Classification and Geometry of Rational Polygons and Surfaces with Torus Action

Dissertation
der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Justus Springer
aus Tübingen

Tübingen
2025

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	04.12.2025
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Jürgen Hausen
2. Berichterstatter:	apl. Prof. Dr. Ivo Radloff

What's reality? I don't know. When my bird was looking at my computer monitor, I thought, "That bird has no idea what he's looking at." And yet, does the bird panic? No – he can't really panic; he just does the best he can. Is he able to live in a world where he's so ignorant? Well, he doesn't really have a choice. Usually, the bird's okay even though he doesn't understand the world. You're that bird looking at the monitor, and you're thinking to yourself, "I can figure this out." Maybe you have some bird ideas. Maybe that's the best you can do.

Terry A. Davis (abridged)

Contents

Introduction

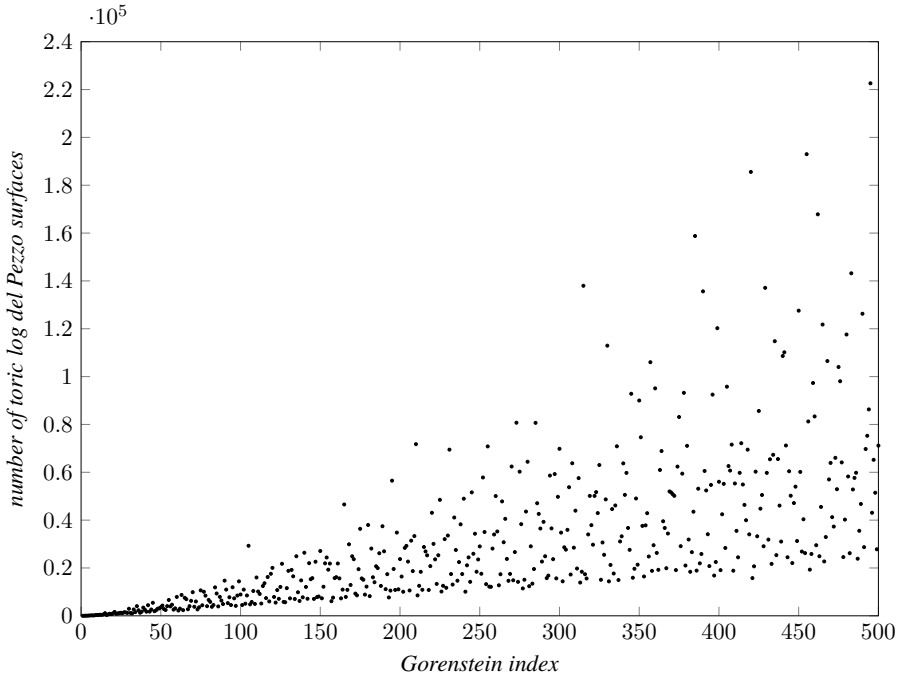
This thesis contributes to the study and classification of log del Pezzo surfaces with torus action on the one hand and to the theory of convex polygons on the other. A *del Pezzo surface* is a two-dimensional Fano variety, i.e. a normal projective surface X with ample anticanonical divisor $-\mathcal{K}_X$. The smooth del Pezzo surfaces are classically known — they are either $\mathbb{P}^1 \times \mathbb{P}^1$ or the blow-up of $\mathbb{P}^2$ in up to eight points in general position. The singular ones form a much larger class and are an area of active research. A common restriction is to consider *log terminal* singularities, which means that all discrepancies of a resolution of singularities are greater than -1. A *log del Pezzo* surface is a del Pezzo surface with at most log terminal singularities.

Focusing on del Pezzo surfaces *with torus action* allows the use of explicit methods. In the toric case (with an effective action of $\mathbb{C}^* \times \mathbb{C}^*$), these surfaces can be described combinatorially by *LDP polygons*, which are convex lattice polygons with primitive vertices having the origin in their interior. Toric surfaces are log terminal and the geometry of a toric log del Pezzo surface is closely connected to combinatorial properties of the LDP polygon. The other case is that of $\mathbb{C}^*$*-surfaces*, where the action has complexity one. In this case, a combinatorial description via toric ambient varieties and Cox Rings is available [28, 32, 33], see also [2, Section 5.4].

A priori, the class of singular log del Pezzo surfaces is unbounded. In order to obtain finite classifications, one imposes suitable conditions on the singularities. A common approach is to require that the singularities are at most ε*-log canonical*, for some real number $\varepsilon > 0$. This means that all discrepancies of a resolution of singularities are greater or equal to $\varepsilon - 1$. It is known that for fixed ε, there are only finitely many families of ε-log canonical del Pezzo surfaces [1]. Alternatively, one can impose bounds on the *Gorenstein index*, which is the smallest positive integer ι such that $\iota\mathcal{K}_X$ is Cartier. A log del Pezzo surface of Gorenstein index ι is also $\frac{1}{\iota}$-log canonical, hence there are only finitely many families of log del Pezzo surfaces with fixed Gorenstein index.

For a toric surface associated to an LDP polygon P, its Gorenstein index equals the smallest positive integer ι such that ιP^* is integral, where P^* is the dual polygon. Kasprzyk, Kreuzer and Nill [36] used this description to obtain a classification algorithm of toric log del Pezzo surfaces, which they ran up to Gorenstein index 17. More recently, Bäuerle [11] classified *Fano simplices*, which correspond to toric Fano varieties with Picard number one. In particular, he classified the toric log del Pezzo surfaces of Picard number one with Gorenstein index at most 1000. We consider the case of Picard number two and contribute the following result, see Classification 2.5.1.

Theorem 1. *There are exactly* 15 669 466 *isomorphism classes of toric log del Pezzo surfaces of Picard number two with Gorenstein index at most* 500*. The number of isomorphism classes for given Gorenstein index develops as follows:*

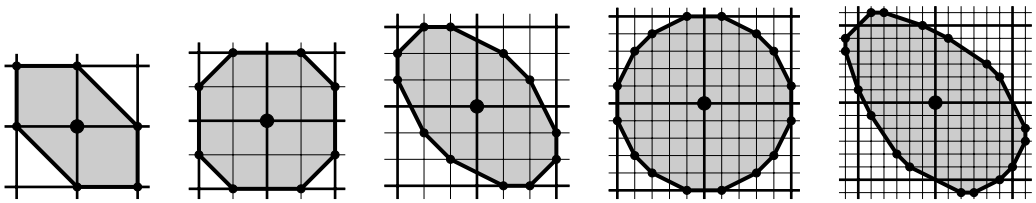


Similar to the Gorenstein index, the log canonicity of a toric surface can also be described by its defining LDP polygon P: The surface is $\frac{1}{k}$-log canonical if and only if $P^\circ \cap k\,\mathbb{Z}^2 = \{0\}$. Equivalently, the toric $\frac{1}{k}$-log canonical del Pezzo surfaces are in one-to-one correspondence with the rational polygons P such that $P^\circ \cap \mathbb{Z}^2 = \{0\}$ and kP is an LDP polygon. In the joint work [27], a classification of $\frac{1}{3}$-log canonical del Pezzo surfaces with torus action was given. In the toric case, we are able to extend this result as follows, see Classification 1.5.6.

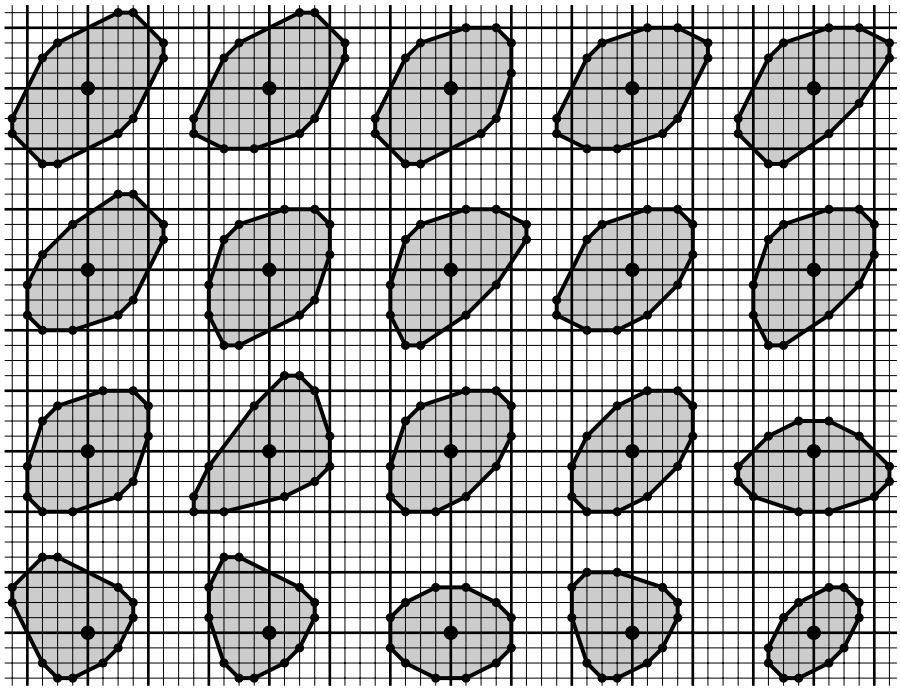
Theorem 2. *Below are the numbers $\#(k)$ of LDP polygons P such that $P^\circ \cap k\,\mathbb{Z}^2 = \{0\}$, up to unimodular equivalence. Moreover, $N(k)$ is the maximally attained number of vertices and $M(k)$ the number of maximizers. Equivalently, $\#(k)$ is the number isomorphism classes of toric $\frac{1}{k}$-log canonical del Pezzo surfaces and $N(k) - 2$ is the maximally attained Picard number.*

k	1	2	3	4	5	6
$\#(k)$	16	505	48 032	1 741 603	154 233 886	2 444 400 116
$N(k)$	6	8	12	12	16	18
$M(k)$	1	1	1	20	1	1

For $k \in \{1, 2, 3, 5, 6\}$, the unique vertex maximizers are the following:



Somewhat surprisingly, for $k = 4$, there are 20 vertex maximizers:



In fact, this result is part of a series of classifications of rational polygons by their denominator and number of interior lattice points. We say a polygon $P \subseteq \mathbb{R}^2$ is *k-rational*, if kP is a lattice polygon, i.e. has integral vertices. It follows from the volume bounds of Lagarias and Ziegler [41] that up to lattice automorphism, there are only finitely many k-rational polygons with fixed number i of interior lattice points, provided $i \geq 1$. For $i = 0$, there are still only finitely *maximal* k-rational polygons, i.e. which are not strictly included in another k-rational polygon with equal number of interior lattice points, see [5]. For lattice polygons ($k = 1$), Castryck [19] gave a recursive classification algorithm, which he ran up to $i = 30$. For higher values of k, we contribute the following classifications, see Theorem 1.1.1.

Theorem 3. *The numbers of k-rational polygons with i interior lattice points are given by the following table. Each cell contains the number of maximal polygons on top, the number of distinct Ehrhart quasipolynomials in the middle and the total number of polygons below. For $i = 0$, only the polygons which cannot be realized in $\mathbb{R} \times [0, 1]$ are counted.*

$k \downarrow i \rightarrow$	0	1	2	3	4	5	6	...
1	1 1 1	3 7 16	4 8 45	6 10 120	9 12 211	11 14 403	13 16 714	...
2	4 34 79	10 270 5 145	25 586 48 639	33 1 060 249 540	63 1 701 893 402	106 2 525 2 798 638	178 3 577 7 299 589	...
3	14 803 6 723	39 8 124 924 042	146 25 892 17 656 886	303 62 347 156 777 687	687 124 021 909 056 858	1 452 218 169 4 211 988 753		
4	39 18 916 399 294	145 320 256 101 267 212	670 1 593 475 3 452 922 966					
5	134 253 631 18 935 385	698 5 034 566 8 544 548 186						
6	299 8 930 335 820 697 679							

Having access to this large dataset has lead to new theoretical insights about the Ehrhart theory of rational polygons. In particular, we are able to provide generalizations of Scott's inequality as well as Pick's formula to the rational setting. Recall that for lattice polygons, Scott's inequality [51] says that the numbers i and b of interior and boundary lattice points satisfy $b \leq 9$ for $i = 1$ and $b \leq 2i + 6$ for $i \geq 2$ and this bound is sharp. In the following, the *denominator* of a rational polygon P is the smallest positive integer k such that it is P is k-rational. See also Theorem 1.1.2.

Theorem 4. *Let P be a rational polygon of denominator $k \geq 2$ with $i \geq 1$ interior and b boundary lattice points. Then $b \leq (k+1)(i+1)+3$ and equality holds if and only if P is affine unimodular equivalent to the triangle* $\operatorname{conv}\big((0, \frac{1}{k}), (0, -1), ((k+1)(i+1), -1)\big)$.

Next, recall that Pick's formula [47] states that the area of a lattice polygon with i interior and b boundary lattice points equals $i + \frac{b}{2} - 1$. For rational polygons of denominator at least two, fixing i and b no longer uniquely determines the area. Instead, we contribute the following sharp lower and upper bounds, see Theorems 1.1.3 and 1.1.4.

Theorem 5. *Let P be a rational polygon of denominator $k \geq 2$ with $i \geq 1$ interior and b boundary lattice points. Set $b_{\max} = (k+1)(i+1)+3$. If $Q := \operatorname{conv}(P \cap \mathbb{Z}^2)$ is two-dimensional and $b \geq 2$, we have*

$$\operatorname{area}(P) \;\geq\; \frac{i(k+1)+1}{2k} + \frac{b}{2} - 1.$$

Moreover, if $\dim(Q) < 2$, we have $b \leq 2$ and

$$\operatorname{area}(P) \geq \begin{cases} \frac{i}{k} - \frac{1}{k} + \frac{3}{2k^2}, & b = 0 \\ \frac{i}{k} + \frac{1}{2k^2}, & b = 1 \\ \frac{i}{k} + \frac{1}{k}, & b = 2. \end{cases}$$

Theorem 6. *Let P be a rational polygon of denominator $k \geq 4$ with $i \geq 1$ interior and b boundary lattice points. Set $b_{\max} := (k+1)(i+1)+3$ and $\tilde{b} := b_{\max} - b$. Then*

$$2k^2 \operatorname{area}(P) \;\leq\; k(k+1)^2(i+1) - \begin{cases} \tilde{b}, & b \in \{b_{\max}, b_{\max}-1\} \\ (2 + k(\tilde{b}-2)), & 1 \leq b \leq b_{\max}-2 \\ (3 + k(\tilde{b}-2)), & b = 0. \end{cases}$$

We return to the classification of log del Pezzo surfaces with torus action. To gain yet another perspective, we will study extensively the *Picard index*, i.e. the index of the Picard group inside the divisor class group. Note that the Gorenstein index divides the Picard index, hence there are only finitely many log del Pezzo surfaces with fixed Picard index. We will prove the following formula, which relates the Picard index to the torsion order of the class group, as well as the local class groups.

Theorem 7. *Let X be a normal rational projective surface with torus action. Then its Picard index is given by*

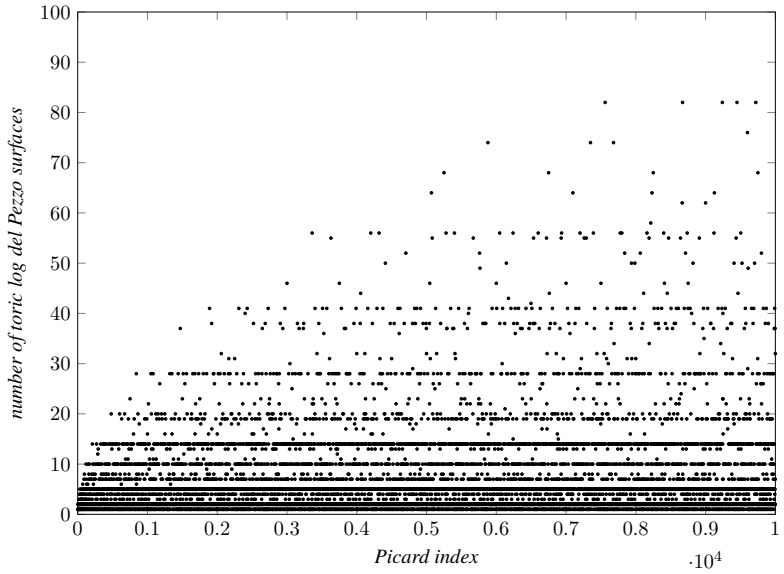
$$\iota_{\mathrm{Pic}}(X) \;=\; \frac{1}{|\operatorname{Cl}(X)|^{\mathrm{tors}}} \prod_{x \in X} |\operatorname{Cl}(X,x)|.$$

Note that this formula does not require the del Pezzo property, but it requires the surface to be rational, which in particular implies $\mathbb{Q}$-factoriality. Hence $\operatorname{Cl}(X)$ is finitely generated, see for instance [2, Thm. 5.4.1.5]. Moreover, by normality of X, there are only finitely many singular points and these are the only possible contributors of non-trivial local class groups. Thus, all terms in our formula are indeed finite.

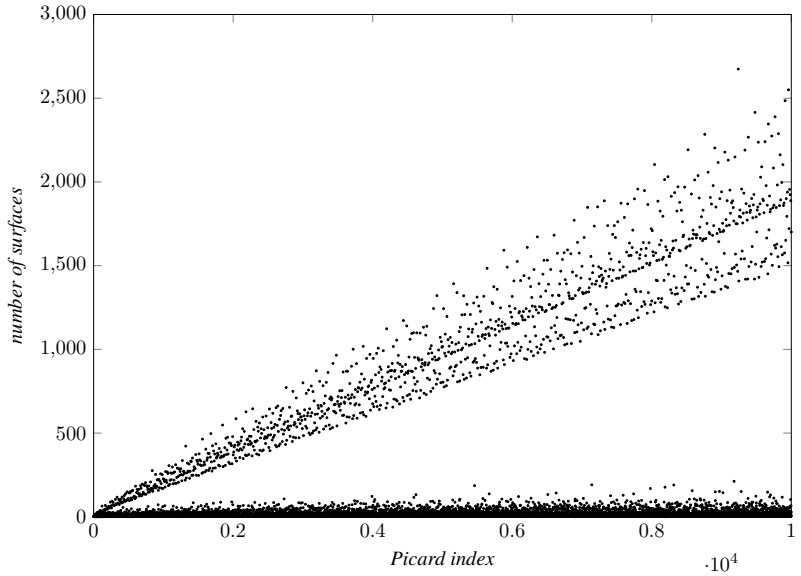
Beyond the $\mathbb{C}^*$-surfaces, the formula trivially holds for all smooth projective surfaces with a finitely generated and torsion free divisor class group. As soon as we allow torsion, the right hand side is longer integral in the smooth case and thus the formula fails. Concrete examples are the Enriques surfaces, having divisor class group $\mathbb{Z}^{10} \times \mathbb{Z}\,/2\,\mathbb{Z}$. A singular counterexample without $\mathbb{C}^*$-action is provided by the D_8-singular log del Pezzo surface of Picard number one: it is $\mathbb{Q}$-factorial with divisor class group $\mathbb{Z} \times \mathbb{Z}\,/2\,\mathbb{Z}$ and doesn't satisfy the formula, see Example 3.3.52.

The idea of filtering by the Picard index has appeared in [29], where not-necessarily log terminal Fano varieties with divisor class group $\mathbb{Z}$ and a torus action of complexity one have been considered. Here, we use Theorem 7 to derive in Picard number one suitable bounds on toric and non-toric log del Pezzo $\mathbb{C}^*$-surfaces and then present classification algorithms. Explicit results are obtained up to Picard index $1\,000\,000$ in the toric case and up to Picard index $10\,000$ in the non-toric case, see Classifications 2.4.2 and 3.4.7.

Theorem 8. *Up to isomorphism, there are $15\,086\,426$ toric log del Pezzo surfaces of Picard number one with Picard index at most $1\,000\,000$. Up to Picard index $10\,000$, the number of isomorphism classes for given Picard index develops as follows:*



Theorem 9. *There are* 1 347 433 *families of non-toric, log del Pezzo* $\mathbb{C}^*$*-surfaces of Picard number one and Picard index at most* 10 000*. The number of families for given Picard index develops as follows:*



A natural question is which positive integers occur as Picard indices of log del Pezzo surfaces. For the toric ones, every positive integer is attained, see Proposition 2.4.3. For non-toric ones with a $\mathbb{C}^*$-action, we will prove that Mersenne prime numbers greater than seven are not attained, see Theorem 3.4.12.

Theorem 10. *There are no non-toric log del Pezzo* $\mathbb{C}^*$*-surfaces of Picard number one whose Picard index is a Mersenne prime number greater than seven.*

This thesis is organized as follows. Chapter 1 studies rational polygons from the perspective of convex geometry. We introduce the classification methods, prove the generalized Scott inequality and our area bounds. We also discuss applications to the Ehrhart theory of half-integral polygons. Chapter 2 concerns toric surfaces. After a brief background on toric geometry (and surfaces in particular), we present the classifications in Picard number one by Picard index and in Picard number two by Gorenstein index. We then prove Theorem 7 in the toric. Chapter 3 is about $\mathbb{C}^*$-surfaces. We introduce the combinatorial background in terms of Cox rings and a normal form for the defining data. We then prove Theorem 7 in full generality, discuss the classification by Picard index and the relation to Mersenne primes. Finally, two appendices document the software developed alongside the thesis. Both packages are written in Julia [13] and are available as open-source software on the author's GitHub page [52, 55]. Appendix A documents `RationalPolygons.jl`, used for computations involving rational polygons and toric del Pezzo surfaces in Chapters 1 and 2. Appendix B documents `CStarSurfaces.jl`, used for computations involving $\mathbb{C}^*$-surfaces in Chapter 3.

CHAPTER 1

Rational polygons

We provide algorithms to classify rational polygons with small denominator and few interior lattice points. Furthermore, we prove generalizations of Scott's inequality and Pick's formula to rational polygons.

This chapter was written jointly by Martin Bohnert and me, with equal contributions. It has been published in our joint works [16, 17].

1.1 Results

A polytope is called *k-rational*, if its k-fold dilation is a lattice polytope, i.e. has integral vertices. Lagarias and Ziegler [41] showed that for a fixed number i of interior lattice points, there exist up to lattice automorphism only finitely many k-rational polytopes in each dimension, provided $i \geq 1$. Even though this does not hold for $i = 0$, Averkov, Wagner and Weismantel [5] showed that in this case there are still only finitely *maximal* k-rational polytopes, i.e. which are not strictly included in another k-rational polytope with equal number of interior lattice points.

Our first result is a classification of rational two-dimensional polytopes by number of interior lattice points. We used Julia [13] to carry out the classification and made the implementation available as part of a software package on rational polygons [55], see also Appendix A for a detailed documentation. Moreover, large parts of our data are available for download [57–62].

Theorem 1.1.1. *The numbers of k-rational polygons with i interior lattice points are given by the following table. Each cell contains the number of maximal polygons on top, the number of distinct Ehrhart quasipolynomials in the middle and the total number of polygons below. For $i = 0$, only the polygons which cannot be realized in $\mathbb{R} \times [0, 1]$ are counted.*

$k \downarrow i \rightarrow$	0	1	2	3	4	5	6	...
1	1 1 1	3 7 16	4 8 45	6 10 120	9 12 211	11 14 403	13 16 714	...
2	4 34 79	10 270 5 145	25 586 48 639	33 1 060 249 540	63 1 701 893 402	106 2 525 2 798 638	178 3 577 7 299 589	...
3	14 803 6 723	39 8 124 924 042	146 25 892 17 656 886	303 62 347 156 777 687	687 124 021 909 056 858	1 452 218 169 4 211 988 753		
4	39 18 916 399 294	145 320 256 101 267 212	670 1 593 475 3 452 922 966					
5	134 253 631 18 935 385	698 5 034 566 8 544 548 186						
6	299 8 930 335 820 697 679							

The first row in the table above is the case of *lattice polygons*, which has already been studied by several authors. Starting from the top left, the unique maximal lattice polygon without interior lattice points is the twofold standard lattice triangle, see e.g. [38, sec. 4.1], [37, §5] and [50, Theorem 2]. The classification of the 16 lattice polygons with exactly one interior lattice point has been done e.g. in [9, 48] and [38, sec. 4.2]. Lattice polygons with collinear interior lattice points, which includes the case $i = 2$, were classified by Koelman [38, sec. 4.3] and a general recursive algorithm was given by Castryck [19], which he ran up to $i = 30$. There are also algorithms classifying lattice polygons by their number of lattice points [38, sec. 4.4] and by their area [6]. Pick's formula [47] relates the area to the numbers of lattice and interior lattice points, hence one can also obtain the classification by interior lattice points from those algorithms.

To the author's knowledge, besides the first row in our table, only the four maximal 2-rational polygons without interior lattice points have directly appeared in the literature before [4]. The classification of almost k-hollow LDP polygons, which was done in [27] for $k \in \{2, 3\}$ is also worth mentioning, as these can be viewed as a subset of all k-rational polygons with exactly one interior lattice point. Hence they appear in our classification among the 5 145 resp. 924 042 polygons in the second column of the table above.

Let us also mention that in three dimensions, the lattice polytopes with one and two interior lattice points have been classified [7, 35] and the case of no interior lattice points has been done in [4, 5]. Moreover, there is the classification of four-dimensional reflexive polytopes [39], which is an important subset of those having exactly one interior lattice point.

Our next series of results provide generalizations of Scott's inequality and Pick's formula to rational polygons. Recall that Scott's inequality [51] says that for lattice polygons, the numbers i and b of interior and boundary lattice points of P satisfy $b \leq 9$ for $i = 1$ and $b \leq 2i + 6$ for $i \geq 2$ and this bound is sharp. In particular, $(i, b) = (1, 9)$ holds if and only if P is affine unimodular equivalent to the threefold standard lattice triangle. In the following, the *denominator* of a rational polygon P is the smallest integer k such that P is k-rational.

Theorem 1.1.2. *Let P be a rational polygon of denominator $k \geq 2$ with $i \geq 1$ interior and b boundary lattice points. Then $b \leq (k+1)(i+1)+3$ and equality holds if and only if P is affine unimodular equivalent to the triangle* $\operatorname{conv}\big((0, \frac{1}{k}), (0, -1), ((k+1)(i+1), -1)\big)$.

Next, we provide area bounds. Recall that Pick's formula [47] states that the area of a lattice polygon with i interior and b boundary lattice points equals $i + \frac{b}{2} - 1$. For rational polygons of denominator at least two, fixing i and b no longer uniquely determines the area. Instead, we provide sharp lower and upper bounds, where we also completely describe minimizers and maximizers.

Theorem 1.1.3 (See also Figures 1.1 and 1.2)**.** *Let P be a rational polygon of denominator $k \geq 2$ with $i \geq 1$ interior and b boundary lattice points. Set $b_{\max} = (k+1)(i+1)+3$. If $Q := \operatorname{conv}(P \cap \mathbb{Z}^2)$ is two-dimensional and $b \geq 2$, we have*

$$\operatorname{area}(P) \;\geq\; \frac{i(k+1)+1}{2k} + \frac{b}{2} - 1.$$

Here, equality holds if and only if P is equivalent to one of the following polygons:

(0a)	$\operatorname{conv}((0,-1),(i(k+1)-k+1,-1),(-\frac{1}{k},\frac{1}{k}))$	*for* $b = b_{\max} - 2(k+1)$
(1a)	$\operatorname{conv}((0,-1),(b-2,-1),(i,0),(-\frac{1}{k},\frac{1}{k}))$	*for* $2 \le b \le b_{\max} - (k+1)$
(1b)	$\operatorname{conv}((0,-2),(2,0),(-\frac{1}{k},\frac{1}{k}))$,	*for* $(i,b) = (3,3)$
(2a)	$\operatorname{conv}((0,0),(0,-1),(b-3,-1),(i+1,0),(\frac{x}{k},\frac{1}{k}))$	*for* $3 \le b \le b_{\max}$
(2b)	$\operatorname{conv}((0,0),(0,-2),(2,0),(\frac{x'}{k},\frac{1}{k}))$	*for* $(i,b) = (1,5)$,

where $x = 0,\dots,\lfloor\frac{b_{\max}-b}{2}\rfloor$ *and* $x' = 0,\dots,k$. *Moreover, if* $\dim(Q) < 2$, *we have* $b \le 2$ *and*

$$\operatorname{area}(P) \ge \begin{cases} \frac{i}{k} - \frac{1}{k} + \frac{3}{2k^2}, & b = 0 \\ \frac{i}{k} + \frac{1}{2k^2}, & b = 1 \\ \frac{i}{k} + \frac{1}{k}, & b = 2. \end{cases}$$

Here, equality holds if and only if P is equivalent to one of the following polygons:

(0c)	$\operatorname{conv}((1,\frac{1}{k}),(1-\frac{1}{k},-\frac{1}{k}),(i+\frac{1}{k},0))$,	*for* $b = 0$,
(1c)	$\operatorname{conv}((1,\frac{1}{k}),(1-\frac{1}{k},-\frac{1}{k}),(i+1,0))$,	*for* $b = 1$,
(2c)	$\operatorname{conv}((0,0),(0,\frac{1}{k}),(\frac{x}{k},-\frac{1}{k}),(i+1,0))$,	*for* $b = 2$,

where $x = 0,\dots,k(i+1)$. *All polygons listed above are pairwise non-equivalent.*

Theorem 1.1.4 (See also Figure 1.3)**.** *Let P be a rational polygon of denominator $k \ge 4$ with $i \ge 1$ interior and b boundary lattice points. Set $b_{\max} := (k+1)(i+1)+3$ and $\tilde{b} := b_{\max} - b$. Then*

$$2k^2 \operatorname{area}(P) \;\le\; k(k+1)^2(i+1) - \begin{cases} \tilde{b}, & b \in \{b_{\max}, b_{\max}-1\} \\ (2 + k(\tilde{b}-2)), & 1 \le b \le b_{\max}-2 \\ (3 + k(\tilde{b}-2)), & b = 0. \end{cases}$$

Here, equality holds if and only if P is equivalent to one the following polygons:

(0a)	$\operatorname{conv}\big((0,\frac{1}{k}),(\frac{1}{k},-1),((k+1)(i+1)-\frac{1}{k},-1)\big)$	*for* $b = b_{\max} - 4$
(0b)	$\operatorname{conv}\big((0,\frac{1}{k}),(\frac{1}{k},-1),(1-\frac{1}{k},-1),(k(i+1)-\frac{1}{k},\frac{1}{k}-1)\big)$	*for* $b = 0$
(1a)	$\operatorname{conv}\big((0,\frac{1}{k}),(\frac{1}{k},-1),(b-\frac{1}{k},-1),(k(i+1),\frac{1}{k}-1)\big)$	*for* $1 \le b \le b_{\max} - 3$,
(2a)	$\operatorname{conv}\big((0,\frac{1}{k}),(0,\frac{1}{k}-1),(x+\frac{1}{k},-1),$ $(x+b-1-\frac{1}{k},-1),(k(i+1),\frac{1}{k}-1)\big)$	*for* $2 \le b \le b_{\max} - 2$,
(2b)	$\operatorname{conv}\big((0,\frac{1}{k}),(0,\frac{1}{k}-1),(\frac{1}{k},-1),((k+1)(i+1),-1)\big)$	*for* $b = b_{\max} - 1$,
(2c)	$\operatorname{conv}\big((0,\frac{1}{k}),(0,-1),((k+1)(i+1),-1)\big)$	*for* $b = b_{\max}$,

where $x = 0,\dots,\lfloor\frac{\tilde{b}}{2}\rfloor - 1$. *All polygons listed above are pairwise non-equivalent.*

We turn to half-integral polygons, i.e. the case of denominator two. We found that in this case, the area bound from Theorem 1.1.4 can be violated precisely if $i = 1$ or $b = 0$. We obtain the following sharp upper bound:

Theorem 1.1.5. *Let P be a rational polygon of denominator two having $i \geq 1$ interior and b boundary lattice points. If $i \geq 2$, we have the sharp bound*

$$\operatorname{area}(P) \leq \frac{3}{2}i + \frac{1}{4}b + \frac{1}{8} \cdot \begin{cases} 8, & 0 \leq b \leq 3i+4 \\ 7, & b = 3i+5 \\ 6, & b = 3i+6. \end{cases}$$

Moreover, if $i = 1$, we have the sharp bound

$$\operatorname{area}(P) \leq \frac{1}{4}b + \frac{1}{8} \cdot \begin{cases} 21, & 0 \leq b \leq 6 \\ 20, & b = 7 \\ 19, & b = 8 \\ 18, & b = 9. \end{cases}$$

The rest of this chapter is organized as follows. In Section 1.2, we provide general background on rational polygons and go over some topics that are necessary for our classifications, namely normal forms, computing subpolygons and the notion of maximality. Sections 1.3, 1.4 and 1.5 describe our classification, first for the case of polygons contained in $\mathbb{R} \times [-1, 1]$, then for polygons with no and exactly one interior lattice point, filling out the first two columns of the table in Theorem 1.1.1. The methods we use are mainly constraints on the lattice width data developed in [14]. In Section 1.6, we discuss a generic (yet inefficient) method of classifying rational polygons for any pair $(k, i) \in \mathbb{Z}^2_{\geq 1}$ that we used to fill in the third and fourth row of the table in Theorem 1.1.1. In Section 1.7, we turn to our generalizations of Scott's inequality and Pick's formula, proving Theorem 1.1.2, as well as the area bounds from Theorems 1.1.3 and 1.1.4. Section 1.8 is about half-integral polygons, i.e. $k = 2$. For the classification, we were able to generalize the idea of moving out the edges from Castryck's algorithm [19] using the classification of two-dimensional Fine interiors of three-dimensional lattice polytopes [15]. Finally, we also prove the area bound of Theorem 1.1.5 and apply this result to Ehrhart theory of half-integral polygons.

Figure 1.1: All area minimizers with collinear lattice points as described by Theorem 1.1.3 for $(k, i) = (3, 1)$. The dashed lines separate the polygons by number of boundary lattice points: In the Theorem's notation, the polygons (2c) are on the left, the polygon (1c) is in the middle and (0c) is on the right.

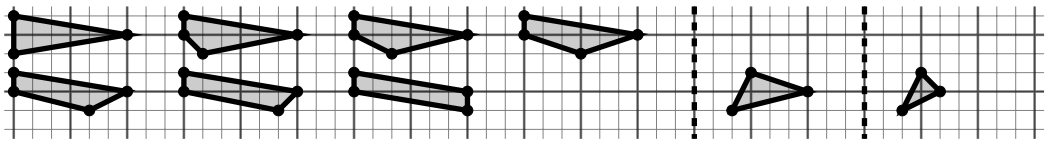


Figure 1.2: All area minimizers with two-dimensional integer hull as described by Theorem 1.1.3 for $(k, i) = (3, 1)$. The polygons are grouped by the number b of boundary lattice points, which is on the left. Each polygon is labeled according to the Theorem's notation.

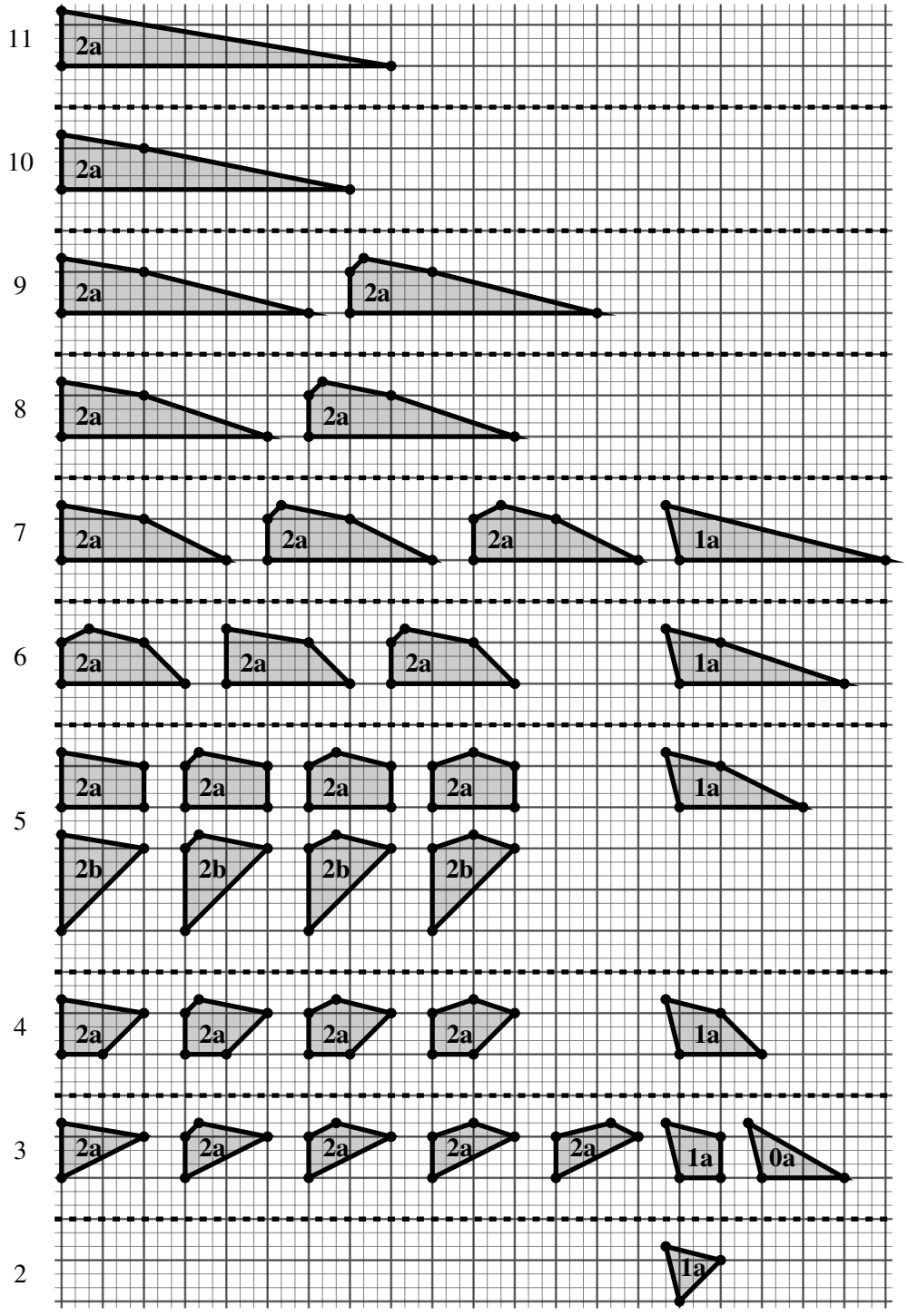
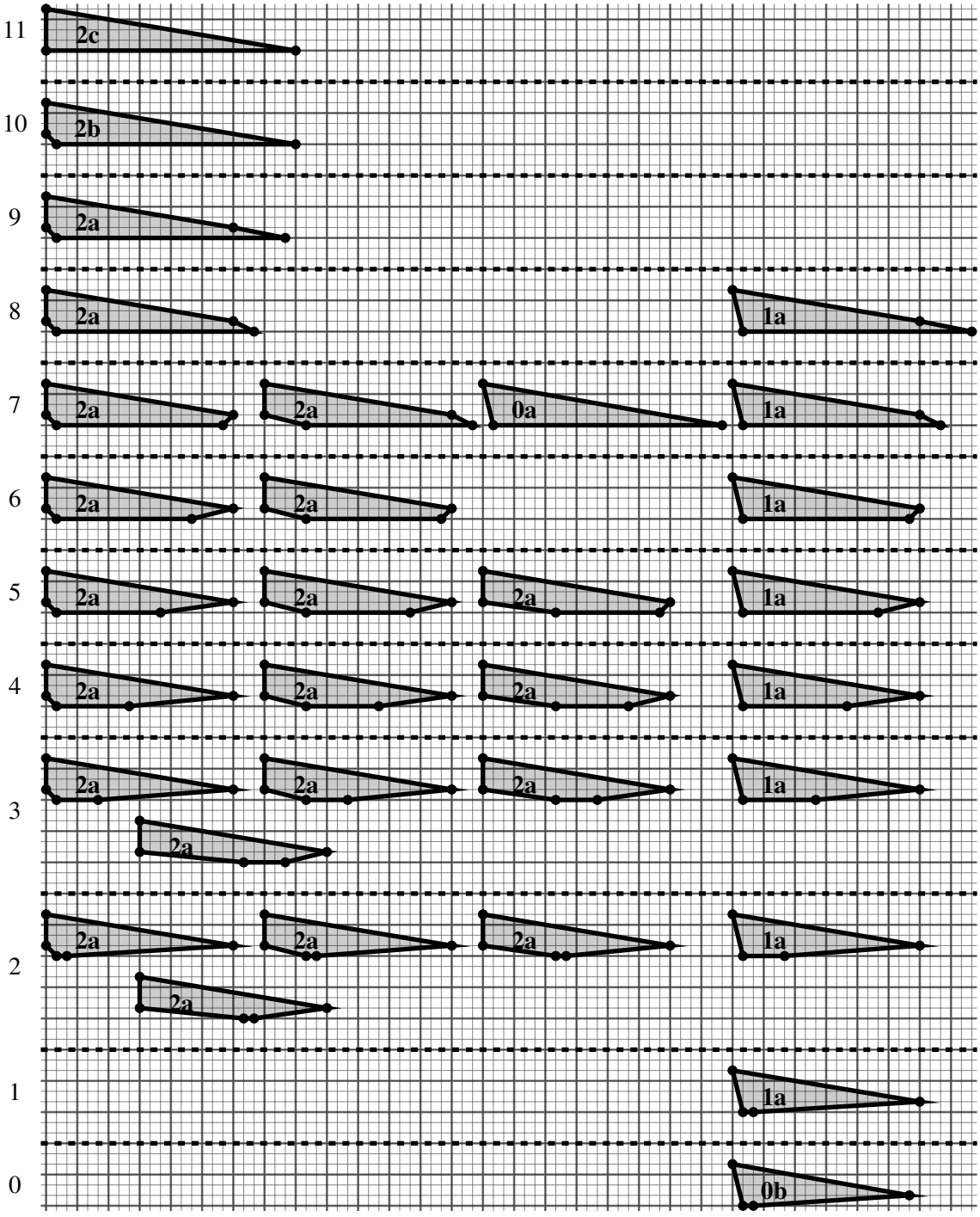


Figure 1.3: All area maximizers as described by Theorem 1.1.4 for $(k, i) = (3, 1)$. The polygons are grouped by the number b of boundary lattice points, which is on the left. Each polygon is labeled according to the Theorem's notation.



1.2 Prerequisites for classifying rational polygons

By a *polygon*, we mean a convex bounded polyhedral set $P \subseteq \mathbb{R}^2$. We call a polygon *rational* (*integral*), if its vertices are contained in $\mathbb{Q}^2$ ($\mathbb{Z}^2$). Integral polygons will also be called *lattice polygons*. For $k \in \mathbb{Z}_{\geq 1}$, we say P is *k-rational*, if kP is integral. The *denominator* $\operatorname{denom}(P)$ of a rational polygon is the smallest positive integer k such that P is k-rational.

We call two polygons P and P' *unimodular equivalent* ($P \sim^{\mathrm{u}} P'$) if $P' = UP$ for some $U \in \mathrm{GL}(2,\mathbb{Z})$. We call them *affine unimodular equivalent* ($P \sim^{\mathrm{a}} P'$), if $P' = UP + b$ for some $U \in \mathrm{GL}(2,\mathbb{Z})$ and $b \in \mathbb{Z}^2$. We say that P can be *realized* in some subset $A \subseteq \mathbb{R}^2$, if $P \sim^{\mathrm{a}} Q$ for some $Q \subseteq A$. We call P a *subpolygon* of P', if P can be realized in P'.

We write $\operatorname{area}(P)$ for the euclidean area of a polygon and $\operatorname{Area}(P)$ for the *normalized area*, which is defined to be twice the euclidean area. Moreover, we define the *k-normalized* area as

$$\operatorname{Area}_k(P) := k^2 \operatorname{Area}(P) = 2k^2 \operatorname{area}(P).$$

Note that if P is k-rational, $\operatorname{Area}_k(P)$ is an integer. By the *integer hull* of a polygon P, we mean the lattice polygon $\operatorname{conv}(P \cap \mathbb{Z}^2)$. Next, we define the number of interior, boundary and total number of lattice points as:

$$i(P) := |P^\circ \cap \mathbb{Z}^2|, \qquad b(P) := |\partial P \cap \mathbb{Z}^2|, \qquad l(P) := |P \cap \mathbb{Z}^2|.$$

All three numbers are invariants under affine unimodular equivalence and there exist several relations between them, e.g. Pick's formula states that $\operatorname{Area}(P) = 2i(P) + b(P) - 2$ holds for lattice polygons [47]. Another important invariant is the *Ehrhart quasipolynomial*, which counts the numbers of lattice points in integral dilations of a rational polygon. It is given by

$$\operatorname{ehr}_P(t) := l(tP) = \operatorname{area}(P)t^2 + c_1(t)t + c_2(t), \qquad t \in \mathbb{Z}_{\geq 1},$$

where $c_1, c_2 : \mathbb{Z} \to \mathbb{Q}$ are uniquely determined periodic functions whose period divides $\operatorname{denom}(P)$. For lattice polygons, $c_1(t) = \frac{b(P)}{2}$ and $c_2(t) = 1$ holds for all t. For the existence of the Ehrhart quasipolynomial and more insights on Ehrhart theory see for example [12].

1.2.1 Normal forms

Since we want to classify polygons up to lattice automorphism, we need an way to decide whether two polygons are equivalent by an affine unimodular transformation, which can be done by introducing a normal form. PALP [40], see also [24], is such a normal form that works for lattice polytopes of any dimension. However, we argue that the two-dimensional case deserves to be treated separately. This is because the vertices of a polygon are ordered naturally by walking along its boundary, up the choice of the starting point and the direction. In this subsection, we present our approach to the normal form of rational polygons, which is based on this natural ordering of the vertices. An implementation of our normal form is available in `RationalPolygons.jl`, see Section A.2.5.

We begin by describing how we encode k-rational polygons by integral matrices.

Definition 1.2.1. Fix $n \in \mathbb{Z}_{\geq 1}$. By a *vertex matrix*, we mean an integral matrix $V = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}$ such that $v_1, \dots, v_n \in \mathbb{Z}^2$ are the vertices of $\operatorname{conv}(v_1, \dots, v_n)$ and v_i is connected to v_{i+1} by an edge for all $i = 1, \dots, n-1$. For $k \in \mathbb{Z}_{\geq 1}$, we obtain a k-rational polygon

$$P_{k,V} \;:=\; \operatorname{conv}(v_1/k, \dots, v_n/k).$$

Remark 1.2.2. Using a planar convex hull algorithm, we can reduce any finite set of lattice points in the plane to a set of vertices ordered by angle. For example, a Graham scan [23] achieves this with time complexity $O(n \log n)$. See also A.1.2 for an implementation in Julia.

Definition 1.2.3. For a vertex matrix $V = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix} \in \mathbb{Z}^{2\times n}$, we obtain new vertex matrices by reversing or shifting the order of its columns:

$$V^{(-1)} \;:=\; \begin{bmatrix} v_n & \dots & v_1 \end{bmatrix}, \qquad V_i := \begin{bmatrix} v_i & \dots & v_n & v_1 & \dots & v_{i-1} \end{bmatrix}.$$

Moreover, we write $V + w := \begin{bmatrix} v_1 + w & \dots & v_n + w \end{bmatrix}$. Two vertex matrices $V, V' \in \mathbb{Z}^{2\times n}$ are called

(i) *strictly equivalent* ($V \sim V'$), if $V' = V_i^{(s)}$ for some $s \in \{\pm 1\}$ and $i = 1, \dots, n$,

(ii) *unimodular equivalent* ($V \sim^{\mathrm{u}} V'$), if $V' \sim UV$ for some $U \in \mathrm{GL}(2, \mathbb{Z})$,

(iii) *k-affine unimodular equivalent* ($V \sim_k^{\mathrm{a}} V'$), if $V' \sim UV + b$ for some $U \in \mathrm{GL}(2, \mathbb{Z})$ and $b \in k\,\mathbb{Z}^2$.

Remark 1.2.4. Let $V, V' \in \mathbb{Z}^{2\times n}$ be vertex matrices and $k \in \mathbb{Z}_{\geq 1}$. Then we have

(i) $P_{k,V} = P_{k,V'}$ if and only if $V \sim V'$,

(ii) $P_{k,V} \sim^{\mathrm{u}} P_{k,V'}$ if and only if $V \sim^{\mathrm{u}} V'$,

(iii) $P_{k,V} \sim^{\mathrm{a}} P_{k,V'}$ if and only if $V \sim_k^{\mathrm{a}} V'$.

Definition 1.2.5. Let $V = [v_1 \ \dots \ v_n]$ be a vertex matrix.

(i) The *unimodular normal form* of V is

$$\operatorname{unf}(V) \;:=\; \max\{\operatorname{hnf}(V_i^{(s)}) \;\mid\; s \in \{\pm 1\},\ i = 1, \dots, n\},$$

where the maximum is taken by lexicographical ordering and we write $\operatorname{hnf}(M)$ for the Hermite normal form of an integral matrix M.

(ii) The *1-affine normal form* of V is

$$\operatorname{anf}_1(V) \;:=\; \max\{\operatorname{hnf}(V_i^{(s)} - v_i) \;\mid\; s \in \{\pm 1\},\ i = 1, \dots, n\}.$$

(iii) Let $k \in \mathbb{Z}_{>1}$ and consider the lexicographically smallest vector $(x, y) \in \{0, \dots, k-1\}^2$ such that $\operatorname{anf}_1(V) + (x, y) \sim_k^{\mathrm{a}} V$. Then the *$k$-affine normal form* of V is

$$\operatorname{anf}_k(V) \;:=\; \operatorname{anf}_1(V) + (x, y).$$

Proposition 1.2.6. *Let $k \in \mathbb{Z}_{\geq 1}$ and consider vertex matrices $V, V' \in \mathbb{Z}^{2\times n}$. Then*

(i) $V \sim^{\mathrm{u}} V'$ *if and only if* $\mathrm{unf}(V) = \mathrm{unf}(V')$,

(ii) $V \sim_1^{\mathrm{a}} V'$ *if and only if* $\mathrm{anf}_1(V) = \mathrm{anf}_1(V')$,

(iii) $V \sim_k^{\mathrm{a}} V'$ *if and only if* $\mathrm{anf}_k(V) = \mathrm{anf}_k(V')$.

