games to the completeness problem for formalized theories. *Fund. Math.*, 49:129–141, 1961. DOI: 10.4064/fm-49-2-129-141.

[62] Tínaz Ekim, Pinar Heggernes, and Daniel Meister. Polar permutation graphs are polynomial-time recognisable. *European J. Combin.*, 34(3):576–592, 2013. DOI: 10.1016/j.ejc.2011.12.007.

[63] Murray Elder and Yoong Kuan Goh. k-pop stack sortable permutations and 2-avoidance. *Electron. J. Combin.*, 28(1):Paper No. 1.54, 15, 2021. DOI: 10.37236/9606.

[64] Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Trans. Amer. Math. Soc.*, 98:21–51, 1961. DOI: 10.2307/1993511.

[65] Paul Erdös and George Szekeres. A combinatorial problem in geometry. *Compositio Math.*, 2:463–470, 1935.

[66] Alastair Farrugia. Vertex-partitioning into fixed additive induced-hereditary properties is NP-hard. *Electron. J. Combin.*, 11(1):Research Paper 46, 9, 2004. DOI: 10.37236/1799.

[67] Jörg Flum and Martin Grohe. Model-checking problems as a basis for parameterized intractability. *Log. Methods Comput. Sci.*, 1(1):1:2, 36, 2005. DOI: 10.2168/LMCS-1(1:2)2005.

[68] Jacob Fox. Stanley–Wilf limits are typically exponential. arXiv:1310.8378v1, 2013.

[69] Roland Fraïssé. Sur quelques classifications des systèmes de relations. *Publ. Sci. Univ. Alger. Sér. A*, 1:35–182 (1955), 1954.

[70] Eugene C. Freuder. Complexity of k-tree structured constraint satisfaction problems. In *Proceedings of the 8th National Conference on Artificial Intelligence. Boston, Massachusetts, USA, July 29 - August 3, 1990, 2 Volumes*, pages 4–9, 1990.

[71] Robert Ganian, Petr Hliněný, Joachim Kneis, Alexander Langer, Jan Obdržálek, and Peter Rossmanith. Digraph width measures in parameterized algorithmics. *Discrete Applied Mathematics*, 168:88–107, 2014. DOI: 10.1016/j.dam.2013.10.038.

[72] Alice L. L. Gao and Sergey Kitaev. On partially ordered patterns of lengths 4 and 5 in permutations. *Electron. J. Combin.*, 26(3):Paper No. 3.26, 31, 2019. DOI: 10.37236/8605.

[73] Pawel Gawrychowski and Mateusz Rzepecki. Faster exponential algorithm for permutation pattern matching. In *5th Symposium on Simplicity in Algorithms, SOSA@SODA 2022, Virtual Conference, January 10-11, 2022*, pages 279–284, 2022. DOI: 10.1137/1.9781611977066.21.

[74] John R. Gilbert, Joan P. Hutchinson, and Robert Endre Tarjan. A separator theorem for graphs of bounded genus. *J. Algorithms*, 5(3):391–407, 1984. DOI: 10.1016/0196-6774(84)90019-1.

[75] Erich Grädel, Phokion G. Kolaitis, Leonid Libkin, Maarten Marx, Joel Spencer, Moshe Y. Vardi, Yde Venema, and Scott Weinstein. *Finite model theory and its applications*. Springer, Berlin, 2007, pages xiv+437.

[76] Martin Grohe and Dániel Marx. On tree width, bramble size, and expansion. *J. Combin. Theory Ser. B*, 99(1):218–228, 2009. DOI: 10.1016/j.jctb.2008.06.004.

[77] Sylvain Guillemot and Dániel Marx. Finding small patterns in permutations in linear time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 82–101, 2014. DOI: 10.1137/1.9781611973402.7.

[78] Sylvain Guillemot and Stéphane Vialette. Pattern matching for 321-avoiding permutations. In *Algorithms and computation*, volume 5878, pages 1064–1073, 2009. DOI: 10.1007/978-3-642-10631-6_107.