Proof. Note that we have $V \sim^{\mathrm{u}} \mathrm{unf}(V)$ as well as $\mathrm{unf}(UV) = \mathrm{unf}(V)$ and $\mathrm{unf}(V_i^{(s)}) = \mathrm{unf}(V)$ for all $U \in \mathrm{GL}(2,\mathbb{Z})$ and $i \in \{1,\dots,n\}, s \in \{\pm 1\}$. Thus (i) follows and (ii) is handled analogously. For (iii), we again have $V \sim_k^{\mathrm{a}} \mathrm{anf}_k(V)$, hence $\mathrm{anf}_k(V) = \mathrm{anf}_k(V')$ implies $V \sim_k^{\mathrm{a}} V'$. For the converse, note that $V \sim_k^{\mathrm{a}} V'$ implies $V \sim_1^{\mathrm{a}} V'$, hence by (ii), we have $\mathrm{anf}_1(V) = \mathrm{anf}_1(V')$. But this means $\mathrm{anf}_1(V) + (x,y) \sim_k^{\mathrm{a}} V$ if and only if $\mathrm{anf}_1(V') + (x,y) \sim_k^{\mathrm{a}} V'$ for all $(x,y) \in \mathbb{Z}^2$, hence $\mathrm{anf}_k(V) = \mathrm{anf}_k(V')$. □

Remark 1.2.7. Using Definition 1.2.5, computing the unimodular or affine normal form requires computing $2n$ Hermite normal forms, where n is the number of vertices. This can be improved by first distinguishing a subset of vertices that maximize a suitable invariant. For example, we can call a vertex v_i *special*, if it maximizes the area $\det(v_{i+1} - v_i, v_i - v_{i-1})$. Then we obtain an alternative unimodular normal form by

$$\mathrm{unf}'(V) \;:=\; \max\{\mathrm{hnf}(V_i^{(s)}) \;\mid\; s \in \{\pm 1\},\; v_i \text{ special}\}.$$

For polygons with few symmetries, there tend to be few special vertices, hence for those polygons $\mathrm{unf}'(V)$ is faster to compute than $\mathrm{unf}(V)$. More sophisticated invariants lead to fewer special vertices on average, however then the computation of the invariant itself becomes more expensive.

Remark 1.2.8. When computing the unimodular or affine normal form of a rational polygon, we get its respective automorphism group for free: Consider the set I of index pairs (i,s) with $i = 1,\dots,n$ and $s \in \{\pm 1\}$ such that $\mathrm{hnf}(V_i^{(s)}) = \mathrm{unf}(V)$. Then the number of rotational symmetries of the polygon is equal to the cardinality of I. Moreover, there exists a reflectional symmetry if and only if I contains two index pairs with opposite second coordinate. In this case, the unimodular automorphism group is the dihedral group $D_{|I|/2}$. Otherwise, it is the cyclic group of order $|I|$. See also A.2.73 for an implementation.

1.2.2 Subpolygons

We address the problem of finding all subpolygons of a given rational polygon up to equivalence. We use the same approach of *successive shaving* that the authors of [18] used to classify lattice polygons contained in a square of given size. In our implementation (see Section A.3), we noticed a significant speed-up by using Hilbert bases in the shaving process, hence we describe this in detail. As an application, we reproduce and extend the classification of [18], see Classification 1.2.13.

Fix $k \in \mathbb{Z}_{\geq 1}$ and let P be a k-rational polygon. Recall that a *subpolygon* of P is a k-rational polygon Q such that $Q \sim^{\mathrm{a}} Q'$ for some $Q' \subseteq P$. Up to affine unimodular equivalence, there

exist only finitely many subpolygons to any given k-rational polygon. For a vertex v, we obtain a subpolygon by taking the convex hull of all k-rational points of P, except v:

$$P\backslash v \;:=\; \operatorname{conv}\left(\left(P \cap \frac{1}{k}\,\mathbb{Z}^2\right)\backslash\{v\}\right).$$

Proposition 1.2.9. *Fix $k \in \mathbb{Z}_{\geq 1}$ and let P and Q be k-rational polygons with Q a subpolygon of P. Then there exists a chain of k-rational polygons*

$$P \;=\; P_0 \;\supsetneq\; P_1 \;\supsetneq\; \ldots \;\supsetneq\; P_n \;\sim^{\mathrm{a}}\; Q$$

such that for all $i = 0, \ldots, n-1$, we have $P_{i+1} = P_i\backslash v_i$ for some vertex v_i of P_i.

Proof. We may assume that $Q \subsetneq P$ holds. Note that the number of k-rational points of P must be strictly greater than the number of k-rational points of Q. Let's consider the case that they differ by one, i.e. there is a unique k-rational point $v \in \frac{1}{k}\,\mathbb{Z}$ with $v \in P\backslash Q$. By convexity, v must be a vertex, hence we have $Q = P\backslash v$. The general case now follows by induction. □

Algorithm 1 Computation of subpolygons

Input: A set of k-rational polygons M.

Output: A set of k-rational polygons S such that for all $P \in M$ and all subpolygons Q of P, there exists precisely one $Q' \in S$ such that $Q \sim^{\mathrm{a}} Q'$.

```
1:  procedure SUBPOLYGONS(M)
2:      A := max{Area_k(P) | P ∈ M}
3:      for a from 1 to A do
4:          S_a := {P ∈ M | Area_k(P) = a}
5:      end for
6:      for a from A to 1 do
7:          for P ∈ S_a do
8:              for v vertex of P do
9:                  Q := anf_k(P\v)
10:                 a' := Area_k(Q)
11:                 if dim(Q) = 2 and Q ∉ S_a' then
12:                     add Q to S_a'
13:                 end if
14:             end for
15:         end for
16:     end for
17:     return S_1 ∪ ··· ∪ S_A
18: end procedure
```

Proof. Proposition 1.2.9 ensures that the algorithm encounters every subpolygon at least once. By Proposition 1.2.6, the resulting set contains each polygon precisely once up to affine unimodular equivalence. □

In line 9 of Algorithm 1, we need to compute the polygon $P\backslash v$. This can be done by first determining all k-rational points of P, removing v and computing the convex hull of the remaining points (for example, by performing a Graham scan). In the rest of this section, we show how to significantly speed this up using an algorithm involving Hilbert bases. First, let us quickly summarize the necessary facts around Hilbert bases and their computation in two dimensions using continued fractions. We refer to [20, Chapter 10] for details.

Let $\sigma \subseteq \mathbb{R}^2$ be a rational pointed cone. After applying a unimodular transformation, we may assume $\sigma = \text{cone}(e_2, de_1 - ke_2)$ where $0 \leq k < d$ are coprime integers. The *Hilbert basis* $\mathcal{H}_\sigma$ of σ is the unique minimal generating set of the monoid $\sigma \cap \mathbb{Z}^2$. It can be computed as follows: First, using the modified euclidean algorithm, we express d/k as a Hirzebruch-Jung continued fraction $[\![b_1, \ldots, b_r]\!]$, which can be defined succinctly as

$$[\![b]\!] \;:=\; b, \qquad\qquad [\![b_1, \ldots, b_r]\!] \;:=\; b_1 - \frac{1}{[\![b_2, \ldots, b_r]\!]}.$$

Given $d/k = [\![b_1, \ldots, b_r]\!]$, the Hilbert basis is given by $\mathcal{H}_\sigma = \{u_0, \ldots, u_{r+1}\}$, where

$$u_0 \;:=\; e_2, \qquad u_1 \;:=\; e_1, \qquad u_{i+1} \;:=\; b_i u_i - u_{i-1}.$$

Proposition 1.2.10. *Let $\sigma \subseteq \mathbb{R}^2$ be a rational pointed cone with Hilbert basis $\mathcal{H}_\sigma = \{u_0, \ldots, u_{r+1}\}$ as constructed above. Consider the unbounded polyhedron*

$$\Theta_\sigma \;:=\; \text{conv}((\sigma \cap \mathbb{Z}^2)\backslash\{0\}).$$

Then the following holds.

(i) $\mathcal{H}_\sigma$ is the set of lattice points on the bounded edges of Θ_σ.

(ii) $\mathcal{H}'_\sigma := \{u_i \in \mathcal{H}_\sigma \;\mid\; b_i \neq 2\}$ is the set of vertices of Θ_σ.

Proof. For (i), we refer to [20, Theorem 10.2.8 (b)]. For part (ii), note that we have $u_{i+1} - u_i = (b_i - 1)u_i - u_{i-1}$ for all $i = 1, \ldots, r$. Hence u_i lies on the line segment between u_{i-1} and u_{i+1} if and only if $b_i = 2$. □

Corollary 1.2.11. *Let P be a lattice polygon with vertices $v_1, \ldots, v_n \in \mathbb{Z}^2$ and assume $v_j = 0$. With $\sigma = \text{cone}(v_{j-1}, v_{j+1})$ and $\mathcal{H}'_\sigma$ as in the proposition, the vertices of $P\backslash v_j$ are given by*

$$(\{v_1, \ldots, v_n\}\backslash\{v_j\}) \;\cup\; \mathcal{H}'_\sigma\,.$$

Remark 1.2.12. The vertices $\mathcal{H}'_\sigma = \{u'_1, \ldots, u'_s\}$ are already ordered naturally, i.e. u'_i and u'_{i+1} are connected by an edge. This means that if we start with ordered vertices $v_1, \ldots, v_n$ of P, Corollary 1.2.11 allows us to easily write down the ordered vertices of $P\backslash v_j$, which ties in nicely with our normal form.

Note that the condition $v_j = 0$ in Corollary 1.2.11 is not a restriction, since we can translate any lattice polygon to move v_j to the origin. Moreover, for any k-rational polygon P, we can apply Corollary 1.2.11 to kP to compute the vertices of $P\backslash v$ for any vertex v of P.

Using this improved algorithm for determining subpolygons, we were able to reproduce and extend a classification by Brown and Kasprzyk [18, Table 1]. See Section A.4.9 for our implementation in `RationalPolygons.jl`.

Classification 1.2.13 (Data available at [60])**.** *Below are the numbers* $\#(m)$ *of lattice subpolygons of* $[0,m]\times[0,m]$ *that are not subpolygons of* $[0,m-1]\times[0,m-1]$*, up to affine unimodular equivalence. We also include the maximally attained number of vertices* $N(m)$ *and the number of vertex maximizers* $M(m)$*.*

m	1	2	3	4	5	6
$\#(m)$	2	15	131	1 369	13 842	129 185
$N(m)$	4	6	8	9	10	12
$M(m)$	1	1	1	1	15	2

m	7	8	9	10	11
$\#(m)$	1 104 895	8 750 964	64 714 465	450 686 225	2 976 189 422
$N(m)$	13	14	16	16	17
$M(m)$	3	13	1	102	153

Remark 1.2.14. In our application, we are rarely interested in all subpolygons of a k-rational polygon P, but only in those sharing the same number of interior lattice points with P. This can be integrated rather easily into our Algorithm 1: Whenever we compute the vertices of $P\backslash v$ using Corollary 1.2.11, we check whether $\mathcal{H}'_\sigma \cap k\,\mathbb{Z}^2 \neq \emptyset$. If so, that means an interior lattice point of P has moved to the boundary of $P\backslash v$. Hence $P\backslash v$ has strictly fewer interior lattice points than P and we can discard it immediately. Our implementation supports this behaviour through a boolean flag, see A.3.6.

1.2.3 Maximal polygons

By a *k-maximal* polygon, we mean a k-rational polygon P such that for any k-rational polygon $Q \supseteq P$ with $i(P) = i(Q)$, we have $P = Q$. Clearly, every k-rational polygon is a subpolygon of a k-maximal one. This means if we classify all k-maximal polygons with $i \in \mathbb{Z}_{\geq 0}$ interior lattice points, we can use Algorithm 1 to obtain all k-rational polygons with i interior lattice points. In Proposition 1.2.16, we give a criterion for k-maximality that can be checked efficiently, see also A.2.20 for our implementation. Let us first fix some notation.

A k-rational polygon P can be written as the intersection of the affine halfplanes H_e associated with its edges, where

$$H_e = \{x \in \mathbb{R}^2 \;\mid\; \langle n_e, x\rangle \geq c_e\}, \qquad n_e \in \mathbb{Z}^2 \text{ primitive}, \qquad c_e \in \mathbb{Q}\,.$$

For any $x \in \mathbb{R}^2$, we define the *lattice distance* of the edge e with x as $\operatorname{ldist}_e(x) := \langle n_e, x\rangle - c_e$. Hence H_e consists of those points having non-negative lattice distance with e. For any rational number $q \in \mathbb{Q}$, we define the *translated halfplane* by

$$H_e + q \;:=\; \{x \in \mathbb{R}^2 \;\mid\; \operatorname{ldist}_e(x) \geq q\}.$$

By moving out the affine halfplanes, we obtain a new rational polygon $P^{(-1)} := \bigcap_e \left(H_e - \frac{1}{k}\right)$, where the intersection runs through all edges of P.

Lemma 1.2.15. *Let $\Delta = \operatorname{conv}(x, y, z) \subseteq \mathbb{R}^2$ be a lattice triangle containing no lattice points other than its vertices. Then $\operatorname{ldist}_e(z) = 1$, where e is the edge connecting x and y.*

Proof. By Pick's formula, Δ has area $i(\Delta) + \frac{b(\Delta)}{2} - 1 = \frac{1}{2}$. This forces $\operatorname{ldist}_e(z) = 1$. □

Proposition 1.2.16. *For a k-rational polygon P, the following are equivalent:*

(i) P is k-maximal,

(ii) for all $v \in \partial P^{(-1)} \cap \frac{1}{k}\mathbb{Z}^2$, we have $i(\operatorname{conv}(P \cup \{v\})) > i(P)$,

(iii) for all $v \in \partial P^{(-1)} \cap \frac{1}{k}\mathbb{Z}^2$, one of the following holds:

(1) There exists an edge e of P and a point $w \in e^\circ \cap \mathbb{Z}^2$ such that $v \notin H_e$,

(2) There exists a vertex $w \in \mathbb{Z}^2$ of P such that $v \notin H_e$ and $v \notin H_{e'}$, where e and e' are the edges adjacent to w.

Proof. "(i) ⇒ (ii)" is clear. To show "(ii) ⇒ (i)", assume P is not k-maximal. Then we have $P \subsetneq Q$ for some k-rational Q with $i(P) = i(Q)$. Let $z \in Q$ be a vertex and e an edge of P such that $z \notin H_e$. Pick two k-rational points $x, y \in e$ containing no k-rational points on the line segment between them and consider the triangle $\Delta := \operatorname{conv}(x, y, z)$. Let $z' \in \Delta$ be a k-rational point with minimal lattice distance to e. Then $\Delta' := \operatorname{conv}(x, y, z')$ contains no k-rational points other than its vertices, hence Lemma 1.2.15 implies $\operatorname{ldist}_e(z') = -\frac{1}{k}$. We obtain

$$i(P) \;\leq\; i(\operatorname{conv}(P \cup \{z'\})) \;\leq\; i(Q).$$

Since $i(P) = i(Q)$, we get equality, which contradicts (ii).

To see that (ii) implies (iii), note that $Q := \operatorname{conv}(P \cup \{v\})$ must contain a lattice point $w \in \partial P \cap \mathbb{Z}^2$ in its interior. If w lies in the relative interior of an edge of P, we are in case (1). If w is a vertex of P, we are in case (2). The converse (iii) ⇒ (ii) is immediate. □

Remark 1.2.17. A k-maximal polygon is not necessarily maximal as a convex set, i.e. there could be a convex set with the same number of interior lattice points which strictly contains the k-maximal polygon. A k-rational polygon is maximal as a convex set if and only if every edge contains an interior lattice point. For example, the quadrilateral

$$\operatorname{conv}\left(\left(0, 1 + \frac{1}{k}\right), (0, 0), (k(i+1) + i, 0), \left(\frac{1}{k}, 1 + \frac{1}{k}\right)\right), \quad i \in \mathbb{Z}_{\geq 0}, k \in \mathbb{Z}_{\geq 1}$$

is k-maximal for $(i, k) \notin \{(0, 1), (0, 2), (1, 1)\}$ by Proposition 1.2.16, but it is not a maximal convex set since there is no interior lattice point on the edge between $\left(0, 1 + \frac{1}{k}\right)$ and $\left(\frac{1}{k}, 1 + \frac{1}{k}\right)$.

We end this section by showing that for lattice polygons, our notion of maximality coincides with the one used by [38] and [19]. Recall that the *interior hull* of a rational polygon is $P^{(1)} := \operatorname{conv}(P^\circ \cap \mathbb{Z}^2)$.

Proposition 1.2.18. *Let P be a lattice polygon with two-dimensional interior hull $P^{(1)}$. Then P is 1-maximal if and only if $P = P^{(1)(-1)}$.*

Proof. Assume P is 1-maximal. Then by [19, Theorem 5], $P^{(1)(-1)}$ is a lattice polygon containing P. Clearly, we have $i(P) = i(P^{(1)(-1)})$, hence we obtain $P = P^{(1)(-1)}$. Conversely, if $P = P^{(1)(-1)}$, pick a 1-maximal polygon Q containing P with $i(P) = i(Q)$. Then we must have $Q^{(1)} = P^{(1)}$, hence $Q = Q^{(1)(-1)} = P^{(1)(-1)} = P$. □

1.3 Rational polygons in $\mathbb{R}\times[-1,1]$ or with collinear interior points

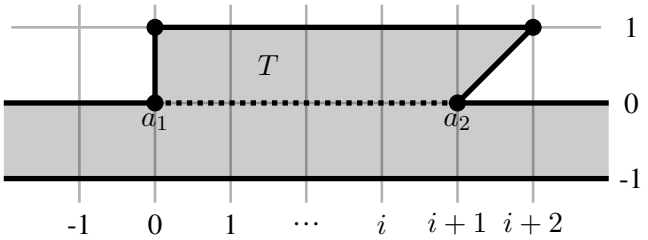
In this short section, we will classify k-maximal polygons that can be realized in $\mathbb{R}\times[-1,1]$ with given number of interior lattice points. This case occurs in our classifications of k-maximal polygons with no interior lattice points in Section 1.4 and exactly one interior lattice point in Section 1.5, hence we treat it here separately. As an application, we classify k-rational polygons with collinear interior lattice points.

Proposition 1.3.1. *Every rational polygon $P \subseteq \mathbb{R}\times[-1,1]$ with $i \in \mathbb{Z}_{\geq 0}$ interior lattice points can be realized in $\mathbb{R}\times[-1,0] \cup T$, with the half-open trapezoid*

$$T := \operatorname{conv}((0,0),(i+1,0),(i+2,1),(0,1)) \setminus (\mathbb{R}\times\{0\}).$$

Proof. After a suitable translation, we can assume the intersection of the interior of P with $\mathbb{R}\times\{0\}$ is contained in $(0,i+1)\times\{0\}$. Furthermore, we can assume this intersection to be non-empty, otherwise we can realize P in $\mathbb{R}\times[-1,0]$. Consider the points $a_1 := (0,0)$ and $a_2 := (i+1,0)$. Then we find unique supporting lines L_1 and L_2 of P with $a_j \in L_j$ such that L_1 and L_2 both contain a point of P with positive second coordinate. Since we can reflect on $\mathbb{R}\times\{0\}$, we can assume L_1 and L_2 to be either parallel or intersect in a point of $\mathbb{R}\times(0,\infty)$. Therefore, after a suitable horizontal shearing, we can assume that L_1 intersects $\mathbb{R}\times\{1\}$ in a point of $[0,1]\times\{1\}$ and L_2 intersects $\mathbb{R}\times\{1\}$ in a point of $(-\infty,i+2]\times\{1\}$. But this implies $P \subseteq \mathbb{R}\times[-1,0]\cup T$. □

Figure 1.4: All rational polygons in $\mathbb{R}\times[-1,1]$ with i interior lattice points can be realized in $\mathbb{R}\times[-1,0]\cup T$



Construction 1.3.2. Consider a k-rational polygon $P \subseteq \mathbb{R}\times[-1,0] \cup T$ with $P \cap T \neq \emptyset$. Then there are supporting lines L_1 and L_2 as in the proof of the proposition above. Consider the associated affine halfplanes H_1 and H_2 with $P \subseteq H_j$. We define the *surrounding polygon* of P to be the k-rational polygon

$$P_{\text{sur}} := \text{conv}\left(Q \cap \frac{1}{k}\mathbb{Z}^2\right), \quad \text{where } Q := \mathbb{R}\times[-1,1] \cap H_1 \cap H_2.$$

Note that we have $P \subseteq P_{\text{sur}} \subseteq \mathbb{R}\times[-1,0] \cup T$ and $i(P) = i(P_{\text{sur}}) = i$. In particular, if P is k-maximal, we have $P = P_{\text{sur}}$.

Definition 1.3.3. For two distinct points $v, w \in \mathbb{R}^2$, we write $H(v,w)$ for the closed affine halfplane given by all points to the right of the line through v and w with respect to the direction vector $w - v$

Algorithm 1.3.4 (See A.4.1 for an implementation)**.** Given integers $k \in \mathbb{Z}_{\geq 1}$ and $i \in \mathbb{Z}_{\geq 0}$, we can classify all k-maximal polygons with i interior lattice points in $\mathbb{R}\times[-1,1]$ as follows: Consider $a_1 := (0,0)$ and $a_2 := (i+1,0)$. To any pair of points $x_1, x_2 \in T$, we associate the polygon

$$P := \text{conv}\left(Q \cap \frac{1}{k}\mathbb{Z}^2\right), \quad \text{where } Q := \mathbb{R}\times[-1,1] \cap H(a_1,x_1) \cap H(x_2,a_2)$$

This defines a possible surrounding polygon. Going through all polygons that arise this way and filtering out those that are k-maximal (for instance, using Proposition 1.2.16) gives us all k-maximal polygons in $\mathbb{R}\times[-1,1]$ having i interior lattice points, excluding those with $P \cap T = \emptyset$, i.e. those that can be realized in $\mathbb{R}\times[0,1]$. However, polygons contained in $\mathbb{R}\times[0,1]$ are not k-maximal, hence we already have all k-maximal polygons.

We turn to the case of collinear interior lattice points, which is tightly related to being realizable in $\mathbb{R}\times[-1,1]$. Clearly, a polygon in $\mathbb{R}\times[-1,1]$ has collinear interior lattice points. The converse holds for lattice polygons with at least two collinear interior lattice points, which Koelman [38, Section 4.3] used to completely describe them. The following proposition generalizes this fact to k-rational polygons.

Proposition 1.3.5. *Let $P \subseteq \mathbb{R}^2$ be k-rational polygon with collinear interior lattice points and $i(P) > k$. Then P can be realized in $\mathbb{R}\times[-1,1]$.*

Proof. After an affine unimodular transformation, we can assume the interior lattice points of P to be of the form $(j,0)$ for $0 \leq j \leq i(P)-1$. Suppose P is not contained in $\mathbb{R}\times[-1,1]$. After a reflection, we can assume that there exists a vertex $v = (x,y)$ of P with $y > 1$. We intend to show that there exists a k-rational point $q \in P \cap (\mathbb{R}\times\{1+\frac{1}{k}\})$. If $y = 1+\frac{1}{k}$, we can take $q = v$. Otherwise, we have $y \geq 1+\frac{2}{k}$ and we consider the line segment $L := P \cap (\mathbb{R}\times\{1+\frac{1}{k}\}$. We can estimate its length using the intercept theorem to obtain $\text{length}(L) \geq \frac{k}{k+2}$. If $k \geq 2$, this implies $\text{length}(L) \geq \frac{1}{k}$ and therefore there exists a k-rational point $q = (\frac{a}{k}, 1+\frac{1}{k}) \in L$. After a horizontal shearing, we can assume $0 \leq a \leq k$. Consider the triangle $Q := \text{conv}(q,(0,0),(i(P)-1,0))$

and note that $Q \setminus \{q\} \subseteq P^\circ$. If $a = 0$, we have $(0,1) \in Q$. If $a > 0$, then $i(P) > k$ implies $(1,1) \in Q$. Thus in both cases, we have an additional interior lattice point of P, a contradiction.

It remains to consider the case $k = 1$. Take any vertex $v = (x, y) \in P$ with $y \geq 1$. Then the triangle $\operatorname{conv}(v, (0,0), (1,0))$ does not contain any lattice points other than its vertices. Lemma 1.2.15 then implies $y = 1$. □

Using Algorithm 1, we can determine all subpolygons of the k-maximal polygons in $\mathbb{R} \times [-1, 1]$. For $i > k$, the preceding proposition implies that this gives us all k-rational polygons with i collinear interior lattice points. We arrive at the following classification.

Classification 1.3.6 (Data available at [58])**.** *Below are the numbers of 2-rational polygons with i collinear interior lattice points.*

i	3	4	5	6	7	8
#Polygons	168 640	504 530	1 279 695	2 881 106	5 924 808	11 343 912
i	9	10	11	12	13	14
#Polygons	20 496 555	35 295 876	58 364 056	93 212 470	144 449 999	218 021 550
i	15	16	17	18	19	20
#Polygons	321 478 832	464 285 436	658 158 267	917 447 376	1 259 556 240	1 705 404 538

Below are the numbers of 3-rational polygons with i collinear interior lattice points.

i	4	5	6
#Polygons	546 989 533	2 150 324 427	7 124 538 405

Recall that lattice polygons with collinear interior lattice points were classified by Koelman [38, Section 4.3]. In particular, he showed that there are exactly $\frac{1}{6}(i+3)(2i^2+15i+16)$ lattice polygons with $i \geq 2$ collinear interior lattice points. Using our values for $k = 2$, we found that they too can be described by a polynomial. Hence we raise the following conjecture, which we have verified up to $i = 20$.

Conjecture 1.3.7. *There are exactly*

$$\frac{1}{1260}(i+1)(512i^6 + 12\,928i^5 + 137\,740i^4 + 685\,145i^3 + 1\,582\,743i^2 + 1\,665\,222i + 710\,640)$$

half-integral polygons with $i \geq 3$ collinear interior lattice points.

1.4 Rational polygons without interior lattice points

We give a classification algorithm for k-maximal polygons with no interior lattice points. Note that for $k = 2$, the four k-maximal polygons with no interior lattice points are known [4, Section 2]. With our approach, we obtain results up to $k = 20$, see Classification 1.4.5.

The main tool that allows us to get good bounds for maximal polygons with no interior lattice points is the lattice width. Recall that the *lattice width* of a rational polygon P in direction $w \in \mathbb{Z}^2 \setminus \{0\}$ is defined as

$$\operatorname{width}_w(P) := \max_{v \in P} \langle v, w \rangle - \min_{v \in P} \langle v, w \rangle \in \mathbb{Z}^2 .$$

The *lattice width* $\mathrm{lw}(P)$ of P is defined to be the minimum over all $\mathrm{width}_w(P)$ with $w \in \mathbb{Z}^2 \setminus \{0\}$. We call $w \in \mathbb{Z}^2 \setminus \{0\}$ a *lattice width direction* of P, if $\mathrm{lw}(P) = \mathrm{width}_w(P)$.

Lemma 1.4.1. *Every rational polygon without interior lattice points can be realized in* $\mathbb{R} \times [-1, 2]$.

Proof. After an affine unimodular transformation, we may assume that $(1, 0)$ is a lattice width direction and $P \subseteq [x_l, x_r] \times \mathbb{R}$ for $x_l, x_r \in \mathbb{R}$. Then [14, Corollary 2.13] gives lower bounds for the numbers $|P^\circ \cap (\{h\} \times \mathbb{Z})|$, where $x_l \leq h \leq x_r$ and $h \in \mathbb{Z}$. Since P does not have interior lattice points, these imply that there are at most two interior integral vertical lines of P, hence $x_r - x_l \leq 3$ and the claim follows. □

Proposition 1.4.2. *Every rational polygon P without interior lattice points can be realized in $A \cup (\mathbb{R} \times [0, 1]) \cup B$, with the two half-open trapezoids*

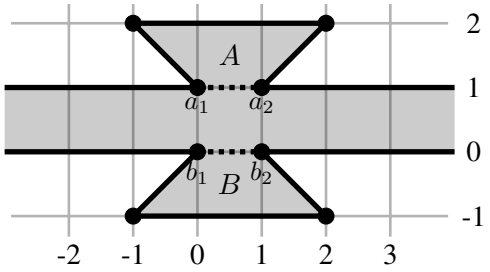
$$\begin{aligned} A &:= \mathrm{conv}\,((0,1),(1,1),(2,2),(-1,2)) \setminus (\mathbb{R} \times \{1\}), \\ B &:= \mathrm{conv}\,((0,0),(-1,-1),(2,-1),(1,0)) \setminus (\mathbb{R} \times \{0\}). \end{aligned}$$

Proof. By Lemma 1.4.1, we can assume $P \subseteq \mathbb{R} \times [-1, 2]$. Consider the line segments

$$P_0 := P^\circ \cap (\mathbb{R} \times \{0\}), \qquad P_1 := P^\circ \cap (\mathbb{R} \times \{1\}).$$

We may assume P_0 and P_1 to be non-empty, otherwise we are in the situation of Proposition 1.3.1. Since P_0 and P_1 do not contain lattice points, we can apply a translation and shearing to achieve $P_0 \subseteq [0, 1] \times \{0\}$ and $P_1 \subseteq [0, 1] \times \{1\}$. By symmetry, it is enough to show that P does not contain a point $v = (x, y)$ with $x < 0$ and $y > x$. Suppose there is such a point. Then $\mathrm{conv}(P_0 \cup P_1 \cup \{v\}) \subseteq P$ has $(0, 0)$ as an interior lattice point, a contradiction. □

Figure 1.5: All polygons without interior lattice points can be realized in $A \cup \mathbb{R} \times [0, 1] \cup B$.



Construction 1.4.3. Consider a k-rational polygon $P \subseteq A \cup \mathbb{R} \times [0, 1] \cup B$ with $P \cap A \neq \emptyset$ and $P \cap B \neq \emptyset$ where A and B are as in the last Proposition. Consider the points

$$a_1 := (0, 1), \quad a_2 := (1, 1), \quad b_1 := (0, 0), \quad b_2 := (1, 0).$$

Then we have unique supporting lines $L_{a_1}, L_{a_2}, L_{b_1}, L_{b_2}$ of P with $a_j \in L_{a_j}, b_j \in L_{b_j}$ such that L_{a_j} contains a point of $P \cap A$ and L_{b_j} contains a point of $P \cap B$. These define affine halfplanes

$H_{a_1}, H_{a_2}, H_{b_1}, H_{b_2}$ with $P \subseteq H_{a_j}$ and $P \subseteq H_{b_j}$. We define the *surrounding polygon* of P to be the k-rational polygon

$$P_{\text{sur}} := \operatorname{conv}\left(Q \cap \frac{1}{k}\mathbb{Z}^2\right), \qquad Q := \mathbb{R}\times[-1,2] \cap H_{a_1} \cap H_{a_2} \cap H_{b_1} \cap H_{b_2}.$$

Note that we have $P \subseteq P_{\text{sur}} \subseteq A \cup \mathbb{R}\times[0,1] \cup B$ and $i(P) = i(P_{\text{sur}}) = 0$. In particular, if P is k-maximal, we have $P = P_{\text{sur}}$.

Algorithm 1.4.4 (See A.4.5 for an implementation)**.** Given $k \in \mathbb{Z}_{\geq 1}$, we can classify all k-maximal polygons without interior lattice points as follows: To any choice of points $v_1, v_2 \in A$ and $w_1, w_2 \in B$, we associate the polygon $P := \operatorname{conv}\left(Q \cap \frac{1}{k}\mathbb{Z}^2\right)$, where

$$Q := \mathbb{R}\times[-1,2] \cap H(a_1, v_1) \cap H(v_2, a_2) \cap H(w_1, b_1) \cap H(b_2, w_2).$$

This defines a possible surrounding polygon. Going through all polygons that arise this way and filtering out those that are k-maximal (for instance, using Proposition 1.2.16) gives us all k-maximal polygons without interior lattice points, excluding those with $P \cap A = \emptyset$ or $P \cap B = \emptyset$. The latter ones can be realized in $\mathbb{R}\times[-1,1]$ and hence can be obtained with Algorithm 1.3.4 applied to $i = 0$.

We ran Algorithm 1.4.4 up to $k = 20$ and obtained the following classification, see also Figures 1.6, 1.7 and 1.8.

Classification 1.4.5 (Data available at [62])**.** *Below are the numbers of k-maximal polygons without interior lattice points, split into the two cases of Algorithm 1.4.4.*

k	1	2	3	4	5	6	7	8	9	10
$\mathbb{R}\times[-1,1]$	1	4	12	24	54	85	164	244	380	517
$\mathbb{R}\times[-1,2]$	0	0	2	15	80	214	791	1 652	4 101	8 368
Total	1	4	14	39	134	299	955	1 896	4 481	8 885

k	11	12	13	14	15	16	17	18	19	20
$\mathbb{R}\times[-1,1]$	809	1 021	1 506	1 878	2 398	2 987	4 039	4 743	6 239	7 263
$\mathbb{R}\times[-1,2]$	17 757	29 637	57 937	91 009	161 418	258 728	409 990	595 051	930 544	1 304 553
Total	18 566	30 658	59 443	92 887	163 816	261 715	414 029	599 794	936 783	1 311 816

Using Algorithm 1, we computed all subpolygons of the k-maximal polygons from Classification 1.4.5 up to $k = 6$. This gives us all k-rational polygons without interior lattice points that cannot be realized in $\mathbb{R}\times[0,1]$ (there are infinitely many in $\mathbb{R}\times[0,1]$), see Theorem 1.1.1. The data is available at [62].

Figure 1.6: All four 2-maximal rational polygons without interior integral points, see also [4, Section 2]. All of them can be realized in $\mathbb{R}\times[-1,1]$.

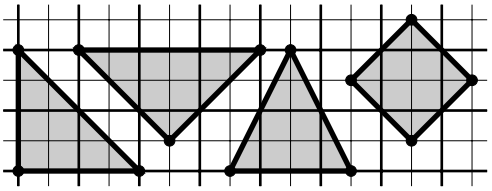


Figure 1.7: All fourteen 3-maximal rational polygons without interior integral points. The two rightmost polygons in the lower row are the only ones that cannot be realized in $\mathbb{R} \times [-1, 1]$.

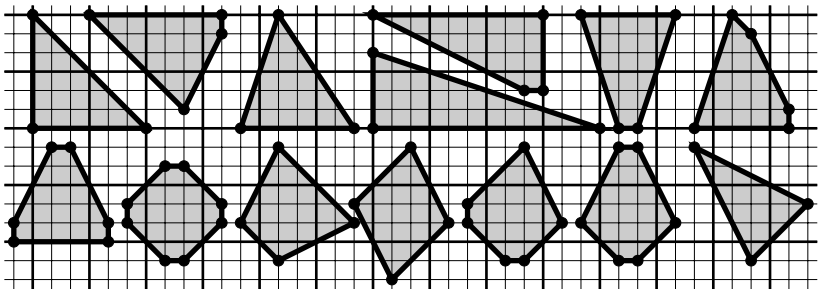
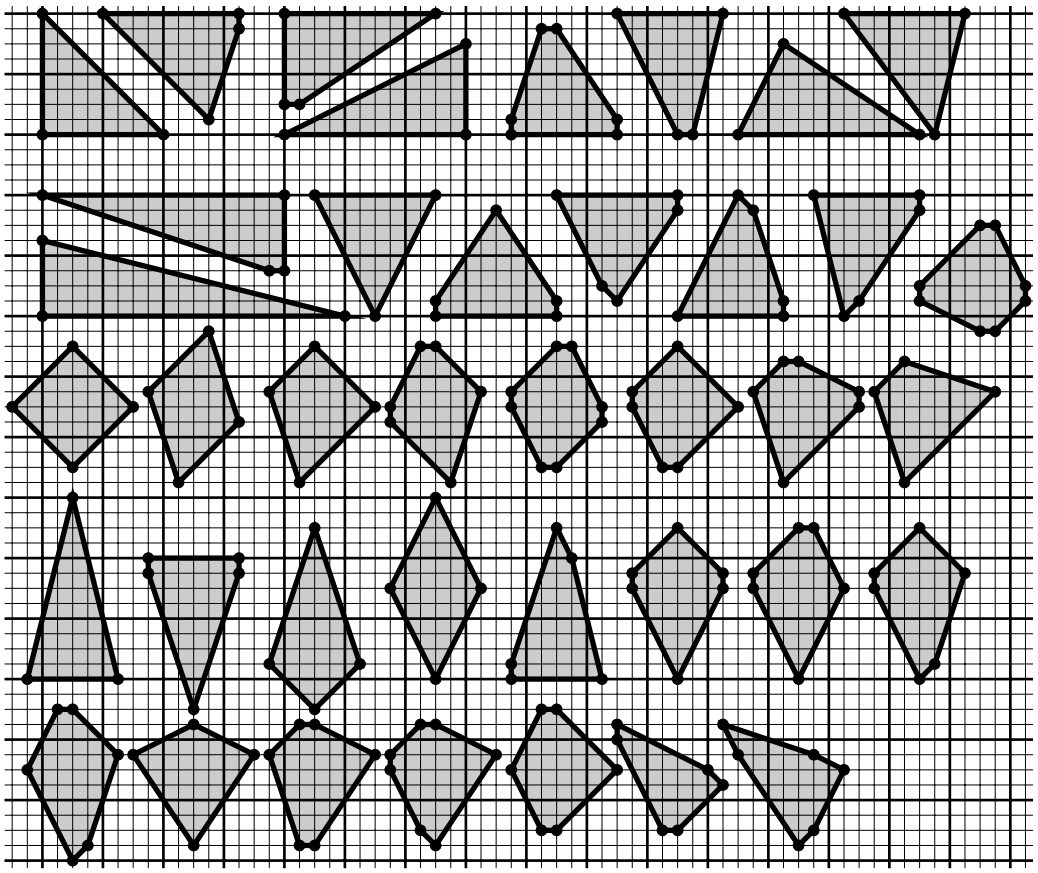


Figure 1.8: All thirty-nine 4-maximal rational polygons without interior lattice points. The 24 polygons in the upper three rows can be realized in $\mathbb{R} \times [-1, 1]$. The remaining 15 polygons cannot.



1.5 Rational polygons with exactly one interior lattice point and LDP polygons

We give a classification algorithm for k-maximal polygons with exactly one interior lattice point and obtain explicit results up to $k = 10$, see Classification 1.5.5. Moreover, we apply this to LDP polygons and extend the classification of k-hollow LDP polygons from [27] to $k = 6$, see Classification 1.5.6.

Lemma 1.5.1. *Every rational polygon P with exactly one interior lattice point can be realized in $\mathbb{R} \times [-2, 2]$.*

Proof. After an affine unimodular transformation, we may assume that $(1, 0)$ is a lattice width direction and $P \subseteq [x_l, x_r] \times \mathbb{R}$ for $x_l, x_r \in \mathbb{R}$. Then [14, Corollary 2.13] gives lower bounds for the numbers $|P^\circ \cap (\{h\} \times \mathbb{Z})|$, where $x_l \leq h \leq x_r$ and $h \in \mathbb{Z}$. Since P only has one interior lattice point, these imply that there are at most three interior integral vertical lines of P, hence $x_r - x_l \leq 4$ and the claim follows. □

Proposition 1.5.2. *Every rational polygon with exactly one interior lattice point can be realized in $\mathbb{R} \times [0, 1] \cup A \cup B \cup C_q$ for some $q \in \{1, 2, 3, 4, 5\}$, where*

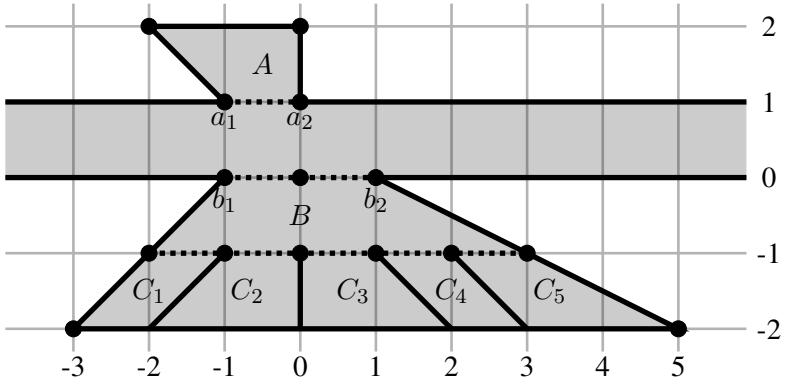
$$\begin{aligned}
A &:= \operatorname{conv}((-1,1),(0,1),(0,2),(-2,2)) \setminus (\mathbb{R} \times \{1\}),\\
B &:= \operatorname{conv}((-1,0),(-2,-1),(3,-1),(1,0)) \setminus (\mathbb{R} \times \{0\},\\
C_1 &:= \operatorname{conv}((-3,-2),(-2,-2),(-1,-1),(-2,-1)) \setminus (\mathbb{R} \times \{-1\}),\\
C_2 &:= \operatorname{conv}((-2,-2),(0,-2),(0,-1),(-1,-1)) \setminus (\mathbb{R} \times \{-1\}),\\
C_3 &:= \operatorname{conv}((0,-2),(2,-2),(1,-1),(0,-1)) \setminus (\mathbb{R} \times \{-1\}),\\
C_4 &:= \operatorname{conv}((2,-2),(4,-2),(5,-2),(1,0)) \setminus (\mathbb{R} \times \{-1\}),\\
C_5 &:= \operatorname{conv}((-1,0),(-3,-2),(5,-2),(1,0)) \setminus (\mathbb{R} \times \{-1\}).
\end{aligned}$$

Proof. By Lemma 1.5.1, we can assume $P \subseteq \mathbb{R} \times [-2, 2]$ and after translation, we can assume the interior lattice point of P to be the origin. Consider the line segments

$$P_1 := P^\circ \cap (\mathbb{R} \times \{1\}), \quad P_0 := P^\circ \cap (\mathbb{R} \times \{0\}), \quad P_{-1} := P^\circ \cap (\mathbb{R} \times \{-1\}).$$

Clearly, we have $P_0 \neq \emptyset$ and convexity forces $P_0 \subseteq [-1, 1] \times \{0\}$. If $P_1 = P_{-1} = \emptyset$, we are in the situation of Proposition 1.3.1, hence we are done after suitable translation and reflection. Thus, after a reflection, we can assume $P_1 \neq \emptyset$. If P were to contain any point outside of $\mathbb{R} \times [0, 1] \cup A \cup B \cup \bigcup_q C_q$, convexity would force P to contain an additional lattice point, which is impossible. Moreover, if there were points $v \in P^\circ \cap C_q$ and $w \in P^\circ \cap C_r$ for $q \neq r$, we would have an interior lattice point in the triangle $\operatorname{conv}(v, w, (0, 0)) \subseteq P$, which is impossible. Hence $P^\circ \cap C_q \neq \emptyset$ for at most one $q \in \{1, 2, 3, 4, 5\}$ and we arrive at the claim. □

Figure 1.9: All rational polygons with exactly one interior lattice point can be realized in $\mathbb{R}\times[0,1]\cup A\cup B\cup C_q$ for some $q\in\{1,2,3,4,5\}$.



Construction 1.5.3. Consider a k-rational polygon $P\subseteq\mathbb{R}\times[0,1]\cup A\cup B\cup C_q$ with $P\cap A\neq\emptyset$, where $q\in\{1,2,3,4,5\}$. Consider the points

$$a_1:=(-1,1),\quad a_2:=(0,1),\quad b_1:=(-1,0),\quad b_2:=(1,0),$$
$$c_1:=(q-3,-1),\quad c_2:=(q-2,-1).$$

Then we have unique supporting lines $L_{a_1},L_{a_2},L_{b_1},L_{b_2}$ of P with $a_j\in L_{a_j},b_j\in L_{b_j}$ such that L_{a_j} contains a point of $P\cap A$ and L_{b_j} contains a point of $P\cap B$. Moreover, if $P\cap C_q\neq\emptyset$, we have unique supporting lines L_{c_1},L_{c_2} defined analogously. All these supporting lines define affine halfplanes H_{a_j},H_{b_j} and H_{c_j} that contain P. We define the *surrounding polygon* of P to be the k-rational polygon $P_{\text{sur}}:=\operatorname{conv}\left(Q\cap\frac{1}{k}\mathbb{Z}^2\right)$ where

$$Q:=\begin{cases}\mathbb{R}\times[-1,2]\cap H_{a_1}\cap H_{a_2}\cap H_{b_1}\cap H_{b_2}, & P\cap C_q=\emptyset,\\ \mathbb{R}\times[-2,2]\cap H_{a_1}\cap H_{a_2}\cap H_{b_1}\cap H_{b_2}\cap H_{c_1}\cap H_{c_2}, & P\cap C_q\neq\emptyset.\end{cases}$$

Note that we have $P\subseteq P_{\text{sur}}\subseteq\mathbb{R}\times[0,1]\cup A\cup B\cup C_q$ and $i(P)=i(P_{\text{sur}})=1$. In particular, if P is k-maximal, we have $P=P_{\text{sur}}$.

Algorithm 1.5.4 (See A.4.9 for an implementation)**.** Given $k\in\mathbb{Z}_{\geq 1}$, we can classify all k-maximal polygons with exactly one interior lattice point as follows: Pick $q\in\{1,\dots,5\}$. To any choice of points $v_1,v_2\in A,w_1,w_2\in B$ and $u_1,u_2\in C_q$, we associate the polygon $P:=\operatorname{conv}\left(Q\cap\frac{1}{k}\mathbb{Z}^2\right)$, where

$$Q:=\mathbb{R}\times[-2,2]\cap H(a_1,v_1)\cap H(v_2,a_2)\cap H(w_1,b_1)\cap H(b_2,w_2)\cap H(u_1,c_1)\cap H(c_2,u_2).$$

This defines a possible surrounding polygon. Going through all polygons that arise this way and filtering out those that are k-maximal (for instance, using Proposition 1.2.16) gives us all k-maximal polygons with one interior lattice point that can be realized in $\mathbb{R}\times[-2,2]$, excluding those that can be realized in $\mathbb{R}\times[-1,2]$. The latter ones are obtained in an analogous way but without the halfplanes given by c_1 and c_2. Finally, the polygons that can be realized in $\mathbb{R}\times[-1,1]$ are obtained by Algorithm 1.3.4 applied to $i=1$.

We ran Algorithm 1.5.4 to $k = 10$ and obtained the following classification, see also Figures 1.10 and 1.11.

Classification 1.5.5 (Data available at [61]). *Below are the numbers of k-rational polygons with exactly one interior lattice point, split into the three cases according to Algorithm 1.5.4.*

k	1	2	3	4	5	6	7	8	9	10
$\mathbb{R}\times[-1,1]$	2	9	26	57	132	199	396	605	937	1 260
$\mathbb{R}\times[-1,2]$	1	1	12	83	470	1 390	4 964	11 426	27 801	56 303
$\mathbb{R}\times[-2,2]$	0	0	1	5	96	329	2 874	8 241	32 176	81 950
Total	3	10	39	145	698	1 918	8 234	20 272	60 914	139 513

Using Algorithm 1, we computed all subpolygons of the k-maximal polygons from Classification 1.5.5 up to $k = 5$. This gives us all k-rational polygons with exactly one interior lattice point, see Theorem 1.1.1.

We turn to LDP polygons, which are of special interest to toric geometry. Recall that a lattice polygon P is called *LDP*, if $0 \in P^\circ$ and its vertices are primitive vectors of $\mathbb{Z}^2$. By taking the fan generated by its faces, an LDP polygon P defines a toric log del Pezzo surface X_P and this assignment is a bijection on isomorphy classes. We refer to [36] for more information on LDP polygons as well as their classification up to index 17. See also Section 2.2.

In [27], the authors describe a classification algorithm for *almost k-hollow* LDP polygons, i.e. LDP polygons with $P \cap k\,\mathbb{Z}^2 = \{0\}$. These correspond to toric $\frac{1}{k}$-log canonical del Pezzo surfaces, which means the discrepancies of the exceptional divisors in the canonical resolution are all greater or equal to $\frac{1}{k} - 1$. Dividing an almost k-hollow LDP polygon by k gives a k-rational polygon having only the origin as an interior lattice point. Hence we can also classify k-hollow LDP polygons by taking our k-maximal polygons with one interior lattice point and computing all subpolygons having primitive vertices. We have done this up to $k = 6$, in particular reproducing the classification of the almost 3-hollow LDP polygons from [27]. See also Figures 1.12, 1.13, 1.14 and 1.15.

Classification 1.5.6 (See A.4.4 for an implementation). *Below are the numbers $\#_{\text{ldp}}(k)$ of almost k-hollow LDP polygons, their maximally attained number of vertices $N_{\text{ldp}}(k)$ and the number of vertex maximizers $M_{\text{ldp}}(k)$. Equivalently, $\#_{\text{ldp}}(k)$ is the number of toric $\frac{1}{k}$-log canonical del Pezzo surfaces and $N_{\text{ldp}}(k) - 2$ is their maximally attained Picard number. For comparison, we include the analogous numbers for all k-rational polygons with exactly one interior lattice points.*

k	1	2	3	4	5	6
$\#_{\text{ldp}}$	16	505	48 032	1 741 603	154 233 886	2 444 400 116
N_{ldp}	6	8	12	12	16	18
M_{ldp}	1	1	1	20	1	1
$\#_{\text{all}}$	16	5 145	924 042	101 267 212	8 544 548 186	
N_{all}	6	9	12	14	18	
M_{all}	1	1	2	23	1	

Figure 1.10: All ten 2-maximal rational polygons with exactly one interior integral point up to affine unimodular equivalence. The threefold standard triangle is the only one that cannot be realized in $\mathbb{R} \times [-1, 1]$.

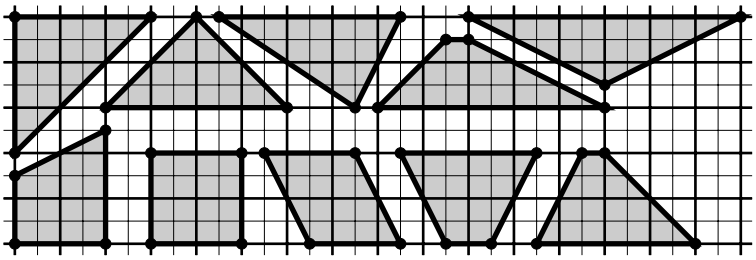


Figure 1.11: All thirty-nine 3-maximal polygons with exactly one interior lattice point. The upper four rows are the 26 polygons in $\mathbb{R} \times [-1, 1]$. The lower two rows contain the 12 polygons in $\mathbb{R} \times [-1, 2]$ and the rightmost polygon in the last row is the only polygon in $\mathbb{R} \times [-2, 2]$.