[79] Arie Nicolaas Habermann. Parallel Neighbor-Sort (or the Glory of the Induction Principle). Technical report, Computer Science Department, Carnegie Mellon University, 1972.

[80] Petr Hliněný and Sang-il Oum. Finding branch-decompositions and rank-decompositions. *SIAM Journal on Computing*, 38(3):1012–1032, 2008. DOI: 10.1137/070685920.

[81] Chinh T. Hoàng and Van Bang Le. P_4-free colorings and P_4-bipartite graphs. *Discrete Math. Theor. Comput. Sci.*, 4(2):109–122, 2001.

[82] Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bull. Amer. Math. Soc. (N.S.)*, 43(4):439–561, 2006. DOI: 10.1090/S0273-0979-06-01126-8.

[83] Sophie Huczynska and Vincent Vatter. Grid classes and the Fibonacci dichotomy for restricted permutations. *Electron. J. Combin.*, 13(1):Research Paper 54, 14, 2006. DOI: 10.37236/1080.

[84] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. System Sci.*, 62(2):367–375, 2001. DOI: 10.1006/jcss.2000.1727. Special issue on the Fourteenth Annual IEEE Conference on Computational Complexity (Atlanta, GA, 1999).

[85] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. System Sci.*, 63(4):512–530, 2001. DOI: 10.1006/jcss.2001.1774. Special issue on FOCS 98 (Palo Alto, CA).

[86] Vít Jelínek and Jan Kynčl. Hardness of permutation pattern matching. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 378–396, 2017. DOI: 10.1137/1.9781611974782.24.

[87] Vít Jelínek and Michal Opler. Splittability and 1-amalgamability of permutation classes. *Discrete Math. Theor. Comput. Sci.*, 19(2):Paper No. 4, 14, 2017. DOI: 10.23638/DMTCS-19-2-4.

[88] Vít Jelínek, Michal Opler, and Jakub Pekárek. A complexity dichotomy for permutation pattern matching on grid classes. In *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, volume 170, 52:1–52:18, 2020. DOI: 10.4230/LIPIcs.MFCS.2020.52.

[89] Vít Jelínek and Pavel Valtr. Splittings and Ramsey properties of permutation classes. *Advances in Applied Mathematics*, 63:41–67, 2015. DOI: 10.1016/j.aam.2014.10.003.

[90] André E. Kézdy, Hunter S. Snevily, and Chi Wang. Partitioning permutations into increasing and decreasing subsequences. *J. Combin. Theory Ser. A*, 73(2):353–359, 1996. DOI: 10.1016/s0097-3165(96)80012-4.

[91] Sergey Kitaev. Introduction to partially ordered patterns. *Discrete Applied Mathematics*, 155(8):929–944, 2007. DOI: 10.1016/j.dam.2006.09.011.

[92] Sergey Kitaev. Partially ordered generalized patterns. *Discrete Math.*, 298(1-3):212–229, 2005. DOI: 10.1016/j.disc.2004.03.017.

[93] Donald E. Knuth. *The art of computer programming. Vol. 1: Fundamental algorithms*. Addison-Wesley Publishing Co., Reading, Mass.-London-Don Mills, Ont., 1969, pages xxi+634. Second printing.

[94] Ephraim Korach and Nir Solel. Tree-width, path-width, and cutwidth. *Discrete Applied Mathematics*, 43(1):97–101, 1993. DOI: 10.1016/0166-218X(93)90171-J.

[95] Tuukka Korhonen. A single-exponential time 2-approximation algorithm for treewidth. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 184–192, 2021. DOI: 10.1109/FOCS52979.2021.00026.

[96] Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979. DOI: 10.1137/0136016.

[97] Vadim V. Lozin. Minimal classes of graphs of unbounded clique-width. *Ann. Comb.*, 15(4):707–722, 2011. DOI: 10.1007/s00026-011-0117-2.