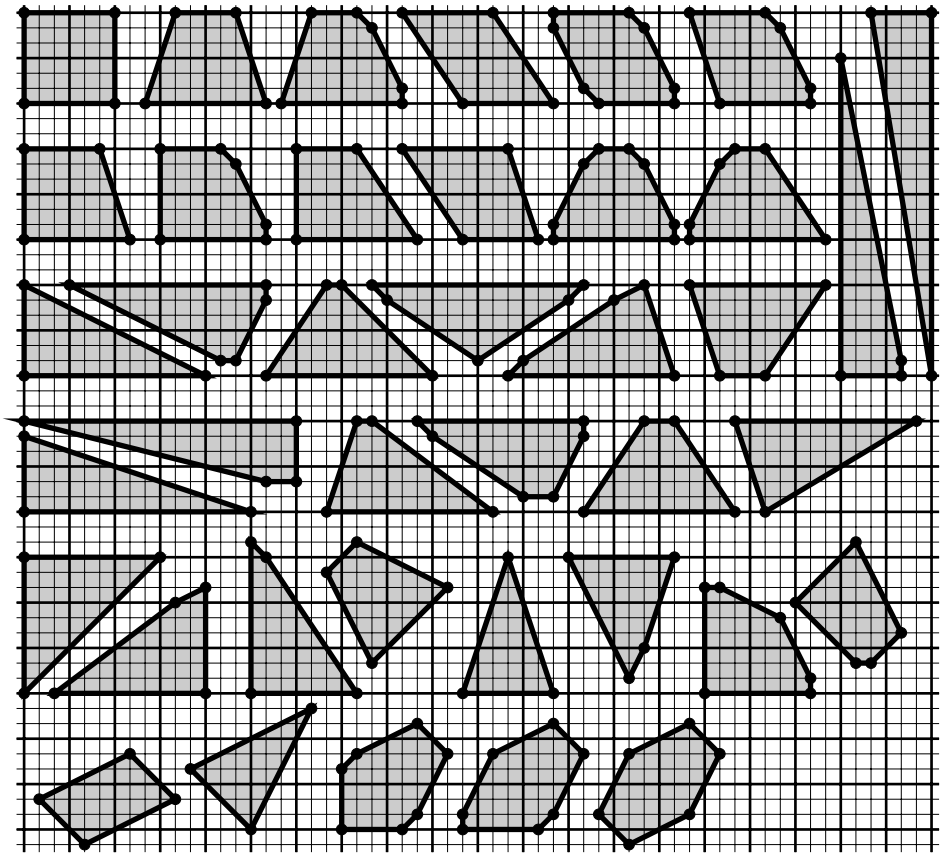


Figure 1.12: The k-rational polygons with exactly one interior lattice point and maximal number of vertices for $k \in \{1, 2, 3, 5\}$, see Classification 1.5.6.

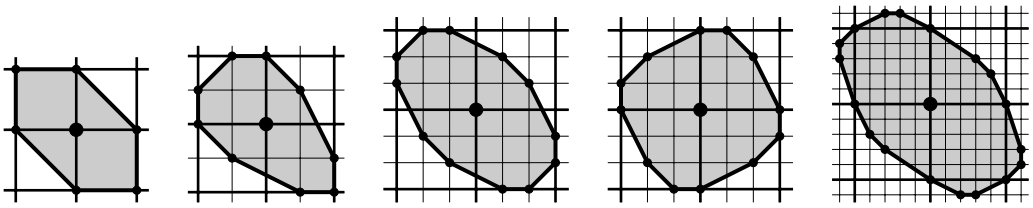


Figure 1.13: The twenty-three 4-rational polygons with exactly one interior lattice point and maximal number of vertices, see Classification 1.5.6.

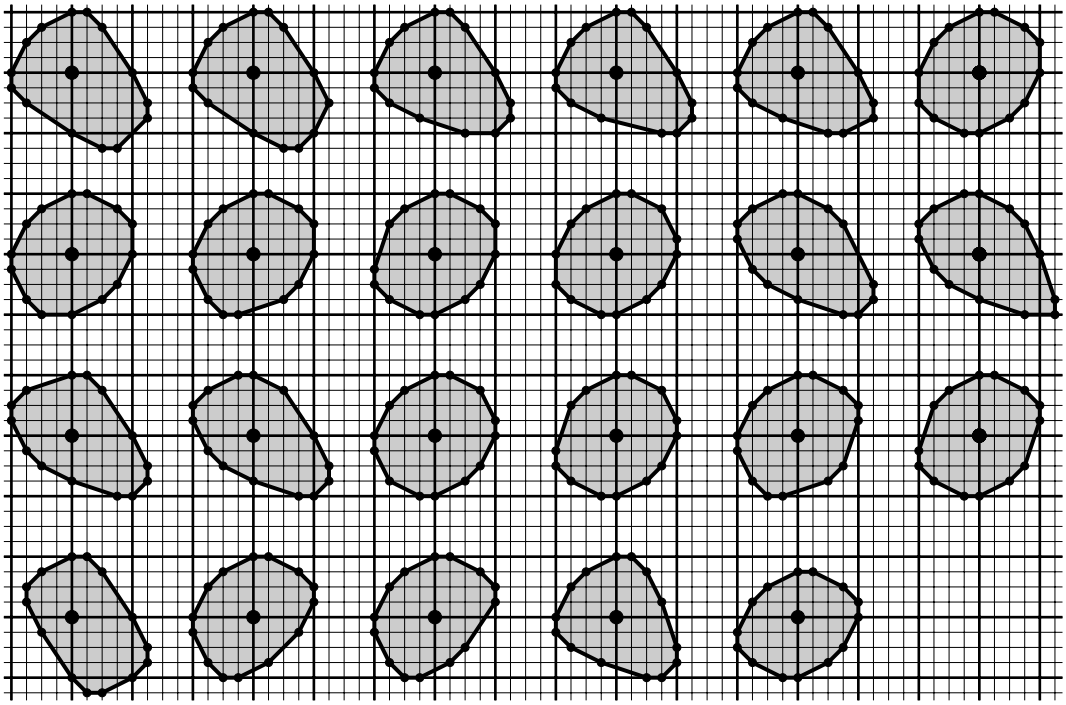


Figure 1.14: The almost k-hollow LDP polygons with maximal number of vertices for $k \in \{1, 2, 3, 5, 6\}$, see Classification 1.5.6.

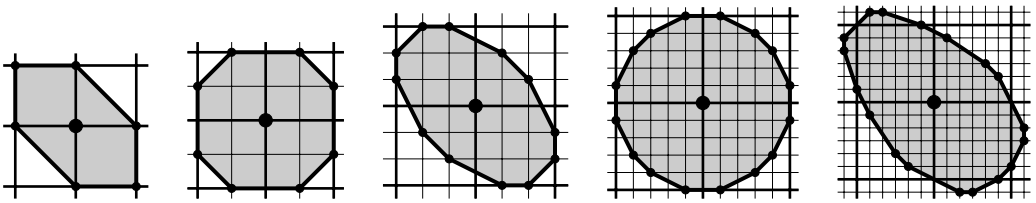
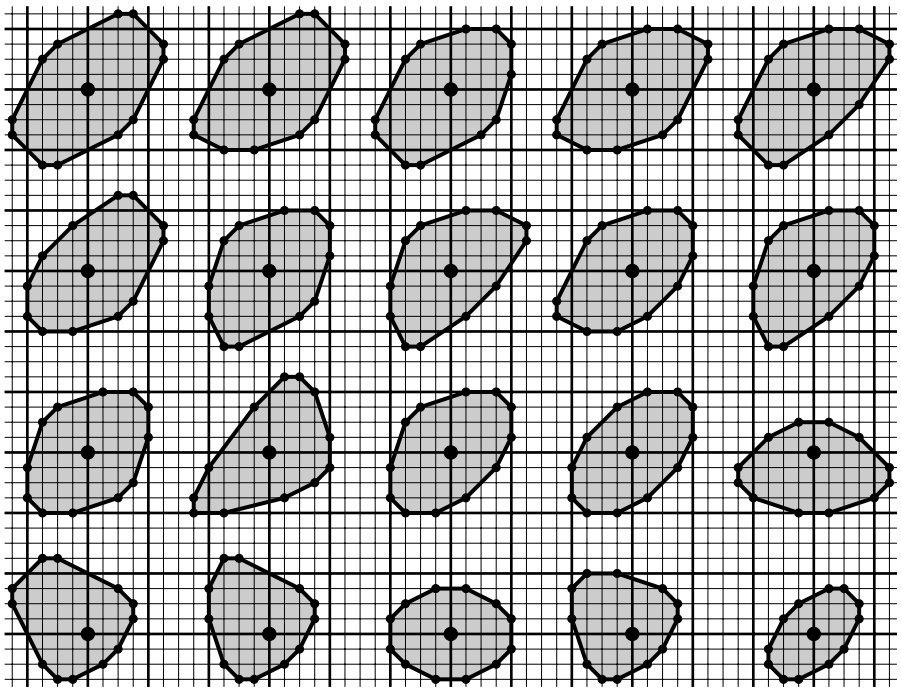


Figure 1.15: The 20 almost 4-hollow LDP polygons with maximal number of vertices, see Classification 1.5.6.



1.6 A general classification procedure

In this section, we show how a classification of k-maximal polygons with $i \in \mathbb{Z}_{\geq 0}$ interior lattice points can be derived from a sufficiently large classification of lattice polygons. First, we need a suitable upper area bound of rational polygons in terms of their number of interior lattice points.

Lemma 1.6.1. *Let P be a rational polygon with $k = \operatorname{denom}(P)$ and $i := i(P)$ which cannot be realized in $\mathbb{R} \times [-1, 1]$. Then we have*

$$\operatorname{Area}_k(P) \leq \max\left(k^2(4i+5),\ \frac{1}{2}k(k+2)^2(i+1)\right).$$

Proof. After an affine unimodular transformation, we can assume that $(0, 1)$ is a lattice width direction and $P \subseteq \mathbb{R} \times [0, \infty)$. Let $m \in \mathbb{Z}$ be minimal such that $P \subseteq \mathbb{R} \times [0, m+1]$. By assumption, we have $m \geq 2$. If $m \geq 3$, [14, Proposition 3.5] implies

$$\operatorname{Area}_k(P) \leq 2k^2\left(2i + 2 + \frac{1}{2}\right) = k^2(4i+5)$$

and we are done. Consider $m = 2$ and let $0 < a, b \leq 1$ be minimal such that $P \subseteq \mathbb{R} \times [1-a, 2+b]$. If $a + b \geq 1$, [14, Proposition 3.3] implies again $\operatorname{Area}_k(P) \leq 4k^2(i+1)$. Otherwise, it gives

$$\operatorname{Area}_k(P) \leq \frac{(a+b+1)^2}{a+b}(i+1)k^2,$$

which attains its maximum for $a = b = \frac{1}{k}$, hence we get $\operatorname{Area}_k(P) \leq \frac{1}{2}k(k+2)^2(i+1)$. □

Proposition 1.6.2. *Let P be a k-maximal polygon with $\operatorname{denom}(P) = k \geq 2$ and $i := i(P)$ that cannot be realized in $\mathbb{R} \times [-1, 1]$. Then there exists a lattice polygon Q such that $P = \frac{1}{k}Q^{(-1)}$, where*

$$l(Q) \ \leq\ \frac{1}{2}\operatorname{Area}_k(P) - \frac{1}{2} \ \leq\ \max\left(k^2(2i+\frac{5}{2}),\ \frac{1}{4}k(k+2)^2(i+1)\right) - \frac{1}{2}.$$

Proof. Since P is k-maximal, the lattice polygon kP is 1-maximal by Proposition 1.2.16. Set $Q := (kP)^{(1)}$. Since P cannot be realized in $\mathbb{R} \times [-1, 1]$ and $k \geq 2$, it follows that Q is two-dimensional. Thus Proposition 1.2.18 implies $P = \frac{1}{k}Q^{(-1)}$. Since $l(Q) = i(kP)$, the bound for $l(Q)$ follows with Pick's formula and then Lemma 1.6.1. □

This proposition tells us that if we know all lattice polygons with sufficiently many lattice points, we can use them to classify k-maximal polygons with $i \in \mathbb{Z}_{\geq 0}$ interior lattice points (those in $\mathbb{R} \times [-1, 1]$ can be classified directly using Algorithm 1.3.4). A classification algorithm for lattice polygons by number of lattice points was given by Koelman [38, Section 4.4]. How far we would have to know this classification for is given by the bound in Proposition 1.6.2, i.e.

$$l_{\max}(k, i) \ :=\ \left\lfloor \max\left(k^2(2i+\frac{5}{2}),\ \frac{1}{4}k(k+2)^2(i+1)\right) - \frac{1}{2}\right\rfloor.$$

Below are some values for $l_{\max}(k, i)$, where k runs vertically and i horizontally. Note however that this bound is not sharp and can likely be improved.

$l_{\max}(k,i)$	0	1	2	3	4	5	6	7	8
2	9	17	25	33	41	49	57	65	73
3	22	40	58	76	94	112	130	149	168
4	39	71	107	143	179	215	251	287	323
5	62	122	183	244	305	367	428	489	550
6	95	191	287	383	479	575	671	767	863
7	141	283	424	566	708	850	991	1133	1275

We have implemented Koelman's classification algorithm in `RationalPolygons.jl`, see A.4.7. In particular, we were able to reproduce his results up to $l = 42$ and extend them up to $l = 112$. In total, we have found 115 449 011 813 lattice polygons with at most 112 lattice points, 79 140 459 of which are internal, i.e. of the form $P^{(1)}$. We made the internal ones as well as all polygons up to 70 lattice points available at [59], see also [43, A371917] for their numbers. Using Proposition 1.6.2, we then obtained the classification of 3-maximal polygons with up to five and 4-maximal polygons with up to two interior lattice points, see Theorem 1.1.1. While this approach works in principle for all pairs (k,i), it is very inefficient compared to our direct classifications for $i \in \{0,1\}$ in Sections 1.4 and 1.5 as well as $k = 2$ in Section 1.8. However, it did prove useful as an independent verification of our results in these sections for low values of k and i respectively.

1.7 Proofs of Theorems 1.1.2, 1.1.3 and 1.1.4

Let us recall the classification of lattice polygons without interior lattice points.

Proposition 1.7.1 (See [38, Theorem 4.1.2])**.** *Let P be a lattice polygon with $i(P) = 0$. Then P is either equivalent to the twofold standard triangle* $\operatorname{conv}((0,0),(2,0),(0,-2))$ *or to a polygon of the form* $\operatorname{conv}((0,0),(0,-1),(n,-1),(m,0))$ *with integers $n \geq m \geq 0$.*

Proof of Theorem 1.1.2. Consider the integer hull $Q := \operatorname{conv}(P \cap \mathbb{Z}^2)$. We may assume Q to be two-dimensional, otherwise we have $b \leq 2$ and the claim holds trivially. If $i(Q) > 0$, then we get with Scott's inequality

$$b \;\leq\; b(Q) \;\leq\; 2i(Q) + 7 \;\leq\; 2i + 7 \;\leq\; (k+1)(i+1) + 3.$$

This can only be an equality for $(k,i) = (2,1)$ and Q being the threefold standard lattice triangle. But since P is not a lattice polygon, it must contain a non-integral vertex outside of Q. Since the threefold standard lattice triangle has a lattice point in the relative interior of each edge, this implies $b < b(Q)$, hence the inequality is strict after all.

If $i(Q) = 0$, Proposition 1.7.1 says that Q is either the twofold standard lattice triangle or of the form $\operatorname{conv}((0,0),(0,-1),(n,-1),(m,0))$ for integers $n \geq m \geq 0$. In the former case, we have $b \leq b(Q) = 6$ and the claim holds trivially. Let us treat the latter case. Since P has at least one interior lattice point, there must be a vertex $v = (x,y)$ of P which lies outside of the strip $\mathbb{R} \times [-1,0]$. If $y < -1$, we have $i \geq n - 1$, hence

$$b \;\leq\; m + 3 \;\leq\; n + 3 \;\leq\; i + 4 \;<\; (k+1)(i+1) + 3.$$

Now assume $y > 0$. Then we maximize the number of boundary lattice points if $(-1, l)$ are all boundary points of P for $0 \leq l \leq n$ and n is as large as possible. This happens when y is as small as possible i.e. $y = \frac{1}{k}$. By convexity, the largest possible n is $(k+1)(i+1)$. Then we obtain the triangle from the assertion and $b = n + 3$ as claimed. □

Remark 1.7.2. Plugging $k = 1$ into Theorem 1.1.2 gives $i \leq 2i + 5$, which is less than Scott's inequality. The difference is explained as follows: First, the threefold standard lattice triangle is an exception which only exists for $k = 1$. Second, if we take $k = 1$ in the triangle from Theorem 1.1.2, the vertex $(0, \frac{1}{k})$ is itself a lattice point, hence it has one more boundary lattice point than expected.

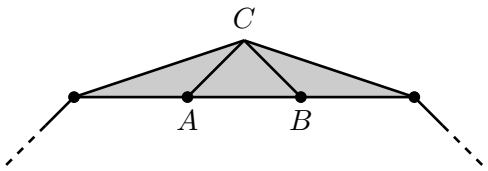
Proof of Theorem 1.1.3. We treat first the case $\dim(Q) = 2$. In terms of the k-normalized area, we have to show the following:

$$\text{Area}_k(P) \;\geq\; k^2(i + b - 2) + k(i + 1).$$

Pick's formula gives us a first bound

$$\text{Area}_k(P) \;\geq\; \text{Area}_k(Q) \;=\; k^2\left(2i(Q) + b(Q) - 2\right).$$

Consider an interior lattice point A of P which lies on the boundary of Q and let B be a boundary lattice point of Q adjacent to A. Pick a rational point $C \in (P \setminus Q) \cap \frac{1}{k}\mathbb{Z}^2$ lying over the edge of Q which contains A and B and consider the triangle with vertices A, B and C. Its base has length one and its height is at least $\frac{1}{k}$. Hence it has k-normalized area of at least k. Choosing the same C for as many pairs (A, B) as possible gives us at least $i - i(Q) + 1$ triangles of this form with disjoint interiors. We obtain $\text{Area}_k(P) - \text{Area}_k(Q) \;\geq\; k(i - i(Q) + 1)$.



Using $b(Q) = b + i - i(Q)$, we compute

$$\begin{aligned}\text{Area}_k(P) &\geq k^2(2i(Q) + b(Q) - 2) + k(i - i(Q) + 1)\\ &= k^2(i + b - 2) + k(i + 1) + i(Q)(k^2 - k)\\ &\geq k^2(i + b - 2) + k(i + 1).\end{aligned}$$

To get equality, we need $i(Q) = 0$ and $\text{Area}_k(P) - \text{Area}_k(Q) = k(i + 1)$. The latter happens if and only if there is a unique point $C \in (P \setminus Q) \cap \frac{1}{k}\mathbb{Z}^2$ having lattice distance $\frac{1}{k}$ to all relevant edges of Q. By Proposition 1.7.1, Q is either the twofold standard triangle or of the form $\text{conv}((0,0),(0,-1),(n,-1),(m,0))$ for $n \geq m \geq 0$. In the former case, we get the polygons (1b) and (2b) from the assertion. In the latter case, we get the polygons (0a), (1a) and (2a). Here, the number before the letter denotes the number of boundary lattice points of P on $\mathbb{R} \times \{0\}$,

which is at most two. Note that by symmetry, for the polygons (2a) it is enough to consider $x = 0, \dots, \lfloor \frac{b_{\max}-b}{2} \rfloor$. Similarly, for (2b), $x' = 0, \dots, k$ is enough. See also Figure 1.2 for an illustration.

Now we consider $\dim(Q) < 2$, i.e. the lattice points of P are collinear. In particular, we have $b \leq 2$. After an affine unimodular transformation, we may assume $Q \subseteq \mathbb{R} \times \{0\}$. In terms of the k-normalized area, we have to show

$$\mathrm{Area}_k(P) \geq \begin{cases} 2k(i+1), & b = 2, \\ 2ki + 1, & b = 1, \\ 2k(i-1) + 3, & b = 0. \end{cases}$$

Note that by Pick's formula, we have

$$\mathrm{Area}_k(P) \;=\; \mathrm{Area}_1(kP) \;=\; 2i(kP) + b(kP) - 2. \tag{1.1}$$

We start with the case $b = 2$. Then kQ has exactly $k(i+1)-1$ lattice points in its relative interior, hence $i(kP) \geq k(i+1) - 1$. Moreover, kP must have at least one lattice point (a vertex) above and below of Q. Together with the two boundary lattice points on Q itself, this gives $b(kP) \geq 4$. Plugging this into Equation (1.1), we arrive at the desired bound and equality holds if and only if $i(kP) = k(i+1) - 1$ and $b(kP) = 4$. Up to affine unimodular equivalence, there are exactly $k(i+1) + 1$ ways to choose boundary points of kP without getting additional interior lattice points, which are exactly the polygons (2c) from the assertion.

For $b = 1$, we obtain with analogous reasoning $i(kP) \geq ki$ and $b(kP) \geq 3$. The bound again follows from Equation (1.1). To get equality, there is up to affine unimodular transformation only one possible choice of boundary points of kP, which gives us the polygon (1c) from the assertion.

For $b = 0$, we have $i(kP) \geq k(i-1)+1$ and $b(kP) \geq 3$, which yields the desired bound. Again, to get equality, there is only one possibility, namely the polygon (0c) from the assertion. See also Figure 1.1 for an illustration. □

Remark 1.7.3. In Theorem 1.1.3, the polygon (1a) for $b = 2$ plays a special role: While it is always an area minimizer with respect to polygons with two boundary lattice points having a two-dimensional integer hull, it is not in general an area minimizer with respect to all polygons with two boundary lattice points. This is because the bound for $\dim(Q) < 2$ and $b = 2$ (namely $\mathrm{area}(P) \geq \frac{i+1}{k}$) is in general smaller. The only exception is the case $(k, i) = (2, 1)$, where both bounds agree. This is the only case where (1a) for $b = 2$ truly is an area minimizer.

We turn to the upper bounds for the area from Theorem 1.1.4. Let $k \geq 2$ and $i \geq 1$. Set $b_{\max} := (k+1)(i+1) + 3$ and let $0 \leq b \leq b_{\max}$. Then we intend to prove that the k-normalized area of a rational polygon with denominator k having i interior and b boundary lattice points is not greater than

$$\mathrm{maxarea}(k, i, b) \;:=\; k(k+1)^2(i+1) - \begin{cases} \tilde{b}, & b \in \{b_{\max}, b_{\max} - 1\}, \\ (2 + k(\tilde{b} - 2)), & 1 \leq b \leq b_{\max} - 2, \\ (3 + k(\tilde{b} - 2)), & b = 0, \end{cases}$$

where $\tilde{b} := b_{\max} - b$.

Lemma 1.7.4. *Let P be a rational polygon with denominator $k \geq 4$ having i interior and b boundary lattice points. Assume P is not equivalent to a polygon in $\mathbb{R} \times [-1, 1]$. Then we have* $\mathrm{Area}_k(P) < \mathrm{maxarea}(k, i, b)$.

Proof. Using Lemma 1.6.1 and $k \geq 4$, we obtain

$$\mathrm{Area}_k(P) \;\leq\; \max\left(k^2(4i+5),\; \frac{1}{2}k(k+2)^2(i+1)\right) \;<\; k^2(k+1)(i+1) - k - 3.$$

Moreover, we have $k^2(k+1)(i+1) - k - 3 = \mathrm{maxarea}(k, i, 0) \leq \mathrm{maxarea}(k, i, b)$. Hence we arrive at the claim. □

Lemma 1.7.5. *Let P be a rational polygon of denominator $k \geq 4$ having i interior and b boundary lattice points. Assume P is not equivalent to a polygon in $\mathbb{R} \times [-1, \frac{1}{k}]$. Then* $\mathrm{Area}_k(P) < \mathrm{maxarea}(k, i, b)$.

Proof. By Lemma 1.7.4, we can assume $P \subseteq \mathbb{R} \times [-1, 1]$. Let $h \in \{2, \ldots, k\}$ be minimal such that $P \subseteq \mathbb{R} \times [-1, \frac{h}{k}]$. Then by [14, Proposition 3.1], we have

$$\mathrm{Area}_k(P) \leq k\left(\frac{k^2}{h} + 2k + h\right)(i+1).$$

For $k \geq 4$, this attains its maximum at $h = 2$. We obtain

$$\mathrm{Area}_k(P) \leq k\left(\frac{k^2}{2} + 2k + 2\right)(i+1) < k^2(k+1)(i+1) - k - 3 \leq \mathrm{maxarea}(k, i, b),$$

where we again used $k \geq 4$ to get the strict inequality in the middle. □

Proof of Theorem 1.1.4. By Lemma 1.7.5, we may assume $P \subseteq \mathbb{R} \times [-1, \frac{1}{k}]$. Throughout our proof, we use the following shorthand notation: For any $-1 \leq y \leq \frac{1}{k}$ as well as $-1 \leq y_1 < y_2 \leq \frac{1}{k}$, we write

$$P_y \;:=\; P \cap (\mathbb{R} \times \{y\}), \qquad P_{[y_1, y_2]} \;:=\; P \cap (\mathbb{R} \times [y_1, y_2]).$$

After a translation, we can assume $P_0 \subseteq [0, i+1] \times \{0\}$. Next, we apply a shearing to achieve that the leftmost vertex in $P_{1/k}$ is $(0, \frac{1}{k})$. Convexity then implies that $P_{[-1,0]}$ is contained in the convex hull of $(0,0), (0,-1), ((k+1)(i+1), -1)$ and $(i+1, 0)$. In particular, the line segment P_{-1} is contained in $[0, (k+1)(i+1)] \times \{-1\}$. Write b_0 and b_{-1} for the number of boundary point of P in P_0 and P_1 respectively. Then $b = b_0 + b_{-1}$ and $b_0 \in \{0, 1, 2\}$. The length of P_{-1} is restricted by the number of lattice points b_{-1} on it, i.e.

$$\mathrm{length}(P_{-1}) \;\leq\; \begin{cases} (k+1)(i+1), & b_{-1} = b_{\max} - 2 \\ (k+1)(i+1) - \frac{1}{k}, & b_{-1} = b_{\max} - 3 \\ b_{-1} + 1 - \frac{2}{k}, & 0 \leq b_{-1} \leq b_{\max} - 4. \end{cases} \tag{1.2}$$

Consider now the line segment $P_{-1+1/k}$, which is contained in $[0, k(i+1)] \times \{-1+1/k\}$. We obtain the following bound:

$$\text{length}(P_{-1+1/k}) \;\leq\; k(i+1) - \frac{2-b_0}{k+1}. \tag{1.3}$$

We consider the decomposition $P = P_{[-1,-1+1/k]} \cup P_{[-1+1/k,1/k]}$. Note that $\text{area}(P_{[-1+1/k,1/k]})$ is maximized if and only if (1.3) is an equality and $(0, \frac{1}{k})$ is the only vertex in $P_{1/k}$. This is because $P_{[-1+1/k,0]}$ cannot be larger anyway and using an additional vertex in $P_{1/k}$ would lose more area in $P_{[-1+1/k,0]}$ than we could gain in $P_{[0,1/k]}$. On the other hand, $P_{[-1,-1+1/k]}$ is just a trapezoid, hence its area is maximized if and only if (1.2) and (1.3) are both equalities. We now proceed by case distinction on $b_0 \in \{0,1,2\}$ and describe all polygons satisfying these conditions, which gives us all area maximizers.

We first treat $b_0 = 2$, which implies $b \geq 2$ and $b_{-1} = b-2$. Here, we achieve equality in (1.3) if we have edges connecting the vertex $(0, \frac{1}{k})$ to $(0, -1+\frac{1}{k})$ and $(k(i+1), -1+\frac{1}{k})$. For $b = b_{\max}$, there is only one possibility to achieve equality in (1.2), namely the polygon (2c). For $b = b_{\max}-1$, there are a priori two possibilities of maximizing the length of P_{-1}. However, they are equivalent by symmetry, hence there is only one polygon (2b). For $2 \leq b \leq b_{\max} - 2$, we obtain the family of polygons (2a), which arises from different translations of the line segment P_{-1}. A priori, we have $x = 0, \ldots, \tilde{b} - 2$. However, by symmetry, it is enough to consider $x = 0, \ldots, \lfloor \frac{\tilde{b}}{2} \rfloor - 1$. Computing the areas of (2a), (2b) and (2c), we arrive at the bound from the assertion.

Next, we treat $b_0 = 1$. By symmetry, we can assume that the unique boundary lattice point on P_0 is $(i+1, 0)$. Then we achieve equality in (1.3) if we have edges connecting the vertex $(0, \frac{1}{k})$ to $(\frac{1}{k}, -1)$ and $(k(i+1), -1+\frac{1}{k})$. By convexity, we can maximize $\text{length}(P_{-1})$ (i.e. achieve equality in (1.2)) only for $b \leq b_{\max} - 3$, which gives us the polygons (1a) from the assertion. Note that these have exactly the same area as the polygons (2a) for fixed $b \leq b_{\max} - 3$ and we cannot reach this bound for b larger and $\text{length}(P_{-1})$ not maximal. Moreover, since $(0, \frac{1}{k}-1)$ is no longer a vertex for (1a), there is no flexibility in moving P_{-1} around. Hence for each $1 \leq b \leq b_{\max} - 3$, we only get one polygon of type (1a).

Lastly, for $b_0 = 0$, simultaneous equality in (1.2) and (1.3) can only be reached for $b = b_{\max} - 4$, in which case connecting $(0, \frac{1}{k})$ to $(\frac{1}{k}, -1)$ and $(b+1-\frac{1}{k}, -1)$ gives the triangle (0a). It has the same area as the polygons corresponding polygons (1a) and (2a). This leaves open the case $b = 0$. Here, we cannot reach equality in (1.3), since this would fix P_{-1} and lead to $b > 0$. The best we can do in this case is to connect $(0, \frac{1}{k})$ to $(\frac{1}{k}, -1)$ and $(k(i+1) - \frac{1}{k}, \frac{1}{k} - 1)$. The line segment P_{-1} is then already uniquely determined, hence we arrive at the polygon (0b). It has the claimed area from the assertion, which is notably smaller than in the other cases by one unit of k-normalized area. □

Remark 1.7.6. We can compare our area bounds to Pick's formula by plugging $k = 1$ into Theorems 1.1.3 and 1.1.4. Ignoring the cases $b \leq 2$ (which cannot happen for lattice polygons), we arrive at $i + \frac{b}{2} - \frac{1}{2}$ for both the lower and upper bound. This is notably $\frac{1}{2}$ more than what Pick's formula says. The difference can be explained as follows: All of our area maximizers and minimizers described in Theorems 1.1.3 and 1.1.4 have a rational vertex at $y = \frac{1}{k}$. For $k = 1$, this would become a boundary lattice point, hence the polygon has one more boundary lattice

point than it would have for $k > 1$. Substituting $b \mapsto b + 1$ into Pick's formula, we recover the same expression $i + \frac{b}{2} - \frac{1}{2}$.

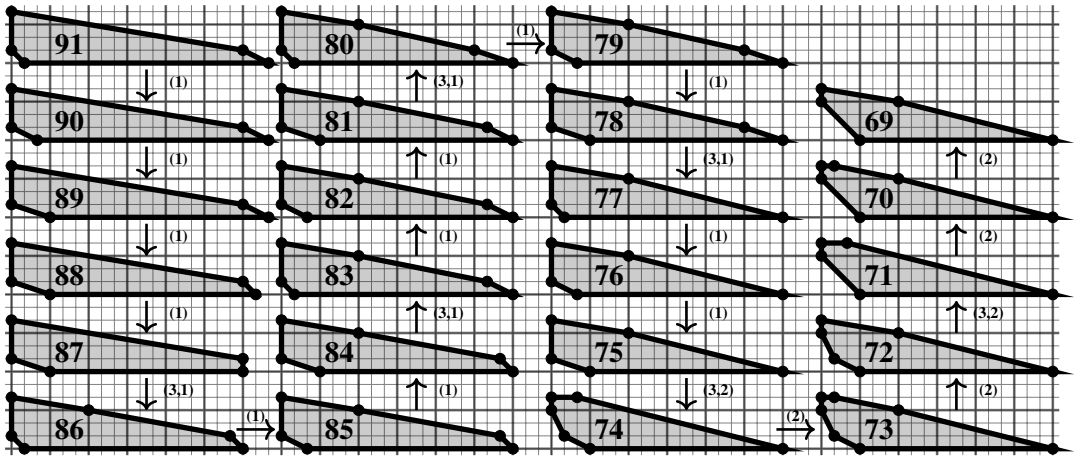
Remark 1.7.7. In the proof of Theorem 1.1.4, the condition $k \geq 4$ was only necessary to ensure $P \subseteq \mathbb{R} \times [-1, \frac{1}{k}]$ by applying the bounds from Lemmas 1.7.4 and 1.7.5. In particular, the area bound of Theorem 1.1.4 holds for all polygons in $\mathbb{R} \times [-1, \frac{1}{k}]$, even if $k \in \{2, 3\}$. However, as we will see in Section 1.8, not all area maximizers for $k = 2$ are contained in $\mathbb{R} \times [-1, \frac{1}{2}]$. For $k = 3$, we used our classification of rational polygons from Theorem 1.1.1 to verify that Theorem 1.1.4 holds completely for $1 \leq i \leq 5$. In particular, there are no maximizers outside of $\mathbb{R} \times [-1, \frac{1}{3}]$. This leads us to conjecture that the bounds from Lemmas 1.7.4 and 1.7.5 can be improved to work also for $k = 3$.

Remark 1.7.8. Let P be an area maximizer of type (2a) as described in Theorem 1.1.4 with at least three boundary lattice points. Taking $Q := \text{conv}(P \cap \mathbb{Z}^2)$, the polygon $P' := \text{conv}(Q \cup \{(0, \frac{1}{k})\})$ is then an area minimizer with the same number of interior and boundary lattice points as P. Moreover, we can find a polygon that attains any area between $\text{Area}_k(P)$ and $\text{Area}_k(P')$ by means of the following three operations:

1. Shorten or lengthen the line segment $P \cap \mathbb{R} \times \{-1\}$ by $\frac{l}{k}$ on either side for $1 \leq l \leq k - 1$. This lowers or increases $\text{Area}_k(P)$ by l.
2. Add or remove the vertex $(\frac{l}{k}, \frac{1}{k})$ for $1 \leq l \leq k - 1$. This increases or lowers $\text{Area}_k(P)$ by l.
3. Move in the vertices on $P \cap \mathbb{R} \times \{-1 + \frac{1}{k}\}$ by $\frac{1}{k}$. This lowers $\text{Area}_k(P)$ by k.

See Figure 1.16 for an example of how these operations can be used to attain any area between the lower and upper bounds.

Figure 1.16: A polygon for each possible area between the lower and upper bounds of Theorems 1.1.3 and 1.1.4. Here, we have $(k, i, b) = (3, 1, 8)$. Each polygon is labeled according to its k-normalized area and the steps between them are labeled according to Remark 1.7.8.



1.8 Half-integral polygons

In this section, we classify half-integral polygons with few interior lattice points. This will fill in the second row of our classification table from Theorem 1.1.1, which we know up to $i = 16$, see Classification 1.8.7. Next, we will prove Theorem 1.1.5 and apply this result to Ehrhart theory, where we also give a lower bound on the number of half-integral boundary points, see Proposition 1.8.10. Lastly, we raise a conjecture on the number of distinct Ehrhart quasipolynomials of half-integral polygons, see Conjecture 1.8.11.

Our classification method is inspired by Castryck's algorithm for classifying lattice polygons [19], where the notion of an *internal polygon* (i.e. a polygon of the form $P^{(1)}$) is replaced by the *Fine interior* as introduced in [21, §4.2.] (see also [49, Chapter II, Appendix to §4]). For more information on the Fine interior and its recent use in combinatorial mirror symmetry and combinatorial birational geometry, see [8, 10].

Recall that for a lattice polytope $P \subseteq \mathbb{R}^d$, we set $P^{(1)} := \operatorname{conv}(P^\circ \cap \mathbb{Z}^d)$. On the other hand $P^{(-1)}$ is the rational polytope obtained by "moving out" all facets of P by one, see Section 1.2.3.

Definition 1.8.1. The *Fine interior* of a rational polytope $P \subseteq \mathbb{R}^d$ is

$$F(P) := \{x \in \mathbb{R}^d \mid \langle x, n\rangle \geq \min\{\langle y, n\rangle \mid y \in P\} + 1 \text{ for all } n \in \mathbb{Z}^d \setminus \{0\}\}.$$

Remark 1.8.2. $F(P)$ is a rational polytope. Moreover, if P is a lattice polytope, we have $P^{(1)} \subseteq F(P)$ by [10, 2.6 f.]. For lattice polygons, we even have equality by [10, p. 2.9]. In this sense, a rational polytope that is a Fine interior of another one can be seen as a generalization of an internal polygon.

We have seen a criterion for the maximality of lattice polygons using internal polygons in Proposition 1.2.18. Here we focus first on the polytopes that are maximal with respect to inclusion among those sharing same Fine interior. The following is a direct consequence of the definition of the Fine interior.

Proposition 1.8.3. *A full-dimensional rational polytope P is the Fine interior of a rational polytope if and only if $F(P^{(-1)}) = P$. In this case $P^{(-1)}$ contains every rational polytope, which has P as its Fine interior.*

This result implies that we can classify all lattice polytopes with given number of interior lattice points by first classifying their Fine interiors, then getting all maximal ones by moving out the facets and finally computing all subpolytopes. Recently, Bohnert [15] gave a classification of two-dimensional Fine interiors of three-dimensional lattice polytopes, which allows us to follow this path with a small detour in dimension three. We give the essential results from [15] for our classification in three steps. The first one gives us a connection between dimension two and three.

Proposition 1.8.4. *The Fine interior of a three-dimensional lattice polytope $P \subseteq \mathbb{R}^2 \times [-1, 1]$ with at least two interior lattice points and $\operatorname{lw}(P) = 2$ has vertices in $\frac{1}{2}\mathbb{Z}^2 \times \{0\}$ and depends only on the half-integral polygon $P \cap (\mathbb{R}^2 \times \{0\})$. Moreover, the Fine interior of such a lattice polytope is of dimension one if and only if $P \cap (\mathbb{R}^2 \times \{0\})$ is affine unimodular equivalent to a polygon in $\mathbb{R} \times [-1, 1] \times \{0\}$.*

The second result gives a suitable notion of maximality in this context.

Proposition 1.8.5. *A half-integral polygon Q is the Fine interior of a three-dimensional lattice polytope if and only if there is a half-integral polygon $P_0 \subseteq \mathbb{R}^2$ such that Q is the Fine interior of $P := \mathrm{conv}(2P_0 \times \{-1\}, (0,0,1))$. Furthermore, there is a unique half-integral polygon Q^- with $F(\mathrm{conv}(2Q^- \times \{-1\}, (0,0,1))) = Q$, which contains every half-integral polygon P_0' for which*

$$F(\mathrm{conv}(2P_0' \times \{-1\}, (0,0,1))) = Q.$$

Finally, the third step is the classification, which was done there.

Classification 1.8.6 (See [15, Theorem 4.1])**.** *There are exactly* 24 324 158 *half-integral polygons with at most* 40 *lattice points, which are Fine interiors of a three-dimensional lattice polytope.*

For each polygon from Classification 1.8.6, we can compute the polygon Q^- from Proposition 1.8.5 and check for 2-maximality. This gives us all 2-maximal polygons with up to 40 interior lattice points that cannot be realized in $\mathbb{R} \times [-1,1]$. The latter ones can be obtained with Algorithm 1.3.4, hence we arrive at the following classification.

Classification 1.8.7 (Data available at [58])**.** *Below are the numbers $\#_{\max}(i)$ of 2-maximal polygons with i interior lattice points.*

i	1	2	3	4	5	6	7	8	9	10
$\#_{\max}(i)$	10	25	33	63	106	178	274	476	693	1 058
i	11	12	13	14	15	16	17	18	19	20
$\#_{\max}(i)$	1 561	2 345	3 431	4 970	6 712	9 345	12 883	17 667	23 809	31 806
i	21	22	23	24	25	26	27	28	29	30
$\#_{\max}(i)$	41 394	54 071	70 088	90 805	116 613	148 877	187 818	236 793	296 708	370 444
i	31	32	33	34	35	36	37	38	39	40
$\#_{\max}(i)$	459 667	569 098	698 986	857 540	1 047 830	1 277 349	1 549 049	1 875 058	2 257 565	2 714 004

Moreover, we obtain the numbers $\#_{\mathrm{all}}(i)$ of all 2-rational polygons as well as the numbers $\#_{\mathrm{ehr}}(i)$ of distinct Ehrhart quasipolynomials:

i	1	2	3	4	5	6
$\#_{\mathrm{all}}(i)$	5 145	48 639	249 540	893 402	2 798 638	7 299 589
$\#_{\mathrm{ehr}}(i)$	270	586	1 060	1 701	2 525	3 577
i	7	8	9	10	11	12
$\#_{\mathrm{all}}(i)$	17 556 059	41 005 529	83 848 960	170 081 198	339 219 561	617 457 338
$\#_{\mathrm{ehr}}(i)$	4 875	6 459	8 343	10 565	13 138	16 111
i	13	14	15	16		
$\#_{\mathrm{all}}(i)$	1 140 740 735	2 040 032 893	3 454 142 981	5 957 874 274		
$\#_{\mathrm{ehr}}(i)$	19 488	23 307	27 596	32 381		

We turn to the area bound of half-integral polygons given in Theorem 1.1.5. Note that it differs from the bound given in Theorem 1.1.4 exactly in the cases $i = 1$ and $b = 0$, where it exceeds the general bound by $\frac{1}{8}$ (= one unit of 2-normalized area). For $i = 1$, the half-integral polygons that violate the bound from Theorem 1.1.4 are all subpolygons of the threefold standard lattice triangle, see Figure 1.18. For $b = 0$, the polygons violating Theorem 1.1.4 are all realized in $\mathbb{R} \times [-1,1]$. This class of half-integral area maximizers also occurs for $b > 0$, but here, their area agrees with the general bound. The following lemma completely describes them.

Lemma 1.8.8 (See also Figure 1.17)**.** *Let $P \subseteq \mathbb{R} \times [-1,1]$ be a rational polygon of denominator two with i interior and b boundary lattice points, which cannot be realized in $\mathbb{R} \times [-1, \frac{1}{2}]$. Then we have*

$$b \leq 2i + 6, \qquad \mathrm{Area}_2(P) \leq 12i + 2b + 8.$$

Moreover, for $-1 \leq y \leq 1$, write ℓ_y for the length of the line segment $P \cap (\mathbb{R} \times \{y\})$ and b_y for the number of boundary lattice points on them. Then $\mathrm{Area}_2(P) = 12i + 2b + 8$ if and only if the following hold:

- *$(\ell_1, \ell_{-1}) = (b_1, b_{-1})$ and $\ell_0 = i + \frac{2}{3} + \frac{1}{6} b_0$,*
- *$P \cap (\mathbb{R} \times [-\frac{1}{2}, \frac{1}{2}])$ is a trapezoid, i.e. P has no vertex on $\mathbb{R} \times \{0\}$.*

Proof. We clearly have $b_1 \leq \ell_1 + 1$ and $b_{-1} \leq \ell_{-1} + 1$ as well as $b_0 \leq 2$. Moreover, $\ell_0 \leq i + 1$ holds and by convexity, we have $\ell_{-1} + \ell_1 \leq 2\ell_0$. We obtain

$$b \;=\; b_{-1} + b_0 + b_1 \;\leq\; \ell_{-1} + \ell_1 + 4 \;\leq\; 2\ell_0 + 4 \;\leq\; 2i + 6.$$

To estimate the area, consider the trapezoids $P_{-1} := P \cap (\mathbb{R} \times [-1, -\frac{1}{2}])$ and $P_1 := P \cap (\mathbb{R} \times [\frac{1}{2}, 1])$. Note that these are non-trivial, since we assume P cannot be realized in $\mathbb{R} \times [-1, \frac{1}{2}]$. Their area is given by

$$\mathrm{Area}_2(P_{-1}) \;=\; 2(\ell_{-1} + \ell_{-\frac{1}{2}}), \qquad \mathrm{Area}_2(P_1) \;=\; 2(\ell_{\frac{1}{2}} + \ell_1).$$

Moreover, the area of $P_0 := P \cap (\mathbb{R} \times [-\frac{1}{2}, \frac{1}{2}])$ is

$$\mathrm{Area}_2(P_0) \;=\; 2(\ell_{-\frac{1}{2}} + \ell_0) + 2(\ell_0 + \ell_{\frac{1}{2}}) \;=\; 4\ell_0 + 2(\ell_{-\frac{1}{2}} + \ell_{\frac{1}{2}}).$$

Since P has denominator two, we have $\ell_{-1} \leq b_{-1}$ and $\ell_1 \leq b_1$. Moreover, convexity forces $\ell_{-\frac{1}{2}} + \ell_{\frac{1}{2}} \leq 2\ell_0$. In total, we obtain

$$\mathrm{Area}_2(P) = \mathrm{Area}_2(P_{-1}) + \mathrm{Area}_2(P_0) + \mathrm{Area}_2(P_1) \leq 12\ell_0 + 2(b_{-1} + b_1).$$

Consider now the line segment $[s, s + \ell_0] \times \{0\} := P \cap (\mathbb{R} \times \{0\})$. Since $P \subseteq \mathbb{R} \times [-1, 1]$ and the denominator of P is two, the possible values for the rational part of s are $\{0, \frac{1}{6}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \frac{5}{6}\}$. This implies the bound $\ell_0 \leq i + \frac{2}{3} + \frac{1}{6} b_0$. Hence we obtain

$$\mathrm{Area}_2(P) \leq 12i + 8 + 2(b_0 + b_{-1} + b_1) = 12i + 2b + 8.$$

Moreover, equality holds if and only if the conditions from the assertion hold. Note that P_0 being a trapezoid is equivalent to $\ell_{-\frac{1}{2}} + \ell_{\frac{1}{2}} = 2\ell_0$. See also Figure 1.17 for an illustration of some area maximizers of the described form. □

In our proof of Theorem 1.1.5, we use the concept of lattice width, which we now quickly recall. For a polygon $P \subseteq \mathbb{R}^2$ and a non-zero vector $w \in \mathbb{R}^2$, we define the *width* of P in direction w as

$$\mathrm{width}_w(P) \;:=\; \max_{v \in P} \langle w, v \rangle - \min_{v \in P} \langle w, v \rangle.$$

The *lattice width* $\mathrm{lw}(P)$ of P is defined to be the minimum over $\mathrm{width}_w(P)$ where $w \in \mathbb{Z}^2 \setminus \{0\}$. By a *lattice width direction*, we mean a $w \in \mathbb{Z}^2 \setminus \{0\}$ such that $\mathrm{lw}(P) = \mathrm{width}_w(P)$.

Proof of Theorem 1.1.5. After an affine unimodular transformation, we can assume that $(0,1)$ is a lattice width direction of P. Let n be the number of interior integral horizontal lines, i.e.

$$n \;:=\; |\{j \in \mathbb{Z} \;|\; P^\circ \cap (\mathbb{R}\times\{j\}) \neq \emptyset\}|.$$

We first treat $n=1$, where we distinguish two subcases: Either P can be realized in $\mathbb{R}\times[-1,\frac{1}{2}]$ or it can be realized in $\mathbb{R}\times[-1,1]$ but not in $\mathbb{R}\times[-1,\frac{1}{2}]$. In the former case, the assertion follows from the proof of Theorem 1.1.4, see also Remark 1.7.7. In particular, we get sharpness for our bound if $b \geq 1$. In the latter case, we refer to Lemma 1.8.8, where we also obtain sharpness for $b=0$.

Now we consider $n \geq 2$. We may assume $P \subseteq \mathbb{R}\times[0,n+1]$. For $0 \leq y \leq n+1$, let ℓ_y be the length of the line segment $P \cap (\mathbb{R}\times\{y\})$. Consider the numbers $i_j := |P^\circ \cap (\mathbb{Z}\times\{j\})|$ for $j=1,\dots,n$ as well as $b_j := |\partial P \cap (\mathbb{Z}\times\{j\})|$ for $j=0,\dots,n+1$. Note that we have

$$\ell_0 \leq b_0, \qquad \ell_{n+1} \leq b_{n+1}, \qquad \ell_j \leq i_j + 1 \text{ for } j=1,\dots,n.$$

Consider the subpolygons $P_j := P \cap (\mathbb{R}\times[j-\frac{1}{2}, j+\frac{1}{2}])$ for $j=0,\dots,n+1$. For $j=1,\dots,n$, we can estimate their area by

$$\mathrm{Area}_2(P_j) = 4\ell_j + 2(\ell_{j-\frac{1}{2}} + \ell_{j+\frac{1}{2}}) \leq 8\ell_j \leq 8(i_j+1).$$

In particular, we obtain $\sum_{j=1}^{n} \mathrm{Area}_2(P_j) \leq 8i+8n$. Moreover, we have $\mathrm{Area}_2(P_0) = 2(\ell_0+\ell_{\frac{1}{2}})$ and $\mathrm{Area}_2(P_{n+1}) = 2(\ell_{n+\frac{1}{2}} + \ell_{n+1})$. By convexity, $\ell_{\frac{1}{2}} + \ell_{n+\frac{1}{2}} \leq \ell_k + \ell_{n-k+1}$ holds for all $k=1,\dots,\lfloor\frac{n}{2}\rfloor$. It follows

$$\mathrm{Area}_2(P_0) + \mathrm{Area}_2(P_{n+1}) \;\leq\; 2(b_0 + b_{n+1} + \ell_k + \ell_{n+k+1}) \;\leq\; 2(b + i_k + i_{n+k+1} + 2).$$

By choosing $k=1,\dots,\lfloor\frac{n}{2}\rfloor$ such that $i_k + i_{n+k+1}$ is minimal, we ensure $i_k + i_{n+k+1} \leq \frac{i}{\lfloor n/2\rfloor}$. In total, we obtain

$$\mathrm{Area}_2(P) \;=\; \sum_{j=0}^{n+1} \mathrm{Area}_2(P_j) \;\leq\; 8i + 8n + 2b + \frac{2i}{\lfloor n/2\rfloor} + 4.$$

For $n=2$, this gives $\mathrm{Area}_2(P) \leq 10i + 2b + 20$, which strictly less than $12i+2b+6$ for $i \geq 8$. Using Classification 1.8.7, we verified that the claim holds for the finitely many polygons with $i \leq 7$. In particular, we found that the only maximizers with $n=2$ are the finitely many polygons with $i \in \{1,2\}$ drawn in Figure 1.18. For $i=1$ and $b \leq 6$, their area even exceeds the generic bound, which is the reason we have to list $i=1$ separately in the assertion.