[98] Tomas Ken Magnusson, Émile Nadeau, Christian Bean, Henning Ulfarsson, Jon Steinn Eliasson, and Jay Pantone. Permutatriangle/tilings: version 3.0.0, version v3.0.0, 2021. DOI: 10.5281/zenodo.4948344.

[99] Dániel Marx. Can you beat treewidth? *Theory Comput.*, 6:85–112, 2010. DOI: 10.4086/toc.2010.v006a005.

[100] B. E. Neou. *Permutation pattern matching*. PhD thesis, Université Paris-Est; Università di Verona, 2017.

[101] Both Emerite Neou, Romeo Rizzi, and Stéphane Vialette. Pattern matching for separable permutations. In *String processing and information retrieval*. Volume 9954, pages 260–272. Springer, Cham, 2016. DOI: 10.1007/978-3-319-46049-9_25.

[102] Sang-Il Oum. Approximating rank-width and clique-width quickly. *ACM Transactions on Algorithms*, 5(1):Art. 10, 20, 2009. DOI: 10.1007/11604686_5.

[103] Sang-il Oum and Paul Seymour. Approximating clique-width and branch-width. *J. Combin. Theory Ser. B*, 96(4):514–528, 2006. DOI: 10.1016/j.jctb.2005.10.006.

[104] Neil Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986. DOI: 10.1016/0196-6774(86)90023-4.

[105] Vladislav Rutenburg. Complexity of generalized graph coloring. In *Mathematical Foundations of Computer Science 1986, Bratislava, Czechoslovakia, August 25-29, 1996, Proceedings*, volume 233, pages 573–581, 1986. DOI: 10.1007/BFb0016284.

[106] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 216–226, 1978. DOI: 10.1145/800133.804350.

[107] Neil Sloane and OEIS Foundation Inc. The online encylopedia of integer sequences, 2022.

[108] Zvezdelina E. Stankova. Forbidden subsequences. *Discrete Math.*, 132(1-3):291–316, 1994. DOI: 10.1016/0012-365X(94)90242-9.

[109] Boris Avraamovich Trakhtenbrot. Finite automata and the logic of one-place predicates. *Sibirsk. Mat. Ž.*, 3:103–131, 1962. DOI: 10.1090/trans2/059/02.

[110] Henning Úlfarsson. Describing West-3-stack-sortable permutations with permutation patterns. *Sém. Lothar. Combin.*, 67:Art. B67d, 20, 2012.

[111] Vincent Vatter. An Erdős-Hajnal analogue for permutation classes. *Discrete Math. Theor. Comput. Sci.*, 18(2):Paper No. 4, 5, 2016.

[112] Vincent Vatter. Finding regular insertion encodings for permutation classes. *J. Symbolic Comput.*, 47(3):259–265, 2012. DOI: 10.1016/j.jsc.2011.11.002.

[113] Vincent Vatter. Growth rates of permutation classes: from countable to uncountable. *Proc. Lond. Math. Soc. (3)*, 119(4):960–997, 2019. DOI: 10.1112/plms.12250.

[114] Vincent Vatter. Permutation classes. In Miklos Bona, editor, *Handbook of enumerative combinatorics*, pages 753–833. CRC Press, Boca Raton, FL, 2015.

[115] Vincent Vatter. Small permutation classes. *Proc. Lond. Math. Soc. (3)*, 103(5):879–921, 2011. DOI: 10.1112/plms/pdr017.

[116] Vincent Vatter and Steve Waton. On partial well-order for monotone grid classes of permutations. *Order*, 28(2):193–199, 2011. DOI: 10.1007/s11083-010-9165-1.

[117] Stehpen D. Waton. *On Permutation Classes Defined by Token Passing Networks, Gridding Matrices and Pictures: Three Flavours of Involvement*. PhD thesis, Univ. of St Andrews, 2007.

[118] Julian West. *Permutations with forbidden subsequences and stack-sortable permutations*. PhD thesis, Massachusetts Institute of Technology, 1990, (no paging).

[119] Kai Ting Keshia Yap, David Wehlau, and Imed Zaguia. Permutations avoiding certain partially-ordered patterns. *Electron. J. Combin.*, 28(3):Paper No. 3.18, 41, 2021. DOI: 10.37236/1020.

List of publications

[P1] Michael Albert, Vít Jelínek, and Michal Opler. Two examples of Wilf-collapse. *Discrete Math. Theor. Comput. Sci.*, 22(2):Paper No. 9, 13, 2021. DOI: 10.46298/dmtcs.5986.

[P2] Václav Blažej, Pavel Dvořák, and Michal Opler. Bears with hats and independence polynomials. In *Graph-theoretic concepts in computer science*. Volume 12911, pages 283–295. Springer, Cham, 2021. DOI: 10.1007/978-3-030-86838-3_22.

[P3] Václav Blažej, Michal Opler, Matas Šileikis, and Pavel Valtr. Non-homotopic loops with a bounded number of pairwise intersections. In *Graph Drawing and Network Visualization - 29th International Symposium, GD 2021, Tübingen, Germany, September 14-17, 2021, Revised Selected Papers*, volume 12868, pages 210–222, 2021. DOI: 10.1007/978-3-030-92931-2_15.

[P4] Václav Blažej, Michal Opler, Matas Šileikis, and Pavel Valtr. On the intersections of non-homotopic loops. In *Algorithms and discrete applied mathematics*. Volume 12601, pages 196–205. Springer, Cham, 2021. DOI: 10.1007/978-3-030-67899-9_15.

[P5] Vít Jelínek, Michal Opler, and Jakub Pekárek. Griddings of Permutations and Hardness of Pattern Matching. In *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*, volume 202, 65:1–65:22, 2021. DOI: 10.4230/LIPIcs.MFCS.2021.65.

[P6] Vít Jelínek, Michal Opler, and Jakub Pekárek. Long paths make pattern-counting hard, and deep trees make it harder. In *16th International Symposium on Parameterized and Exact Computation, IPEC 2021, September 8-10, 2021, Lisbon, Portugal*, volume 214, 22:1–22:17, 2021. DOI: 10.4230/LIPIcs.IPEC.2021.22.

[P7] Vít Jelínek, Michal Opler, and Jakub Pekárek. A complexity dichotomy for permutation pattern matching on grid classes. In *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, volume 170, 52:1–52:18, 2020. DOI: 10.4230/LIPIcs.MFCS.2020.52.

[P8] Michael Albert, Vít Jelínek, and Michal Opler. Wilf collapse in permutation classes. arXiv:1909.13348, 2019.

[P9] Vít Jelínek, Michal Opler, and Pavel Valtr. Generalized coloring of permutations. In *26th European Symposium on Algorithms*. Volume 112, Art. No. 50, 14. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2018. DOI: 10.4230/LIPIcs.ESA.2018.50.

[P10] Vít Jelínek and Michal Opler. Splittability and 1-amalgamability of permutation classes. *Discrete Math. Theor. Comput. Sci.*, 19(2):Paper No. 4, 14, 2017. DOI: 10.23638/DMTCS-19-2-4.

[P11] Matěj Konečný, Stanislav Kučera, Michal Opler, Jakub Sosnovec, Štěpán Šimsa, and Martin Töpfer. Squarability of rectangle arrangements. In *Proceedings of the 28th Canadian Conference on Computational Geometry, CCCG 2016, August 3-5, 2016, Simon Fraser University, Vancouver, British Columbia, Canada*, pages 101–106, 2016.

[P12] Michal Opler. Major index distribution over permutation classes. *Adv. in Appl. Math.*, 80:131–150, 2016. DOI: 10.1016/j.aam.2016.06.011.