For $n \geq 3$, we use the estimate $\lfloor n/2\rfloor \geq \frac{n-1}{2}$. It is then enough to show

$$8i + 8n + \frac{4i}{n-1} + 2b + 4 \;<\; 12i + 2b + 6,$$

which is equivalent to

$$f(n) \;:=\; \frac{4n^2 - 5n + 1}{2n-4} \;<\; i.$$

By [14, Proposition 2.3], we get estimates $i_j \geq j-1$ and $i_{n+1-j} \geq j-1$ for $j = 1, \ldots, \lceil \frac{n}{2} \rceil$. Taking the sum, we obtain $i \geq \frac{n^2-2n}{4}$. For $n \geq 11$, this implies the claim. On the other hand, for $n \leq 7$, we have $f(n) < 17$. Again by our classification, we know the claim holds for $i \leq 16$, hence this case is done. For the remaining cases $n \in \{8, 9, 10\}$, we used our classification of maximal half-integral polygons with up to 40 interior lattice points to compute their maximally attained lattice width $\mathrm{lw}_{\max}$. We obtained the following table:

i	1	2	3	4	5	6	7	8	9	10
$\mathrm{lw}_{\max}$	3	3	4	4	4	5	5	5	11/2	6
i	11	12	13	14	15	16	17	18	19	20
$\mathrm{lw}_{\max}$	11/2	6	6	6	7	7	7	15/2	8	15/2
i	21	22	23	24	25	26	27	28	29	30
$\mathrm{lw}_{\max}$	8	8	8	8	17/2	9	17/2	9	9	19/2
i	31	32	33	34	35	36	37	38	39	40
$\mathrm{lw}_{\max}$	10	19/2	19/2	19/2	10	10	10	10	10	11

We now treat $n = 8$. We have $f(8) < 19$, hence it suffices to show $i \geq 19$. Assume $i \leq 18$, then the table above implies $n \leq \mathrm{lw}(P) \leq \frac{15}{2} < 8$, a contradiction. The cases $n \in \{9, 10\}$ are settled analogously. □

Remark 1.8.9. For half-integral polygons with at least three interior integral horizontal lines, the proof of Theorem 1.1.5 shows that the area bound is strict. This means that there are no half-integral area maximizers with $n \geq 3$. Hence the half-integral area maximizers are precisely the following:

- All maximizers in $\mathbb{R} \times [-1, \frac{1}{2}]$, which are described in Theorem 1.1.4,
- maximizers in $\mathbb{R} \times [-1, 1]$ as described by Lemma 1.8.8 and
- the finitely many area maximizers with $n = 2$ for $i \in \{1, 2\}$, which are drawn in Figure 1.18.

In the rest of this section, we show how our area bounds can be applied to Ehrhart theory of half-integral polygons. Recall that a rational polygon $P \subseteq \mathbb{R}^2$ admits an *Ehrhart quasipolynomial*, which counts lattice points in its integral dilations, i.e.

$$\mathrm{ehr}_P(t) := |tP \cap \mathbb{Z}^2| = \mathrm{area}(P)t^2 + c_1(t)t + c_2(t), \qquad \text{for } t \in \mathbb{Z}_{\geq 1},$$

where $c_1, c_2 \colon \mathbb{Z} \to \mathbb{Q}$ are uniquely determined periodic functions whose period divides the denominator of P. For polygons of denominator two, the four coefficients $c_1(1), c_1(2), c_2(1)$ and $c_2(2)$ can be described in terms of the numbers i and b of interior and boundary lattice points, the area and the number $b(2P) = |\partial P \cap \frac{1}{2}\mathbb{Z}^2|$ of half-integral boundary points:

$$c_1(1) = \frac{b}{2}, \qquad c_2(1) = i + \frac{b}{2} - \mathrm{area}(P), \qquad c_1(2) = \frac{b(2P)}{4}, \qquad c_2(2) = 1.$$

We refer to [12] for more information on Ehrhart theory in general. The half-integral case was already studied in [34] with focus on the coefficients $c_1(1)$ and $c_2(1)$ for not necessarily convex polygons. Moreover, the cases with period collapse, i.e. $c_1(1) = c_1(2)$ and $c_2(1) = c_2(2)$, were studied in [15]. Restrictions on the coefficients of Ehrhart quasipolynomials of half-integral polygons, in particular for the case $i = 0$, were also given in [25].

Using our results, we contribute to the description of Ehrhart quasipolynomials for convex half-integral polygons with at least one interior lattice point. Note that by the formulas for the coefficients above, the quadruple $(i, b, \text{area}(P), b(2P))$ completely determines the Ehrhart quasipolynomial. To understand which Ehrhart quasipolynomials are possible, we look for sharp bounds between these four parameters: By Theorem 1.1.2, we have the sharp bound $0 \leq b \leq 3i + 6$ and clearly every intermediate value is attained. Theorems 1.1.3 and 1.1.5 give sharp lower and upper bounds for the area in terms of i and b, while Remark 1.7.8 shows that all intermediate values are attained if $b \geq 3$. It remains to bound $b(2P)$ in terms of the first three parameters. We succeeded in proving the following sharp lower bound:

Proposition 1.8.10. *Let P be a half integral polygon with $i \geq 2$ interior lattice points and $b \geq 3$ boundary lattice points. Set $A := \text{Area}_2(P)$. Then the number of half-integral boundary points of P is bounded sharply as follows:*

$$b(2P) \;\geq\; 2b + r + \begin{cases} 2, & \text{if } A = 12i + 2b + 8 \text{ and } b \neq 2i + 4 \\ 0, & \text{otherwise} \end{cases},$$

where r is 1 if A is odd and 0 if it is even.

Proof. Since there is always a half-integral point between two adjacent boundary lattice points, we get $b(2P) \geq 2b$. By Pick's formula, we have

$$A = \text{Area}_1(2P) = 2i(2P) + b(2P) + 2.$$

This implies that $b(2P) - A$ is an even number, hence $b(2P) \geq 2b + r$. We now argue that this is a sharp inequality for $A \neq 12i + 2b + 8$ by giving examples. Note that by Theorem 1.1.3, we have $A \geq 6i + 4b - 6$. Consider $\tilde{A} := 12i + 2b - 7 - A$. Then for $A \neq 6i + 4b - 5$, the convex hull of

$$(0, 1/2), (0, -1), (b - 3 + r/2, -1), \left(2i + 2 - \frac{1}{2}\lfloor \tilde{A}/2 \rfloor, -1/2\right), (i + 1, 0)$$

satisfies $b(2P) = 2b + r$. For $A = 6i + 4b - 5$, the convex hull of

$$(0, 1/2), (1/2, 1/2), (i + 1, 0), (b - 3, -1), (0, -1)$$

works. Moreover, for $A = 12i + 2b + 8$ and $b = 2i + 4$, we can realize $b(2P) = 2b + r$ by the following triangle:

$$\text{conv}((-1/2, -1), (2i + 3/2, -1), (1/2, 1)).$$

It remains to show that for $A = 12i + 2b + 8$ and $b \neq 2i + 4$, we have the sharp inequality $b(2P) \geq 2b + 2$ (note that $r = 0$ in this case). By Remark 1.8.9, there are three kinds of half-integral area maximizers: Those in $\mathbb{R} \times [-1, \frac{1}{2}]$ are described by Theorem 1.1.4 and we can check

that $b(2P) = 2b+2$ holds for all of them. The same is true for the finitely many area maximizers with two interior integral lines and $i = 2$ drawn in Figure 1.18. The remaining ones are described by Lemma 1.8.8. Here, $b \neq 2i+4$ implies that we have at least one vertex with second coordinate $\pm\frac{1}{2}$. Hence there is at least one pair of adjacent non-integral vertices with no lattice point between them, which implies $b(2P) \geq 2b + 2$. □

It remains to bound $b(2P)$ from above in terms of i, b and area(P) and to understand which intermediate values are attained. Together with our other bounds, this would allow the complete description of the four-parameter family $(i, b, \text{area}(P), b(2P))$ and hence all possible Ehrhart quasipolynomials of half-integral polygons having $i \geq 1$. In particular, this should provide a path to prove that the number of distinct Ehrhart quasipolynomial of half-integral polygons is given by the following polynomial, which we have verified by our classification for $i \leq 16$:

Conjecture 1.8.11. *There are exactly*

$$\frac{9}{2}i^3 + 36i^2 + \frac{175}{2}i + 53$$

Ehrhart quasi-polynomials of half-integral polygons with $i \in \mathbb{Z}_{\geq 2}$ interior lattice points and at least 3 boundary lattice points.

Figure 1.17: All half-integral area maximizers in $\mathbb{R} \times [-1, 1]$ with two interior lattice points and five boundary lattice points, which cannot be realized in $\mathbb{R} \times [-1, \frac{1}{2}]$, as described by Lemma 1.8.8. The upper two rows of polygons have $b_0 = 2$, the polygons in the lower two rows have $b_0 = 1$

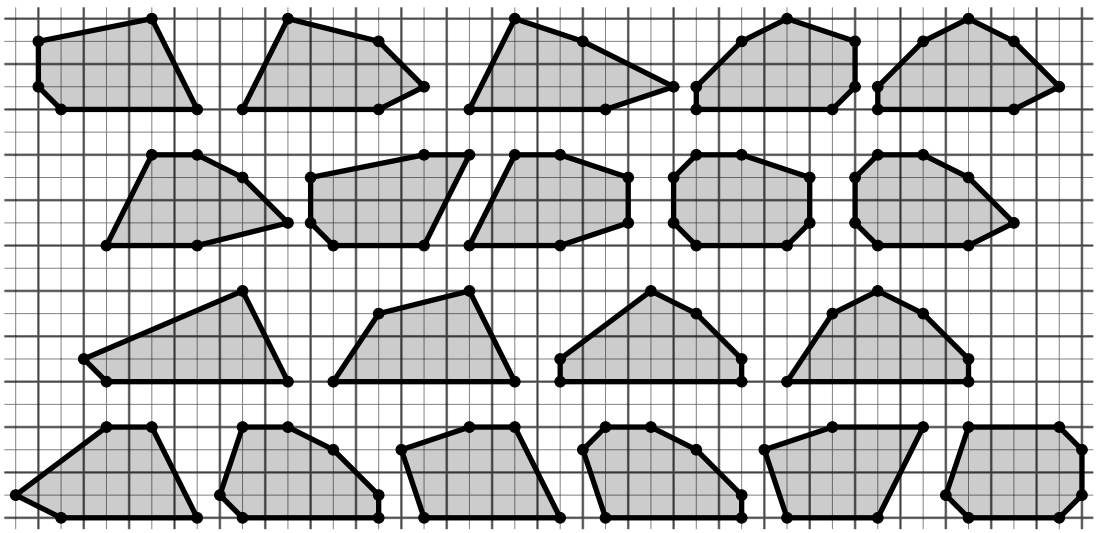
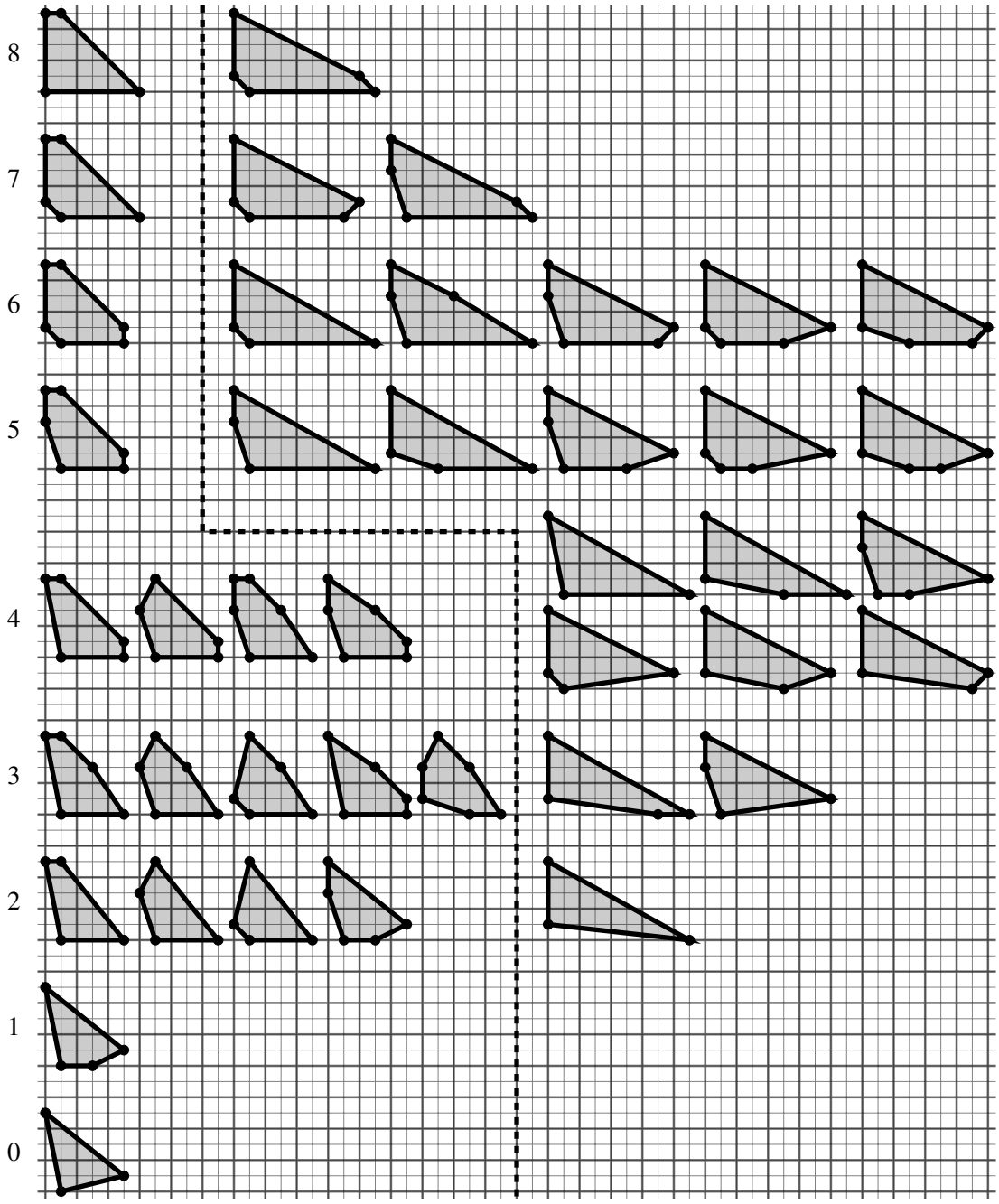


Figure 1.18: All half-integral area maximizers with two interior integral lines. On the left is the number b of boundary lattice points. The dashed line separates polygons with $i = 1$ from polygons with $i = 2$.



CHAPTER 2

Toric del Pezzo surfaces

2.1 Background on toric geometry

We start by briefly recalling some general background on toric geometry, in particular discussing Cox's quotient construction. We refer to the standard introductory texts on toric geometry for more details, for instance [20, 22], see also [2, Chapter 2]. In Subsection 2.1.1, which also contains new results, we will consider the Picard group of a toric variety in more detail.

A *torus* is an algebraic group isomorphic to a standard torus $\mathbb{T}^n := (\mathbb{C}^*)^n$. By a *toric variety*, we mean a normal variety X containing a torus T as an open dense subset, such that the action by multiplication $T \times T \to T$ extends to a morphical action $T \times X \to X$. A *toric morphism* between toric varieties $T \subseteq X$ and $T' \subseteq X'$ is a morphism $X \to X'$ that restricts to a homomorphisms of tori $T \to T'$.

We recall the correspondence between affine toric varieties and lattice cones. A *lattice* is a free abelian group of finite rank. For a lattice N, we write $N_{\mathbb{Q}} := N \otimes_{\mathbb{Z}} \mathbb{Q}$ for the associated rational vector space. By a *cone in* N, we mean a convex polyhedral cone $\sigma \subseteq N_{\mathbb{Q}}$. Writing $M := N^*$ for the dual lattice and $\sigma^\vee \subseteq M_{\mathbb{Q}}$ for the dual cone, a cone σ in N defines a $\mathbb{C}$-algebra

$$\mathbb{C}[\sigma^\vee \cap M] := \bigoplus_{u \in \sigma^\vee \cap M} \mathbb{C}\,\chi^u, \qquad \chi^u \cdot \chi^{u'} := \chi^{u+u'}.$$

We obtain an affine toric variety $Z_\sigma := \operatorname{Spec} \mathbb{C}[\sigma^\vee \cap M]$ with acting torus $T := \operatorname{Spec} \mathbb{C}[M]$, where the torus action is given by its comorphism

$$\mathbb{C}[\sigma^\vee \cap M] \to \mathbb{C}[\sigma^\vee \cap M] \otimes \mathbb{C}[M], \qquad \chi^u \mapsto \chi^u \otimes \chi^u.$$

Note that have we have the identifications $\mathcal{O}(T) = \mathbb{Z}[M] \cong \mathbb{Z}[T_1^{\pm 1}, \ldots, T_r^{\pm 1}]$ and $\mathcal{O}(Z_\sigma) = \mathbb{C}[\sigma^\vee \cap M]$, where $r = \operatorname{rank}(N)$. To any face $\tau \preceq \sigma$, one associates a *distinguished point* $z_\tau \in Z_\sigma$, which is determined by

$$\chi^u(z_\tau) = \begin{cases} 1, & u \in \tau^\perp \cap \sigma^\vee, \\ 0, & u \in \sigma^\vee \backslash \tau^\perp. \end{cases}$$

The *rays* $\rho_1, \ldots, \rho_n$ of σ (i.e. its one-dimensional faces) correspond to the T-invariant prime divisors via $D_i := \overline{T \cdot z_{\rho_i}}$.

We turn to the correspondence between fans and toric varieties. A *fan* in N is a collection of pointed convex polyhedral cones in N such that for all $\sigma \in \Sigma$, each face of σ is also in Σ, and

for all $\sigma, \sigma' \in \Sigma$, the intersection $\sigma \cap \sigma'$ is a face of each. We write $\Sigma_{\max}$ for the set of maximal cones. A *map of fans* $(\Sigma, N) \to (\Sigma', N')$ is a lattice map $\varphi\colon N \to N'$ such that for all $\sigma \in \Sigma$, there exists a $\sigma' \in \Sigma'$ such that $\varphi(\sigma) \subseteq \sigma'$. By appropriately gluing the affine toric varieties Z_σ along the open toric subvarieties $Z_{\sigma\cap\sigma'} \subseteq Z_\sigma, Z_{\sigma'}$, a fan gives rise to a toric variety Z_Σ and this construction is functorial. We will mostly work with *non-degenerate* fans, which means that the support of Σ is not contained in a proper linear subspace of $N_\mathbb{Q}$. On the geometric side, this means the toric variety Z_Σ has no torus factor, or equivalently, $\mathcal{O}(Z_\Sigma)^* = \mathbb{C}^*$. As in the affine case, the rays $\rho_1, \ldots, \rho_n$ of Σ correspond to the T-invariant prime divisors by $D_i := \overline{T \cdot z_{\rho_i}}$. Let $F := \mathbb{Z}^n$ and write $v_i \in N$ for the primitive generators of the rays $\rho_i \in \Sigma$. Note that Σ is non-degenerate if and only if the v_i span $N_\mathbb{Q}$ as a vector space. By the *generator map*, we mean

$$P\colon F \to N, \qquad f_i \mapsto v_i,$$

where $f_1, \ldots, f_n$ is the canonical basis of F. If $N = \mathbb{Z}^r$, we also speak of the *generator matrix* $P = \begin{bmatrix} v_1 & \ldots & v_n \end{bmatrix}$. The next proposition shows how the divisor class group (the group of Weil divisors modulo principal divisors) can be computed in terms of the generator map.

Proposition 2.1.1 (See [20, Theorem 4.1.3])**.** *Let Z_Σ be a toric variety associated to a non-degenerate fan Σ with generator map $P\colon F \to N$. Then we have an exact sequence.*

$$0 \longrightarrow M \xrightarrow{P^\top} E \xrightarrow{Q} \mathrm{Cl}(Z_\Sigma) \longrightarrow 0,$$

where $M := N^*$ and $E := F^* \cong \mathbb{Z}^n$ and $P^\top$ is the transpose of P.

We call $Q\colon E \to \mathrm{Cl}(Z_\Sigma)$ from Proposition 2.1.1 the *grading map*. It sends the i-th canonical basis vector e_i to $\omega_i := Q(e_i) = [D_i]$. Given a splitting $\mathrm{Cl}(Z_\Sigma) \cong \mathbb{Z}^\rho \times \mathrm{Cl}(Z_\Sigma)^{\mathrm{tors}}$, we also speak of the *grading matrix* $Q = \begin{bmatrix} \omega_1 & \ldots & \omega_n \end{bmatrix}$, where $\omega_i = (w_i, \eta_i)$, with $w_i \in \mathbb{Z}^\rho$ and $\eta_i \in \mathrm{Cl}(Z_\Sigma)^{\mathrm{tors}}$.

Proposition 2.1.2 (See [20, Theorem 8.2.3])**.** *For the canonical class of a toric variety, we have*

$$\mathcal{K}_{Z_\Sigma} = -(Q(e_1) + \cdots + Q(e_n))$$

Next, we discuss Cox's quotient construction for toric varieties. Recall that for a normal variety X with finitely generated divisor class group and $\mathcal{O}(X)^* = \mathbb{C}^*$, the *Cox ring* is the $\mathrm{Cl}(X)$-graded $\mathbb{C}$-algebra

$$\mathcal{R}(X) := \bigoplus_{[D]\in \mathrm{Cl}(X)} \Gamma(X, \mathcal{O}(D)).$$

Construction 2.1.3 (See [20, Sections 5.1, 5.2] and [2, Section 2.1.3])**.** Let $Z = Z_\Sigma$ be a toric variety associated to a non-degenerate fan Σ in N with generator map P and let $K := \mathrm{Cl}(Z)$. Then we have

$$\mathcal{R}(Z) \cong \mathbb{C}[T_1, \ldots, T_n],$$

where n is the number of rays of Σ. Moreover, the K-grading is given by $\deg(T_i) = Q(e_i) = \omega_i = [D_i]$. Geometrically, this means we have an action of the quasitorus $H := \operatorname{Spec} \mathbb{C}[K]$ on $\overline{Z} := \operatorname{Spec} \mathcal{R}(Z) = \mathbb{C}^n$. Consider the fan

$$\hat{\Sigma} := \{\hat{\sigma} \preceq \mathbb{Q}^n_{\geq 0} \mid P(\hat{\sigma}) \subseteq \sigma \text{ for some } \sigma \in \Sigma\}.$$

The associated toric variety $\hat{Z} = Z_{\hat{\Sigma}}$ is an open $\mathbb{T}^n$-invariant subvariety of $\overline{Z}$. Moreover, P defines a map of fans $(\hat{\Sigma}, \mathbb{Z}^n) \to (\Sigma, N)$ and hence a toric morphism $p\colon \hat{Z} \to Z$, which is a good quotient with respect to H. We arrive at the following diagram.

$$\begin{array}{ccccc} \mathbb{T}^n & \subseteq & \hat{Z} & \subseteq & \overline{Z}. \\ \downarrow p & & /\!/ H \downarrow p & & \\ T & \subseteq & Z & & \end{array}$$

2.1.1 The Picard group of a toric variety

In this subsection, we take a closer look at the Picard group of toric varieties, which yields in Corollary 2.1.7 a criterion for torsion-freeness and in Proposition 2.1.9 a first formula involving the Picard index and local class groups. This subsection has been published as [56, Section 2].

Recall that the Picard group of a normal variety X is the group of Cartier divisors modulo principal divisors. It sits naturally inside the class group, which is the group of Weil divisors modulo principal divisors. If the class group of is finitely generated and X is $\mathbb{Q}$-factorial (i.e. a multiple of every Weil divisor is Cartier), we can define the *Picard index* as

$$\iota_{\mathrm{Pic}}(X) \;:=\; [\mathrm{Cl}(X) : \mathrm{Pic}(X)].$$

A toric variety Z_Σ is $\mathbb{Q}$-factorial if and only if all cones in Σ are simplicial.

Construction 2.1.4. Let $Z = Z_\Sigma$ be a toric variety coming from a non-degenerate a fan Σ in the lattice $N := \mathbb{Z}^r$. Let $P\colon F \to N$ be the generator map, where $F := \mathbb{Z}^n$ and n is the number of rays of Σ. To any maximal cone $\sigma = \mathrm{cone}(v_{i_1}, \ldots, v_{i_{n_\sigma}}) \in \Sigma_{\max}$, we associate the lattices

$$N_\sigma \;:=\; \mathrm{lin}_{\mathbb{Q}}(\sigma) \cap N, \qquad\qquad F_\sigma \;:=\; \mathbb{Z}^{n_\sigma}\,.$$

We define the *local generator map* associated to σ by

$$P_\sigma\colon F_\sigma \to N_\sigma, \qquad f_j \mapsto v_{i_j}.$$

With the inclusion $\alpha_\sigma\colon N_\sigma \hookrightarrow N$ and the map $\beta_\sigma\colon F_\sigma \to F$ sending f_j to f_{i_j}, we obtain a commutative diagram

$$\begin{array}{ccc} F & \xrightarrow{P} & N \\ {\scriptstyle\beta_\sigma}\uparrow & & \uparrow{\scriptstyle\alpha_\sigma} \\ F_\sigma & \xrightarrow{P_\sigma} & N_\sigma. \end{array}$$

Consider the dual lattices

$$M := N^*, \qquad E := F^*, \qquad M_\sigma := N_\sigma^*, \qquad E_\sigma := F_\sigma^*.$$

Setting $K := E/\operatorname{im}(P^\top)$ and $K_\sigma := E_\sigma/\operatorname{im}(P_\sigma^\top)$, we obtain a map $\pi_\sigma\colon K \to K_\sigma$ fitting into the commutative diagram with exact rows

$$\begin{array}{ccccccccc} 0 & \longrightarrow & M & \xrightarrow{P^\top} & E & \longrightarrow & K & \longrightarrow & 0 \\ & & \downarrow{\scriptstyle\alpha_\sigma^\top} & & \downarrow{\scriptstyle\beta_\sigma^\top} & & \downarrow{\scriptstyle\pi_\sigma} & & \\ 0 & \longrightarrow & M_\sigma & \xrightarrow{P_\sigma^\top} & E_\sigma & \longrightarrow & K_\sigma & \longrightarrow & 0. \end{array}$$

By Proposition 2.1.1, we have isomorphisms $K \cong \mathrm{Cl}(Z)$ and $K_\sigma \cong \mathrm{Cl}(U_\sigma)$, where U_σ is the affine toric chart associated to σ. Moreover, the map π_σ corresponds to the restriction of divisor classes $[D] \mapsto [D|_{U_\sigma}]$. In particular, its kernel consists of those divisor classes that are principal on U_σ.

Construction 2.1.5. In the setting of Construction 2.1.4, we define lattices $\mathbf{N}$ and $\mathbf{F}$ and a lattice homomorphism $\mathbf{P}\colon\ \mathbf{N} \to \mathbf{F}$ by

$$\mathbf{N} := \bigoplus_{\sigma\in\Sigma_{\max}} N_\sigma, \qquad \mathbf{F} := \bigoplus_{\sigma\in\Sigma_{\max}} F_\sigma, \qquad \mathbf{P} := \bigoplus_{\sigma\in\Sigma_{\max}} P_\sigma.$$

Furthermore, we define lattice homomorphisms

$$\begin{aligned} \alpha\colon\ \mathbf{N} &\to N, & N_\sigma \ni v &\mapsto \alpha_\sigma(v),\\ \beta\colon\ \mathbf{F} &\to F, & F_\sigma \ni w &\mapsto \beta_\sigma(w). \end{aligned}$$

Let $\gamma\colon \hat{N} \to \mathbf{N}$ be a kernel of α and $\delta\colon \hat{F} \to \mathbf{F}$ be a kernel of β. We obtain an induced map $\hat{P}\colon \hat{F} \to \hat{N}$ making the following diagram commute:

$$\begin{array}{ccc} F & \xrightarrow{P} & N \\ {\scriptstyle\beta}\uparrow & & \uparrow{\scriptstyle\alpha} \\ \mathbf{F} & \xrightarrow{\mathbf{P}} & \mathbf{N} \\ {\scriptstyle\delta}\uparrow & & \uparrow{\scriptstyle\gamma} \\ \hat{F} & \xrightarrow{\hat{P}} & \hat{N} \end{array}$$

Now consider the dual lattices $\mathbf{M} := \mathbf{N}^*$ and $\mathbf{E} := \mathbf{F}^*$ as well as the abelian group $\mathbf{K} := \bigoplus_{\sigma\in\Sigma_{\max}} K_\sigma$. We define the map

$$\pi\colon K \to \mathbf{K}, \quad w \mapsto (\pi_\sigma(w))_{\sigma\in\Sigma_{\max}}.$$

Setting $\hat{M} := \mathbf{M}\,/\,\mathrm{im}(\alpha^\top)$ and $\hat{E} := \mathbf{E}\,/\,\mathrm{im}(\beta^\top)$ as well as $\hat{K} := \mathbf{K}\,/\,\mathrm{im}(\pi)$, we obtain a map $\hat{P}'\colon \hat{M} \to \hat{E}$ fitting into the following commutative diagram with exact rows:

$$\begin{array}{ccccccccc} 0 & \longrightarrow & M & \xrightarrow{P^\top} & E & \longrightarrow & K & \longrightarrow & 0 \\ & & \downarrow{\scriptstyle\alpha^\top} & & \downarrow{\scriptstyle\beta^\top} & & \downarrow{\scriptstyle\pi} & & \\ 0 & \longrightarrow & \mathbf{M} & \xrightarrow{\mathbf{P}^\top} & \mathbf{E} & \longrightarrow & \mathbf{K} & \longrightarrow & 0 \\ & & \downarrow & & \downarrow & & \downarrow & & \\ & & \hat{M} & \xrightarrow{\hat{P}'} & \hat{E} & \longrightarrow & \hat{K} & \longrightarrow & 0. \end{array}$$

Proposition 2.1.6. *In Construction 2.1.5, the map β is surjective and there is an exact sequence*

$$0 \longrightarrow \mathrm{Pic}(Z) \longrightarrow \hat{M} \xrightarrow{\hat{P}'} \hat{E} \longrightarrow \hat{K} \longrightarrow 0.$$

Moreover, if α is surjective, $\hat{M}$ is torsion-free and $\hat{P}' = \hat{P}^\top$.

Proof. Every primitive generator of a ray of Σ is a generator of some maximal cone. This implies that β is surjective, hence $\beta^\top$ is injective. As a subgroup of K, the Picard group consists of the Cartier divisor classes, i.e. those that are principal on all affine toric charts U_σ for $\sigma \in \Sigma_{\max}$. This means

$$\mathrm{Pic}(Z) = \bigcap_{\sigma \in \Sigma_{\max}} \ker(\pi_\sigma) = \ker(\pi).$$

Applying the snake lemma to the lower diagram of Construction 2.1.5, gives the exact sequence of the Proposition. The last statement is clear. □

Corollary 2.1.7. *Assume that in Construction 2.1.5, the map α is surjective. Then the Picard group* $\mathrm{Pic}(Z)$ *is torsion-free.*

Remark 2.1.8. Corollary 2.1.7 generalizes the well-known fact that if Z has a toric fixed point, its Picard group is torsion-free. Indeed, having a toric fixed point means having a cone $\sigma \in \Sigma$ of maximal dimension. This implies $N_\sigma = N$, hence α is surjective.

Proposition 2.1.9. *Let $Z = Z_\Sigma$ be a toric variety with a non-degenerate simplicial fan Σ. In the notation of Construction 2.1.5, we have*

$$\iota_{\mathrm{Pic}}(Z) \;=\; \frac{1}{|\hat{K}|} \prod_{\sigma \in \Sigma_{\max}} |K_\sigma|.$$

Proof. Recall that $\mathrm{Pic}(Z) = \ker(\pi)$ and $\hat{K} = \mathbf{K} \,/\, \mathrm{im}(\pi)$. Since Σ is simplicial, each K_σ is finite, hence so is $\mathbf{K}$. We obtain

$$\iota_{\mathrm{Pic}}(Z) = [K : \ker(\pi)] = |\,\mathrm{im}(\pi)| = \frac{|\,\mathbf{K}\,|}{|\hat{K}|} = \frac{1}{|\hat{K}|} \prod_{\sigma \in \Sigma_{\max}} |K_\sigma|.$$

□

2.2 LDP polygons and toric log del Pezzo surfaces

In this section, we discuss properties of LDP polygons and their relations to the geometry of the associated toric del Pezzo surfaces. In particular, we discuss class group, Picard group, Picard index and Gorenstein index, log canonicity, resolution of singularities as well as intersection theory. We refer to [20, Chapter 10] for more background. Here, we place special emphasis on explicit formulas and algorithms. Moreover, we provide references to our implementation in `RationalPolygons.jl` throughout the section.

Recall that a toric variety Z_Σ associated to a fan Σ is Fano if and only if its primitive ray generators form the vertices of a convex polytope. Conversely, if we start with a *Fano polytope*, that is a convex lattice polytope having only primitive vertices and the origin in its interior, its face fan defines toric Fano variety. Specializing to surfaces, two-dimensional Fano varieties are called *del*

Pezzo surfaces, while two-dimensional Fano polytopes are called *LDP polygons*. Hence we have a bijection on isomorphism classes:

$$\{\text{LDP polygons}\}/\sim^{\text{u}} \quad \overset{1:1}{\longleftrightarrow} \quad \{\text{toric del Pezzo surfaces}\}/\cong .$$

Recall from Section 1.2 that we write $\sim^{\text{u}}$ for *unimodular equivalence*, i.e. $P \sim^{\text{u}} P'$ iff $P' = UP$ holds for some unimodular $U \in \mathbb{Z}^{2\times 2}$.

When there is no confusion, we will denote the LDP polygon $P \subseteq \mathbb{R}^2$ and its vertex matrix $P = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}$ by the same symbol. Moreover, we will always consider vertex indices modulo n, e.g. $v_{n+1} = v_1$.

2.2.1 Class group and Picard group

Recall that in algebraic geometry, the divisor class group of a normal variety is the group of Weil divisors modulo linear equivalence, while the Picard group is Cartier divisors modulo linear equivalence. For toric varieties, both are finitely generated abelian groups that can be computed in terms of the defining lattice fan, see Proposition 2.1.1 and Subsection 2.1.1. In this subsection, we will consider class group, Picard group and related notions for toric log del Pezzo surfaces via their LDP polygons.

We start with a numerical invariant called the *multiplicity*, which will turn out to be the torsion order of the class group, see Proposition 2.2.5.

Definition 2.2.1 (See A.2.49, A.2.50)**.** Let $P = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}$ be an LDP polygon. The *multiplicity* of P is

$$\mu(P) \;:=\; [\mathbb{Z}^2 : \mathbb{Z}\, v_1 + \cdots + \mathbb{Z}\, v_n]$$

For $i = 1, \dots, n$, the *local multiplicities* are $\mu_i(P) := [\mathbb{Z}^2 : \mathbb{Z}\, v_i + \mathbb{Z}\, v_{i+1}]$.

The following elementary lemma about minors will be useful going forward:

Lemma 2.2.2. *Let* $v_i = (x_i, y_i) \in \mathbb{Z}^2$ *for* $i = 1, 2, 3$ *such that* $\gcd(x_1, y_1) = 1$ *and write* $\mu_{ij} := \det(v_i, v_j)$*. Then there exist integers* α *and* β *such that*

$$\mu_{23} = \alpha\mu_{12} + \beta\mu_{13}.$$

In particular, $\gcd(\mu_{12}, \mu_{13}, \mu_{23}) = \gcd(\mu_{12}, \mu_{13})$.

Proof. Since x_1 and y_1 are coprime, we find integers a and b with $ax_1 + by_1 = 1$. Setting $\alpha := ax_2 + by_2$ and $\beta := -ax_3 - by_3$, the rows of the following 3×3-matrix are linearly dependent

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & \alpha & -\beta \end{bmatrix}.$$

Its determinant is $\mu_{23} - \alpha\mu_{13} - \beta\mu_{12} = 0$, which proves the claim. □

Proposition 2.2.3. *For an LDP polygon* $P = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}$*, we have*

$$\mu_i(P) \;=\; \det(v_i, v_{i+1}), \qquad \mu(P) \;=\; \gcd(\mu_1(P), \dots, \mu_n(P)).$$

Proof. That $\mu_i(P) = \det(v_i, v_{i+1})$ is clear by definition. As for $\mu(P)$, it is the index of the sublattice spanned by the columns of P. This equals the index of the sublattice spanned by the columns of the Smith normal form of P, which is of the form

$$\begin{bmatrix} s_1 & 0 & 0 & \dots 0 \\ 0 & s_2 & 0 & \dots 0 \end{bmatrix},$$

where s_1 and s_2 are the first and second *invariant factors* of P. By elementary matrix theory (see for instance [42, Chapter II]), these are given as $s_1 = d_1$ and $s_2 = d_2/d_1$, where d_i is the greatest common divisor of minors of size $i = 1, 2$. In particular, we have $\mu(P) = s_1 s_2 = d_2$. Since all columns of P are primitive, we can apply Lemma 2.2.2 multiple times and see that it is enough to take the greatest common divisor of minors with adjacent column indices, hence $d_2 = \gcd(\mu_1(P), \dots, \mu_n(P))$. □

By Propositions 2.1.1 and 2.1.2, the following definitions of class group and canonical class of an LDP polygon coincide with the respective notions of the associated toric del Pezzo surface.

Definition 2.2.4 (See A.2.53)**.** Viewing an LDP polygon P as a map $P\colon\ \mathbb{Z}^n \to \mathbb{Z}^2$ sending e_i to the i-th vertex v_i, we define the *class group* as $\mathrm{Cl}(P) \;:=\; \mathbb{Z}^n \,/\, \mathrm{im}(P^\top)$. The *grading matrix* is the canonical quotient map $Q\colon\ \mathbb{Z}^n \to \mathrm{Cl}(P)$. Moreover, we define the *canonical class* as

$$\mathcal{K}_P \;:=\; -(Q(e_1) + \dots + Q(e_n)) \;\in\; \mathrm{Cl}(P).$$

Proposition 2.2.5. *For an LDP polygon* P*, we have* $\mathrm{Cl}(P) \;\cong\; \mathbb{Z}^{n-2} \times \mathbb{Z}\,/\mu(P)\,\mathbb{Z}$*.*

Proof. Let $\mu := \mu(P)$. Consider again the Smith normal form of P, which by the proof of Proposition 2.2.3 is

$$S := \begin{bmatrix} 1 & 0 & 0 & \dots 0 \\ 0 & \mu & 0 & \dots 0 \end{bmatrix}.$$

We obtain $\mathrm{Cl}(P) \cong \mathbb{Z}^n \,/\, \mathrm{im}(S^\top) \cong \mathbb{Z}^{n-2} \times \mathbb{Z}\,/\mu\,\mathbb{Z}$. □

Remark 2.2.6. Under the isomorphism from Proposition 2.2.5, we can view the grading matrix as $Q = \begin{bmatrix} \omega_1 & \dots & \omega_n \end{bmatrix}$, where $\omega_i = (w_i, \eta_i) \in \mathbb{Z}^{n-2} \times \mathbb{Z}\,/\mu(P)\,\mathbb{Z}$. In particular, the canonical class is $\mathcal{K}_P = -(\omega_1 + \dots + \omega_n)$.

Lemma 2.2.7. *Let* $P = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}$ *be an LDP polygon with grading matrix* $Q = \begin{bmatrix} \omega_1 & \dots & \omega_n \end{bmatrix}$*, where* $\omega_i \in \mathrm{Cl}(P)$*. Let* $I = \{i_1, \dots, i_k\} \subseteq \{1, \dots, n\}$ *and set* $N_I := \mathbb{Z}\,v_{i_1} + \dots + \mathbb{Z}\,v_{i_k}$ *and* $N_I^{\mathbb{Q}} := N_I \otimes \mathbb{Q} = \mathrm{span}_{\mathbb{Q}}(v_{i_1}, \dots, v_{i_k})$*. Then*

$$\mathrm{Cl}(P)/\langle \omega_i; i \notin I \rangle \quad\cong\quad \mathbb{Z}^{k - \dim(N_I^{\mathbb{Q}})} \oplus (N_I^{\mathbb{Q}} \cap \mathbb{Z}^2)/N_I.$$

Proof. [2, Lemma 2.1.4.1]. □

The following property of the grading matrix is also known as being *almost free*, see for instance [2, Definition 3.2.1.1].

Corollary 2.2.8. *Let $Q = \begin{bmatrix}\omega_1 & \dots & \omega_n\end{bmatrix}$ be the grading matrix of an LDP polygon. Then any $n-1$ of the ω_i generate $\mathrm{Cl}(P)$ as a group.*

Proof. Applying Lemma 2.2.7 to $I = \{i\}$, we see that $\mathrm{Cl}(P)/\langle w_j; j \neq i\rangle$ is isomorphic to $(\mathbb{Q}\, v_i \cap \mathbb{Z}^2)/\,\mathbb{Z}\, v_i$. But v_i is primitive, hence $\mathbb{Q}\, v_i \cap \mathbb{Z}^2 = \mathbb{Z}\, v_i$. Thus $\{w_j \ ; \ j \neq i\}$ generate $\mathrm{Cl}(P)$. □

The following should be compared to Construction 2.1.4 and the following discussion.

Construction 2.2.9. Let $P = \begin{bmatrix}v_1 & \dots & v_n\end{bmatrix}$ be an LDP polygon. Write $P_i := \begin{bmatrix}v_i & v_{i+1}\end{bmatrix} \in \mathbb{Z}^{2\times 2}$. Then the *local class groups* are $\mathrm{Cl}_i(P) := \mathbb{Z}^2 \,/\, \mathrm{im}(P_i^\top)$. Writing $Q_i\colon\ \mathbb{Z}^2 \to \mathrm{Cl}_i(P)$ for the quotient maps, we define surjections

$$\pi_i\colon\ \mathrm{Cl}(P) \to \mathrm{Cl}_i(P), \qquad \pi_i(Q(x)) := Q_i(x_i, x_{i+1}).$$

In other words, π_i fits into the following commutative diagram with exact rows

$$\begin{array}{ccccccccc} 0 & \longrightarrow & \mathbb{Z}^2 & \xrightarrow{P^\top} & \mathbb{Z}^n & \xrightarrow{Q} & \mathrm{Cl}(P) & \longrightarrow & 0 \\ & & \| & & \downarrow & & \downarrow{\scriptstyle \pi_i} & & \\ 0 & \longrightarrow & \mathbb{Z}^2 & \xrightarrow{P_i^\top} & \mathbb{Z}^2 & \xrightarrow{Q_i} & \mathrm{Cl}_i(P) & \longrightarrow & 0 \end{array}$$

Here, the map $\mathbb{Z}^n \to \mathbb{Z}^2$ sends e_i and e_{i+1} to e_1 and e_2 respectively, and everything else to zero. We define the *local Picard groups* as $\mathrm{Pic}_i(P) := \ker(\pi_i) = \langle \omega_j \ ; \ j \notin \{i, i+1\}\rangle$ and the *Picard group* as

$$\mathrm{Pic}(P) \ := \ \bigcap_{i=1}^{n} \mathrm{Pic}_i(P) \ \leq \ \mathrm{Cl}(P).$$

The *Picard index* is $\iota_{\mathrm{Pic}}(P) := [\mathrm{Cl}(P) : \mathrm{Pic}(P)]$.

Proposition 2.2.10. *Let $P = \begin{bmatrix}v_1 & \dots & v_n\end{bmatrix}$ be an LDP polygon. Then for $i = 1, \dots, n$, we have*

$$\mu_i(P) \ = \ |\,\mathrm{Cl}_i(P)| \ = \ \mu(P)\tilde{\mu}_i(P),$$

where $\tilde{\mu}_i(P) := |\det(w_j \ ; \ j \notin \{i, i+1\})|$ and $w_j \in \mathbb{Z}^{n-2}$ are the free parts of the columns of the grading matrix.

Proof. Recall that the local class group is defined to be the cokernel of $P_i^\top$. In particular, we have $|\,\mathrm{Cl}_i(P)| = \det(P_i^\top) = \det(P_i) = \mu_i(P)$. For the second equality, we treat exemplarily the case $i = n-1$. Then $\mathrm{Pic}_i(P) = \langle \overline{\omega}_1, \dots, \overline{\omega}_{n-2}\rangle$, where we write $\overline{\omega}_j = (w_j, \overline{\eta}_j) \in \mathrm{Cl}(P) = \mathbb{Z}^{n-2} \times \mathbb{Z}\,/\,\mu\,\mathbb{Z}$ for the columns of Q and $\mu := \mu(P)$. Note that $\mathrm{Cl}(P) \cong \mathbb{Z}^{n-1}\,/\,\mathbb{Z}\,\vec{\mu}$, where $\vec{\mu} := (0, \dots, 0, \mu) \in \mathbb{Z}^{n-1}$. By the third isomorphism theorem for abelian groups, we get

$$\mathrm{Cl}(P)/\,\mathrm{Pic}_i(P) \ \cong \ \mathbb{Z}^{n-1}\,/(\mathbb{Z}\,\omega_1 + \dots + \mathbb{Z}\,\omega_{n-2} + \mathbb{Z}\,\vec{\mu}).$$

In particular, we obtain

$$[\mathrm{Cl}(P) : \mathrm{Pic}_i(P)] \;=\; |\det(\omega_1, \ldots, \omega_{n-2}, \vec{\mu})| \;=\; \mu |\det(w_1, \ldots, w_{n-2})| \;=\; \mu\tilde{\mu}_i(P).$$

□

Theorem 2.2.11 (See A.2.55)**.** *The Picard index of an LDP polygon P is*

$$\iota_{\mathrm{Pic}}(P) \;=\; \frac{\mu_1(P)\ldots\mu_n(P)}{\mu(P)} \;=\; \mu(P)^{n-1}\tilde{\mu}_1(P)\ldots\tilde{\mu}_n(P),$$

where $\tilde{\mu}_i(P)$ is defined as in Proposition 2.2.10.

Theorem 2.2.11 is the combinatorial version of the Picard index formula from Theorem 7, specialized to toric surfaces. A proof will be given in Section 2.3.

2.2.2 Gorenstein index

Recall that the *Gorenstein index* of a normal variety X with canonical divisor $\mathcal{K}_X$ is the smallest positive integer ι such that $\iota\mathcal{K}_X$ is Cartier. The following definition of the Gorenstein index of an LDP polygon coincides with the Gorenstein index of its associated toric del Pezzo surface.

Definition 2.2.12 (See A.2.56)**.** The *Gorenstein index* $\iota(P)$ of an LDP polygon P is the smallest positive integer ι such that $\iota\,\mathcal{K}_P \in \mathrm{Pic}(P)$. For $i = 1, \ldots, n$, the *local Gorenstein index* $\iota_i(P)$ is the smallest positive integer ι_i such that $\iota_i\,\mathcal{K}_P \in \mathrm{Pic}_i(P)$.

Remark 2.2.13. The Gorenstein index of an LDP polygon is the order of the canonical class $\mathcal{K}_P$ as an element of the factor group $\mathrm{Cl}(P)/\mathrm{Pic}(P)$. In particular, by Lagrange's theorem, the Gorenstein index divides the Picard index.

Proposition 2.2.14. *Let $P = \begin{bmatrix} v_1 & \ldots & v_n \end{bmatrix}$ be an LDP polygon with vertices $v_i = (x_i, y_i) \in \mathbb{Z}^2$. Then we have*

$$\iota_i(P) = \frac{\mu_i(P)}{\gcd(x_{i+1} - x_i, y_{i+1} - y_i)}, \qquad \iota(P) = \mathrm{lcm}(\iota_1(P), \ldots, \iota_n(P)).$$

Proof. By definition, the i-th local Gorenstein index is the smallest positive integer ι_i such that $\iota_i Q_i(1,1) \;=\; 0$ in $\mathrm{Cl}_i(P)$, which is equivalent to $(\iota_i, \iota_i) \;\in\; \mathrm{im}(P_i^*)$. In other words, it is the smallest positive integer such that there exist a $u \;\in\; \mathbb{Z}^2$ with $\iota_i \;=\; \langle u, v_i\rangle \;=\; \langle u, v_{i+1}\rangle$. This minimum is attained by setting

$$u \;:=\; \frac{1}{\gcd(x_{i+1} - x_i, y_{i+1} - y_i)}(y_{i+1} - y_i, x_{i+1} - x_i).$$

In particular, we get the formula for ι_i from the assertion. Since the Picard group of an LDP polygon is defined as the intersection of the local Picard groups, the Gorenstein index equals the least common multiple of the local Gorenstein indices. □

Remark 2.2.15. Another definition of the Gorenstein index goes via the dual polytope P^*: It is the smallest positive integer ι such that ιP^* is a lattice polytope. For a lattice polygon $P = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}$, we can directly compute that the vertices of P^* are

$$v_i^* = \frac{1}{x_i y_{i+1} - x_{i+1} y_i}(y_{i+1} - y_i, x_i - x_{i+1}), \qquad i = 1, \dots, n.$$

In particular, the smallest positive integer ι_i such that $\iota_i v_i^*$ is integral is

$$\iota_i = \frac{x_i y_{i+1} - x_{i+1} y_i}{\gcd(x_{i+1} - x_i, y_{i+1} - y_i)},$$

which coincides with Proposition 2.2.14.

2.2.3 Log canonicity and resolution of singularities

Recall that for a $\mathbb{Q}$-Gorenstein variety X, a resolution of singularities $\pi\colon X' \to X$ gives rise to a *ramification formula* for the canonical divisor classes:

$$\mathcal{K}_{X'} = \pi^* \mathcal{K}_X + \sum_E a_E E,$$

where E runs through the exceptional prime divisors and $a_E \in \mathbb{Q}$ are the *discrepancies*. For $0 < \varepsilon \leq 1$, the singularities are called *ε-log canonical*, if $a_E \geq \varepsilon - 1$ for all E. For a toric Fano variety associated to an LDP polytope P, it is known that X is ε-log canonical if and only if the only interior lattice point of εP is the origin.

Definition 2.2.16 (See A.2.60)**.** We say that an LDP polygon $P \subseteq \mathbb{Q}^2$ is *ε-log canonical* for some $0 < \varepsilon \leq 1$, if the origin is the only interior lattice point of εP. Furthermore, the *log canonicity* $\varepsilon(P)$ is the maximum over all $0 < \varepsilon \leq 1$ such that P is ε-log canonical.

Remark 2.2.17. In [27], the authors call an LDP polygon *almost k-hollow* for some integer k, if $P^\circ \cap k\,\mathbb{Z}^2 = \{0\}$. This is clearly equivalent to being $\frac{1}{k}$-log canonical in the sense of Definition 2.2.16. Thus, we have equivalences

$$P \text{ almost } k\text{-hollow} \iff P \ \frac{1}{k}\text{-log canonical} \iff X_P \ \frac{1}{k}\text{-log canonical}.$$

Moreover, as was remarked in Section 1.5, dividing by k gives a bijection between $\frac{1}{k}$-log canonical LDP polygons and k-rational polygons $P \subseteq \mathbb{R}^2$ having $P^\circ = \{0\}$.

Like the Gorenstein index, log canonicity is a local property. That means we can compute it by considering the edges of the polygon individually. We make the following auxiliary definition.

Definition 2.2.18. Let $v, w \in \mathbb{Z}^2$ be non-collinear primitive vectors. Consider the triangle $\Delta := \operatorname{conv}(0, v, w)$. We define $\varepsilon_{v,w}$ to be the maximum over all $0 < \varepsilon \leq 1$ such that $(\varepsilon\Delta)^\circ = \emptyset$. For an LDP polygon $P = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}$, we also write $\varepsilon_i := \varepsilon_{v_i, v_{i+1}}$.

With this definition, we clearly have $\varepsilon(P) = \min(\varepsilon_1, \ldots, \varepsilon_n)$. Hence to compute the log canonicity of an LDP polygon, it suffices to compute the discrepancy $\varepsilon_{v,w}$ of a pair of primitive vectors. This can be done using Hilbert bases and continued fractions. We again refer to [20, Chapter 10] for more details. See also Section 1.2.2, where we used Hilbert bases to compute subpolygons.

Let $v, w \in \mathbb{Z}^2$ non-collinear primitive vectors. There exists a unique unimodular transformation $U \in \mathbb{Z}^{2\times 2}$ such that $Uv = e_2$ and $Uw = de_1 - ke_2$, where $0 \leq k < d$. Following [20], we call (d, k) the *parameters* of the pair (v, w). Using a modified euclidean algorithm, we can express the ratio d/k as a Hirzebruch-Jung continued fraction $[\![b_1, \ldots, b_r]\!]$, which is defined as

$$[\![b]\!] := b, \qquad [\![b_1, \ldots, b_r]\!] := b_1 - \frac{1}{[\![b_2, \ldots, b_r]\!]}.$$

Proposition 2.2.19 (See A.2.59)**.** *Let $v, w \in \mathbb{Z}^2$ be non-collinear primitive vectors with parameters $0 \leq k < d$ and let $d/k = [\![b_1, \ldots, b_r]\!]$. Define the rational numbers:*

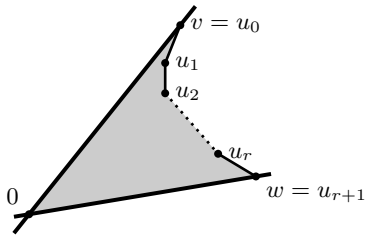
$$a_0 := 1, \qquad a_1 := \frac{k+1}{d}, \qquad a_{i+1} := b_i a_i - a_{i-1}.$$

Then we have $\varepsilon_{v,w} = \min(a_0, \ldots, a_{r+1})$.

Proof. Since a unimodular transformation leaves $\varepsilon_{v,w}$ invariant, we may assume $v = e_2$ and $w = de_1 - ke_2$. Then the Hilbert basis of the cone $\sigma = \text{cone}(v, w)$ is $\mathcal{H}_\sigma = \{u_0, \ldots, u_{r+1}\}$, where

$$u_0 = e_2, \qquad u_1 = e_1, \qquad u_{i+1} = bu_i - u_{i-1}.$$

The following is a schematic drawing. Note that by definition of the Hilbert basis, the shaded area contains no lattice points.



In particular, $\varepsilon_{v,w}$ equals the minimum over the ratios $\|u_i\|/\|z_i\|$, where z_i is the unique intersection point of the ray through u_i with the line through v and w. We compute $z_0 = u_0 = e_2$ and $z_1 = \frac{d}{k+1}e_1$, hence $a_i = \|u_i\|/\|z_i\|$ for $i = 0, 1$. Since $u_{i+1} = b_i u_i - u_{i-1}$, we get $a_i = \|u_i\|/\|z_i\|$ for all $i = 0, \ldots, r+1$, which proves the claim. □

Remark 2.2.20. A resolution of toric surface singularities is obtained by adding a ray through each element of the Hilbert basis of each maximal cone $\sigma = \text{cone}(v, w)$. In particular, the exceptional prime divisors correspond to the elements of the Hilbert basis and their discrepancies are exactly the numbers $a_1, \ldots, a_r$ from the previous proof.

Remark 2.2.21. It is not hard to show that an LDP polygon of Gorenstein index ι is $\frac{1}{\iota}$-log canonical. Recall also from Remark 2.2.13 that the Gorenstein index divides the Picard index. Together, we get the following inequalities:

$$\iota_{\mathrm{Pic}}(P) \;\geq\; \iota(P) \;\geq\; \frac{1}{\varepsilon(P)}.$$

In terms of classification, this means we have inclusions

$$\{\iota_{\mathrm{Pic}}(P) \leq k\} \quad\subseteq\quad \{\iota(P) \leq k\} \quad\subseteq\quad \{\varepsilon(P) \geq 1/k\}.$$

Recall that in 1.5.6, a classification of almost k-hollow LDP polygons was given up to $k = 6$. Using this data, we can compare the numbers of LDP polygons in the three sets above. In particular, except for $k = 1$, both inclusions are strict and the overlap is very small.

k	$\#\{\varepsilon(P) \geq 1/k\}$	$\#\{\iota(P) \leq k\}$	$\#\{\iota_{\mathrm{Pic}}(P) \leq k\}$
1	16	16	5
2	505	30	8
3	48 032	99	11
4	1 741 603	91	16
5	154 233 886	250	19
6	2 444 400 116	379	26

2.2.4 Intersection numbers

For a toric del Pezzo surface, the intersection numbers of the torus-invariant prime divisors $D_1, \ldots, D_n$ are given by

$$D_i \cdot D_{i+1} = D_{i+1} \cdot D_i = \frac{1}{\det(v_i, v_{i+1})}, \qquad D_i^2 = \frac{\det(v_{i+1}, v_{i-1})}{\det(v_{i-1}, v_i)\det(v_i, v_{i+1})},$$

where $D_i \cdot D_j = 0$ if $\mathrm{mod}(|i-j|, n) > 1$. Here D_i is the torus-invariant prime divisor associated to the i-th vertex v_i of the defining LDP polygon. See for instance [20, Chapter 10.4] for more background.

We may define intersection numbers purely in terms of the polygon as follows.

Definition 2.2.22 (See A.2.61, A.2.62, A.2.63)**.** Let $P = \begin{bmatrix} v_1 & \ldots & v_n \end{bmatrix}$ be an LDP polygon. Recall that the class group $\mathrm{Cl}(P)$ is generated by $\omega_i := Q(e_i)$, where $Q\colon \mathbb{Z}^n \to \mathrm{Cl}(P)$ is the grading matrix. We define

$$\omega_i \cdot \omega_{i+1} := \omega_{i+1} \cdot \omega_i := \frac{1}{\det(v_i, v_{i+1})}, \qquad \omega_i^2 := \frac{\det(v_{i+1}, v_{i-1})}{\det(v_{i-1}, v_i)\det(v_i, v_{i+1})},$$

and $\omega_i \cdot \omega_j := 0$ if $\mathrm{mod}(|i-j|, n) > 1$. Extending linearly, we obtain a symmetric bilinear map $\cdot\colon \mathrm{Cl}(P) \times \mathrm{Cl}(P) \to \mathbb{Q}$. By the *anticanonical self-intersection* or the *degree* of P, we mean the rational number

$$(-K_P)^2 = (\omega_1 + \cdots + \omega_n)^2 = \sum_{i=1}^{n} \omega_i^2 + 2\sum_{i=1}^{n} \omega_i \cdot \omega_{i+1}.$$

For a toric n-dimensional Fano variety X_P, the anticanonical self intersection $(-\mathcal{K}_P)^n$ equals the normalized volume of the dual polytope P^* (see e.g. [22, Chapter 5.3]). In the following, we give an elementary proof of this fact for surfaces.

Proposition 2.2.23. *Let P be an LDP polygon and let P^* be its dual. Then*

$$(-\mathcal{K}_P)^2 = 2\,\mathrm{area}(P^*).$$

Proof. Recall from Remark 2.2.15 that the vertices of P^* are $v_i^* = \frac{1}{\mu_i}(y_{i+1} - y_i, x_i - x_{i+1})$ for $i = 1, \ldots, n$, where $\mu_i := \det(v_i, v_{i+1})$ are the local multiplicities. Consider the triangles $\Delta_i := \mathrm{conv}(0, v_{i-1}^*, v_i^*)$. We compute

$$\begin{aligned} 2\,\mathrm{area}(\Delta_i) &= \det(v_{i-1}^*, v_i^*) \\ &= \frac{1}{\mu_{i-1}\mu_i}((y_i - y_{i-1})(x_i - x_{i+1}) - (x_{i-1} - x_i)(y_{i+1} - y_i)) \\ &= \frac{1}{\mu_{i-1}\mu_i}(\det(v_{i+1}, v_{i-1}) + \mu_{i-1} + \mu_i) \\ &= \omega_i^2 + \omega_{i-1} \cdot \omega_i + \omega_i \cdot \omega_{i+1}. \end{aligned}$$

Since $\mathrm{area}(P^*) = \mathrm{area}(\Delta_1) + \cdots + \mathrm{area}(\Delta_n)$, we arrive at the claim. □

2.3 Proof of the Picard index formula

We give an elementary proof of the Picard index formula for LDP polygons as stated in Theorem 2.2.11. Note that it is a special case of the more general Picard index formula for surfaces with torus action, which will be discussed in Section 3.3 and has been published in [56]. The proof given here follows essentially the same pattern as the general one, but is much simpler as it is tailored to the toric case. We begin by specializing the observation about the Picard index made in Proposition 2.1.9 to surfaces.

As in previous sections, we identify an LDP polygon $P \subseteq \mathbb{R}^2$ with its vertex matrix $P = \begin{bmatrix} v_1 & \ldots & v_n \end{bmatrix}$, where we require the vertices to be ordered counterclockwise. For ease of notation, indices corresponding to vertices are always cyclical, e.g. $v_{n+1} := v_1$.

Proposition 2.3.1. *Let $P = \begin{bmatrix} v_1 & \ldots & v_n \end{bmatrix}$ be an LDP polygon. Then we have*

$$\iota_{\mathrm{Pic}}(P) = \frac{\mu_1(P) \ldots \mu_n(P)}{|\hat{K}|},$$

where $\hat{K}$ is defined as in Construction 2.1.5 applied to the face fan of P.

Proof. The face fan of P has maximal cones $\sigma_i = \det(v_i, v_{i+1})$, where $i = 1, \ldots, n$. Moreover, we have $K_{\sigma_i} = \mathrm{Cl}_i(P)$ for the local class groups from Construction 2.1.4. Since the Picard group of P coincides with the Picard group of the associated toric surface, Proposition 2.1.9 yields the claim. □

In order to prove the Picard index formula 2.2.11, we still need to show that the group order $|\hat{K}|$ equals the multiplicity $\mu(P)$. This comes down to a careful study of minors of both P and the map $\hat{P}$ from Construction 2.1.5. Indeed, since $\hat{K}$ is the cokernel of $\hat{P}^\top$, its order equals the greatest common divisor of the maximal minors of $\hat{P}$. On the other hand, $\mu(P)$ is the torsion order of the cokernel of $P^\top$, hence equals the greatest common divisor of the 2×2-minors of P (see also Proposition 2.2.3). Before we can study minors of $\hat{P}$, we need to fix a matrix representation, which means choosing bases of the involved lattices. In the following, we replay Construction 2.1.4 specialized to a toric surface, fixing lattice bases along the way.

Construction 2.3.2. Consider lattices $F := \mathbb{Z}\, f_1 \oplus \cdots \oplus \mathbb{Z}\, f_n$ and $N := \mathbb{Z}\, e \oplus \mathbb{Z}\, u$. Then we can view an LDP polygon P with vertices $v_i \in N$ as a lattice map

$$P\colon F \to N, \qquad f_i \mapsto v_i = x_i e + y_i u.$$

For $i = 1, \ldots, n$, set $N_i := \mathbb{Z}\, e_i \oplus \mathbb{Z}\, u_i$ and $F_i := \mathbb{Z}\, f_i^- \oplus \mathbb{Z}\, f_{i+1}^+$ (using cyclical indices, i.e. $f_{n+1}^+ = f_1^+$). Then we have the local generator maps

$$P_i\colon F_i \to N_i, \qquad f_i^- \mapsto x_i e_i + y_i u_i, \quad f_{i+1}^+ \mapsto x_{i+1} e_i + y_{i+1} u_i.$$

Together with the maps $\alpha\colon N_i \to N$ sending e_i, u_i to e, u and $\beta\colon F_i \to F$ sending f_i^-, f_{i+1}^+ to f_i, f_{i+1}, we obtain a commutative diagram

$$\begin{array}{ccc} F & \xrightarrow{P} & N \\ {\scriptstyle\beta_i}\uparrow & & \uparrow{\scriptstyle\alpha_i} \\ F_i & \xrightarrow{P_i} & N_i. \end{array}$$

Next, we replay Construction 2.1.5. Choosing explicit representations of the kernels of α and β, we will obtain an explicit description of $\hat{P}\colon \hat{F} \to \hat{N}$.

Construction 2.3.3. In the setting of Construction 2.3.2, we define lattices

$$\begin{aligned} \mathbf{N} &:= \bigoplus_{i=1}^n N_i &&= \mathbb{Z}\, e_1 \oplus \mathbb{Z}\, u_1 \oplus \cdots \oplus \mathbb{Z}\, e_n \oplus \mathbb{Z}\, u_n \\ \mathbf{F} &:= \bigoplus_{i=1}^n F_i &&= \mathbb{Z}\, f_1^- \oplus \mathbb{Z}\, f_2^+ \oplus \mathbb{Z}\, f_2^- \oplus \cdots \oplus \mathbb{Z}\, f_n^+ \oplus \mathbb{Z}\, f_n^- \oplus \mathbb{Z}\, f_1^+ \end{aligned}$$

as well as the map $\mathbf{P} := \bigoplus_{i=1}^n P_i\colon \mathbf{N} \to \mathbf{F}$. The lattice maps α and β are defined as in Construction 2.1.4, i.e.

$$\begin{aligned} \alpha\colon \mathbf{N} &\to N, \quad e_i \mapsto e, \quad u_i \mapsto u, \\ \beta\colon \mathbf{F} &\to F, \quad f_i^-, f_i^+ \mapsto f_i. \end{aligned}$$

Next, define lattices

$$\hat{F} := \mathbb{Z}\, \hat{f}_1 \oplus \cdots \oplus \mathbb{Z}\, \hat{f}_n, \qquad \hat{N} := \mathbb{Z}\, \hat{e}_1 \oplus \mathbb{Z}\, \hat{u}_1 \oplus \cdots \oplus \mathbb{Z}\, \hat{e}_{n-1} \oplus \mathbb{Z}\, \hat{u}_{n-1},$$

as well as maps

$$\begin{aligned} \gamma\colon \hat{N} &\to \mathbf{N}, \quad \hat{e}_i \mapsto e_i - e_{i+1}, \qquad \hat{u}_i \mapsto u_i - u_{i+1}, \\ \delta\colon \hat{F} &\to \mathbf{F}, \quad \hat{f}_i \mapsto f_i^+ - f_i^-. \end{aligned}$$

We claim that γ is a kernel of α and δ is a kernel of β. This gives us a unique map $\hat{P}\colon \hat{F} \to \hat{N}$ making the obvious diagram commute. Explicitly, the map is given as follows:

$$\hat{P}\colon \hat{F} \to \hat{N}, \qquad \hat{f}_i \mapsto \begin{cases} -x_1(\hat{e}_1 + \cdots + \hat{e}_{n-1}) - y_1(\hat{u}_1 + \cdots + \hat{u}_{n-1}), & i = 1 \\ x_i\hat{e}_{i-1} + y_i\hat{u}_{i-1}, & 2 \le i \le n. \end{cases}$$

Proof. Clearly, we have $\alpha \circ \gamma = 0$ and $\beta \circ \delta = 0$. Moreover, γ and δ are both injective and we have

$$\begin{aligned} \operatorname{rank}(\hat{N}) &= 2n-2 &&= \operatorname{rank}(\mathbf{N}) - \operatorname{rank}(N), \\ \operatorname{rank}(\hat{F}) &= n &&= \operatorname{rank}(\mathbf{F}) - \operatorname{rank}(F). \end{aligned}$$

This shows that γ and δ are kernels of α and β respectively. An explicit computation shows that $\gamma \circ \hat{P} = \mathbf{P} \circ \delta$, hence $\hat{P}$ is the induced map between the kernels. □

Remark 2.3.4. Let P be an LDP polygon with vertices $v_i = x_i e + y_i u \in N$. With respect to the lattice bases of Construction 2.3.3, the matrix representation of $\hat{P}$ is

$$\hat{P} = \begin{bmatrix} -x_1 & x_2 & 0 & & 0 \\ -y_1 & y_2 & 0 & \dots & 0 \\ -x_1 & 0 & x_3 & & 0 \\ -y_1 & 0 & y_3 & & 0 \\ & \vdots & & \ddots & \\ -x_1 & 0 & 0 & & x_n \\ -y_1 & 0 & 0 & \dots & y_n \end{bmatrix}.$$

Proposition 2.3.5. *Let P be an LDP polygon with vertices $v_i = x_i e + y_i u \in N$ and let $\hat{P}$ be as in Construction 2.3.3. Then*

$$\gcd(M(P)) = \gcd(M(\hat{P})),$$

where $M(\cdot)$ is the set of maximal minors of a matrix.

Proof. Define the set

$$M' := \{\det(v_1, v_i) \mid i = 2, \dots, n\} \subseteq M(P).$$

Applying Lemma 2.2.2 to P, we see that $\gcd(M(P)) = \gcd(M')$. We claim that also $\gcd(M(\hat{P}))$ equals $\gcd(M')$. A maximal minor of $\hat{P}$ is described by a subset of row indices $I \subseteq \{\hat{e}_1, \hat{u}_1, \dots, \hat{e}_{n-1}, \hat{u}_{n-1}\}$ of the lattice basis of $\hat{N}$, where $|I| = n$. Considering the matrix representation of $\hat{P}$

from Remark 2.3.4, we see that for a maximal minor to be nonzero, we must have $I \cap \{\hat{e}_j, \hat{u}_j\} \neq \emptyset$ for all $j = 1, \ldots, n-1$. In particular, a nonzero maximal minor is of the form

$$\pm \det(v_1, v_i) \prod_{\substack{j=1 \\ j \neq i}}^{n-1} z_j, \tag{2.1}$$

for some $i = 2, \ldots, n$, where $z_j := x_j$ if $\hat{e}_j \in I$ and $z_j := y_j$ if $\hat{u}_j \in I$. Since $\gcd(x_j, y_j) = 1$, taking the greatest common divisors of all expressions of the form (2.1) reduces to the greatest common divisor of all $\det(v_1, v_i)$ for $i = 2, \ldots, n$. Hence $\gcd(M(\hat{P})) = \gcd(M')$. □

Proof of Theorem 2.2.11. By Proposition 2.3.1, we need to show $|\hat{K}| = \mu(P)$. In general, the torsion order of the cokernel of an integral matrix equals the product of its invariant factors, which equals the greatest common divisor of its maximal minors. Thus $\mu(P) = \gcd(M(P^\top)) = \gcd(M(P))$ and $|\hat{K}| = \gcd(M(\hat{P}^\top)) = \gcd(M(\hat{P}))$. Proposition 2.3.5 now yields the claim. □

2.4 Classification of LDP triangles by Picard index

Using the Picard index formula from Theorem 2.2.11, we obtain a very simple classification algorithm for LDP triangles, which in terms of toric geometry, correspond to fake weighted projective planes. We give explicit results up to Picard index 1 000 000, see also Section A.4.5 for an implementation in `RationalPolygons.jl`. This classification has been published in [56, Section 8].

We view an LDP triangle as an integral matrix $P = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}$, where $v_i = (x_i, y_i) \in \mathbb{Z}^2$ are the primitive vertices and they are ordered counterclockwise. Write $\mu := \mu(P)$ and $\mu_i := \mu_i(P)$ for the (local) multiplicities and $\omega_i = (w_i, \eta_i) \in \mathbb{Z} \times \mathbb{Z}/\mu\mathbb{Z}$ for the columns of the grading matrix. Propositions 2.2.3 and 2.2.10 yield

$$\mu_1 = \det(v_1, v_2) = \mu w_3, \qquad \mu_2 = \det(v_2, v_3) = \mu w_1, \qquad \mu_3 = \det(v_3, v_1) = \mu w_2. \tag{2.2}$$

Note that Corollary 2.2.8 implies that the w_i are pairwise coprime. Next, we may assume P to be in Hermite normal form, which means $v_1 = (1, 0)$ and $0 \leq x_2 < y_2$. Using Equations (2.2), we can thus write

$$P = \begin{bmatrix} 1 & x_2 & -\frac{w_1 + x_2 w_2}{w_3} \\ 0 & \mu w_3 & -\mu w_2 \end{bmatrix}, \qquad 0 \leq x_2 < \mu w_3.$$

Furthermore, by Theorem 2.2.11, the Picard index of P is $\mu^2 w_1 w_2 w_3$. We arrive at the following classification algorithm:

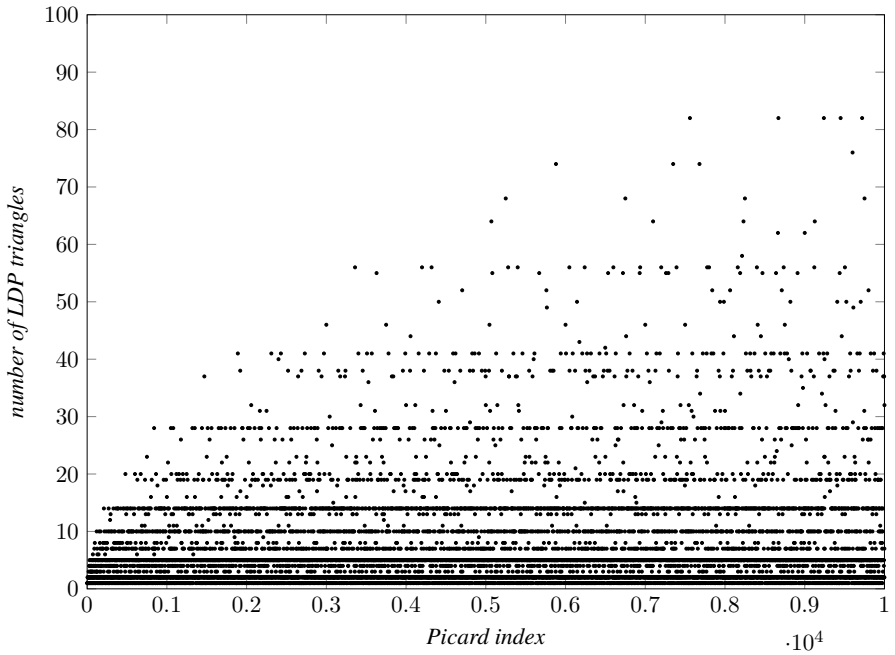
Algorithm 2.4.1 (See A.4.16 for an implementation)**.** Given a prospective Picard index $\iota \in \mathbb{Z}_{\geq 1}$ we can classify all LDP triangles with that Picard index as follows: First, consider all decompositions of ι into four integers $\iota = M w_1 w_2 w_3$, where $M = \mu^2$ is a square and w_i are pairwise

coprime with $w_1 \leq w_2 \leq w_3$. Then, go through all $0 \leq x < \mu w_3$ such that w_3 divides $w_1 + xw_2$ and $\gcd(x, \mu w_3) = 1$ and $\gcd(\frac{w_1+xw_2}{w_3}, \mu w_2) = 1$. Consider the LDP triangle

$$P = \begin{bmatrix} 1 & x & -\frac{w_1+xw_2}{w_3} \\ 0 & \mu w_3 & -\mu w_2 \end{bmatrix}$$

and add it to the list of results. Duplicate entries can be avoided by bringing the triangles into unimodular normal form, see Section 1.2.1.

Classification 2.4.2 (Data available at [54])**.** *Up to unimodular equivalence, there are* 15 086 426 *LDP triangles with Picard index at most* 1 000 000*. Up to Picard index* 10 000*, the numbers of LDP triangles develop as follows:*



Write $\triangle(\iota)$ for the number of LDP triangles with Picard index ι. From Classification 2.4.2, we observe that $\triangle(\iota) = 1$ holds exactly for ι prime and $\iota \in \{1, 4\}$. This observation can be proven to hold in general:

Proposition 2.4.3. *We have* $\triangle(\iota) \geq 1$ *for all* $\iota \in \mathbb{Z}_{\geq 1}$ *and* $\triangle(\iota) = 1$ *if and only if* ι *is prime or* $\iota \in \{1, 4\}$.

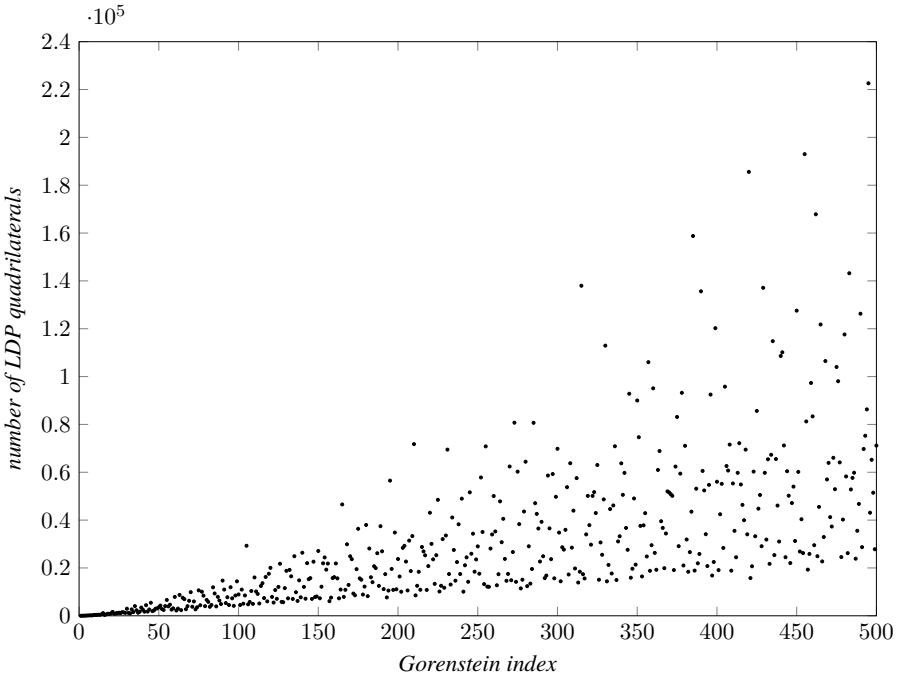
Proof. For any $\iota \in \mathbb{Z}_{\geq 1}$, the LDP triangle $P_\iota := \mathrm{conv}((1, 0), (0, 1), (-\iota, -1))$ has Picard index ι, hence $\triangle(\iota) \geq 1$. The fact that $\triangle(1) = \triangle(4) = 1$ follows from the classification. Moreover, if $\iota = \mu^2 w_1 w_2 w_3$ is prime, we must have $\mu = 1$ and after renumbering vertices, we can assume $(w_1, w_2, w_3) = (\iota, 1, 1)$. By Algorithm 2.4.1, the only LDP triangle having ι as Picard index is P_ι, hence $\triangle(\iota) = 1$. For the converse, we need to show that for $\iota \notin \{1, 4\}$ not prime, we can find another LDP triangle with Picard index ι which is not equivalent to P_ι. If ι is not a prime power,

we can write $\iota = ab$ for some coprime integers $a, b > 1$. Then $\mathrm{conv}((1,0),(0,1),(-a,-b))$ has Picard index ι. Since its local multiplicities differ from P_ι, they are not equivalent. Now suppose $\iota = p^k$ for a prime p and an integer $k \geq 2$. Then we define $Q := \mathrm{conv}((1,0),(1,p),(-(p^{k-2}+1),-p^{k-1}))$. This is an LDP polygon if and only if $(p,k) \neq (2,2)$ and it's Picard index equals ι. Moreover, we have $\mu(Q) = p \neq 1 = \mu(P_\iota)$, hence Q and P_ι are not equivalent. □

2.5 Classifications by Gorenstein index

In this section, we are concerned with classifications of LDP polygons (toric log del Pezzo surfaces) by Gorenstein index. Our main contribution is a classification algorithm for LDP quadrangles, obtaining the following result:

Classification 2.5.1 (Data available at [53])**.** *Up to unimodular equivalence, there are* 15 669 466 *LDP quadrangles of Gorenstein index at most* 500*. Their numbers develop as follows:*



Recall that in [36], Kasprzyk, Kreuzer and Nill gave an algorithm to classify LDP polygons with arbitrary number of vertices by Gorenstein index and obtained results up to index 17. More recently, the authors of [26] classified log del Pezzo surfaces of Picard number one with torus action up to Gorenstein index 200, which in the toric case are precisely the LDP triangles. Furthermore, Bäuerle [11] gave an algorithm classifying Fano simplices in any dimension by Gorenstein index. In particular, he extended the classification of LDP triangles to Gorenstein index 1 000.

Similar to [11, 26], we work with the description of LDP polygons by their grading matrices, splitting the classification into two independent steps:

1) Given $\iota \in \mathbb{Z}_{\geq 1}$, classify all possible free parts of grading matrices Q of LDP polygons with Gorenstein index ι.

2) Given a pair (Q, ι) classify all LDP polygons having Gorenstein index ι and Q as the free part of its grading matrix.

The rest of this section is organized as follows: In Subsection 2.5.1, we present our approach to step 1), which involves Gorenstein matrices and their connection to certain unit fraction equations. In Subsection 2.5.2, we give an algorithm for step 2), which works for polygons of any number of vertices. In Subsection 2.5.3, we revisit the case of LDP triangles and explain Bäuerle's method, which we use to extend his classification to Gorenstein index $45\,000$. Finally, Subsection 2.5.4 treats step 1) for LDP quadrangles, resulting in our Classification 2.5.1.

2.5.1 Gorenstein matrices and unit fraction partitions

In this section, we develop the notion of Gorenstein matrices for LDP polygons as well as their relation to certain unit fraction equations. We begin by introducing yet again Picard group and Gorenstein index, but this time purely in terms of grading matrices.

Construction 2.5.2. Let $Q = \begin{bmatrix} w_1 & \dots & w_n \end{bmatrix} \in \mathbb{Z}^{\rho \times n}$ be an integral matrix, where $n = \rho + 2$. Column indices will be considered modulo n, e.g. $w_{n+1} := w_1$. We define the *Picard group* as

$$\operatorname{Pic}(Q) \;:=\; K_1 \cap \cdots \cap K_n, \qquad K_i \;:=\; \langle w_j \,;\, j \notin \{i, i+1\} \rangle \;\leq\; \mathbb{Z}^\rho \,.$$

Setting $w := w_1 + \cdots + w_n$, we say that Q is *ι-Gorenstein*, if $\iota w \in \operatorname{Pic}(Q)$. In this case, we find unique integers a_{ij} such that

$$\iota w = \sum_{j=1}^{\rho} a_{ij} w_{j+i-1} \tag{2.3}$$

holds for all $i = 1, \dots n$. We call $A := (a_{ij}) \in \mathbb{Z}^{n \times \rho}$ the *Gorenstein coefficients* of the pair (Q, ι). We call (Q, ι) an *LDQ pair*, if Q is ι-Gorenstein and $a_{ij} > 0$ holds for all i and j.

Proposition 2.5.3. *Let P be an LDP polygon with class group $\operatorname{Cl}(P) = \mathbb{Z}^\rho \times \mathbb{Z} \,/\, \mu\, \mathbb{Z}$ and Gorenstein index ι. Let Q be the free part of its grading matrix. Then we have $\pi(\operatorname{Pic}(P)) \subseteq \operatorname{Pic}(Q)$, where $\pi \colon \operatorname{Cl}(P) \to \mathbb{Z}^\rho$ is the projection. Moreover, (Q, ι) is an LDQ pair.*

Proof. For the local Picard groups, we have $\pi(\operatorname{Pic}_i(P)) = K_i$, hence

$$\pi(\operatorname{Pic}(P)) \;=\; \pi\left(\bigcap_{i=1}^{n} \operatorname{Pic}_i(P)\right) \;\subseteq\; \bigcap_{i=1}^{n} K_i \;=\; \operatorname{Pic}(Q).$$

Since $\pi(\mathcal{K}_P) = -w$, and $\iota \mathcal{K}_P \in \operatorname{Pic}(P)$, it follows that Q is ι-Gorenstein. It remains to see that the Gorenstein coefficients are all positive. Recall that P defines a toric del Pezzo surface X_P.

Being del Pezzo, its anticanonical divisor class is in the ample cone, which for toric surfaces is given by

$$\operatorname{Ample}(X_P) \;=\; \bigcap_{i=1}^{n} \operatorname{cone}_{\mathbb{Q}}(\langle w_j \;;\; j \notin \{i, i+1\}\rangle)^\circ \;\subseteq\; \mathbb{Q}^\rho,$$

see for instance [20, Proposition 2.4.2.6]. Hence w can be written as a linear combination over $\{w_j \;;\; j \notin \{i, i+1\}\}$ with positive rational coefficients. Since the Gorenstein coefficients are uniquely determined, they must be positive. □

Example 2.5.4. In Proposition 2.5.3, if P has nontrivial multiplicity, we can have $\pi(\operatorname{Pic}(P)) \subsetneq \operatorname{Pic}(Q)$. An example is given by the LDP triangle with vertex matrix

$$P = \begin{bmatrix} 3 & -1 & -1 \\ -1 & 1 & -5 \end{bmatrix}.$$

It has multiplicity $\mu(P) = 2$ and grading matrix

$$Q = \begin{bmatrix} 3 & 8 & 1 \\ 1 & 1 & 0 \end{bmatrix}.$$

In particular, we compute $\pi(\operatorname{Pic}(P)) = 48\,\mathbb{Z} \subsetneq 24\,\mathbb{Z} = \operatorname{Pic}(Q)$. Moreover, P has Gorenstein index 4, hence $\iota w = 4 \cdot 12 = 48$. Since $(w_1, w_2, w_3) = (3, 8, 1)$, the Gorenstein coefficients are the 3×1-matrix $\begin{bmatrix} 16 & 6 & 48 \end{bmatrix}^\top$.

Next, we study how the Gorenstein coefficients change under renumbering of columns as well as unimodular transformations of Q. In the next proposition, column indices of Q and row indices of A are both considered modulo n.

Proposition 2.5.5. *Let (Q, ι) be an LDQ pair with $Q = \begin{bmatrix} w_1 & \ldots & w_n \end{bmatrix} \in \mathbb{Z}^{\rho \times n}$.*

(i) If $Q' = UQ$ for some unimodular $U \in \mathbb{Z}^{\rho\times\rho}$, we have $A(Q', \iota) = A(Q, \iota)$.

(ii) If $Q' = \begin{bmatrix} w'_1 & \ldots & w'_n \end{bmatrix}$ with $w'_i = w_{-i}$, we have $A(Q', \iota) = (a_{-i+3,\rho-j+1})$.

(iii) If $Q' = \begin{bmatrix} w'_1 & \ldots & w'_n \end{bmatrix}$ with $w'_i = w_{i+k}$, we have $A(Q', \iota) = (a_{i+k,j})$.

Proof. In all three cases, we have to verify Equation (2.3) for the modified matrices Q' and $A(Q', \iota)$. For (i), write $w'_i = Uw_i$ for the columns of Q'. Then also $w' = Uw$, hence $\iota w' = \sum_j a_{ij} w'_{j+i-1}$ for all i. For (ii), we have to show $\sum_{j=1}^{\rho} a'_{ij} w'_{j+i-1} = \iota w$ holds for all i, where $a'_{ij} := a_{-i+3,\rho-j+1}$. Substituting j for $\rho - j + 1$ in the sum, we obtain

$$\sum_{j=1}^{\rho} a'_{ij} w'_{j+i-1} = \sum_{j=1}^{\rho} a_{-i+3,\rho-j+1} w_{-j-i+1} = \sum_{j=1}^{\rho} a_{-i+3,j} w_{j-i-\rho}.$$

Since $-\rho \equiv 2 \mod n$, the right hand side is $\sum_{j=1}^{\rho} a_{-i+3,j} w_{j+(-i+3)-1}$, which equals ιw by assumption. Part (iii) can be shown analogously. □

This proposition allows us to define equivalence and a normal form for Gorenstein coefficients:

Definition 2.5.6. We call two Gorenstein coefficients $A, A' \in \mathbb{Z}^{\rho \times n}$ *equivalent* and write $A \sim A'$, if A' arises from A by finitely many operations from Proposition 2.5.5 i.e.

$$(a_{ij}) \sim (a_{-i+3,\rho-j+1}), \qquad (a_{ij}) \sim (a_{i+k,j}),$$

where $k = 1, \ldots, n$. Note that any equivalence class of Gorenstein coefficients contains at most $2n$ elements. We say A is in *normal form*, if it is lexicographically minimal among its equivalence class.

By converting Equation (2.3) into a matrix equation of the form $GQ^\top = 0$, we arrive at the main concept of this section, the *Gorenstein matrix*:

Definition 2.5.7. Let $A = (a_{ij}) \in \mathbb{Z}^{n \times \rho}$ Gorenstein coefficients of some LDQ pair (Q, ι). The *Gorenstein matrix* is be the $n \times n$-matrix

$$G(Q,\iota) := \begin{bmatrix} \iota - a_{11} & \iota - a_{12} & \iota - a_{13} & \ldots & \iota - a_{1\rho} & \iota & \iota \\ \iota & \iota - a_{21} & \iota - a_{22} & \ldots & \iota - a_{2\rho-1} & \iota - a_{2\rho} & \iota \\ \iota & \iota & \iota - a_{31} & \ldots & \iota - a_{3\rho-2} & \iota - a_{3\rho-1} & \iota - a_{3\rho} \\ \iota - a_{4\rho} & \iota & \iota & \ldots & \iota - a_{4\rho-3} & \iota - a_{4\rho-2} & \iota - a_{4\rho-1} \\ & \vdots & & \ddots & & \vdots & \\ \iota - a_{n2} & \iota - a_{n3} & \iota - a_{n4} & \ldots & \iota & \iota & \iota - a_{n1} \end{bmatrix}.$$

That is, the entries of $G(Q, \iota)$ are given as

$$g_{ij} := \begin{cases} \iota - a_{ir}, & r \in \{1, \ldots, \rho\} \\ \iota, & r \in \{\rho+1, \rho+2\} \end{cases},$$

where $r \in \{1, \ldots, n\}$ is the unique integer such that $r + i - 1 \equiv j \mod n$.

Proposition 2.5.8. *Let $G = G(Q, \iota)$ be a Gorenstein matrix. Then $GQ^\top = 0$. Moreover, we have* $\operatorname{rank}(G) = 2$ *and all 3×3-minors of G vanish.*

Proof. That $GQ^\top = 0$ follows directly from Equation (2.3). In particular, this implies $\operatorname{rank}(G) \leq 2$. On the other hand, since the a_{ij} are positive, any two rows of G are linearly independent, thus $\operatorname{rank}(G) = 2$. The fact that all 3×3-minors vanish is an immediate consequence. □

Proposition 2.5.8 is the key for our classification, as the vanishing of all 3×3-minors gives plenty of relations between the Gorenstein coefficients. The task now becomes to use these relations to get effective bounds on all Gorenstein coefficients. By computing the kernels of the associated Gorenstein matrices, we then get all possible LDQ pairs for a given Gorenstein index. The following two examples describe this approach for $\rho \in \{1, 2\}$.

Example 2.5.9. In the case $\rho = 1$, we are concerned with the classification of LDP triangles, which in the language of toric geometry are fake weighted projective planes. Here, there are three Gorenstein coefficients a_1, a_2, a_3 and the Gorenstein matrix is of the form

$$G = \begin{bmatrix} \iota - a_1 & \iota & \iota \\ \iota & \iota - a_2 & \iota \\ \iota & \iota & \iota - a_3 \end{bmatrix}.$$

Proposition 2.5.8 gives us the condition $\det(G) = 0$, which is equivalent to the following unit fraction identity:

$$\frac{1}{\iota} = \frac{1}{a_1} + \frac{1}{a_2} + \frac{1}{a_3}. \tag{2.4}$$

Since there are only finitely many ways in which to write a rational number as the sum of three positive unit fractions, Equation (2.4) is enough to enumerate all triples (a_1, a_2, a_3) that occur as Gorenstein coefficients of LDP triangles with given Gorenstein index. We will elaborate on this and provide an actual classification algorithm, which is based on the one given by Bäuerle [11], in Section 2.5.3.

Example 2.5.10. We consider now the case $\rho = 2$, i.e. the classification of LDP quadrangles. Here, there are eight Gorenstein coefficients and the Gorenstein matrix is of the form

$$G = \begin{bmatrix} \iota - a_{11} & \iota - a_{12} & \iota & \iota \\ \iota & \iota - a_{21} & \iota - a_{22} & \iota \\ \iota & \iota & \iota - a_{31} & \iota - a_{32} \\ \iota - a_{42} & \iota & \iota & \iota - a_{41} \end{bmatrix}.$$

By Proposition 2.5.8, all sixteen 3×3-minors of this matrix vanish. Note that a 3×3-minor of G contains either four or five distinct Gorenstein coefficients. An example of a minor with four coefficients is the one obtained by erasing the third column and fourth row of G. An example with five coefficients is the minor obtained by erasing the third column and third row of G. The condition that these two minors vanish can be stated in terms of the following identities:

$$\frac{1}{\iota} = \frac{1}{a_{11}} + \frac{1}{a_{21}} + \frac{1}{a_{32}} - \frac{a_{12}}{a_{11}a_{21}}. \tag{2.5}$$

$$\frac{1}{\iota} = \frac{1}{a_{11}} + \frac{1}{a_{21}} + \frac{1}{a_{41}} - \frac{a_{12}}{a_{11}a_{21}} - \frac{a_{42}}{a_{11}a_{41}} + \frac{a_{12}a_{42}}{a_{11}a_{21}a_{41}}. \tag{2.6}$$

Unlike Equation (2.4), these are no longer simple sums of unit fractions, as they also contain subtraction. We will come back to this in Section 2.5.4, where we will give a classification algorithm to enumerate all Gorenstein coefficients of LDP quadrangles.

In general, a 3×3-minor of a Gorenstein matrix contains between three and nine distinct Gorenstein coefficients, where the case of three Gorenstein coefficients only occurs for $\rho = 1$. Minors with exactly four Gorenstein coefficients as in the previous example occur in every Picard number greater than one. We end this section by providing a complete description of such minors for arbitrary Picard number and deriving some useful equalities.

Proposition 2.5.11. *Let $A \in \mathbb{Z}^{n\times\rho}$ be the Gorenstein coefficients of some LDQ pair (Q,ι). Then every 3×3-submatrix of the Gorenstein matrix $G := G(Q,\iota)$ containing precisely four distinct Gorenstein coefficients is of one of the following forms:*

$$\begin{bmatrix} \iota & \iota & \iota - a_{k1} \\ \iota - a_{k+1,\rho} & \iota & \iota \\ \iota - a_{k+2,\rho-1} & \iota - a_{k+2,\rho} & \iota \end{bmatrix} \qquad \begin{bmatrix} \iota & \iota - a_{k1} & \iota - a_{k2} \\ \iota & \iota & \iota - a_{k+1,1} \\ \iota - a_{k+2,\rho} & \iota & \iota \end{bmatrix}.$$

Here, all row indices of A are considered modulo n. Moreover, for $k = 1,\dots,n$, the following identities and inequalities hold:

(i) $\frac{1}{\iota} = \frac{1}{a_{k1}} + \frac{1}{a_{k+1,\rho}} + \frac{1}{a_{k+2,\rho}} - \frac{a_{k+2,\rho-1}}{a_{k+1,\rho}a_{k+2,\rho}}$,

(ii) $\frac{1}{\iota} = \frac{1}{a_{k1}} + \frac{1}{a_{k+1,1}} + \frac{1}{a_{k+2,\rho}} - \frac{a_{k2}}{a_{k+1,1}a_{k1}}$,

(iii) $\min(a_{k1}, a_{k+1,\rho}, a_{k+2,\rho}) \le 3\iota$,

(iv) $\min(a_{k1}, a_{k+1,1}, a_{k+2,\rho}) \le 3\iota$,

(v) $a_{k+1,\rho}a_{k+2,\rho}(a_{k2} - a_{k1}) = a_{k1}a_{k+1,1}(a_{k+2,\rho-1} - a_{k+2,\rho})$,

(vi) $a_{k+2,\rho}a_{k+3,\rho}(a_{k+1,1} - a_{k2}) = a_{k1}a_{k+1,1}(a_{k+2,\rho} - a_{k+3,\rho-1})$.

Proof. The main statement is a direct consequence of the structure of the Gorenstein matrix. By Proposition 2.5.8, the determinants of the displayed submatrices vanish, which gives us equations (i) and (ii). For (iii), notice that for $a := \min(a_{k1}, a_{k+1,\rho}, a_{k+2,\rho})$, we have

$$\frac{1}{\iota} = \frac{1}{a_{k1}} + \frac{1}{a_{k+1,\rho}} + \frac{1}{a_{k+2,\rho}} - \frac{a_{k+2,\rho-1}}{a_{k+1,\rho}a_{k+2,\rho}} \le \frac{1}{a} + \frac{1}{a} + \frac{1}{a} = \frac{3}{a},$$

hence we get $a \le 3\iota$. Part (iv) is handled analogously. Combining equations (i) and (ii), we get

$$\frac{1}{a_{k+1,\rho}} - \frac{a_{k+2,\rho-1}}{a_{k+1,\rho}a_{k+2,\rho}} = \frac{1}{a_{k+1,1}} - \frac{a_{k2}}{a_{k+1,1}a_{k1}}.$$

Multiplying both sides by the common denominator yields (v). On the other hand, we can also first substitute k with $k+1$ in (i) and then combine it with (ii) to get

$$\frac{1}{a_{k1}} - \frac{a_{k2}}{a_{k+1,1}a_{k1}} = \frac{1}{a_{k+3,\rho}} - \frac{a_{k+3,\rho-1}}{a_{k+2,\rho}a_{k+3,\rho}}.$$

Again, we multiply by the common denominator and obtain (vi). □

2.5.2 LDP polygons from LDQ pairs

In this subsection, we develop an algorithm that, given a prospective LDQ pair (Q,ι), computes all LDP polygons having Gorenstein index ι and Q as the free part of their grading matrix. Our

approach works for polygons of any number of vertices and is essentially a generalization of the one described in [26, Prop. 3.11].

Let $P = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}$ be an LDP polygon with vertices $v_i = (x_i, y_i) \in \mathbb{Z}^2$. We may assume P to be in Hermite normal form, i.e. $v_1 = (1, 0)$ and $0 \leq x_2 < y_2$. Moreover, let $Q = \begin{bmatrix} w_1 & \dots & w_n \end{bmatrix}$ be the free part of the grading matrix of P, where $w_i \in \mathbb{Z}^{n-2}$. For $1 \leq i < j \leq n$, we define the integers

$$m_{ij} = \det(w_1, \dots, \cancel{w_i}, \dots, \cancel{w_j}, \dots, w_n).$$

Note that by Proposition 2.2.10, we have $\mu_i(P) = \mu(P)|m_{i,i+1}|$. Using Lemma 2.2.7, one can even show generally that $|\det(v_i, v_j)| = \mu(P)|m_{ij}|$.

Lemma 2.5.12. *In the above setting, we have*

$$x_k = (-1)^k \frac{m_{2k} + x_2 m_{1k}}{m_{12}}, \qquad y_k = (-1)^k \frac{y_2 m_{1k}}{m_{12}}.$$

for all $3 \leq k \leq n$.

Proof. By definition of the grading matrix, we have $QP^\top = 0$. In particular, $(x_3, \dots, x_n)$ and $(y_3, \dots, y_n)$ satisfy

$$\begin{bmatrix} w_3 & \dots & w_n \end{bmatrix} \begin{bmatrix} x_3 \\ \vdots \\ x_n \end{bmatrix} = -w_1 - x_2 w_2, \qquad \begin{bmatrix} w_3 & \dots & w_n \end{bmatrix} \begin{bmatrix} y_3 \\ \vdots \\ y_n \end{bmatrix} = -y_2 w_2.$$

Applying Cramer's rule to these systems of linear equations, we obtain

$$\begin{aligned} x_k &= \frac{\det(w_2, \dots, w_{k-1}, -w_1 - x_2 w_2, w_{k+1}, \dots, w_n)}{\det(w_3, \dots, w_n)} &&= (-1)^k \frac{m_{2k} + x_2 m_{1k}}{m_{12}} \\ y_k &= \frac{\det(w_2, \dots, w_{k-1}, -y_2 w_2, w_{k+1}, \dots, w_n)}{\det(w_3, \dots, w_n)} &&= (-1)^k \frac{y_2 m_{1k}}{m_{12}}. \end{aligned}$$

□

Now consider the Gorenstein index $\iota \in \mathbb{Z}_{\geq 1}$ of P. Recall from Proposition 2.2.14 that it is equal to the least common multiple of the local Gorenstein indices $\iota_1, \dots, \iota_n$, which in turn are given by

$$\iota_i = \frac{\det(v_i, v_{i+1})}{c_i}, \qquad c_i := \gcd(y_{i+1} - y_i, x_{i+1} - x_i).$$

Let $d_i \in \mathbb{Z}_{\geq 1}$ be the unique integers such that $c_i d_i = x_{i+1} - x_i$. Then we can prove the following identity:

Lemma 2.5.13. *In the above setting, we have*

$$c_1 m_{1n}(\iota_1 d_n + \iota_n d_1) = \iota_n((-1)^n m_{12} - m_{2n} - m_{1n}).$$

Proof. Since we assume P to be in Hermite normal form, we have $c_1\iota_1 = \det(v_1, v_2) = y_2$ and $c_1 d_1 = x_2 - 1$ as well as $c_n\iota_n = \det(v_n, v_1) = -y_n$ and $c_n d_n = x_n - 1$. Hence

$$c_1 m_{1n}(\iota_1 d_n + \iota_n d_1) = m_{1n}(d_n y_2 + \iota_n(x_2 - 1)). \tag{2.7}$$

Applying Lemma 2.5.12 to $k = n$, we obtain $y_2 m_{1n} = (-1)^n m_{12} y_n$ and $x_2 m_{1n} = (-1)^n m_{12} x_n - m_{2n}$. Substituting this into Equation (2.7), we get

$$\begin{aligned} c_1 m_{1n}(\iota_1 d_n + \iota_n d_1) &= (-1)^n d_n m_{12} y_n + \iota_n((-1)^n m_{12} x_n - m_{2n} - m_{1n}) \\ &= -(-1)^n d_n m_{12} c_n \iota_n + \iota_n((-1)^n m_{12} x_n - m_{2n} - m_{1n}) \\ &= -(-1)^n m_{12}\iota_n(x_n - 1) + \iota_n((-1)^n m_{12} x_n - m_{2n} - m_{1n}) \\ &= \iota_n((-1)^n m_{12} - m_{2n} - m_{1n}) \end{aligned}$$

□

The previous two lemmas provide everything we need for our classification algorithm: Since we start with a given free part of the grading matrix, the numbers m_{ij} are known. Lemma 2.5.12 then says that all entries of the vertex matrix can be expressed in terms of just x_2 and y_2. Since we have $0 \leq x_2 < y_2$, we only need to find a bound for y_2. But we have $y_2 = c_1\iota_1$, and since we know the Gorenstein index, we have a bound for ι_1. Hence bounding y_2 is equivalent to bounding c_1. Lemma 2.5.13 provides such a bound, since it states that c_1 is a divisor of $\iota_n((-1)^n m_{12} - m_{2n} - m_{1n})$, which again is bounded in terms of the Gorenstein index. Before we describe this classification algorithm in more detail, let us mention a few more useful properties of the involved variables:

Lemma 2.5.14. *(i)* $\gcd(\iota_1, d_1) = \gcd(\iota_n, d_n) = 1$,

(ii) $-\frac{1}{c_1} \leq d_1 < \iota_1 - \frac{1}{c_1}$,

(iii) $\gcd(\iota_1, \iota_n) \mid \frac{\iota_n((-1)^n m_{12} - m_{2n} - m_{1n})}{c_1 m_{1n}}$.

Proof. Recall that we have $c_1\iota_1 = y_2$ and $c_1 d_1 = x_2 - 1$ as well as $c_n\iota_n = -y_n$ and $c_n d_n = x_n - 1$. We obtain

$$c_1 = \gcd(y_2, x_2 - 1) = \gcd(c_1\iota_1, c_1 d_1)$$

and

$$c_n = \gcd(-y_n, 1 - x_n) = \gcd(c_n\iota_n, c_n d_n),$$

hence (i) follows. Part (ii) is just a restatement of $0 \leq x_2 < y_2$ using $x_2 = c_1 d_1 + 1$. Part (iii) is a direct consequence of Lemma 2.5.13. □

Algorithm 2.5.15 (See A.4.21 for an implementation for $n = 4$)**.** Given an LDQ pair (Q, ι) with $Q \in \mathbb{Z}^{(n-2)\times n}$, an algorithm to classify all LDP n-gons with Gorenstein index ι having Q as the free part of its grading matrix is given as follows. First, go through all divisors ι_n of ι such that m_{1n} divides $\iota_n((-1)^n m_{12} - m_{2n} - m_{1n})$. For each choice, go through all divisors c_1 of $\frac{\iota_n((-1)^n m_{12} - m_{2n} - m_{1n})}{m_{1n}}$ and set $b := \frac{\iota_n((-1)^n m_{12} - m_{2n} - m_{1n})}{c_1 m_{1n}}$. Then go through all divisors ι_1 of ι such that $\gcd(\iota_1, \iota_n)$ divides b and set $y_2 := c_1\iota_1$. Next, go through all integers d_1 such that $-\frac{1}{c_1} \leq d_1 < \iota_1 - \frac{1}{c_1}$ and set $x_2 := 1 + c_1 d_1$. We now check that the following conditions hold:

- $\gcd(\iota_1, d_1) = 1$,
- $\iota_1 \mid (b - \iota_n d_1)$ and $\gcd(\iota_n, d_n) = 1$, where $d_n := \frac{b-\iota_n d_1}{\iota_1}$,
- m_{12} divides $m_{2k} + x_2 m_{1k}$ and $y_2 m_{1k}$ for all $3 \leq k \leq n$,
- $\gcd(x_k, y_k) = 1$ for all $2 \leq k \leq n$, where for $k \geq 3$, we set

$$x_k := (-1)^k \frac{m_{2k} + x_2 m_{1k}}{m_{12}}, \qquad y_k := (-1)^k \frac{y_2 m_{1k}}{m_{12}}.$$

If any of these conditions does not hold, we ignore the choice of $(\iota_n, c_1, \iota_1, d_1)$ currently considered and continue with the next one. If the conditions all hold, we have an LDP polygon $P := \operatorname{conv}((1,0), (x_2, y_2), \ldots, (x_n, y_n))$. We then check that it truly has n vertices and has Gorenstein index ι. If so, we add it to our list of results, where we avoid duplicate entries by bringing the vertex matrices into unimodular normal form.

2.5.3 LDP triangles

In this subsection, we treat the classification of LDP triangles by Gorenstein index. Recall the two steps from the introduction to Section 2.5:

1) Given $\iota \in \mathbb{Z}_{\geq 1}$, classify all possible LDQ pairs (Q, ι) with $Q = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}$.
2) Given (Q, ι), classify all LDP triangles having Gorenstein index ι and Q as the free part of its grading matrix.

For step 2), we could use our algorithm from Section 2.5.2, which works for any number of vertices. However, for LDP triangles, there is a more efficient method by Bäuerle [11], which he used in his classification of Fano simplices in arbitrary dimension. We will repeat his method here (specialized to dimension two) and extend the classification up to Gorenstein index $45\,000$, see Classification 2.5.20.

Let us first discuss step 1). Recall from Example 2.5.9 that the Gorenstein coefficients $A = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}^\top$ of an LDP triangle with Gorenstein index ι satisfy

$$\frac{1}{\iota} = \frac{1}{a_1} + \frac{1}{a_2} + \frac{1}{a_3}. \tag{2.8}$$

Since the a_i are positive, this equation has a finite number of solutions. In fact, there are only finitely many ways in which to write a positive rational number as the sum of positive unit fractions with fixed length. Below, we describe a simple recursive algorithm for enumerating them, which turns out to be sufficient for our application.

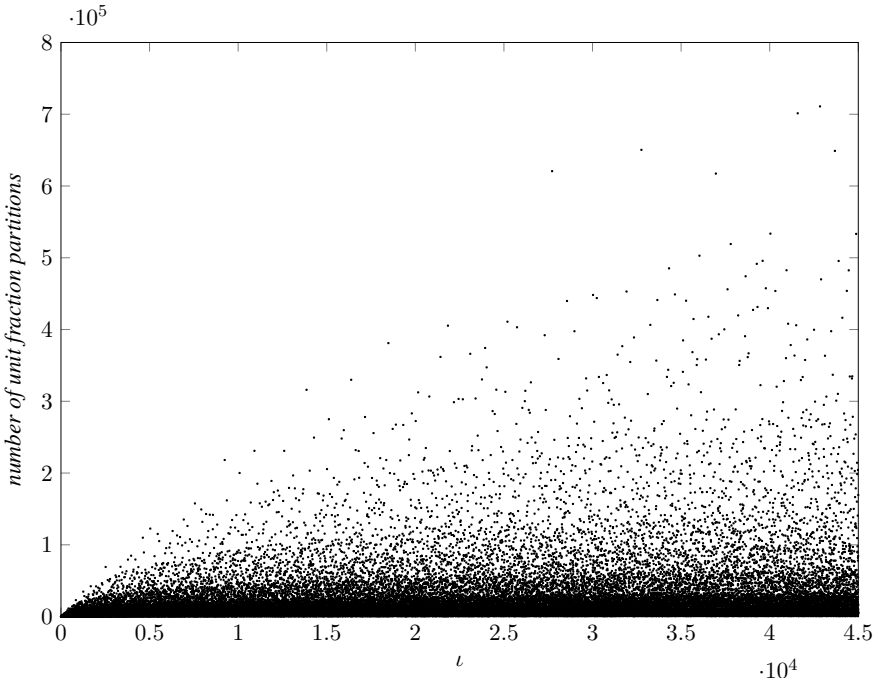
Suppose we have coprime $r, s > 1$ and want to find all integers $0 < a_1 \leq \cdots \leq a_n$ satisfying

$$\frac{r}{s} = \frac{1}{a_1} + \frac{1}{a_2} + \cdots + \frac{1}{a_n}, \tag{2.9}$$

This implies the bound $\frac{r}{s} \leq \frac{n}{a_1}$, hence $a_1 \leq \frac{ns}{r}$. The problem of finding solutions to (2.9) is thus reduced to finding solutions of $\frac{r}{s} - \frac{1}{a_1} = \frac{1}{a_2} + \cdots + \frac{1}{a_n}$, which can be handled recursively. Since we want the next term a_2 to be greater or equal to a_1, it is useful to pass a_1 into the recursive calls to be used as a lower bound for the next terms, see also the pseudocode in Algorithm 2.

Applying this to our original problem (2.8), we get the following:

Classification 2.5.16 (See A004194 on OEIS [43])**.** *Up to permutation, there are* $952\,458\,597$ *partitions* $\frac{1}{\iota} = \frac{1}{a_1} + \frac{1}{a_2} + \frac{1}{a_3}$ *for* $\iota \leq 45\,000$*. Their number develops as follows:*



This settles step 1) of our strategy. The next observation is that for LDP triangles, we can directly compute the grading matrix from the Gorenstein coefficient:

Lemma 2.5.17. *Let* $A = (a_1, a_2, a_3)$ *be a unit fraction partition of* $\frac{1}{\iota}$ *for some* $\iota \in \mathbb{Z}_{\geq 1}$*. Then* A *are the Gorenstein coefficients of* $Q = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \in \mathbb{Z}^{1\times 3}$ *with*

$$w_1 := \frac{a_2a_3}{g}, \quad w_2 := \frac{a_1a_3}{g}, \quad w_3 := \frac{a_1a_2}{g}, \qquad g := \gcd(a_2a_3, a_1a_3, a_1a_2).$$

Proof. We need to verify $\iota w = a_i w_i$ for $i = 1, 2, 3$, where $w := w_1 + w_2 + w_3$. Solving the unit fraction equation (2.8) for ι, we get

$$\iota = \frac{a_1a_2a_3}{a_2a_3 + a_1a_3 + a_1a_2}.$$

Hence we obtain,

$$\iota w = \iota \frac{a_2a_3 + a_1a_3 + a_2a_1}{g} = \frac{a_1a_2a_3}{g},$$

which equals $a_i w_i$ for all $i = 1, 2, 3$. □

Algorithm 2 Computation of unit fraction partitions

Input: A rational q, an integer n and an auxiliary variable B (zero by default).

Output: The list of ordered tuples $0 < a_1 \leq a_2 \leq ... \leq a_n$ such that $q = \frac{1}{a_1} + \cdots + \frac{1}{a_n}$.

```
1:  procedure UFP(q, n, B = 0)
2:      (r, s) := (numerator(q), denominator(q))
3:      if n = 1 or r ≤ 0 then
4:          if r = 1 and s ≥ B then
5:              return [(s)]
6:          else
7:              return []
8:          end if
9:      end if
10:     result := []
11:     for a_1 from max(⌈s/r⌉, B) to ⌊ns/r⌋ do
12:         for (a_2, ..., a_n) in UFP(q − 1/a_1, n − 1, a_1) do
13:             Add (a_1, a_2, ..., a_n) to result.
14:         end for
15:     end for
16:     return result
17: end procedure
```

We now treat step 2) of the classification. Let P be an LDP triangle with vertices $v_i = (x_i, y_i)$. We may assume the vertex matrix to be in Hermite normal form, i.e.

$$P = \begin{bmatrix} v_0 & v_1 & v_2 \end{bmatrix} = \begin{bmatrix} 1 & x_2 & x_3 \\ 0 & y_2 & y_3 \end{bmatrix}, \qquad 0 \leq x_2 < y_2$$

Let $Q = \begin{bmatrix} w_1, w_2, w_3 \end{bmatrix}$ be the free part of the grading matrix. Since $PQ^\top = 0$, we have

$$w_1 + x_2 w_2 + x_3 w_3 = 0 \tag{2.10}$$

$$y_2 w_2 + y_3 w_3 = 0. \tag{2.11}$$

By Proposition 2.2.14, the Gorenstein index is $\iota = \operatorname{lcm}(\iota_1, \iota_2, \iota_3)$, with the local Gorenstein indices $\iota_i = \mu_i / c_i$, where $c_i = \gcd(x_{i+1} - x_i, y_{i+1} - y_i)$. Specifically, we have

$$\iota_1 = \frac{y_2}{\gcd(x_2 - 1, y_2)}, \qquad \iota_2 = \frac{x_2 y_3 - x_3 y_2}{\gcd(x_3 - x_2, y_3 - y_2)}, \qquad \iota_3 = -\frac{y_3}{\gcd(x_3 - 1, y_3)}.$$

Let $b_i \in \mathbb{Z}$ be such that $\iota = b_i \iota_i$ and $d_i \in \mathbb{Z}$ such that $c_i d_i = x_{i+1} - x_i$ for $i = 1, 2, 3$. Furthermore, let $Q = (w_1, w_2, w_3)$ be the free part of the grading matrix and $A = (a_1, a_2, a_3)$ the associated Gorenstein coefficients. The following identity is the key observation for efficiently handling step 2) of our classification, as it directly relates entries of P with the Gorenstein coefficients:

Lemma 2.5.18 (Compare [11, Proposition 5.1 (i)])**.** *In the situation above, we have*

$$a_2 = -y_2(b_1 d_1 + b_3 d_3)$$

In particular, y_2 divides a_2.

Proof. We claim that $-y_2w_2(b_1d_1 + b_3d_3) = \iota w$, where $w = w_1 + w_2 + w_3$. Since the a_i are the unique integers with $\iota w = a_i w_i$, this proves the claim. We compute

$$\begin{aligned}
-y_2w_2(b_1d_1 + b_3d_3) &= -y_2w_2b_1d_1 + y_3w_3b_3d_3 \\
&= -w_2\iota_1c_1b_1d_1 - w_3\iota_3c_3b_3d_3 \\
&= -\iota(w_2c_1d_1 + w_3c_3d_3) \\
&= \iota(w_2(1 - x_2) + w_3(1 - x_3)) \\
&= \iota(w_2 + w_3 - w_2x_2 - w_3x_3) \\
&= \iota(w_1 + w_2 + w_3) \\
&= \iota w.
\end{aligned}$$

Here, we used (2.11) in the first line, $\iota_i c_i = \mu_i$ in the second, $b_i\iota_i = \iota$ in the third, $c_id_i = x_{i+1} - x_i$ in the fourth and (2.10) in the sixth. □

Algorithm 2.5.19 (See A.4.48 for an implementation)**.** Given $\iota \in \mathbb{Z}_{\geq 1}$, we can classify the LDP triangles with Gorenstein index ι as follows: First, use Algorithm 2 to compute the unit fraction partitions $\frac{1}{\iota} = \frac{1}{a_1} + \frac{1}{a_2} + \frac{1}{a_3}$. Then compute the associated grading matrices $Q = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}$ according to Lemma 2.5.17 and filter out those where the w_i are pairwise coprime. Then, go through all divisors y_2 of a_2 and all $0 \leq x_2 < y_2$ with $\gcd(x_2, y_2) = 1$. Finally, check that w_3 divides y_2w_2 and that w_2 divides $w_1 + x_3w_3$ (if it does not hold, continue with the next choice (y_2, x_2)). We set

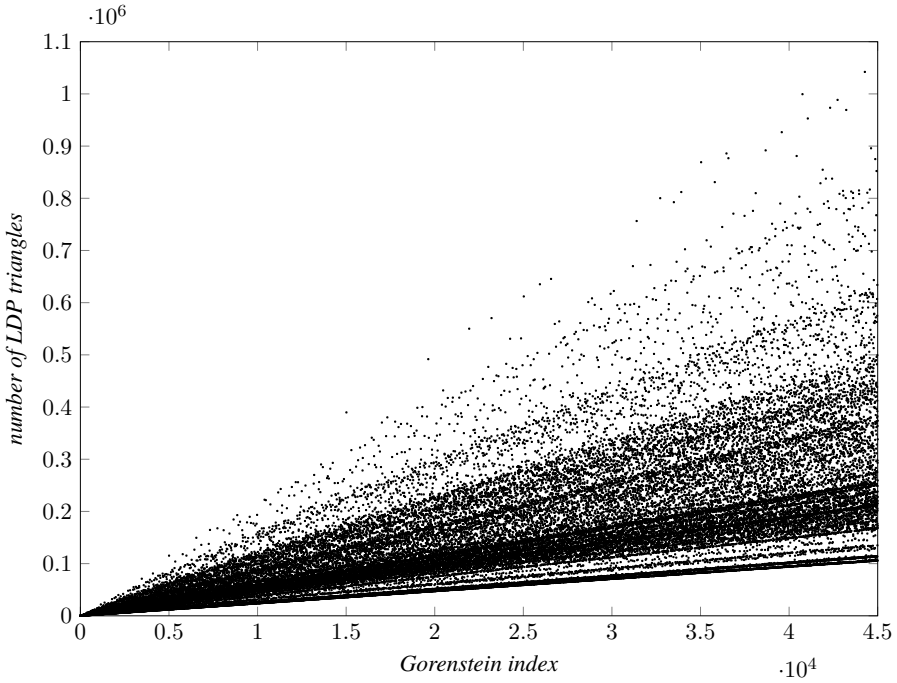
$$y_3 := -\frac{y_2w_2}{w_3}, \qquad x_3 := -\frac{w_1 + x_3w_3}{w_2}.$$

Next, check that $\gcd(x_3, y_3) = 1$ and that the LDP triangle

$$P := \begin{bmatrix} 1 & x_2 & x_3 \\ 0 & y_2 & y_3 \end{bmatrix}$$

truly has Gorenstein index ι. If so, add it to the list of results, where we avoid duplicate entries by using a unimodular normal form (see Section 1.2.1).

Classification 2.5.20. *Up to unimodular equivalence, there are* 6 497 749 373 *LDP triangles of Gorenstein index* $\leq 45\,000$*. Their numbers develop as follows:*



2.5.4 LDP quadrangles

In this subsection, we provide a classification for Gorenstein coefficients of LDP quadrangles, up to equivalence. Explicit results are obtained up to Gorenstein index 500, see Classification 2.5.32. Combining this with Algorithm 2.5.15, we arrive at our classification of LDP quadrangles from 2.5.1.

Recall from Example 2.5.10 that the Gorenstein matrix of an LDP quadrangle is of the form

$$G = \begin{bmatrix} \iota - a_{11} & \iota - a_{12} & \iota & \iota \\ \iota & \iota - a_{21} & \iota - a_{22} & \iota \\ \iota & \iota & \iota - a_{31} & \iota - a_{32} \\ \iota - a_{42} & \iota & \iota & \iota - a_{41} \end{bmatrix},$$

where $\iota \in \mathbb{Z}_{\geq 1}$ is the Gorenstein index and $A = (a_{ij}) \in \mathbb{Z}^{4\times 2}$ are the Gorenstein coefficients. By Proposition 2.5.8, all sixteen 3×3-minors of G vanish. Furthermore, applying Proposition 2.5.11 (v) and (vi) give us four equations each for $k = 1, 2, 3, 4$. We list them here for later reference:

$$a_{22}a_{32}(a_{12} - a_{11}) = a_{11}a_{21}(a_{31} - a_{32}) \tag{2.12}$$

$$a_{32}a_{42}(a_{22} - a_{21}) = a_{21}a_{31}(a_{41} - a_{42}) \tag{2.13}$$

$$a_{42}a_{12}(a_{32} - a_{31}) = a_{31}a_{41}(a_{11} - a_{12}) \tag{2.14}$$

$$a_{12}a_{22}(a_{42} - a_{41}) = a_{41}a_{11}(a_{21} - a_{22}) \tag{2.15}$$

$$a_{32}a_{42}(a_{21} - a_{12}) = a_{11}a_{21}(a_{32} - a_{41}) \tag{2.16}$$

$$a_{42}a_{12}(a_{31} - a_{22}) = a_{21}a_{31}(a_{42} - a_{11}) \tag{2.17}$$

$$a_{12}a_{22}(a_{41} - a_{32}) = a_{31}a_{41}(a_{12} - a_{21}) \tag{2.18}$$

$$a_{22}a_{32}(a_{11} - a_{42}) = a_{41}a_{11}(a_{22} - a_{31}) \tag{2.19}$$

Lemma 2.5.21. *Let $A = A(Q, \iota) = (a_{ij}) \in \mathbb{Z}^{4\times 2}$ be Gorenstein coefficients. Then*

(i) $a_{12} \leq a_{11}$ *if and only if* $a_{31} \leq a_{32}$,

(ii) $a_{22} \leq a_{21}$ *if and only if* $a_{41} \leq a_{42}$,

(iii) $a_{21} \leq a_{12}$ *if and only if* $a_{32} \leq a_{41}$,

(iv) $a_{31} \leq a_{22}$ *if and only if* $a_{42} \leq a_{11}$.

Proof. Since the a_{ij} are positive, the statements follow from equations (2.12), (2.13), (2.16) and (2.17). □

Proposition 2.5.22. *Let $A = A(Q, \iota) = (a_{ij}) \in \mathbb{Z}^{4\times 2}$ be the Gorenstein coefficients. Then there exists an $i \in \{1, 2, 3, 4\}$ such that* $\max(a_{i1}, a_{i2}) \leq 3\iota$.

Proof. By Proposition 2.5.11 (iii) and (iv), all of the following values are bounded from above by 3ι:

$$\begin{array}{llll} \min(a_{11}, a_{22}, a_{32}), & \min(a_{21}, a_{32}, a_{42}), & \min(a_{31}, a_{42}, a_{12}), & \min(a_{41}, a_{12}, a_{22}), \\ \min(a_{11}, a_{21}, a_{32}), & \min(a_{21}, a_{31}, a_{42}), & \min(a_{31}, a_{41}, a_{12}), & \min(a_{41}, a_{11}, a_{22}). \end{array}$$

We now go through the $8! = 40\,320$ possible total orderings of the eight-element set $\{a_{11}, a_{12}, a_{21}, a_{22}, a_{31}, a_{32}, a_{41}, a_{42}\}$. We find that 3 776 of them satisfy the conditions of Lemma 2.5.21. For each of them, we verify that

$$\min(\max(a_{11}, a_{12}),\ \max(a_{21}, a_{22}),\ \max(a_{31}, a_{32}),\ \max(a_{41}, a_{42}))$$

is less than or equal to

$$\begin{aligned} \max(\ &\min(a_{11}, a_{22}, a_{32}), \min(a_{21}, a_{32}, a_{42}), \min(a_{31}, a_{42}, a_{12}), \min(a_{41}, a_{12}, a_{22}) \\ &\min(a_{11}, a_{21}, a_{32}), \min(a_{21}, a_{31}, a_{42}), \min(a_{31}, a_{41}, a_{12}), \min(a_{41}, a_{11}, a_{22})). \end{aligned}$$

Since the latter is less than or equal to 3ι, we arrive at the claim. □

Definition 2.5.23. Let $A = A(Q, \iota) = (a_{ij}) \in \mathbb{Z}^{4\times 2}$ be the Gorenstein coefficients. We call A *adjusted*, if the following conditions hold:

(i) $\max(a_{11}, a_{12}) \leq \max(a_{i1}, a_{i2})$ for $i = 2, 3, 4$,

(ii) $a_{11} \leq a_{12}$,

(iii) If $a_{11} = a_{12}$, then $a_{21} \leq a_{42}$.

Clearly, every Gorenstein coefficient matrix is equivalent to an adjusted one (see 2.5.6 for the definition of equivalence). Moreover, Proposition 2.5.22 implies that for adjusted Gorenstein coefficients, we have $a_{11} \leq a_{12} \leq 3\iota$. In the rest of this section, we will explain how to bound the remaining entries of A. We split the classification into four cases:

Definition 2.5.24. Let $A = (a_{ij}) \in \mathbb{Z}^{4\times 2}$ be adjusted Gorenstein coefficients. Then A falls into one of the four following mutually exclusive types:

(I) $a_{11} < a_{12}$,

(II) $a_{11} = a_{12}$ and $a_{22} \leq a_{21}$,

(III) $a_{11} = a_{12}$ and $a_{21} < a_{22}$ and $a_{22} \leq a_{31}$,

(IV) $a_{11} = a_{12}$ and $a_{21} < a_{22}$ and $a_{31} < a_{22}$.

Example 2.5.25. Two adjusted Gorenstein coefficients can belong to different types but still be equivalent. For example,

$$\begin{bmatrix} 1 & 2 \\ 2 & 2 \\ 4 & 2 \\ 2 & 2 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 2 & 2 \\ 2 & 1 \\ 2 & 2 \\ 2 & 4 \end{bmatrix}$$

are both adjusted Gorenstein coefficients. They are equivalent, but the first is of type (I) and the second of type (II).

Generalizing the occurring unit fraction equations of the form (2.5), we arrive at the following: For $q \in \mathbb{Q}_{>0}$ and $c, d \in \mathbb{Z}_{\geq 1}$, write

$$S(q,c,d) := \left\{ (x,y) \in \mathbb{Z}^2_{\geq 1} \mid q = \frac{1}{x} + \frac{1}{y} - \frac{c}{dy} \right\}$$

for the set of solutions to the *modified unit fraction equation* with parameters q, c and d.

Lemma 2.5.26. *Let $q \in \mathbb{Q}_{>0}$ and $c, d \in \mathbb{Z}_{\geq 1}$ with $c \neq d$. Then $S(q,c,d)$ is finite.*

Proof. Let $(x,y) \in S(q,c,d)$. Since $q < \frac{1}{x} + \frac{1}{y}$, we have $\min(x,y) < \frac{2}{q}$. If $y \leq x$, we thus have $y < \frac{2}{q}$. Solving the equation for x then gives

$$x = \frac{dy}{qdy + c - d}.$$

On the other hand, if $x < y$, we have $x < \frac{2}{q}$. Using $c \neq d$, we can solve for y and get

$$y = \frac{x(d-c)}{d(qx-1)}.$$

In particular, we can enumerate explicitly the solutions to the modified unit fraction equation and there are finitely many. See also A.4.17 for an implementation. □

Lemma 2.5.27. *Let $A = (a_{ij}) \in \mathbb{Z}^{4\times 2}$ be adjusted Gorenstein coefficients of type (I). Then (a_{31}, a_{42}) is contained in the finite set $S(\frac{1}{\iota} - \frac{1}{a_{12}}, a_{11}, a_{12})$. Moreover, we have $a_{32} < a_{31}$ and*

$$a_{41} = \frac{a_{12}a_{42}(a_{32} - a_{31})}{a_{31}(a_{11} - a_{12})}, \qquad a_{21} = \frac{a_{12}a_{32}a_{42}}{a_{11}a_{41} + a_{32}a_{42} - a_{11}a_{32}},$$

$$a_{22} = \frac{a_{11}a_{21}(a_{31} - a_{32})}{a_{32}(a_{12} - a_{11})}.$$

Proof. Let $q := \frac{1}{\iota} - \frac{1}{a_{12}}$. We first show that $q > 0$. Applying Proposition 2.5.11 (i) to $k = 3$, we get

$$\frac{1}{\iota} = \frac{1}{a_{12}} + \frac{1}{a_{31}} + \frac{1}{a_{42}} - \frac{a_{11}}{a_{12}a_{42}}. \tag{2.20}$$

In particular, using $a_{11} < a_{12}$, we obtain

$$\frac{1}{\iota} > \frac{1}{a_{12}} + \frac{1}{a_{31}} + \frac{1}{a_{42}} - \frac{a_{12}}{a_{12}a_{42}} = \frac{1}{a_{12}} + \frac{1}{a_{31}} > \frac{1}{a_{12}}.$$

This shows $q > 0$. Equation (2.20) now exactly states that $(a_{31}, a_{42}) \in S(q, a_{11}, a_{12})$. Moreover, this set is finite by Lemma 2.5.26. Next, since $a_{11} < a_{12}$, Equation (2.14) implies $a_{32} < a_{31}$ as claimed. Furthermore, we get the stated formula for a_{41}. Rearranging terms in Equation (2.16), we get

$$a_{21}(a_{11}a_{41} + a_{32}a_{42} - a_{11}a_{32}) = a_{12}a_{32}a_{42}.$$

This implies both $a_{11}a_{41} + a_{32}a_{42} - a_{11}a_{32} \neq 0$ and the stated formula for a_{21}. Lastly, the formula for a_{22} follows from Equation (2.12). □

Lemma 2.5.28. *Let $A = (a_{ij}) \in \mathbb{Z}^{4\times 2}$ be adjusted Gorenstein coefficients with $a_{11} = a_{12}$. Then*

$$a_{31} = a_{32} = \frac{\iota a_{11}}{a_{11} - \iota}.$$

Proof. Substituting $a_{12} = a_{11}$ into equations (2.20) and (2.14) yields $a_{31} = \frac{\iota a_{11}}{a_{11} - \iota}$ and $a_{32} = a_{31}$ respectively. □

Lemma 2.5.29. *Let $A = (a_{ij}) \in \mathbb{Z}^{4\times 2}$ be adjusted Gorenstein coefficients of type (II). Then $a_{41} \leq a_{31}$ and $a_{22} \leq a_{31}$ as well as*

$$a_{21} = a_{11} - \frac{a_{11}a_{22}(a_{41} - a_{31})}{a_{31}a_{41}}, \qquad a_{42} = \frac{a_{21}a_{41}}{a_{22}}.$$

Proof. Note that we have $a_{12} = a_{11}$ and $a_{32} = a_{31}$ by Lemma 2.5.28. Since we assume $a_{22} \leq a_{21}$, Equation (2.13) implies $a_{41} \leq a_{42}$. Since A is adjusted, we hence get $a_{12} \leq \max(a_{21}, a_{22}) = a_{21}$ and $a_{11} \leq \max(a_{41}, a_{42}) = a_{42}$. Equations (2.16) and (2.19) then imply $a_{41} \leq a_{32} = a_{31}$ and $a_{22} \leq a_{31}$ as claimed. The formulas for a_{21} and a_{42} follow from equations (2.18) and (2.13). □

Lemma 2.5.30. *Let $A = (a_{ij}) \in \mathbb{Z}^{4\times 2}$ be adjusted Gorenstein coefficients of type (III). Then $a_{21} < a_{22} \leq a_{31}$ and*

$$a_{41} = \frac{a_{11}a_{22}a_{31}}{a_{11}a_{22} + a_{21}a_{31} - a_{11}a_{31}}, \qquad a_{42} = \frac{a_{21}a_{41}}{a_{22}}.$$

Proof. $a_{21} < a_{22} \leq a_{31}$ is the definition of type (III). The formulas for a_{41} and a_{42} follow from equations (2.18) and (2.13). □

Lemma 2.5.31. *Let $A = (a_{ij}) \in \mathbb{Z}^{4\times 2}$ be adjusted Gorenstein coefficients of type (IV). Then $a_{21} < 2\iota$ and $a_{42} < a_{11}$ as well as*

$$a_{22} = a_{31} - \frac{a_{21}a_{31}(a_{42} - a_{11})}{a_{11}a_{42}}, \qquad a_{41} = \frac{a_{22}a_{42}}{a_{21}}.$$

Proof. Applying Proposition 2.5.11 (ii) to $k = 2$ yields

$$\frac{1}{\iota} = \frac{1}{a_{21}} + \frac{1}{a_{31}} + \frac{1}{a_{42}} - \frac{a_{22}}{a_{21}a_{31}}. \tag{2.21}$$

Since $a_{11} = a_{12}$ and A is adjusted, we have $a_{21} \leq a_{42}$. Together with $a_{21} < a_{22}$, we obtain

$$\frac{1}{\iota} < \frac{1}{a_{21}} + \frac{1}{a_{42}} \leq \frac{2}{a_{21}}.$$

Hence we arrive at $a_{21} \leq 2\iota$. The bound $a_{42} < a_{11}$ follows from $a_{31} < a_{22}$ with Equation (2.17). The formulas for a_{22} and a_{41} follow from equations (2.17) and (2.15). □

Together with the initial bound $a_{11} \leq a_{12} \leq 3\iota$, Lemmas 2.5.27 - 2.5.31 provide effective bounds for all Gorenstein coefficients, according to the cases (I)-(IV). An implementation of the classification algorithm is available in `RationalPolygons.jl`, see A.4.19. We obtain the following results:

Classification 2.5.32. *Up to equivalence, there are 1 074 672 725 distinct pairs (A, ι) of Gorenstein coefficients associated to LDQ pairs (Q, ι) with $\iota \leq 500$. Their initial numbers are given as follows:*

ι	1	2	3	4	5	6	7	8	9	10	...
(I)	4	60	197	476	549	1868	937	2728	2906	4685	...
(II)	3	14	24	41	34	111	39	107	99	166	...
(III)	1	10	23	52	54	173	64	241	237	382	...
(IV)	1	9	18	37	34	113	42	133	127	207	...
Total	6	79	237	568	634	2164	1040	3108	3262	5290	...

Note that since adjusted Gorenstein coefficients of different types might be equivalent, the total is smaller than the sum of the four individual numbers.

Once we know all possible Gorenstein coefficients up to a certain Gorenstein index ι, we can compute the kernel of the associated Gorenstein matrices to get all LDQ pairs (Q, ι) (see A.4.18). Using Algorithm 2.5.15, we then obtained our classification of LDP quadrangles from 2.5.1.

CHAPTER 3

Log del Pezzo $\mathbb{C}^*$-surfaces

3.1 Background on $\mathbb{C}^*$-surfaces

We start by recalling some aspects of the geometry of $\mathbb{C}^*$-surfaces and their fixed points, the major part of which has been developed in [44–46]. A $\mathbb{C}^*$-*surface* is an irreducible, normal surface X coming with an effective morphical action $\mathbb{C}^* \times X \to X$. Let X be a projective $\mathbb{C}^*$-surface. For each point $x \in X$, the orbit map $t \mapsto t \cdot x$ extends to a morphism $\varphi_x \colon \mathbb{P}_1 \to X$. This allows one to define

$$x_0 := \varphi_x(0), \qquad x_\infty := \varphi_x(\infty).$$

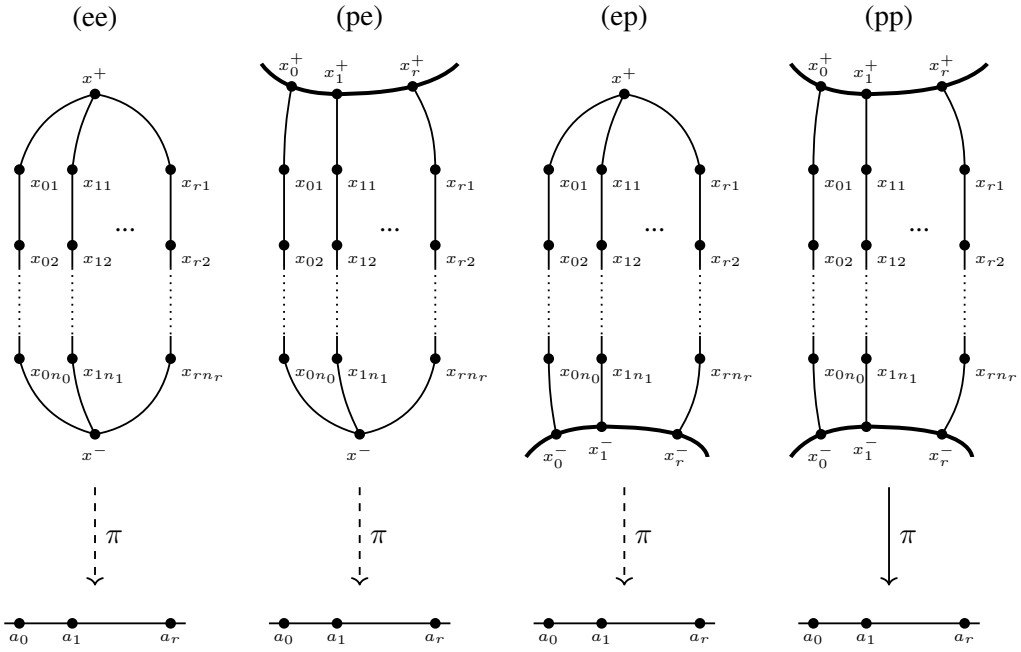
The points x_0 and x_∞ are fixed points for the $\mathbb{C}^*$-action and they lie in the closure of the orbit $\mathbb{C}^* \cdot x$. There are three types of fixed points: A fixed point is called *parabolic* (*hyperbolic*, *elliptic*), if it lies in the closure of precisely one (precisely two, infinitely many) non-trivial $\mathbb{C}^*$-orbits. Every projective $\mathbb{C}^*$-surface has a *source* and a *sink*, i.e. two irreducible components of the fixed point set $F^+, F^- \subseteq X$ such that there exist non-empty $\mathbb{C}^*$-invariant open subsets $U^+, U^- \subseteq X$ with

$$x_0 \in F^+ \text{ for all } x \in U^+, \qquad x_\infty \in F^- \text{ for all } x \in U^-.$$

The source either consists of a single elliptic fixed point x^+ or it is a smooth curve D^+ of parabolic fixed points. Similarly, the sink either consists of a single elliptic fixed point x^- or it is a smooth curve D^- of parabolic fixed points. The general $\mathbb{C}^*$-orbit connects the source with the sink. Besides the general orbits, there are special orbits whose closures are rational curves $D_{ij} \subseteq X$ forming the *arms*

$$\mathcal{A}_i := D_{i1} \cup \cdots \cup D_{in_i} \subseteq X, \qquad i = 0, \ldots, r.$$

The two curves D_{ij} and D_{ij+1} intersect uniquely in a hyperbolic fixed point x_{ij}. Moreover, we have unique intersections $\{x_i^+\} = D_{i1} \cap D^+$ and $\{x_i^-\} = D_{in_i} \cap D^-$ with parabolic fixed points $x_i^\pm$. Finally, we obtain a rational map $\pi \colon X \dashrightarrow C$ to a projective curve C given by the inclusion of the field of invariant rational functions $\mathcal{L} \subseteq \mathbb{C}(X)$. We call $a_i := \pi(\mathcal{A}_i) \in C$ the *critical values*. Here, π is defined everywhere except at the elliptic fixed points $x^\pm$. Moreover, $C = \mathbb{P}^1$ holds if and only if X is rational. According to the four possible constellations of source and sink, the geometric picture looks as follows:



In this rest of this section, we provide the combinatorial framework for rational projective $\mathbb{C}^*$-surfaces that we will use throughout this chapter. It is based on the description via Cox rings from [28, 32, 33].

3.1.1 Toric ambient spaces

In a first step, we produce in this subsection certain toric varieties that will serve as ambient spaces for our $\mathbb{C}^*$-surfaces. A crucial feature is that the $\mathbb{C}^*$-action given by the inclusion into the acting torus as the last coordinate admits a categorical quotient to projective space, see Construction 3.1.9 and Proposition 3.1.10. We start by introducing the combinatorial data, which we will call *defining triples*.

Construction 3.1.1. Fix positive integers $r, n_0, \ldots, n_r$ and set $n := n_0 + \cdots + n_r$. We start with tuples $l = (l_0, \ldots, l_r)$ and $d = (d_0, \ldots, d_r)$, where $l_i \in \mathbb{Z}^{n_i}_{\geq 1}$ and $d_i \in \mathbb{Z}^{n_i}$ such that $\gcd(l_{ij}, d_{ij}) = 1$ and

$$\frac{d_{i1}}{l_{i1}} > \cdots > \frac{d_{in_i}}{l_{in_i}}, \qquad i = 0, \ldots, r.$$

Consider the integral matrix

$$P' := \begin{bmatrix} -l_0 & l_1 & 0 & \ldots & 0 \\ -l_0 & 0 & l_2 & \ldots & 0 \\ \vdots & \vdots & & \ddots & \\ -l_0 & 0 & 0 & & l_r \\ d_0 & d_1 & d_2 & \ldots & d_r \end{bmatrix} \in \mathbb{Z}^{(r+1)\times n}.$$

Note that with the canonical basis vectors $e_1, \ldots, e_{r+1}$ of $\mathbb{Z}^{r+1}$ and $e_0 := -(e_1 + \cdots + e_r)$, the columns of P' are

$$v_{ij} := l_{ij}e_i + d_{ij}e_{r+1}.$$

Given a case $\mathfrak{c} \in \{(\text{ee}), (\text{pe}), (\text{ep}), (\text{pp})\}$, we define the associated *generator matrix*

$$(\text{ee}) \quad P := P', \qquad (\text{pe}) \quad P := \begin{bmatrix} P' & v^+ \end{bmatrix},$$

$$(\text{ep}) \quad P := \begin{bmatrix} P' & v^- \end{bmatrix}, \qquad (\text{pp}) \quad P := \begin{bmatrix} P' & v^+ & v^- \end{bmatrix},$$

where $v^+ := e_{r+1}$ and $v^- := -e_{r+1}$. We call $(\mathfrak{c}, l, d)$ a *defining triple*, if the columns of P are pairwise distinct and generate $\mathbb{Q}^{r+1}$ as a convex cone.

Definition 3.1.2. Let $\mathfrak{c} \in \{(\text{ee}), (\text{pe}), (\text{ep}), (\text{pp})\}$ be a case. The *number of parabolic fixed point curves* $m(\mathfrak{c}) \in \{0, 1, 2\}$ is defined as

$$m((\text{ee})) := 0, \qquad m((\text{pe})) := 1, \qquad m((\text{ep})) := 1, \qquad m((\text{pp})) := 2.$$

Remark 3.1.3. Let $(\mathfrak{c}, l, d)$ be a defining triple with $m := m(\mathfrak{c})$. Write $\{f_{ij}, f^{\pm}\}$ for the canonical lattice basis of $\mathbb{Z}^{n+m}$. Here, f^+ is understood to be present only in the cases (pe) and (pp), while f^- is only present for (ep) and (pp). Then we can view the generator matrix as a lattice map:

$$P\colon\ \mathbb{Z}^{n+m} \to \mathbb{Z}^{r+1}, \qquad f_{ij} \mapsto v_{ij} = l_{ij}e_i + d_{ij}e_{r+1}, \quad f^{\pm} \mapsto v^{\pm} = \pm e_{r+1}.$$

Construction 3.1.4. Let $(\mathfrak{c}, l, d)$ be a defining triple with generator matrix P. Setting $v_{i0} := v^+$ and $v_{in_i+1} := v^-$ for all i, we define the cones

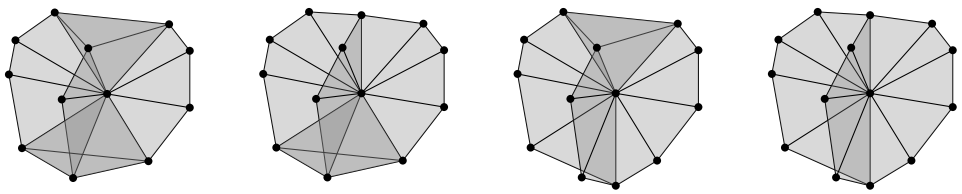
$$\sigma^+ := \operatorname{cone}(v_{01}, \ldots, v_{r1}), \qquad \sigma^- := \operatorname{cone}(v_{0n_0}, \ldots, v_{rn_r}),$$

$$\tau_{ij} := \operatorname{cone}(v_{ij}, v_{ij+1}), \quad \text{for } i = 0, \ldots, r \text{ and } j = 0, \ldots, n_i.$$

According to the case $\mathfrak{c}$, we define Σ to be the fan with the following maximal cones:

$$\begin{array}{lccccccc}
(\text{ee}) & \{\sigma^+\} & \cup & \{\tau_{i1}, \ldots, \tau_{in_i-1}\ ;\ i = 0, \ldots, r\} & \cup & \{\sigma^-\}, \\
(\text{pe}) & \{\tau_{00}, \ldots, \tau_{r0}\} & \cup & \{\tau_{i1}, \ldots, \tau_{in_i-1}\ ;\ i = 0, \ldots, r\} & \cup & \{\sigma^-\}, \\
(\text{ep}) & \{\sigma^+\} & \cup & \{\tau_{i1}, \ldots, \tau_{in_i-1}\ ;\ i = 0, \ldots, r\} & \cup & \{\tau_{0n_0}, \ldots, \tau_{rn_r}\}, \\
(\text{pp}) & \{\tau_{00}, \ldots, \tau_{r0}\} & \cup & \{\tau_{i1}, \ldots, \tau_{in_i-1}\ ;\ i = 0, \ldots, r\} & \cup & \{\tau_{0n_0}, \ldots, \tau_{rn_r}\}.
\end{array}$$

Note that Σ is a non-degenerate simplicial lattice fan in $\mathbb{Z}^{r+1}$ with the columns of P as primitive ray generators. As such, it defines a $\mathbb{Q}$-factorial $(r+1)$-dimensional toric variety Z_Σ. Below are schematic pictures of the four types of fans just defined. Note that the fans, and hence the toric varieties, are *not* complete.



(ee) (pe) (ep) (pp)

The following is a replay of Cox's quotient construction (see Construction 2.1.3) applied to the toric varieties from above.

Construction 3.1.5. Let $Z = Z_\Sigma$ be a toric variety from Construction 3.1.4. Writing $Q\colon\ \mathbb{Z}^{n+m} \to K$ for the cokernel of $P^\top$, Proposition 2.1.1 tells us that $\mathrm{Cl}(Z) \cong K$. Moreover, the Cox ring of Z is the polynomial ring in $n+m$ variables:

$$\mathbb{C}[T_{ij}, S^\pm] \ := \ \mathbb{C}[T_{ij}, S^\pm \ ; \ i = 0, \ldots, r, j = 1, \ldots, n_i].$$

Here, we understand S^+ to be present only in the cases (pe) and (pp) and S^- only in the cases (ep) and (pp). The Cox ring comes with a K-grading given by

$$\deg(T_{ij}) \ := \ Q(f_{ij}), \quad \deg(S^+) \ := \ Q(f^+), \quad \deg(S^-) \ := \ Q(f^-).$$

Geometrically, this means we have an action of the quasitorus $H := \operatorname{Spec} \mathbb{C}[K]$ on $\overline{Z} := \mathbb{C}^{n+m}$. To any cone $\sigma \in \Sigma$, we associate a face of the positive orthant $\mathbb{Q}^{n+m}_{\geq 0}$ by

$$\hat{\sigma} \ := \ \mathrm{cone}(f_{ij}, f^\pm \ ; \ v_{ij} \in \sigma, v^\pm \in \sigma).$$

This defines a toric variety $\hat{Z} \subseteq Z$ associated to the fan $\hat{\Sigma} := \{\hat{\sigma} \ ; \ \sigma \in \Sigma\}$. The generator matrix defines a map of lattice fans $P\colon\ (\mathbb{Z}^{n+m}, \hat{\Sigma}) \to (\mathbb{Z}^{r+1}, \Sigma)$. The associated toric morphism $p\colon\ \hat{Z} \to Z$ is a good quotient by the action of the quasitorus H.

Remark 3.1.6. As with any toric variety, Cox's quotient map $p\colon\ \hat{Z} \to Z$ allows us to describe points on Z using explicit coordinates: We identify a point $z \in Z$ with its *Cox coordinates* $[x_{ij}, x^\pm] := p(x_{ij}, x^\pm)$, where

$$(x_{ij}, x^\pm) \ \in \ p^{-1}(z) \ \subseteq \ \hat{Z} \ \subseteq \ \mathbb{C}^{n+m}.$$

In Cox coordinates, the distinguished points z_σ associated to cones $\sigma \in \Sigma$ are $z_\sigma = [x^\sigma_{ij}, x^\sigma_\pm]$, where

$$x^\sigma_{ij} := \begin{cases} 0, & v_{ij} \in \sigma \\ 1, & v_{ij} \notin \sigma \end{cases}, \qquad x^\sigma_+ := \begin{cases} 0, & v^+ \in \sigma \\ 1, & v^+ \notin \sigma \end{cases}, \qquad x^\sigma_- := \begin{cases} 0, & v^- \in \sigma \\ 1, & v^- \notin \sigma \end{cases}.$$

Definition 3.1.7. Let $Z = Z_\Sigma$ be a toric variety from Construction 3.1.4. We write $\rho_{ij} := \mathrm{cone}(v_{ij})$ and $\rho^\pm := \mathrm{cone}(v^\pm)$ for the rays of Σ. Each ray defines a torus-invariant prime divisor

$$D^{ij}_Z \ := \ \overline{\mathbb{T}^{r+1} \cdot z_{\rho_{ij}}}, \qquad D^\pm_Z \ := \ \overline{\mathbb{T}^{r+1} \cdot z_{\rho^\pm}},$$

where the $z_{\rho_{ij}}, z_{\rho^\pm}$ are the distinguished points associated to the rays of Σ.

Remark 3.1.8. In terms of Cox coordinates $z = [x_{ij}, x^\pm]$, we have

$$\begin{aligned} z \in D^{ij}_Z \ &\iff \ \text{only } x_{ij} = 0, \\ z \in D^+_Z \ &\iff \ \text{only } x^+ = 0, \\ z \in D^-_Z \ &\iff \ \text{only } x^- = 0. \end{aligned}$$

Construction 3.1.9. Let $Z = Z_\Sigma$ be a toric variety from Construction 3.1.4. Consider the projection $B\colon\ \mathbb{Z}^{r+1} \to \mathbb{Z}^r$ onto the first r coordinates. We obtain a commutative diagram

$$\begin{array}{ccccc} & & \mathbb{Z}^{n+m} & & \\ & \swarrow^{P} & & \searrow^{P_0} & \\ \mathbb{Z}^{r+1} & & \xrightarrow{\qquad B \qquad} & & \mathbb{Z}^r, \end{array}$$

where as a matrix, P_0 consists of the upper r rows of P. Writing $Q_0\colon\ \mathbb{Z}^{n+m} \to K_0$ for the cokernel of P_0, we obtain a commutative diagram of short exact sequences:

$$\begin{array}{ccccccccc} & & & & 0 & & & & \\ & & & & \downarrow & & & & \\ & & 0 & & \tilde{K} & & & & \\ & & \downarrow & & \downarrow & & & & \\ 0 & \longrightarrow & \mathbb{Z}^r & \xrightarrow{P_0^\top} & \mathbb{Z}^{n+m} & \xrightarrow{Q_0} & K_0 & \longrightarrow & 0 \\ & & \downarrow{\scriptstyle B^\top} & & \| & & \downarrow & & \\ 0 & \longrightarrow & \mathbb{Z}^{r+1} & \xrightarrow{P^\top} & \mathbb{Z}^{n+m} & \xrightarrow{Q} & K & \longrightarrow & 0 \\ & & \downarrow & & & & \downarrow & & \\ & & \mathbb{Z} & & & & 0 & & \\ & & \downarrow & & & & & & \\ & & 0 & & & & & & \end{array}$$

Recall from Construction 3.1.5 that the Cox ring $\mathbb{C}[T_{ij}, S^{\pm}]$ of Z is K-graded. We also obtain a K_0-grading by

$$\deg(T_{ij}) \ := \ Q_0(f_{ij}), \qquad \deg(S^+) \ := \ Q_0(f^+), \qquad \deg(S^-) \ := \ Q_0(f^-).$$

Geometrically, this means $H_0 := \operatorname{Spec}[K_0]$ acts on $\hat{Z}$. Passing to the quotient, we obtain an action of $T := H_0/H = \operatorname{Spec}[\tilde{K}]$. on Z. By the snake lemma, we may identify $\tilde{K}$ with the cokernel of $B^\top$. This implies $T \cong \mathbb{C}^*$. Moreover, the action of T is given by the inclusion $\mathbb{C}^* \hookrightarrow \mathbb{T}^{r+1}$ into the acting torus as the last coordinate, i.e.

$$T \times Z \to Z, \qquad (t, z) \mapsto (1, \ldots, 1, t) \cdot z.$$

Proposition 3.1.10. *The $\mathbb{C}^*$-action from Construction 3.1.9 admits a categorical quotient $\pi\colon\ Z_0 \to \mathbb{P}_r$, where $Z_0 \subseteq Z$ is an open toric subset and π is given in terms of Cox coordinates by*

$$\pi\colon\ Z_0 \to \mathbb{P}_r, \qquad [x_{ij}, x^{\pm}] \mapsto [x_0^{l_0}, \ldots, x_r^{l_r}],$$

where $x_i^{l_i} := x_{i1}^{l_{i1}} \cdots x_{in_i}^{l_{in_i}}$. In particular, a toric prime divisor D_Z^{ij} is sent into the projective hyperplane $V_{\mathbb{P}_r}(U_i)$.

Proof. Let Σ_0 be the subfan of Σ having as maximal cones the rays of Σ, i.e. $\rho_{ij} = \operatorname{cone}(v_{ij})$ and $\rho^{\pm} := \operatorname{cone}(v^{\pm})$. Denote by $B\colon\ \mathbb{Z}^{r+1} \to \mathbb{Z}^r$ the projection onto the first r coordinates. Then by definition, we have $B(v_{ij}) = l_{ij}e_i$ and $B(v^{\pm}) = 0$, where $e_1, \ldots, e_r$ are the canonical basis of $\mathbb{Z}^r$ and $e_0 := -(e_1 + \cdots + e_r)$. Hence B is a map of lattice fans $(\mathbb{Z}^{r+1}, \Sigma_0) \to (\mathbb{Z}^r, \Sigma_{\mathbb{P}_r})$, which defines a toric morphism $\pi\colon\ Z_0 \to \mathbb{P}_r$. Moreover, since by Construction 3.1.9, the $\mathbb{C}^*$-action on Z corresponds to the kernel of B, the map π is a categorical quotient. Setting $\hat{Z}_0 := p^{-1}(Z)$, we have a commutative diagram of toric morphisms

$$\begin{array}{ccc} & \hat{Z}_0 & \\ {\scriptstyle p}\swarrow & & \searrow{\scriptstyle p_0} \\ Z_0 & \xrightarrow{\quad\pi\quad} & \mathbb{P}_r, \end{array}$$

where p, p_0 and π are the toric morphisms associated to P, P_0 and B respectively. By definition, p_0 sends a point $(x_{ij}, x^{\pm})$ to $[x_0^{l_0}, \ldots, x_r^{l_r}]$. This proves the given presentation of π in terms of Cox coordinates. For the supplement, use Remark 3.1.8. □

We end this section by showing that the l_{ij} encode the orders of the isotropy group of the $\mathbb{C}^*$-action:

Proposition 3.1.11. *Consider the $\mathbb{C}^*$-action on a toric variety Z as in Construction 3.1.9. For the isotropy groups $T_x \leq \mathbb{C}^*$, we have*

(i) $T_x \cong \mathbb{Z}/l_{ij}\,\mathbb{Z}$ *for* $x \in \mathbb{T}^{r+1} \cdot z_{\rho_{ij}}$,

(ii) $T_x \cong \mathbb{C}^*$ *for* $x \in \mathbb{T}^{r+1} \cdot z_{\rho^{\pm}}$.

Proof. It is enough to show that the isotropy groups of the distinguished points $z_{\rho_{ij}}$ and $z_{\rho^{\pm}}$ are $\mathbb{Z}/l_{ij}\,\mathbb{Z}$ and $\mathbb{C}^*$ respectively. We may consider the $\mathbb{C}^*$-action locally on the affine toric pieces Z_δ, where $\delta \leq \mathbb{Z}^{r+1}$ is a pointed cone. Recall that the $\mathbb{C}^*$-action is given by the inclusion

$$\mathbb{C}^* \;=\; \operatorname{Spec}\mathbb{C}[\mathbb{Z}] \;\hookrightarrow\; \operatorname{Spec}\mathbb{C}[\mathbb{Z}^{r+1}] \;=\; \mathbb{T}^{r+1}$$

associated to the projection $Q\colon\ \mathbb{Z}^{r+1} \to \mathbb{Z}$ onto the last coordinate. Proposition 2.1.4.2 of [2] tells us that, the inclusion of the isotropy group $T_{z_\delta} \leq \mathbb{C}^*$ is given by the projection $\mathbb{Z} \to K_\delta$, where

$$K_\delta \;\cong\; (\{0\}^r \times \mathbb{Z}) \cap \operatorname{lin}_{\mathbb{Q}}(\delta) \;\oplus\; B(\operatorname{lin}_{\mathbb{Q}}(\delta) \cap \mathbb{Z}^r)/B(\operatorname{lin}_{\mathbb{Q}}(\delta) \cap \mathbb{Z}^{r+1}).$$

Here, $B\colon\ \mathbb{Z}^{r+1} \to \mathbb{Z}^r$ is the projection onto the first r coordinates. For $\delta = \rho_{ij}$, the left summand is zero, while the right summand is

$$\mathbb{Z}\,e_i/B(\rho_{ij}) \;\cong\; \mathbb{Z}/l_{ij}\,\mathbb{Z}\,.$$

On the other hand, for $\delta = \rho^{\pm}$, the right summand is zero and the left summand is $\mathbb{Z}$. In particular, the isotropy group is $\mathbb{C}^*$. □

3.1.2 $\mathbb{C}^*$-surfaces from defining data

In the previous section, we studied certain toric varieties Z_Σ built from defining triples. We have seen that these come naturally with a $\mathbb{C}^*$-action given by the inclusion into the acting torus as the last coordinate. In this section, we start with one additional piece of data, namely a complex $2 \times (r+1)$-matrix A and construct a $\mathbb{C}^*$-surface $X \subseteq Z$. This will yield all normal rational projective $\mathbb{C}^*$-surfaces, see Proposition 3.1.17.

Construction 3.1.12. Let $(\mathfrak{c}, l, d)$ be a defining triple and $Z = Z_\Sigma$ the associated toric variety from Construction 3.1.4. Recall that its Cox ring is the K-graded polynomial ring $\mathbb{C}[T_{ij}, S^{\pm}]$, where $Q\colon \mathbb{Z}^{n+m} \to K$ is the cokernel of $P^\top$. Given a complex matrix $A = \begin{bmatrix} a_0 & \ldots & a_r \end{bmatrix} \in \mathbb{C}^{2\times(r+1)}$ with pairwise linearly independent columns, we define trinomials

$$g_i \;:=\; \det \begin{bmatrix} T_i^{l_i} & T_{i+1}^{l_{i+1}} & T_{i+2}^{l_{i+2}} \\ a_i & a_{i+1} & a_{i+2} \end{bmatrix} = \alpha_{i+1,i+2} T_i^{l_i} - \alpha_{i,i+2} T_{i+1}^{l_{i+1}} + \alpha_{i,i+1} T_{i+2}^{l_{i+2}} \;\in\; \mathbb{C}[T_{ij}, S_k],$$

where $T_i^{l_i} := T_{i1}^{l_{i1}} \cdots T_{in_i}^{l_{in_i}}$ and $\alpha_{ii'} := \det(a_i, a_{i'}) \neq 0$. We claim that with respect to the K-grading, the g_i are all homogeneous and of the same degree. Hence we can define the K-graded $\mathbb{C}$-algebra:

$$R \;:=\; R(\mathfrak{c}, l, d; A) \;:=\; \mathbb{C}[T_{ij}, S_k]/\langle g_0, \ldots, g_{r-2}\rangle.$$

Proof. We verify that the g_i are homogeneous and of the same degree. Write $w_{ij} := Q(f_{ij}) \in K$. Then the degree of a monomial in g_i is given by

$$\deg(T_i^{l_i}) = l_{i1} w_{i1} + \cdots + l_{in_i} w_{in_i}.$$

Since $QP^\top = 0$, we have

$$l_{i1} w_{i1} + \cdots + l_{in_i} w_{in_i} - (l_{01} w_{01} + \cdots + l_{0n_0} w_{0n_0}) = 0.$$

for all $i = 1, \ldots, r$. Thus the degrees $\deg(T_i^{l_i})$ are all equal, which proves the claim. □

Construction 3.1.13. We continue Construction 3.1.12. Let $p\colon \hat{Z} \to Z$ be Cox's quotient presentation from Construction 3.1.5. We may identify the total coordinate space with $\overline{Z} = \mathbb{C}^{n+m} = \operatorname{Spec} \mathbb{C}[T_{ij}, S_k]$. Consider the affine variety

$$\overline{X} \;:=\; V(g_0, \ldots, g_{r-2}) \;=\; \operatorname{Spec} R \;\subseteq\; \overline{Z}.$$

We obtain a commutative diagram

$$\begin{array}{ccc} \overline{X} & \hookrightarrow & \overline{Z} \\ \cup & & \cup \\ \hat{X} & \hookrightarrow & \hat{Z} \\ \downarrow p & & \downarrow p \\ X & \hookrightarrow & Z \end{array}$$

where we set $\hat{X} := \overline{X} \cap \hat{Z}$ and

$$X \;:=\; X(\mathfrak{c}, l, d; A) \;:=\; p(\hat{X}) \;\subseteq\; Z$$

Here, X is an irreducible normal surface and it is left invariant by the $\mathbb{C}^*$-action on Z. In particular, it is a $\mathbb{C}^*$-surface.

Proof. For the fact that X is irreducible and normal, we refer to [2, Theorem 3.2.1.4]. Next, we show that X is left invariant by the $\mathbb{C}^*$-action on Z. Working in the notation of Construction 3.1.9, the action is given by the quotient $H_0/H \cong \mathbb{C}^*$, where $H = \operatorname{Spec} \mathbb{C}[K]$ and $H_0 = \operatorname{Spec} \mathbb{C}[K_0]$. Analogous to the above argument, we can see that the trinomials g_i are even homogeneous with respect to the finer K_0-grading on $\mathbb{C}[T_{ij}, S^{\pm}]$. Geometrically, this means $\hat{X}$ is left invariant by the action of H_0. Passing to the quotient, we get that X is left invariant by $H_0/H \cong \mathbb{C}^*$. □

Definition 3.1.14. Let $X \subseteq Z$ be a $\mathbb{C}^*$-surface arising from Construction 3.1.13. Restricting the toric prime divisors from Definition 3.1.7, we obtain prime divisors

$$D_X^{ij} \;:=\; D_Z^{ij} \cap X, \qquad D_X^{\pm} \;:=\; D_Z^{\pm} \cap Z.$$

Proposition 3.1.15. *Let $X = X(\mathfrak{c}, l, d; A) \subseteq Z$ be a $\mathbb{C}^*$-surface arising from Construction 3.1.13. Then there exists an open subset X_0 such that the $\mathbb{C}^*$-action admits a categorical quotient*

$$X_0 \to \mathbb{P}_1\,.$$

Moreover, the prime divisors D_X^{ij} are sent to $[a_i] \in \mathbb{P}_1$, where $a_i \in \mathbb{C}^2$ is the i-th column of A.

Proof. Consider the categorical quotient $\pi\colon Z_0 \to \mathbb{P}_r$ from Proposition 3.1.10 and set $X_0 := Z_0 \cap X$. Using the presentation of π in terms of Cox coordinates, we see that $\pi(X_0)$ is cut out by linear equations $h_0, \ldots, h_{r-2}$, where

$$h_i \;:=\; \begin{bmatrix} U_i & U_{i+1} & U_{i+2} \\ a_i & a_{i+1} & a_{i+2} \end{bmatrix} \;=\; \alpha_{i+1,i+2} U_i - \alpha_{i,i+2} U_{i+1} + \alpha_{i,i+1} U_{i+2}.$$

Here, we have written $U_0, \ldots, U_r$ for the variables in projective space and $\alpha_{ii'} := \det(a_i, a_{i'})$. We can parameterize this image by

$$\iota\colon \mathbb{P}_1 \hookrightarrow \mathbb{P}_r, \qquad x \mapsto [\det(x, a_0), \ldots, \det(x, a_r)].$$

Indeed, $\iota(\mathbb{P}_1) \subseteq \pi(X_0)$ is verified by direct computation and since both sides are one-dimensional and connected, we get equality. Hence we obtain the quotient $\iota^{-1} \circ \pi\colon X_0 \to \mathbb{P}_1$. The fact that D_X^{ij} are mapped to $[a_i] \in \mathbb{P}_1$ follows from

$$\iota([a_i]) = [\alpha_{i0}, \ldots, \alpha_{ir}] \in V_{\mathbb{P}_r}(U_i) = \pi(D_Z^{ij}).$$

□

Proposition 3.1.16. *Let $X = X(\mathfrak{c}, l, d; A) \subseteq Z$ arise from Construction 3.1.13. Then X is a normal rational projective $\mathbb{C}^*$-surface. Moreover, its Cox ring is $R(\mathfrak{c}, l, d; A)$ and we have an isomorphism between divisor class groups:*

$$\mathrm{Cl}(Z) \to \mathrm{Cl}(X), \qquad [D] \mapsto [D \cap X].$$

Proof. Most claims follow from [2, Theorem 3.4.3.7]. Rationality of X follows from the existence of a birational quotient $X \dashrightarrow \mathbb{P}_1$, see Proposition 3.1.15. Finally, projectivity follows from the assumption that the columns of the generator matrix P generate $\mathbb{Q}^{r+1}$ as a cone. □

Proposition 3.1.17 (See [2, Theorem 5.4.1.5])**.** *Every normal projective rational $\mathbb{C}^*$-surface is isomorphic to some $X(\mathfrak{c}, l, d; A)$ from Construction 3.1.13.*

Remark 3.1.18. For $r = 1$, Construction 3.1.13 produces precisely the projective toric surfaces. Indeed, starting with any projective toric surface Z, we can renumber the primitive ray generators $v_1, \ldots, v_n$ such that $v_i = (-l_{0i}, d_{0i})$ for $i = 1, \ldots, n_0$ and $v_{n_0+i} = (l_{1i}, d_{1n_i})$ for $i = 1, \ldots, n_1$, where l_{ij} are positive integers and $d_{ij} \in \mathbb{Z}$. According to the existence of $v^+ = (0, 1)$ and $v^- = (0, -1)$ among the primitive ray generators, we can define an appropriate case $\mathfrak{c}$ and obtain $Z \cong Z(\mathfrak{c}, l, d)$. Let $A \in \mathbb{C}^{2\times 2}$ be the identity matrix. Since $r = 1$, there are no relations in the Cox ring $R(\mathfrak{c}, l, d; A)$, hence $Z \cong X(\mathfrak{c}, l, d; A)$.

Proposition 3.1.19. *Let $X \subseteq Z$ be a $\mathbb{C}^*$-surface arising from Construction 3.1.13. Then for the isotropy groups $T_x \leq \mathbb{C}^*$ of the $\mathbb{C}^*$-action, we have*

(i) $T_x \cong \mathbb{Z}/l_{ij}\,\mathbb{Z}$ *for a generic* $x \in D^{ij}_X$,

(ii) $T_x \cong \mathbb{C}^*$ *for a generic* $x \in D^{\pm}_X$.

Proof. Since the $\mathbb{C}^*$-action on X comes from the $\mathbb{C}^*$-action on Z from Construction 3.1.9, the statements follow directly from Proposition 3.1.11. □

3.2 The isomorphy problem and a normal form

When classifying $\mathbb{C}^*$-surfaces using the combinatorial framework from Section 3.1, one needs to be able to decide when two sets of defining data give rise to isomorphic $\mathbb{C}^*$-surfaces. We call this the *isomorphy problem*. Recall that in the toric case, two toric del Pezzo surfaces are isomorphic if and only if their LDP polygons are equivalent by a unimodular transformation. Thus, the isomorphy problem for toric del Pezzo surfaces is solved by the unimodular normal form, see Section 1.2.1. For $\mathbb{C}^*$-surfaces, the combinatorics is more involved. The goal is to find operations on the defining data leaving the isomorphy type of the $\mathbb{C}^*$-surface invariant (called *admissible operations*), and then show that these are *complete* in the sense that two $\mathbb{C}^*$-surfaces are equivalent if and only if their defining data are equivalent by admissible operations.

This section is organized as follows. We first discuss operations on the coefficient matrix in Subsection 3.2.1, followed by operations on defining triples in Subsection 3.2.2. In Subsection 3.2.3,

we discuss the possibility of erasing certain entries of the defining triple, while keeping the isomorphy type of the $\mathbb{C}^*$-surface invariant. In Subsection 3.2.4, we prove completeness of admissible operations. Finally, in Subsection 3.2.5, we introduce a normal form for defining triples, which allows us to efficiently check when two $\mathbb{C}^*$-surfaces are isomorphic. An implementation of the normal form is available in the Julia package `CStarSurfaces.jl`, see Section B.2.3 for details.

Parts of this section have been published in shortened form as Section 6 of the joint work with Hättig and Hausen [27].

3.2.1 A standard form for the coefficient matrix

Given a defining triple $(\mathfrak{c}, l, d)$ and a matrix A as in Construction 3.1.12, we discuss operations on the coefficient matrix A that leave the isomorphy type of the graded $\mathbb{C}$-algebra $R(\mathfrak{c}, l, d; A)$ invariant. This leads to a standard form for the coefficient matrices, see Proposition 3.2.4.

Definition 3.2.1. Let $r \in \mathbb{Z}_{\geq 1}$. By a *coefficient matrix*, we mean a complex matrix $A \in \mathbb{C}^{2\times(r+1)}$ with pairwise linearly independent columns. We call two coefficient matrices A and A' *equivalent* and write $A \sim A'$, if $A' = UAD$ for some $U \in \mathrm{GL}(2, \mathbb{C})$ and a diagonal matrix $D = \mathrm{diag}(d_0, \ldots, d_r) \in \mathrm{GL}(r+1, \mathbb{C})$.

Proposition 3.2.2. *Let $(\mathfrak{c}, l, d)$ be a defining triple, and let A and A' be equivalent coefficient matrices. Then there exists an isomorphism of graded $\mathbb{C}$-algebras:*

$$R(\mathfrak{c}, l, d; A) \;\cong\; R(\mathfrak{c}, l, d; A').$$

Proof. Let $R = R(\mathfrak{c}, l, d; A)$ and $R' = R(\mathfrak{c}, l, d; A')$. We work in the notation of Construction 3.1.12. First, assume $A' = UA$ for a matrix $U \in \mathrm{GL}(2, \mathbb{C})$. Then $g'_i = \det(U) g_i$, which implies $R = R'$. Now consider $A' = AD$ where $D = \mathrm{diag}(d_0, \ldots, d_r) \in \mathrm{GL}(r+1, \mathbb{C})$ is a diagonal matrix. Here, we find:

$$g'_i = d_{i+1}d_{i+2}\alpha_{i+1,i+2}T_i^{l_i} - d_i d_{i+2}\alpha_{i,i+2}T_{i+1}^{l_{i+1}} + d_i d_{i+1}\alpha_{i,i+1}T_{i+2}^{l_{i+2}}.$$

Define the morphism $\varphi\colon\ \mathbb{C}[T_{ij}, S^{\pm}] \to \mathbb{C}[T_{ij}, S^{\pm}]$ such that it maps T_{i1} to $\prod_{k\neq i} \sqrt[l_i]{d_k} \cdot T_{i1}$, keeping other variables unchanged. Hence we have

$$\varphi(T_i^{l_i}) = \left(\prod_{k\neq i} d_k\right) T_i^{l_i},$$

and thus $\varphi(g_i) = \left(\prod_{k\notin\{i,i+1,i+2\}} d_k\right) g'_i$. In particular, φ sends g_i to a scalar multiple of g'_i. Since φ respects the K-grading by merely rescaling variables, it defines an isomorphism of K-graded algebras $R \cong R'$. □

Definition 3.2.3. A coefficient matrix $A \in \mathbb{C}^{2\times(r+1)}$ is said to be in *standard form* if

$$A \;=\; \begin{bmatrix} 1 & 0 & -1 & -1 & \ldots & -1 \\ 0 & 1 & -1 & \lambda_1 & \ldots & \lambda_{r-2} \end{bmatrix}$$

for some pairwise distinct $\lambda_i \in \mathbb{C}\setminus\{0,-1\}$.

Proposition 3.2.4. *Let $r \in \mathbb{Z}_{\geq 1}$ and let $A \in \mathbb{C}^{2\times(r+1)}$ have pairwise linearly independent columns. Then A is equivalent to a unique matrix in standard form.*

Proof. Write $A = \begin{bmatrix} a_0 & a_1 & \dots & a_r \end{bmatrix}$ with $a_i = (a_{i1}, a_{i2}) \in \mathbb{C}^2$. Multiplying from the left with an invertible matrix $U \in \mathrm{GL}(2,\mathbb{C})$, we achieve $a_0 = (1,0)$ and $a_1 = (0,1)$. Since the a_i are pairwise linearly independent, we may assume $a_{i1}, a_{i2} \neq 0$ for $i = 2,\dots,r$. Hence we may scale the last $r-2$ columns to achieve $a_{21} = \cdots = a_{r1} = -1$. We are left with

$$A = \begin{bmatrix} 1 & 0 & -1 & -1 & \dots & -1 \\ 0 & 1 & \lambda_0 & \lambda_1 & \dots & \lambda_{r-2} \end{bmatrix}$$

where $\lambda_i \in \mathbb{C}\setminus\{0\}$. Multiplying the second row by $-\frac{1}{\lambda_0}$ and the second column by $-\lambda_0$, we achieve standard form. Linear independence of the columns guarantees $\lambda_i \neq 0, -1$. □

Corollary 3.2.5. *Two coefficient matrices are equivalent if and only if they have the same standard form.*

Example 3.2.6. The converse of Proposition 3.2.2 is not true generally: Take the defining triple $(\mathfrak{c}, l, d)$ where

$$\mathfrak{c} := (\mathrm{ee}), \qquad l := ((1,1),(1,1),(2),(2)), \qquad d := ((0,-1),(0,-2),(1),(1)).$$

The generator matrix is

$$P = \begin{bmatrix} -1 & -1 & 1 & 1 & 0 & 0 \\ -1 & -1 & 0 & 0 & 2 & 0 \\ -1 & -1 & 0 & 0 & 0 & 2 \\ 0 & -1 & 0 & -2 & 1 & 1 \end{bmatrix}$$

It is readily verified that the columns of P generate $\mathbb{Q}^4$ as a cone, hence this is a valid defining triple. Now consider the coefficient matrices:

$$A := \begin{bmatrix} 1 & 0 & -1 & -1 \\ 0 & 1 & -1 & -2 \end{bmatrix}, \qquad A' := \begin{bmatrix} 1 & 0 & -1 & -1 \\ 0 & 1 & -1 & -\frac{1}{2} \end{bmatrix}.$$

Both are in standard form, hence they are not equivalent. However, the $\mathbb{C}^*$-surfaces $X(\mathfrak{c}, l, d; A)$ and $X(\mathfrak{c}, l, d; A')$ are isomorphic. Indeed, swapping the variables T_{21} and T_{31} gives a morphism of $\mathbb{C}$-algebras

$$R(\mathfrak{c}, l, d; A) \to R(\mathfrak{c}, l, d; A''),$$

where A'' arises from A by swapping the last two columns, i.e.

$$A'' := \begin{bmatrix} 1 & 0 & -1 & -1 \\ 0 & 1 & -2 & -1 \end{bmatrix}.$$

Bringing A'' into standard form, we see that $A'' \sim A'$, hence $R(\mathfrak{c}, l, d; A)$ and $R(\mathfrak{c}, l, d; A')$ are indeed isomorphic.

Remark 3.2.7. Example 3.2.6 shows that to get a converse statement for Proposition 3.2.2, it is not enough to consider just operations on the coefficient matrix alone. Instead, one needs to treat operations on the defining triple and on the coefficient matrix together. We will discuss operations on the defining triple in the next section, see also Proposition 3.2.19 for the precise statement.

3.2.2 Admissible operations on defining triples

We study operations on defining triples that leave the isomorphy type of the associated $\mathbb{C}^*$-surface invariant, see Proposition 3.2.14. The operations come in three types, which we will call *inversion*, *permutation* and *addition*.

Definition 3.2.8. Given a defining triple $(\mathfrak{c}, l, d)$, we consider the following operations:

(i) *Inversion*: We define

$$\mathrm{inv}(\mathfrak{c}, l, d) \;:=\; (-\,\mathfrak{c}, (l'_0, \ldots, l'_r), (d'_0, \ldots, d'_r)),$$

where $l'_i := (l_{in_i}, \ldots, l_{i1})$ and $d'_i := (-d_{in_i}, \ldots, -d_{i1})$. Here, negation on the set of cases is defined as

$$-\,(\mathrm{ee}) \;:=\; (\mathrm{ee}), \quad -\,(\mathrm{pe}) \;:=\; (\mathrm{ep}), \quad -\,(\mathrm{ep}) \;:=\; (\mathrm{pe}), \quad -\,(\mathrm{pp}) \;:=\; (\mathrm{pp})\,.$$

We also write $\mathrm{inv}_1 := \mathrm{id}$ and $\mathrm{inv}_{-1} := \mathrm{inv}$.

(ii) *Permutation*: Given a permutation $\sigma\colon \{0, \ldots, r\} \to \{0, \ldots, r\}$, we define

$$\mathrm{perm}_\sigma(\mathfrak{c}, l, d) \;:=\; (\mathfrak{c}, (l_{\sigma(0)}, \ldots, l_{\sigma(r)}), (d_{\sigma(0)}, \ldots, d_{\sigma(r)})).$$

(iii) *Addition*: Given an integral vector $c = (c_1, \ldots, c_r)$, set $c_0 := -(c_1 + \cdots + c_r)$ and define

$$\mathrm{add}_c(\mathfrak{c}, l, d) \;:=\; (\mathfrak{c}, l, (d'_0, \ldots, d'_r)),$$

where $d'_i := d_i + c_i l_i$.

By an *admissible operation*, we mean any composition of operations (i), (ii) and (iii). We call two defining triples *equivalent* and write $(\mathfrak{c}, l, d) \sim (\mathfrak{c}', l', d')$, if one arises from the other by an admissible operation.

Lemma 3.2.9. *Any admissible operation is equal to some*

$$\psi_{c,\sigma,o} \;:=\; \mathrm{add}_c \,\circ\, \mathrm{perm}_\sigma \,\circ\, \mathrm{inv}_o,$$

where $c \in \mathbb{Z}^r$, $\sigma\colon \{0, \ldots, r\} \to \{0, \ldots, r\}$ is a permutation and $o \in \{\pm 1\}$.

Proof. The following identities are easy to check:

$$\mathrm{inv} \circ \mathrm{add}_c \;=\; \mathrm{add}_{-c} \circ \mathrm{inv}, \quad \mathrm{inv} \circ \mathrm{perm}_\sigma \;=\; \mathrm{perm}_\sigma \circ \mathrm{inv}, \quad \mathrm{perm}_\sigma \circ \mathrm{add}_c \;=\; \mathrm{add}_c \circ \mathrm{perm}_\sigma\,.$$

This shows that we can rearrange any composition of admissible operations to have inversions first, followed by permutations and then additions. Moreover, since we have

$$\mathrm{inv}_{o\cdot o'} = \mathrm{inv}_o \circ \mathrm{inv}_{o'}, \quad \mathrm{add}_c \circ \mathrm{add}_{c'} = \mathrm{add}_{c+c'}, \quad \mathrm{perm}_{\sigma\circ\tau} = \mathrm{perm}_\sigma \circ \mathrm{perm}_\tau,$$

it is sufficient to have at most one inversion, permutation and addition respectively. □

Next, we study how admissible operations affect the columns of the generator matrix.

Proposition 3.2.10. *Let $(\mathfrak{c}, l, d)$ and $(\mathfrak{c}', l', d)$ be defining triples with generator matrices P and P'. Write $v_{ij}, v^{\pm}$ and $v'_{ij}, v'^{\pm}$ for the columns of P and P' respectively.*

(i) Assume $(\mathfrak{c}', l', d') = \mathrm{inv}(\mathfrak{c}, l, d)$ and define $U := \mathrm{diag}(1, \ldots, 1, -1) \in \mathrm{GL}(r+1, \mathbb{Z})$. Then we have

$$v'_{ij} \;=\; U v_{i, n_i - j + 1}.$$

(ii) Assume $(\mathfrak{c}', l', d') = \mathrm{perm}_\sigma(\mathfrak{c}, l, d)$ and define $U := \begin{bmatrix} e_{\sigma^{-1}(1)} & \ldots & e_{\sigma^{-1}(r)} & u \end{bmatrix}$, where $\{e_1, \ldots, e_r, u\}$ is the canonical basis of $\mathbb{Z}^{r+1}$ and $e_0 := -(e_1 + \cdots + e_r)$. Then

$$v'_{ij} \;=\; U v_{\sigma(i) j}.$$

(iii) Assume $(\mathfrak{c}', l', d') = \mathrm{add}_c(\mathfrak{c}, l, d)$ with $c \in \mathbb{Z}^r$ and set

$$U := \begin{bmatrix} 1 & & 0 & 0 \\ & \ddots & & \vdots \\ 0 & & 1 & 0 \\ c_1 & \ldots & c_r & 1 \end{bmatrix} \in GL(r+1, \mathbb{Z}).$$

Then we have

$$v'_{ij} \;=\; U v_{ij}.$$

Proof. All three claims are immediate computations using the definitions of inversion, permutation and addition and $v'_{ij} = l'_{ij} e_i + d'_{ij} u$. □

Corollary 3.2.11. *Let $(\mathfrak{c}, l, d)$ and $(\mathfrak{c}', l', d)$ be defining triples with generator matrices P and P' and assume $(\mathfrak{c}', l', d') = \psi_{c,\sigma,o}(\mathfrak{c}, l, d)$, where $\psi_{c,\sigma,o}$ is defined as in Lemma 3.2.9. Then there is a unimodular matrix $U \in \mathrm{GL}(r+1, \mathbb{Z})$ such that*

$$P' = U P E_{\sigma,o},$$

where $E_{\sigma,o} \in \mathrm{GL}(n+m, \mathbb{Z})$ denotes the permutation matrix given by

$$f_{ij} \mapsto \begin{cases} f_{\sigma(i),j}, & o = 1 \\ f_{\sigma(i), n'_i - j + 1}, & o = -1 \end{cases}, \qquad f^{\pm} \mapsto \begin{cases} f^{\pm}, & o = 1 \\ f^{\mp}, & o = -1 \end{cases}.$$

Proposition 3.2.12. *An admissible operation sends defining triples to defining triples.*

Proof. Let $(\mathfrak{c}', l', d')$ arise from the defining triple $(\mathfrak{c}, l, d)$ by an admissible operation. We need to show that $(\mathfrak{c}', l', d')$ is a defining triple, i.e. that the following hold:

(1) $\gcd(l'_{ij}, d'_{ij}) = 1$ for all $i = 0, \ldots, r$ and $j = 1, \ldots, n'_i$,

(2) $\frac{d'_{i1}}{l'_{i1}} > \cdots > \frac{d'_{in'_i}}{l'_{in'_i}}$ for all $i = 0, \ldots, r$,

(3) the columns of P' are pairwise distinct and generate $\mathbb{Q}^{r+1}$ as a cone.

Property (1) holds trivially for an inversion and a permutation. For an addition, we have

$$\gcd(l'_{ij}, d'_{ij}) = \gcd(l_{ij}, d_{ij} + c_i l_{ij}) = \gcd(l_{ij}, d_{ij}) = 1.$$

Property (2) holds for an inversion, since it both flips the signs of d_{ij} and reverses the order of the l_{ij} and d_{ij}. It also holds trivially for a permutation, since it does not change the order of the l_{ij} for a fixed i. For an addition, we have

$$\frac{d'_{ij}}{l'_{ij}} = \frac{d_{ij}}{l_{ij}} + c_i,$$

hence the order stays invariant. Property (3) holds for all admissible operations, since by Corollary 3.2.11, the columns of P' arise from the columns of P by applying unimodular matrix. □

In order to prove that admissible operations leave the isomorphy type of the associated graded $\mathbb{C}$-algebra invariant, we need the following auxiliary lemma:

Lemma 3.2.13. *In Construction 3.1.12, define more generally the trinomials*

$$g_{i_1 i_2 i_3} := \alpha_{i_2 i_3} T_{i_1}^{l_{i_1}} - \alpha_{i_1 i_3} T_{i_2}^{l_{i_2}} + \alpha_{i_1 i_2} T_{i_3}^{l_{i_3}},$$

where $0 \leq i_1, i_2, i_3 \leq r$ are pairwise distinct. Then $g_{i_1 i_2 i_3} \in \langle g_0, \ldots, g_{r-2} \rangle$.

Proof. By definition, we have $g_i = g_{i,i+1,i+2}$. Let $0 \leq i_1, i_2, i_3 \leq r$ be arbitrary and pairwise distinct. Observe that the trinomials are alternating under permutations:

$$g_{i_1 i_2 i_3} = g_{i_2 i_3 i_1} = g_{i_3 i_1 i_2} = -g_{i_2 i_1 i_3} = -g_{i_3 i_2 i_1} = -g_{i_1 i_3 i_2}.$$

Hence, without loss of generality, we may assume that $i_1 < i_2 < i_3$. If $i_3 = i_2 + 1 = i_1 + 2$, then $g_{i_1 i_2 i_3}$ is one of the generators g_i, and the claim follows immediately. Otherwise, there exists an index i_4 such that either $i_1 < i_4 < i_2 < i_3$ or $i_1 < i_2 < i_4 < i_3$. In the first case, a straightforward computation shows:

$$\alpha_{i_4 i_2} g_{i_1 i_2 i_3} = \alpha_{i_1 i_2} g_{i_4 i_2 i_3} - \alpha_{i_3 i_2} g_{i_1 i_4 i_2}.$$

Similarly, in the second case:

$$\alpha_{i_4 i_2} g_{i_1 i_2 i_3} = \alpha_{i_3 i_2} g_{i_1 i_2 i_4} - \alpha_{i_1 i_2} g_{i_2 i_4 i_3}.$$

Repeating this reduction finitely many times, we can express $g_{i_1 i_2 i_3}$ as a $\mathbb{C}$-linear combination of the generators $g_0, \ldots, g_{r-2}$. Therefore, $g_{i_1 i_2 i_3} \in \langle g_0, \ldots, g_{r-2} \rangle$. □

Proposition 3.2.14. *Let $(\mathfrak{c}, l, d)$ be a defining triple, and let $(\mathfrak{c}', l', d') = \psi_{c,\sigma,o}(\mathfrak{c}, l, d)$ for some admissible operation $\psi_{c,\sigma,o}$. Then, there exists an isomorphism between the graded $\mathbb{C}$-algebras:*

$$R(\mathfrak{c}, l, d; A) \cong R(\mathfrak{c}', l', d'; \sigma(A)),$$

where $\sigma(A)$ is defined as $\begin{bmatrix} a_{\sigma(0)} & \ldots & a_{\sigma(r)} \end{bmatrix}$.

Proof. Write $R := R(\mathfrak{c}, l, d; A)$ and $R' := R(\mathfrak{c}', l', d'; A')$. We work in the notation of Construction 3.1.12. Recall that an isomorphism between graded algebras consists of a pair $(\varphi, \tilde{\varphi})$, where $\tilde{\varphi}\colon K' \to K$ is an isomorphism of groups, and $\varphi\colon R' \to R$ is an isomorphism of $\mathbb{C}$-algebras such that $\deg_R(\varphi(f)) = \tilde{\varphi}(\deg_{R'}(f))$ for all $f \in R'$. According to Corollary 3.2.11, we have the relation $P' = UPE_{\sigma,o}$, where $U \in \mathrm{GL}(r+1, \mathbb{Z})$ and $E_{\sigma,o}$ is a permutation matrix. As $E_{\sigma,o}^\top = E_{\sigma,o}^{-1}$, this gives rise to a commutative diagram with exact rows:

$$\begin{array}{ccccccccc} 0 & \longrightarrow & \mathbb{Z}^{r+1} & \xrightarrow{P'^\top} & \mathbb{Z}^{n+m} & \xrightarrow{Q'} & K' & \longrightarrow & 0 \\ & & \downarrow{\scriptstyle U^\top} & & \downarrow{\scriptstyle E_{\sigma,o}} & & \vdots{\scriptstyle \tilde{\varphi}} & & \\ 0 & \longrightarrow & \mathbb{Z}^{r+1} & \xrightarrow{P^\top} & \mathbb{Z}^{n+m} & \xrightarrow{Q} & K & \longrightarrow & 0 \end{array}$$

Hence we get a homomorphism $\tilde{\varphi}\colon K' \to K$, making the right square commute. Writing $w_{ij}, w^\pm$ and $w'_{ij}, w'^\pm$ for the columns of Q and Q' respectively, we get

$$\tilde{\varphi}(w'_{ij}) = \begin{cases} w_{\sigma(i)j}, & \text{if } o = 1, \\ w_{\sigma(i)n'_i - j + 1}, & \text{if } o = -1, \end{cases} \qquad \tilde{\varphi}(w'^\pm) = \begin{cases} w^\pm, & \text{if } o = 1, \\ w^\mp, & \text{if } o = -1. \end{cases}$$

We define the $\mathbb{C}$-algebra homomorphism $\varphi\colon\ \mathbb{C}[T_{ij}, S^\pm] \to \mathbb{C}[T_{ij}, S^\pm]$ by:

$$T_{ij} \mapsto \begin{cases} T_{\sigma(i),j}, & \text{if } o = 1, \\ T_{\sigma(i),n'_i - j + 1}, & \text{if } o = -1, \end{cases} \qquad S^\pm \mapsto \begin{cases} S^\pm, & \text{if } o = 1, \\ S^\mp, & \text{if } o = -1. \end{cases}$$

We obtain

$$\varphi(T_i^{l'_i}) = \varphi(T_{i1})^{l'_{i1}} \cdots \varphi(T_{in'_i})^{l'_{in'_i}} = T_{\sigma(i)1}^{l_{\sigma(i)1}} \cdots T_{\sigma(i)n_{\sigma(i)}}^{l_{\sigma(i)n_{\sigma(i)}}} = T_{\sigma(i)}^{l_{\sigma(i)}}.$$

Moreover, since $\alpha'_{i_1 i_2} = \alpha_{\sigma(i_1)\sigma(i_2)}$, Lemma 3.2.13 implies

$$\varphi(g_i) = \varphi(g_{i,i+1,i+2}) = g'_{\sigma(i_1)\sigma(i_2)\sigma(i_3)} \in \langle g'_0, \ldots, g'_{r-2} \rangle.$$

Therefore, φ descends to a $\mathbb{C}$-algebra homomorphism $R' \to R$. Additionally, the degree compatibility holds:

$$\deg_R(\varphi(T_{ij})) = \tilde{\varphi}(\deg_{R'}(T_{ij})), \qquad \deg_R(\varphi(S^\pm)) = \tilde{\varphi}(\deg_{R'}(S^\pm)).$$

This shows that $(\varphi, \tilde{\varphi})$ is a morphism of graded algebras from R' to R. Since the admissible operation $\psi_{c,\sigma,o}$ is invertible, both $\tilde{\varphi}$ and φ are isomorphisms. □

3.2.3 Erasable entries and irredundancy

If a defining triple $(\mathfrak{c}, l, d)$ contains an entry l_i consisting of a single one, we may erase this entry while not changing the isomorphy type of the associated graded $\mathbb{C}$-algebra. Let us make this precise:

Definition 3.2.15. Let $(\mathfrak{c}, l, d)$ be a defining triple with $l_i \in \mathbb{Z}_{\geq 0}^{n_i}$ and $d_i \in \mathbb{Z}^{n_i}$. We call the i-th entry of l *erasable* if $n_i = l_{i1} = 1$ and $d_{i1} = 0$. We call $(\mathfrak{c}, l, d)$ *irredundant*, if $n_i l_{i1} > 1$ for all $i = 0, \ldots, r$ and *redundant* otherwise.

Remark 3.2.16. Given any redundant defining triple $(\mathfrak{c}, l, d)$ with $n_i = l_{i1} = 1$, we may apply the admissible operation add_c with $c_i = -d_{i1}$ and $c_j = 0$ for $j \neq i$. The resulting defining triple will have $d_{i1} = 0$, hence its i-th entry is erasable.

Proposition 3.2.17. *Let $(\mathfrak{c}, l, d)$ be a defining triple with $r > 1$ whose i-th entry is erasable. Let $A \in \mathbb{C}^{2\times(r+1)}$ be a coefficient matrix and set*

$$l' := (l_0, \ldots, \cancel{l_i}, \ldots, l_r), \qquad d' := (d_0, \ldots, \cancel{d_i}, \ldots, d_r), \qquad A' := \begin{bmatrix} a_0 & \ldots & \cancel{a_i} & \ldots & a_r \end{bmatrix}.$$

Then $(\mathfrak{c}, l', d')$ is again a defining triple and we have an isomorphism of graded $\mathbb{C}$-algebras:

$$R(\mathfrak{c}, l, d; A) \;\cong\; R(\mathfrak{c}, l', d'; A').$$

Proof. We may assume without loss of generality that $i = r$. We work in the notation of Construction 3.1.12. Using $d_{r1} = 0$, we see that the columns of the P' generate $\mathbb{Q}^r$ as a convex cone, hence $(\mathfrak{c}, l', d')$ is a defining triple. Recall that a homomorphism of graded algebras is a pair $(\varphi, \tilde{\varphi})$ where $\varphi\colon R \to R'$ is an algebra homomorphism and $\tilde{\varphi}\colon K \to K'$ is a group homomorphism such that for all $f \in R$, we have $\deg_{R'}(\varphi(f)) = \tilde{\varphi}(\deg_R(f))$. Let us first construct the isomorphism $\tilde{\varphi}$: Write $\{e_1, \ldots, e_r, u\}$ and $\{e_1, \ldots, e_{r-1}, u\}$ for the canonical bases of $\mathbb{Z}^{r+1}$ and $\mathbb{Z}^r$ respectively. Moreover, write $\{f_{ij}, f^{\pm}\}$ for the canonical bases if $\mathbb{Z}^{n+m}$ and $\mathbb{Z}^{n+m-1}$, where f_{r1} is not present in the latter. We consider the following commutative diagram with exact rows:

$$\begin{array}{ccccccccc}
0 & \longrightarrow & \mathbb{Z}^{r+1} & \xrightarrow{P^\top} & \mathbb{Z}^{n+m} & \xrightarrow{Q} & K & \longrightarrow & 0 \\
 & & \iota_1 \uparrow \downarrow \pi_1 & & \iota_2 \uparrow \downarrow \pi_2 & & \tilde{\psi} \uparrow \vdots \downarrow \tilde{\varphi} & & \\
0 & \longrightarrow & \mathbb{Z}^{r} & \xrightarrow{P'^\top} & \mathbb{Z}^{n+m-1} & \xrightarrow{Q'} & K' & \longrightarrow & 0.
\end{array}$$

Here, we define ι_1 and ι_2 to be the canonical inclusions sending $e_i, u \mapsto e_i, u$ and $f_{ij}, f^{\pm} \mapsto f_{ij}, f^{\pm}$. Moreover, π_1 is the projection sending $e_i, u \mapsto e_i, u$ and $e_r \mapsto 0$, while π_2 is the projection sending $f_{ij}, f^{\pm} \mapsto f_{ij}, f^{\pm}$ and $f_{r1} \mapsto l_{01} f_{01} + \cdots + l_{0n_0} f_{0n_0}$. It is immediate that $P^\top \circ \iota_1 = \iota_2 \circ P'^\top$. Using $d_{r1} = 0$, we see that also $P'^\top \circ \pi_1 = \pi_2 \circ P^\top$. This implies that we get group homomorphisms $\tilde{\psi}$ and $\tilde{\varphi}$ as drawn, making the appropriate squares commute. Writing

$$w_{ij} := Q(f_{ij}), \quad w^{\pm} := Q(f^{\pm}), \qquad w'_{ij} := Q'(f_{ij}), \quad w'^{\pm} := Q'(f^{\pm}),$$

we thus have

$$\begin{array}{rclcrcl}
\tilde{\varphi}(w^{\pm}) & = & w'^{\pm}, & \qquad & \tilde{\varphi}(w_{ij}) & = & \begin{cases} w'_{ij} & i = 0, \ldots, r-1 \\ l_{01} w'_{01} + \cdots + l_{0n_0} w'_{0n_0} & i = r. \end{cases} \\
\tilde{\psi}(w'^{\pm}) & = & w^{\pm}, & & \tilde{\psi}(w'_{ij}) & = & w_{ij}.
\end{array}$$

Clearly, $\tilde{\varphi} \circ \tilde{\psi} = \mathrm{id}_{K'}$. Moreover, recall from Construction 3.1.12 that the monomials $T_i^{l_i}$ all have the same degree, hence

$$\tilde{\psi}(\tilde{\varphi}(w_{r1})) = l_{01} w_{01} + \cdots + l_{0n_0} w_{0n_0} = w_{r1}.$$

This implies $\tilde{\psi} \circ \tilde{\varphi} = \mathrm{id}_K$. We have shown that $\tilde{\varphi}\colon K \to K'$ is an isomorphism. Next, we construct the ring isomorphism $\varphi\colon R \to R'$. Note that we have

$$g_{r-2} \;=\; \alpha_{r-1,r} T_{r-2}^{l_{r-2}} - \alpha_{r-2,r} T_{r-1}^{l_{r-1}} + \alpha_{r-2,r-1} T_{r1}.$$

We define $\varphi\colon R \to R'$ by sending T_{r1} to $\alpha_{r-2,r-1}^{-1}(\alpha_{r-2,r} T_{r-1}^{l_{r-1}} - \alpha_{r-1,r} T_{r-2}^{l_{r-2}})$ and leaving the other variables unchanged. This achieves $\varphi(g_i) = g_i'$ for $i = 0, \ldots, r-3$ and $\varphi(g_{r-2}) = 0$, hence φ is well-defined. An inverse of φ is the map $\psi\colon R' \to R$ sending $T_{ij} \mapsto T_{ij}$ and $S^{\pm} \mapsto S^{\pm}$. It remains to show that φ is compatible with the grading. For $i = 0, \ldots, r-1$, we have

$$\deg_{R'}(\varphi(T_{ij})) = \deg_{R'}(T_{ij}) = w_{ij}' = \tilde{\varphi}(w_{ij}) = \tilde{\varphi}(\deg_R(T_{ij}))$$

and similarly $\deg_{R'}(\varphi(S^{\pm})) = \tilde{\varphi}(\deg_R(S^{\pm}))$. Lastly, we compute

$$\deg_{R'}(\varphi(T_{r1})) = \deg_{R'}(T_{r-1}^{l_{r-1}}) = \sum_{j=1}^{n_{r-1}} l_{r-1,j} w_{r-1,j}' = \sum_{j=1}^{n_0} l_{0j} w_{0j}' = \tilde{\varphi}(w_{r1}) = \tilde{\varphi}(\deg_R(T_{r1})).$$

□

Corollary 3.2.18. *Every graded $\mathbb{C}$-algebra $R(\mathfrak{c}, l, d; A)$ arising from Construction 3.1.12 is isomorphic to some $R(\mathfrak{c}', l', d', A')$ with $(\mathfrak{c}', l', d')$ irredundant.*

Proof. Combine Remark 3.2.16 and Proposition 3.2.17. □

3.2.4 Completeness of admissible operations

We are ready to prove the main theorem about admissible operations:

Theorem 3.2.19. *Let $(\mathfrak{c}, l, d)$ and $(\mathfrak{c}', l', d')$ be irredundant defining triples with $r = r' > 1$ and let $A, A' \in \mathbb{C}^{2\times(r+1)}$ coefficient matrices. Then the following are equivalent:*

(i) $(\mathfrak{c}', l', d') = \psi_{c,\sigma,o}(\mathfrak{c}, l, d)$ *for some admissible operation $\psi_{c,\sigma,o}$ and $A' \sim \sigma(A)$, where $\sigma(A)$ is defined to be $\begin{bmatrix} a_{\sigma(0)} & \ldots & a_{\sigma(r)} \end{bmatrix}$.*

(ii) $R \cong R'$ *as graded $\mathbb{C}$-algebras,*

(iii) $X \cong X'$ *as surfaces,*

(iv) $X \cong X'$ *as $\mathbb{C}^*$-surfaces,*

where $R := R(\mathfrak{c}, l, d; A)$ and $R' := R(\mathfrak{c}', l', d'; A')$ as well as $X := X(\mathfrak{c}, l, d; A)$ and $X' := X(\mathfrak{c}', l', d'; A')$.

Proof. The implication "(i) ⇒(ii)" has been established in Propositions 3.2.2 and 3.2.14. For the implication "(ii) ⇒(iii)", consider that by Proposition 3.1.16, the Cox rings of X and X' are R and R', respectively. By [2, Theorem 4.3.3.5], the moving cone of X is equal to its semiample

cone. This means that any surface with Cox ring R must be isomorphic to X, see for instance [2, Remark 3.3.4.2]. Next, we show "(iii) $\Rightarrow$(iv)". By [3, Corollary 2.4], the automorphism group $\mathrm{Aut}(X)$ is linear algebraic and acts morphically. In particular, any $\mathbb{C}^*$-action on X defines a one-dimensional torus in $\mathrm{Aut}(X)$. Since we assume the defining triples to be irredundant with $r = r' > 1$, Remark 3.1.18 implies that X and X' are non-toric. Hence any two one-dimensional tori in $\mathrm{Aut}(X)$ are maximal and thus conjugate to each other. This implies that $X \cong X'$ as $\mathbb{C}^*$-surfaces.

We are left with showing "(iv) $\Rightarrow$(i)". Suppose we have an isomorphism $\varphi\colon X' \to X$ of $\mathbb{C}^*$-surfaces. Since φ respects isotropy groups, by Proposition 3.1.19, it must send prime divisors $D^{ij}_{X'}$ into D^{ij}_X and $D^{\pm}_{X'}$ into $D^{\pm}_X$. Moreover, recall that an arm $D^{i1}_X \cup \cdots \cup D^{in_i}_X$ is connected, hence divisors in the same arm must stay in the same arm. This means we find a permutation $\sigma\colon \{0,\ldots,r\} \to \{0,\ldots,r\}$ and $o \in \{\pm 1\}$ such that

$$\varphi_*([D^{ij}_{X'}]) = \begin{cases} [D^{\sigma(i),j}_X], & o = 1 \\ [D^{\sigma(i),n'_i-j+1}_X], & o = -1 \end{cases}, \qquad \varphi_*([D^{\pm}_{X'}]) = \begin{cases} [D^{\pm}_X], & o = 1 \\ [D^{\mp}_X], & o = -1 \end{cases},$$

where $\varphi_*\colon \mathrm{Cl}(X') \to \mathrm{Cl}(X)$ denotes the pushforward on divisor classes. We obtain a commutative diagram with exact rows

$$\begin{array}{ccccccccc} 0 & \longrightarrow & \mathbb{Z}^{r+1} & \xrightarrow{P'^*} & \mathbb{Z}^{n+m} & \xrightarrow{Q'} & \mathrm{Cl}(X') & \longrightarrow & 0 \\ & & \downarrow{\scriptstyle U} & & \downarrow{\scriptstyle E_{\sigma,o}} & & \downarrow{\scriptstyle \varphi_*} & & \\ 0 & \longrightarrow & \mathbb{Z}^{r+1} & \xrightarrow{P^\top} & \mathbb{Z}^{n+m} & \xrightarrow{Q} & \mathrm{Cl}(X) & \longrightarrow & 0 \end{array},$$

where $E_{\sigma,o}$ is the permutation matrix defined in Corollary 3.2.11. The map U must be an isomorphism making the left square commute, hence we obtain $P' = U^\top P E_{\sigma,o}$. By the structure of P and P', this forces $U^\top$ to be of the form

$$U^\top = \begin{bmatrix} B_\sigma & 0 \\ c & o \end{bmatrix},$$

where $B_\sigma = \begin{bmatrix} e_{\sigma^{-1}(1)} & \ldots & e_{\sigma^{-1}(r)} \end{bmatrix}$ and $c \in \mathbb{Z}^r$ is arbitrary. This implies that $(\mathfrak{c}', l', d') = \psi_{c,\sigma,o}(\mathfrak{c}, l, d)$, as desired. It remains to show that $A' \sim \sigma(A)$. By Proposition 3.1.15, we get a commutative diagram involving the rational quotients

$$\begin{array}{ccc} X' & \xrightarrow{\varphi} & X \\ \downarrow{\scriptstyle \pi'} & & \downarrow{\scriptstyle \pi} \\ \mathbb{P}_1 & \xrightarrow{U} & \mathbb{P}_1 \end{array},$$

where the induced map U is an isomorphism, hence given by a matrix $U \in \mathrm{GL}_2(\mathbb{C})$. Recall that π sends D^{ij}_X to $[a_i]$, while π' sends $D^{ij}_{X'}$ to $[a'_i]$. Thus by commutativity of the diagram, $U(a'_i)$ and $a_{\sigma(i)}$ differ only by a nonzero scalar. Hence we obtain $UA' = \sigma(A)D$ for a diagonal matrix $D \in \mathrm{GL}_{r+1}(\mathbb{C})$. This implies $A' \sim \sigma(A)$. □

3.2.5 The normal form

We describe a normal form for defining triples and show that every defining triple can be brought uniquely into normal form, see Proposition 3.2.25. An implementation of the normal form is available in `CStarSurfaces.jl`, see Section B.2.3.

Recall that the *slopes* of a defining triple $(\mathfrak{c}, l, d)$ are rational numbers

$$m_{ij} \;:=\; \frac{d_{ij}}{l_{ij}}, \qquad i = 0, \ldots, r,\ j = 1, \ldots n_i$$

and they satisfy $m_{i1} > \cdots > m_{in_i}$.

Definition 3.2.20 (See B.2.35 – B.2.38)**.** To any defining triple $(\mathfrak{c}, l, d)$, we associate integers

$$\mathfrak{m}^+ \;:=\; \lfloor m_{01} \rfloor + \cdots + \lfloor m_{r1} \rfloor, \qquad \mathfrak{m}^- \;:=\; -(\lceil m_{0n_0} \rceil + \cdots + \lceil m_{rn_r} \rceil).$$

Moreover, we define rational numbers

$$\beta^+_{ij} \;:=\; m_{ij} - \lfloor m_{i1} \rfloor, \qquad \beta^-_{ij} \;:=\; \lceil m_{in_i} \rceil - m_{i,n_i-j+1}.$$

and set $\beta^\pm_i := (\beta^\pm_{i1}, \ldots, \beta^\pm_{in_i})$. Since the slopes are ordered descendingly, we have

$$\beta^+_{i1} > \cdots > \beta^+_{in_i} \qquad \text{and} \qquad \beta^-_{i1} > \cdots > \beta^-_{in_i}.$$

We define multivectors $\beta^\pm$ by sorting $(\beta^\pm_0, \ldots, \beta^\pm_r)$ lexicographically:

$$\beta^+ \;:=\; \operatorname{sort}(\beta^+_0, \ldots, \beta^+_r), \qquad \beta^- \;:=\; \operatorname{sort}(\beta^-_0, \ldots, \beta^-_r).$$

Lemma 3.2.21. *Let $(\mathfrak{c}, l, d)$ and $(\mathfrak{c}', l', d')$ be defining triples with $\mathfrak{m}^\pm, \beta^\pm$ and $\mathfrak{m}'^\pm, \beta'^\pm$ defined as above.*

(i) If $(\mathfrak{c}', l', d')$ arises from $(\mathfrak{c}, l, d)$ by permutation or addition, we have

$$\mathfrak{m}'^+ = \mathfrak{m}^+, \quad \mathfrak{m}'^- = \mathfrak{m}^-, \qquad \beta'^+ = \beta^+, \quad \beta'^- = \beta^-.$$

(ii) If $(\mathfrak{c}', l', d')$ arises from $(\mathfrak{c}, l, d)$ by an inversion, we have

$$\mathfrak{m}'^+ = \mathfrak{m}^-, \quad \mathfrak{m}'^- = \mathfrak{m}^+, \qquad \beta'^+ = \beta^-, \quad \beta'^- = \beta^+.$$

Proof. A permutation permutes the numbers $\lfloor m_{i1} \rfloor$ and $\lceil m_{in_i} \rceil$ accordingly, thus it leaves $\mathfrak{m}^\pm$ invariant. Moreover, since $\beta^\pm$ are defined by sorting the vectors $\beta^\pm_i$ lexicographically, they too stay invariant under permutation. Let's consider an addition add_c with $c = (c_1, \ldots, c_r) \in \mathbb{Z}^r$ and $c_0 := -(c_1 + \cdots + c_r)$. By definition, we have $d'_{ij} = d_{ij} + c_i l_{ij}$. Hence the slopes transform as

$$m'_{ij} = \frac{d_{ij} + c_i l_{ij}}{l_{ij}} = m_{ij} + c_i.$$

Note that since the c_i are integers, we have $\lfloor m_{i1} + c_i \rfloor = \lfloor m_{i1} \rfloor + c_i$. We obtain

$$\mathfrak{m}'^{+} = \mathfrak{m}^{+} + c_0 + \cdots + c_r = \mathfrak{m}^{+},$$

and similarly, $\mathfrak{m}'^{-} = \mathfrak{m}^{-}$. Moreover, we have

$$\beta'^{+}_{ij} = m_{ij} + c_i - (\lfloor m_{i1} \rfloor + c_i) = m_{ij} - \lfloor m_{i1} \rfloor = \beta^{+}_{ij}.$$

Thus $\beta'^{+} = \beta^{+}$ and similarly $\beta'^{-} = \beta^{-}$. We have shown (i).

For (ii), let $(\mathfrak{c}', l', d') = \mathrm{inv}(\mathfrak{c}, l, d)$. By definition, $l'_{ij} = l_{i,n_i-j+1}$ and $d'_{ij} = -d_{i,n_i-j+1}$. Hence the slopes transform as $m'_{ij} = -m_{i,n_i-j+1}$. We obtain

$$\mathfrak{m}'^{+} = \lfloor m'_{01} \rfloor + \cdots + \lfloor m'_{r1} \rfloor = -(\lceil m_{0n_0} \rceil + \ldots \lceil m_{rn_r} \rceil) = \mathfrak{m}^{-},$$

and similarly $\mathfrak{m}'^{-} = \mathfrak{m}^{+}$. Moreover, we have

$$\beta'^{+}_{ij} = -m_{i,n_i-j+1} - \lfloor -m_{in_i} \rfloor = \lceil m_{in_i} \rceil - m_{i,n_i-j+1} = \beta^{-}_{ij}.$$

Thus $\beta'^{+} = \beta^{-}$ and similarly $\beta'^{-} = \beta^{+}$. □

Definition 3.2.22 (See B.2.40)**.** To a defining triple $(\mathfrak{c}, l, d)$, we associate an *orientation* $o \in \{-1, 0, 1\}$ as follows: We set $o := 1$ if and only if one of the following holds:

$(O^{+}1)$ $\mathfrak{c} = (\mathrm{pe})$,

$(O^{+}2)$ $\mathfrak{c} \in \{(\mathrm{ee}), (\mathrm{pp})\}$ and $\mathfrak{m}^{+} > \mathfrak{m}^{-}$,

$(O^{+}3)$ $\mathfrak{c} \in \{(\mathrm{ee}), (\mathrm{pp})\}$ and $\mathfrak{m}^{+} = \mathfrak{m}^{-}$ and $\beta^{+} >_{\mathrm{lex}} \beta^{-}$.

Accordingly, we set $o := -1$ if and only if one of the following holds:

$(O^{-}1)$ $\mathfrak{c} = (\mathrm{ep})$,

$(O^{-}2)$ $\mathfrak{c} \in \{(\mathrm{ee}), (\mathrm{pp})\}$ and $\mathfrak{m}^{-} > \mathfrak{m}^{+}$,

$(O^{-}3)$ $\mathfrak{c} \in \{(\mathrm{ee}), (\mathrm{pp})\}$ and $\mathfrak{m}^{+} = \mathfrak{m}^{-}$ and $\beta^{-} >_{\mathrm{lex}} \beta^{+}$.

The remaining case is $\mathfrak{c} \in \{(\mathrm{ee}), (\mathrm{pp})\}$ and $\mathfrak{m}^{+} = \mathfrak{m}^{-}$ and $\beta^{+} = \beta^{-}$, where we set $o := 0$.

Lemma 3.2.23. *Let $(\mathfrak{c}, l, d)$ and $(\mathfrak{c}', l', d')$ be defining triples with orientations o and o'.*

(i) If $(\mathfrak{c}', l', d')$ arises from $(\mathfrak{c}, l, d)$ by permutation or addition, we have $o' = o$.

(ii) If $(\mathfrak{c}', l', d')$ arises from $(\mathfrak{c}, l, d)$ by an inversion, we have $o' = -o$.

Proof. By Lemma 3.2.21 (i), addition and permutation leave $\mathfrak{c}, \mathfrak{m}^{\pm}$ and $\beta^{\pm}$ invariant. Hence also the orientation stays invariant, which shows (i). For (ii), recall that inversion flips the case $\mathfrak{c}$ and swaps $(\mathfrak{m}^{+}, \mathfrak{m}^{-})$ as well as (β^{+}, β^{-}). Hence the conditions $(O^{+}1), (O^{+}2)$ and $(O^{+}3)$ are transformed into $(O^{-}1), (O^{-}2)$ and $(O^{-}3)$ respectively. Finally, if $o = 0$, then after an inversion, we clearly still have $o' = 0$. □

Definition 3.2.24 (See B.2.41)**.** A defining triple $(\mathfrak{c}, l, d)$ is said to be in *normal form*, if it satisfies the following conditions:

(i) $o \geq 0$,

(ii) $\beta_0^+ \geq_{\text{lex}} \cdots \geq_{\text{lex}} \beta_r^+$,

(iii) $\lfloor m_{i1} \rfloor = 0$ for all $i = 1, \ldots, r$.

Proposition 3.2.25. *Every defining triple is equivalent to a unique defining triple in normal form.*

Proof. We describe how to bring a defining triple $(\mathfrak{c}, l, d)$ into normal form by admissible operations. By Lemma 3.2.23, an inversion flips the sign of the orientation, hence we may achieve $o \geq 0$. Next, we may apply a permutation to achieve $\beta_0^+ \geq_{\text{lex}} \cdots \geq_{\text{lex}} \beta_r^+$. Finally, the addition add_c with $c := (-\lfloor m_{11} \rfloor, \ldots, -\lfloor m_{r1} \rfloor)$ achieves $\lfloor m_{i1} \rfloor = 0$ for $i = 1, \ldots, r$. Recall that permutation and addition do not change the orientation, and addition does not change the values β_{ij}^+. Hence we arrive at a defining triple in normal form. See also B.2.48 for an implementation of this algorithm.

To show uniqueness, assume $(\mathfrak{c}, l, d)$ and $(\mathfrak{c}', l'd')$ are both in normal form and equivalent to each other by an admissible operation ψ. Property (i) of the normal form says $o, o' \geq 0$. If $o = 1$, the operation ψ cannot contain an inversion, as that would imply $o' = -1$ by Lemma 3.2.23 (ii). Hence we have $(\mathfrak{c}, \mathfrak{m}^+, \beta^+) = (\mathfrak{c}', \mathfrak{m}'^+, \beta'^+)$ by Lemma 3.2.21 (i). On the other hand, if $o = 0$, we have by definition $\mathfrak{c} \in \{(\text{ee}), (\text{pp})\}$ and $\mathfrak{m}^+ = \mathfrak{m}^-$ and $\beta^+ = \beta^-$. Thus, regardless of the existence of an inversion in ψ, we get $(\mathfrak{c}, \mathfrak{m}^+, \beta^+) = (\mathfrak{c}', \mathfrak{m}'^+, \beta'^+)$ also for $o = 0$. Together with property (ii) of the normal form $\beta^+ = \beta'^+$ implies $\beta_{ij}^+ = \beta_{ij}'^+$ for all i and j. Next, property (iii) of the normal form implies $\lfloor m_{i1} \rfloor = 0 = \lfloor m'_{i1} \rfloor$ for $i = 1, \ldots, r$ and hence also $\lfloor m_{01} \rfloor = \mathfrak{m}^+ = \mathfrak{m}'^+ = \lfloor m'_{01} \rfloor$. Together, we obtain $m_{ij} = m'_{ij}$ for all i and j. Since $\gcd(l_{ij}, d_{ij}) = \gcd(l'_{ij}, d'_{ij}) = 1$, this already implies $l_{ij} = l'_{ij}$ and $d_{ij} = d'_{ij}$. Thus, the defining triples coincide. □

3.3 Proof of the Picard index formula

In this section, we prove the formula for the Picard index of surfaces with torus action given in Theorem 7. Note that in the toric case, this formula matches the one in Proposition 2.2.11. A much simpler proof is available in that case, as detailed in Section 2.3.

Let us give an outline of the proof strategy, which closely follows the one used in the toric case. Let $X \subseteq Z$ be a normal rational projective $\mathbb{C}^*$-surface arising from Construction 3.1.13 and let P be its generator matrix. Applying Proposition 2.1.9 to the toric ambient, we obtain

$$\iota_{\text{Pic}}(Z) = \frac{1}{|\hat{K}|} \prod_{\sigma \in \Sigma_{\max}} |K_\sigma|. \tag{3.1}$$

Here, $\hat{K}$ is the cokernel of the transpose of the map $\hat{P}$ defined in Construction 2.1.5. Moreover, by [2, Corollary 3.3.1.7], we have $\iota_{\text{Pic}}(Z) = \iota_{\text{Pic}}(X)$ and $|\operatorname{Cl}(X, z_\sigma)| = |\operatorname{Cl}(Z, z_\sigma)| = |K_\sigma|$ for

a maximal cone $\sigma \in \Sigma_{\max}$. The points $\{z_\sigma \mid \sigma \in \Sigma_{\max}\} \subseteq X$ are precisely the fixed points of the $\mathbb{C}^*$-action. Hence, the only thing left to prove Theorem 7 is to establish the equality

$$|\hat{K}| = |\operatorname{Cl}(X)^{\text{tors}}|. \tag{3.2}$$

Since $\hat{K}$ is the cokernel of $\hat{P}^\top$, its group order equals the greatest common divisor of all maximal minors of $\hat{P}$. On the other hand, the class group of X is the cokernel of $P^\top$, hence the order of its torsion part equals the greatest common divisor of all maximal minors of P. Writing $M(A)$ for the set of maximal minors of a matrix, Equation (3.2) thus reduces to

$$\gcd(M(\hat{P})) = \gcd(M(P)). \tag{3.3}$$

This section is organized as follows. In Subsection 3.3.1, we will introduce notions of class group, Picard group, multiplicities and Picard index in terms of defining triples. This will lead to a purely combinatorial version of the Picard index formula, see Theorem 3.3.13. Next, we will study maximal minors of P in Subsection 3.3.2. An explicit construction of the map $\hat{P}$ in this setting will be given in Subsection 3.3.3, after which we study its maximal minors in Subsection 3.3.4. Finally, Subsection 3.3.5 completes the proof and provides some examples.

Parts of this section have been published in [56].

3.3.1 The Picard group of a defining triple

Recall that a defining triple $(\mathfrak{c}, l, d)$ consists of a case $\mathfrak{c} \in \{(\text{ee}), (\text{pe}), (\text{ep}), (\text{pp})\}$ and vectors $l = (l_0, \ldots, l_r)$ and $d = (d_0, \ldots, d_r)$, where $l_i \in \mathbb{Z}^{n_i}_{\geq 1}$ and $d_i \in \mathbb{Z}^{n_i}$. Here, we require $\gcd(l_{ij}, d_{ij}) = 1$ and that the slopes $\frac{d_{ij}}{l_{ij}}$ are ordered descendingly for each $i = 0, \ldots, r$. By Construction 3.1.1, the *generator matrix* of $(\mathfrak{c}, l, d)$ has the form

$$P = \begin{bmatrix} -l_0 & l_1 & & 0 & 0 & 0 \\ \vdots & & \ddots & & \vdots & \vdots \\ -l_0 & 0 & & l_r & 0 & 0 \\ d_0 & d_1 & \ldots & d_r & 1 & -1 \end{bmatrix},$$

where the column $v^+ = (0, \ldots, 0, 1)$ is only present if $\mathfrak{c} \in \{(\text{pe}), (\text{pp})\}$ and $v^- = (0, \ldots, 0, -1)$ is only present if $\mathfrak{c} \in \{(\text{ep}), (\text{pp})\}$. Writing $m \in \{0, 1, 2\}$ for the number of parabolic fixed point curves and setting $n = n_0 + \cdots + n_r$, we can view the generator matrix as a lattice map $P\colon \mathbb{Z}^{n+m} \to \mathbb{Z}^{r+1}$. As in previous sections, we will write $\{f_{ij}, f^\pm\}$ for the lattice basis of $\mathbb{Z}^{n+m}$ and $e_1, \ldots, e_r, u$ for the lattice basis of $\mathbb{Z}^{r+1}$.

Definition 3.3.1 (See B.2.21, B.2.25)**.** Let $(\mathfrak{c}, l, d)$ be a defining triple with generator map $P\colon \mathbb{Z}^{n+m} \to \mathbb{Z}^{r+1}$ as above. We define its *class group* as $\operatorname{Cl}(\mathfrak{c}, l, d) := \mathbb{Z}^{n+m} / \operatorname{im}(P^\top)$. The *grading matrix* is the canonical quotient map $Q\colon \mathbb{Z}^{n+m} \to \operatorname{Cl}(\mathfrak{c}, l, d)$. Moreover, we define the *canonical class* as

$$\mathcal{K} := -\left(\sum_{i,j} Q(f_{ij})\right) - Q(f^+) - Q(f^-) \in \operatorname{Cl}(\mathfrak{c}, l, d).$$

Example 3.3.2. Unlike the class group of an LDP polygon (see Proposition 2.2.5), the class group of a defining triple of a $\mathbb{C}^*$-surface can have more than one torsion factor. An example is given by the triple $((\mathrm{pe}),((2),(2),(2)),((5),(1),(1)))$ with generator matrix

$$P = \begin{bmatrix} -2 & 2 & 0 & 0 \\ -2 & 0 & 2 & 0 \\ -5 & 1 & 1 & 1 \end{bmatrix}.$$

Here, the class group is isomorphic to $\mathbb{Z} \times \mathbb{Z}/2\,\mathbb{Z} \times \mathbb{Z}/2\,\mathbb{Z}$.

Definition 3.3.3 (See B.2.22)**.** By the *Picard number* of a defining triple, we mean $\rho := n+m-(r+1) = \operatorname{rank}(\mathrm{Cl}(\mathfrak{c},l,d))$. The *multiplicity* is $\mu := |\,\mathrm{Cl}(\mathfrak{c},l,d)^{\mathrm{tors}}| = \gcd(M(P))$.

Like for LDP polygons, the multiplicity of a defining triple equals the greatest common divisor of all maximal minors of the generator matrix. Recall that in Proposition 2.2.3, we saw that it was enough to take the greatest common divisor of minors with adjacent column indices, i.e. the local multiplicities. When we study maximal minors of P in more detail in Subsection 3.3.2, we will show that a similar result holds, see Proposition 3.3.24.

Remark 3.3.4. Under the splitting $\mathrm{Cl}(\mathfrak{c},l,d) \cong \mathbb{Z}^\rho \times \mathrm{Cl}(\mathfrak{c},l,d)^{\mathrm{tors}}$, we can consider the columns of the grading matrix as $Q(f_{ij}) = \omega_{ij} = (w_{ij},\eta_{ij})$ and $Q(f^\pm) = \omega^\pm = (w^\pm,\eta^\pm)$, where $w_{ij}, w^\pm \in \mathbb{Z}^\rho$ and $\eta_{ij},\eta^\pm \in \mathrm{Cl}(\mathfrak{c},l,d)$. In particular, the canonical class is $\mathcal{K} = -\sum_{i,j}\omega_{ij} - \omega^+ - \omega^-$.

Lemma 3.3.5. *Let $(\mathfrak{c},l,d)$ be a defining triple with generator matrix $P\colon \mathbb{Z}^{n+m} \to \mathbb{Z}^{r+1}$ and grading matrix $Q\colon \mathbb{Z}^{n+m} \to \mathrm{Cl}(\mathfrak{c},l,d)$. Let $I \subseteq \{f_{ij}, f^\pm\}$ be a subset of the canonical basis of $\mathbb{Z}^{n+m}$. Consider*

$$N_I := \bigoplus_{f\in I} P(f) \le \mathbb{Z}^{r+1}, \qquad N_I^{\mathbb{Q}} := N_I \otimes \mathbb{Q} = \operatorname{span}_{\mathbb{Q}}(P(f)\,;\, f\in I) \le \mathbb{Q}^{r+1}.$$

Then we have

$$\mathrm{Cl}(\mathfrak{c},l,d)/\langle Q(f); f\notin I\rangle \quad\cong\quad \mathbb{Z}^{|I|-\dim(N_I^{\mathbb{Q}})} \oplus (N_I^{\mathbb{Q}} \cap \mathbb{Z}^{r+1})/N_I.$$

Proof. [2, Lemma 2.1.4.1]. □

Recall that fixed points of $\mathbb{C}^*$-surfaces are either *parabolic*, *hyperbolic* or *elliptic*, which means they are in the closure of precisely one, two or infinitely many $\mathbb{C}^*$-orbits respectively. In order to define local properties of $\mathbb{C}^*$-surfaces purely in terms of their defining triples, we now introduce sets of formal symbols that represent fixed points.

Definition 3.3.6 (See Subsection B.3.1)**.** Let $(\mathfrak{c},l,d)$ be a defining triple. We define sets of formal symbols $\mathcal{X}_{\mathrm{h}}$, $\mathcal{X}_{\mathrm{e}}$ and $\mathcal{X}_{\mathrm{p}}$, which we will call *hyperbolic*, *elliptic* and *parabolic fixed points* respectively. The hyperbolic fixed points are

$$\mathcal{X}_{\mathrm{h}} := \{x_{ij} \mid i = 0,\dots,r, \quad j = 1,\dots n_i - 1\}.$$

According to the case $\mathfrak{c}$, the elliptic and parabolic fixed points are

$$\begin{array}{llllll}
\text{(ee)} & \mathcal{X}_{\mathrm{e}} & := & \{x^+, x^-\} & \mathcal{X}_{\mathrm{p}} & := & \emptyset, \\
\text{(pe)} & \mathcal{X}_{\mathrm{e}} & := & \{x^-\} & \mathcal{X}_{\mathrm{p}} & := & \{x_0^+, \dots, x_r^+\}, \\
\text{(ep)} & \mathcal{X}_{\mathrm{e}} & := & \{x^+\} & \mathcal{X}_{\mathrm{p}} & := & \{x_0^-, \dots, x_r^-\}, \\
\text{(pp)} & \mathcal{X}_{\mathrm{e}} & := & \emptyset & \mathcal{X}_{\mathrm{p}} & := & \{x_0^+, \dots, x_r^+, x_0^-, \dots, x_r^-\}.
\end{array}$$

Finally, the set of *fixed points* is $\mathcal{X} := \mathcal{X}_{\mathrm{h}} \cup \mathcal{X}_{\mathrm{e}} \cup \mathcal{X}_{\mathrm{p}}$.

Definition 3.3.7 (See B.3.15)**.** To each fixed point $x \in \mathcal{X}$ of a defining triple $(\mathfrak{c}, l, d)$, we will define a *local generator matrix* P_x as follows. For hyperbolic fixed points, we set

$$P_{x_{ij}} := \begin{bmatrix} l_{ij} & l_{ij+1} \\ d_{ij} & d_{ij+1} \end{bmatrix}.$$

For the elliptic fixed points, we set

$$P_{x^+} := \begin{bmatrix} -l_{01} & l_{11} & & 0 \\ \vdots & & \ddots & \\ -l_{01} & 0 & & l_{r1} \\ d_{01} & d_{11} & \dots & d_{r1} \end{bmatrix}, \qquad P_{x^-} := \begin{bmatrix} -l_{0n_0} & l_{1n_1} & & 0 \\ \vdots & & \ddots & \\ -l_{0n_0} & 0 & & l_{rn_r} \\ d_{0n_0} & d_{1n_1} & \dots & d_{rn_r} \end{bmatrix}.$$

Finally, for parabolic fixed points, we set

$$P_{x_i^+} := \begin{bmatrix} l_{i1} & 0 \\ d_{i1} & 1 \end{bmatrix}, \qquad P_{x_i^-} := \begin{bmatrix} l_{in_i} & 0 \\ d_{in_i} & -1 \end{bmatrix}.$$

We will also use shorthand notation $P_{ij} = P_{x_{ij}}$ as well as $P^{\pm} = P_{x^{\pm}}$ and $P_i^{\pm} = P_{x_i^{\pm}}$.

Definition 3.3.8 (See B.3.17)**.** The *local multiplicity* of a fixed point $x \in \mathcal{X}$ is the positive integer $\mu_x := |\det(P_x)|$. We will also use shorthand notation $\mu_{ij} = \mu_{x_{ij}}$ as well as $\mu^{\pm} = \mu_{x^{\pm}}$ and $\mu_i^{\pm} = \mu_{x_i^{\pm}}$.

Remark 3.3.9. We get the following formulas for the local multiplicities:

(i) $\mu^+ = \sum_{i=0}^{r} d_{i1} \prod_{j \neq i} l_{j1}$,

(ii) $\mu^- = -\sum_{i=0}^{r} d_{in_i} \prod_{j \neq i} l_{jn_j}$,

(iii) $\mu_{ij} = l_{ij} d_{ij+1} - d_{ij} l_{ij+1}$,

(iv) $\mu_i^+ = l_{i1}$,

(v) $\mu_i^- = l_{in_i}$.

Construction 3.3.10 (See B.3.16, B.2.26)**.** Let $(\mathfrak{c}, l, d)$ be a defining triple. Consider the lattices

$$F_{x^{\pm}} := N_{x^{\pm}} := \mathbb{Z}^{r+1}, \qquad F_{x_{ij}} := N_{x_{ij}} := F_{x_i^{\pm}} := N_{x_i^{\pm}} := \mathbb{Z}^2.$$

Then the local generator map P_x associated to a fixed point $x \in \mathcal{X}$ can be viewed as a lattice map fitting into the commutative diagram

$$\begin{array}{ccc} \mathbb{Z}^{r+1} & \xrightarrow{P} & \mathbb{Z}^{n+m} \\ \uparrow & & \uparrow \\ F_x & \xrightarrow{P_x} & N_x \end{array},$$

where the upwards inclusions have to be chosen appropriately. We define the *local class groups* as $\mathrm{Cl}_x(\mathfrak{c}, l, d) := F_x^* / \operatorname{im}(P_x^\top)$. Writing Q_x for the quotient maps, we obtain a commutative diagram with exact rows

$$\begin{array}{ccccccccc} 0 & \longrightarrow & \mathbb{Z}^{r+1} & \xrightarrow{P^\top} & \mathbb{Z}^{n+m} & \xrightarrow{Q} & \mathrm{Cl}(\mathfrak{c}, l, d) & \longrightarrow & 0 \\ & & \downarrow & & \downarrow & & \downarrow{\scriptstyle \pi_x} & & \\ 0 & \longrightarrow & N_x^* & \xrightarrow{P_x^\top} & F_x^* & \xrightarrow{Q_x} & \mathrm{Cl}_x(\mathfrak{c}, l, d) & \longrightarrow & 0 \end{array}.$$

We define the *local Picard groups* as $\mathrm{Pic}_x(\mathfrak{c}, l, d) := \ker(\pi_x)$ and the *Picard group* as

$$\mathrm{Pic}(\mathfrak{c}, l, d) := \bigcap_{x \in \mathcal{X}} \mathrm{Pic}_x(\mathfrak{c}, l, d) \leq \mathrm{Cl}(\mathfrak{c}, l, d).$$

The *Picard index* is $\iota_{\mathrm{Pic}}(\mathfrak{c}, l, d) := [\mathrm{Cl}(\mathfrak{c}, l, d) : \mathrm{Pic}(\mathfrak{c}, l, d)]$.

Proposition 3.3.11. *Let $(\mathfrak{c}, l, d)$ be a defining triple. The local multiplicities satisfy*

$$\mu_x = |\mathrm{Cl}_x(\mathfrak{c}, l, d)|.$$

Moreover, we have $\mu^+ = \mu\tilde{\mu}^+$ and $\mu^- = \mu\tilde{\mu}^-$, where

$$\begin{aligned} \tilde{\mu}^+ &:= |\det(w_{ij}, w^- ;\ i = 0, \ldots, r, j = 2, \ldots, n_i)| && \text{if } \mathfrak{c} \in \{(\mathrm{ee}), (\mathrm{ep})\}, \\ \tilde{\mu}^- &:= |\det(w_{ij}, w^+ ;\ i = 0, \ldots, r, j = 1, \ldots, n_i - 1)| && \text{if } \mathfrak{c} \in \{(\mathrm{ee}), (\mathrm{pe})\}. \end{aligned}$$

Here, $w_{ij}, w^\pm$ are the free parts of the columns of the grading matrix, as in Remark 3.3.4.

Proof. By the diagram in Construction 3.3.10, the local class group $\mathrm{Cl}_x(\mathfrak{c}, l, d)$ is the cokernel of $P_x^\top$, hence its order is $|\det(P_x)| = \mu_x$. For the second assertion, note that the local Picard group $\mathrm{Pic}_{x^+}(\mathfrak{c}, l, d) = \ker(\pi_{x^+})$ is generated by ω_{ij}, ω^- with $i = 0, \ldots, r$ and $j = 2, \ldots, n_i$. The class group is of the form

$$\mathrm{Cl}(\mathfrak{c}, l, d) = \mathbb{Z}^\rho \times \mathbb{Z}/\mu_1\mathbb{Z} \times \cdots \times \mathbb{Z}/\mu_k\mathbb{Z} \cong \mathbb{Z}^{\rho+k}/\langle \vec{\mu}_1, \ldots, \vec{\mu}_k \rangle,$$

where $\mu = \mu_1 \cdots \mu_k$ and $\vec{\mu}_j = \mu_j e_{\rho+j}$. By the third isomorphism theorem, we get

$$\mathrm{Cl}(\mathfrak{c}, l, d)/\mathrm{Pic}_x(\mathfrak{c}, l, d) \cong \mathbb{Z}^{\rho+k}/\langle \{\omega_{ij}, \omega^- ; i = 0, \ldots, r, j = 2, \ldots, n_i\} \cup \{\vec{\mu}_1, \ldots, \vec{\mu}_k\} \rangle.$$

We obtain

$$\mu^+ = \mu_1 \cdots \mu_k \cdot |\det(w_{ij}, w^- ; i = 0, \ldots, r, j = 2, \ldots, n_i)| = \mu\tilde{\mu}^+.$$

The equality $\mu^- = \mu\tilde{\mu}^-$ is shown analogously. □

Remark 3.3.12. Let X be a $\mathbb{C}^*$-surface arising from a defining triple $(\mathfrak{c}, l, d)$. Then the combinatorial versions of class group, Picard group and local class groups coincide with the geometric ones, i.e.

(i) $\mathrm{Cl}(X) \cong \mathrm{Cl}(\mathfrak{c}, l, d)$,

(ii) $\mathrm{Pic}(X) \cong \mathrm{Pic}(\mathfrak{c}, l, d)$,

(iii) $\iota_{\mathrm{Pic}}(X) \cong \iota_{\mathrm{Pic}}(\mathfrak{c}, l, d)$,

(iv) $\mathrm{Cl}(X, x) \cong \mathrm{Cl}_x(\mathfrak{c}, l, d)$ for a fixed point $x \in X$.

This leads to the following purely combinatorial version of the Picard index formula:

Theorem 3.3.13. *The Picard index of a defining triple is*

$$\iota_{\mathrm{Pic}}(\mathfrak{c}, l, d) \;=\; \frac{1}{\mu} \prod_{x \in \mathcal{X}} \mu_x \;=\; \mu^{1-m} \tilde{\mu}^+ \tilde{\mu}^- \prod_{x \in \mathcal{X}_{\mathrm{h}} \cup \mathcal{X}_{\mathrm{p}}} \mu_x,$$

where $\tilde{\mu}^{\pm}$ are defined as in 3.3.11 and we set $\tilde{\mu}^+ := 1$ if $\mathfrak{c} \in \{(\mathrm{pe}), (\mathrm{pp})\}$ and $\tilde{\mu}^- := 1$ if $\mathfrak{c} \in \{(\mathrm{ep}), (\mathrm{pp})\}$.

By Remark 3.3.12, Theorem 3.3.13 implies Theorem 7.

3.3.2 Maximal minors of P

As a first step towards the equality of the cokernel orders of P and $\hat{P}^\top$, we study in this section the set $M(P)$ of maximal minors of our generator matrices. We figure out relations among the maximal minors of P and, based on that, show that $\gcd(M(P))$ equals $\gcd(M'(P))$ for a proper, more accessible subset $M'(P) \subset M(P)$; see Proposition 3.3.24. First, we introduce a suitable lattice basis for $\mathbb{Z}^n$, reflecting the block structure of P.

Construction 3.3.14. Let $(\mathfrak{c}, l, d)$ be a defining triple with generator matrix P and set $n := n_0 + \cdots + n_r$. Then P is an integral $(r+1) \times (n+m)$-matrix, where $m \in \{0, 1, 2\}$ is the number of parabolic fixed point curves. Write $e_1, \ldots, e_{r+1}$ for the canonical basis vectors of $\mathbb{Z}^{r+1}$ and set $u := e_{r+1}$. Then we define

$$N \;:=\; \mathbb{Z}^{r+1} \;=\; \mathbb{Z}\, e_1 \oplus \cdots \oplus \mathbb{Z}\, e_r \oplus \mathbb{Z}\, u.$$

For each $i = 0, \ldots, r$, we introduce a set of symbols to be used as a lattice basis:

$$\mathcal{F}_i \;:=\; \{f_{i1}, \ldots, f_{in_i}\}, \qquad F_i \;:=\; \mathbb{Z}\, f_{i1} \oplus \cdots \oplus \mathbb{Z}\, f_{in_i} \cong \mathbb{Z}^{n_i},$$

$$\mathcal{F}' \;:=\; \begin{cases} \emptyset, & (\mathrm{ee}) \\ \{f^+\}, & (\mathrm{pe}) \\ \{f^-\}, & (\mathrm{ep}) \\ \{f^+, f^-\}, & (\mathrm{pp}) \end{cases}, \qquad F' \;:=\; \bigoplus\nolimits_{f \in \mathcal{F}'} \mathbb{Z}\, f \cong \mathbb{Z}^m.$$

Setting

$$\mathcal{F} := \mathcal{F}_0 \cup \cdots \cup \mathcal{F}_r \cup \mathcal{F}', \qquad F := F_0 \oplus \cdots \oplus F_r \oplus F',$$

we obtain an isomorphism $F \cong \mathbb{Z}^{n+m}$. Set $e_0 := -(e_1 + \cdots + e_r)$. Then we can view P as a lattice map given by

$$P\colon F \to N, \quad \begin{cases} f_{ij} \mapsto v_{ij} := l_{ij}e_i + d_{ij}u & \\ f^+ \mapsto u, & \text{if } f^+ \in \mathcal{F}' \\ f^- \mapsto -u, & \text{if } f^- \in \mathcal{F}'. \end{cases}$$

Definition 3.3.15. Let a subset $A \subseteq \mathcal{F}$ with $|A| = r+1$ be given. Then we have a sublattice $F_A := \bigoplus_{f\in A} \mathbb{Z}\cdot f \subseteq F$ and an induced map $P_A\colon F_A \to N$ as in the commutative diagram

$$\begin{array}{ccc} F & \xrightarrow{P} & N \\ \uparrow & \nearrow_{P_A} & \\ F_A. & & \end{array}$$

We call $|\det(P_A)| \in \mathbb{Z}_{\geq 0}$ the *maximal minor of P associated to A*. The set of maximal minors of P is

$$M(P) := \{|\det(P_A)|\,;\ A \subseteq \mathcal{F},\ |A| = r+1\}.$$

Example 3.3.16. Consider the defining triple $(((\mathrm{ee})), ((1,1), 8, 4), ((-1,-2), 7, 3))$. We have $\mathcal{F} = \{f_{01}, f_{02}, f_{11}, f_{21}\}$ and the generator matrix is

$$P = \begin{bmatrix} -1 & -1 & 8 & 0 \\ -1 & -1 & 0 & 4 \\ -1 & -2 & 7 & 3 \end{bmatrix}.$$

Setting $A_{ij} := \mathcal{F}\backslash\{f_{ij}\}$, the maximal minors of P are

$$\begin{aligned} |\det(P_{A_{01}})| &= |d_{02}l_{11}l_{21} + l_{02}d_{11}l_{21} + l_{02}l_{11}d_{21}| = 12, \\ |\det(P_{A_{02}})| &= |d_{01}l_{11}l_{21} + l_{01}d_{11}l_{21} + l_{01}l_{11}d_{21}| = 20, \\ |\det(P_{A_{11}})| &= |l_{21}(d_{01}l_{02} - l_{01}d_{02})| = 4, \\ |\det(P_{A_{21}})| &= |l_{11}(d_{01}l_{02} - l_{01}d_{02})| = 8. \end{aligned}$$

Hence, we have $M(P) = \{12, 20, 4, 8\}$.

The following lemma gives a vanishing criterion for maximal minors of P. In particular, it will allow us to describe the nonzero maximal minors of P explicitly.

Lemma 3.3.17. *Let $A \subseteq \mathcal{F}$ with $|A| = r+1$. Assume that there exist $0 \leq i_0 < i_1 \leq r$ such that $A \cap \mathcal{F}_{i_0} = A \cap \mathcal{F}_{i_1} = \emptyset$. Then* $\det(P_A) = 0$.

Proof. Consider the dual bases $\{f_{ij}^*\}$ and $\{e_1^*, \dots, e_r^*, u^*\}$ of F^* and N^* respectively. Then we have

$$P^\top(e_i^*) = \left(\sum_{j=1}^{n_i} l_{ij} f_{ij}^*\right) - \left(\sum_{j=1}^{n_0} l_{0j} f_{0j}^*\right).$$

If $i_0 = 0$, this implies $P_A^\top(e_{i_1}^*) = 0$. If i_0 and i_1 are both nonzero, we have

$$P_A^\top(e_{i_0}^*) = -\left(\sum_{j=1}^{n_0} l_{0j} f_{0j}^*\right) = P_A^\top(e_{i_1}^*).$$

Thus in both cases, $\det(P_A) = \det(P_A^\top) = 0$. □

Definition 3.3.18. (i) Let numbers $1 \le j_i \le n_i$ for all $i = 0, \dots, r$ be given. We define

$$\mu(j_0, \dots, j_r) := \sum_{i_0=0}^{r} d_{i_0 j_{i_0}} \prod_{i \neq i_0} l_{ij_i}, \qquad \hat{\mu} := \mu(n_0, \dots, n_r).$$

(ii) Let $0 \le i \le r$ and $0 \le j, j' \le n_i$ be given. We define

$$\nu(i, j, j') := l_{ij} d_{ij'} - l_{ij'} d_{ij}, \qquad \hat{\nu}(i, j) := \nu(i, j, n_i).$$

Lemma 3.3.19. *Let $A \subseteq \mathcal{F}$ with $|A| = r + 1$ such that $\det(P_A) \neq 0$. Assume that $A \cap \mathcal{F}' = \emptyset$ holds. Then either (i) or (ii) must hold.*

(i) *We have $|A \cap \mathcal{F}_i| = 1$ for all $i = 0, \dots, r$ and $|\det(P_A)| = |\mu(j_0, \dots, j_r)|$ for some numbers $1 \le j_i \le n_i$.*

(ii) *There exist $0 \le i_0, i_1 \le r$ with*

$$|A \cap \mathcal{F}_i| = \begin{cases} 2, & i = i_0 \\ 0, & i = i_1 \\ 1, & \text{otherwise} \end{cases}$$

and we have $|\det(P_A)| = |\nu(i_0, j_{i_0}, j'_{i_0})| \prod_{i \neq i_0, i_1} l_{ij_i}$ for some numbers $1 \le j_i \le n_i$ as well as $1 \le j'_{i_0} \le n_{i_0}$.

Proof. Lemma 3.3.17 implies that there is at most one $i = 0, \dots, r$ with $A \cap \mathcal{F}_i = \emptyset$. Since $A \cap \mathcal{F}' = \emptyset$ and $|A| = r + 1$, the conditions on $|A \cap \mathcal{F}_i|$ follow. The expressions for $\det(P_A)$ then come from cofactor expansion. □

Example 3.3.20. Applying Lemma 3.3.19 to Example 3.3.16, we see that the minors associated to A_{01} and A_{02} satisfy (i), while those associated to A_{11} and A_{21} satisfy (ii).

Lemma 3.3.21. *Let $A \subseteq \mathcal{F}$ with $|A| = r+1$ such that $\det(P_A) \neq 0$. Assume that $A \cap \mathcal{F}' \neq \emptyset$ holds. Then there exists an $i_1 = 0, \ldots, r$ such that*

$$|A \cap \mathcal{F}_i| = \begin{cases} 0, & i = i_1 \\ 1, & \textit{otherwise} \end{cases}$$

and we have $|\det(P_A)| = \prod_{i \neq i_1} l_{ij_i}$.

Proof. Note that $|A \cap \mathcal{F}'| = 2$ cannot hold, since this would imply $\det(P_A) = 0$. Hence $|A \cap \mathcal{F}'| = 1$. Lemma 3.3.17 forces the condition on $|A \cap \mathcal{F}_i|$. Cofactor expansion gives the expression for $\det(P_A)$. □

Lemma 3.3.22. (i) *Let $i = 0, \ldots, r$ and $1 \leq j, j' \leq n_i$. There exist integers $\alpha, \beta \in \mathbb{Z}$ such that*

$$\nu(i, j, j') \;=\; \alpha\hat{\nu}(i, j) - \beta\hat{\nu}(i, j').$$

(ii) *Let $i = 0, \ldots, r$ and $j_i = 1, \ldots, n_i$ for all i. There exist integers β_i such that*

$$\begin{aligned} \mu(j_0, \ldots, j_r) \;&= \beta_{i_0}\mu(j_0, \ldots, j_{i_0-1}, n_{i_0}, j_{i_0+1}, \ldots, j_r) \\ &\quad + \sum_{i_1 \neq i_0} \beta_{i_1}\hat{\nu}(i_0, j_{i_0}) \prod_{i \notin \{i_0, i_1\}} l_{ij_i}. \end{aligned}$$

Proof. We show (i). Since $\gcd(l_{in_i}, d_{in_i}) = 1$, we find integers $x, y \in \mathbb{Z}$ such that $xl_{in_i} + yd_{in_i} = 1$. Set $\alpha := xl_{ij} + yd_{ij}$ and $\beta := xl_{ij'} + yd_{ij'}$. Then the rows of 3×3-matrix

$$\begin{bmatrix} l_{ij} & l_{ij'} & l_{in_i} \\ d_{ij} & d_{ij'} & d_{in_i} \\ \alpha & \beta & 1 \end{bmatrix}$$

are linearly dependent, hence its determinant vanishes. Cofactor expansion by the last row gives the desired equality. For (ii), we consider exemplarily the case $i_0 = 0$. Since $\gcd(l_{0n_0}, d_{0n_0}) = 1$, we find $x, y \in \mathbb{Z}$ such that $-xl_{0n_0} + yd_{0n_0} = 1$. Now set

$$\beta_0 := -xl_{0j_0} + yd_{0j_0}, \qquad \beta_1 := xl_{1j_1} + yd_{1j_1},$$

as well as $\beta_i := yd_{ij_i}$ for $i = 2, \ldots, r$. Then the rows of the $(r+2) \times (r+2)$ matrix

$$\begin{bmatrix} -l_{0j_0} & -l_{0n_0} & l_{1j_1} & 0 & \ldots & 0 \\ -l_{0j_0} & -l_{0n_0} & 0 & l_{2j_2} & \ldots & 0 \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ -l_{0j_0} & -l_{0n_0} & 0 & \ldots & \ldots & l_{rj_r} \\ d_{0j_0} & d_{0n_0} & d_{1j_1} & d_{2j_2} & \ldots & d_{rj_r} \\ \beta_0 & 1 & \beta_1 & \beta_2 & \ldots & \beta_r \end{bmatrix}$$

are linearly dependent, hence its determinant vanishes. Cofactor expansion by the last row and adjusting the signs of the β_i gives the desired equality. □

Definition 3.3.23. According to the case $\mathfrak{c} \in \{(\text{ee}), (\text{pe}), (\text{ep}), (\text{pp})\}$, we define

$$M'(P) := \begin{cases} \{|\hat{\mu}|\} \cup \left\{ |\hat{\nu}(i_0, j_{i_0})| \prod_{i \neq i_0, i_1} l_{ij_i} \; ; \; \begin{array}{l} 0 \leq i_0, i_1 \leq r, \text{ with } i_0 \neq i_1, \\ 1 \leq j_i \leq n_i \text{ for all } i \neq i_1 \end{array} \right\}, & \text{(ee)} \\ \left\{ \prod_{i \neq i_1} l_{ij_i} \; ; \; 0 \leq i_1 \leq r, \; 1 \leq j_i \leq n_i \text{ for all } i \neq i_1 \right\}, & \text{(pe), (ep), (pp).} \end{cases}$$

Proposition 3.3.24. *We have* $\gcd(M(P)) = \gcd(M'(P))$.

Proof. Consider first the case (ee). Then $\mathscr{F}' = \emptyset$, hence Lemma 3.3.19 implies $M'(P) \subseteq M(P)$. This shows that $\gcd(M(P))$ divides $\gcd(M'(P))$. On the other hand, by repeated application of Lemma 3.3.22, we see that every maximal minor of P can be written as a $\mathbb{Z}$-linear combination of elements of $M'(P)$. This implies that $\gcd(M'(P))$ divides $\gcd(M(P))$. For cases (pe), (ep) and (pp), Lemma 3.3.21 implies $M'(P) \subseteq M(P)$, hence $\gcd(M(P))$ divides $\gcd(M'(P))$. On the other hand, Lemma 3.3.19 implies that every maximal minor of P is a $\mathbb{Z}$-linear combination of elements of $M'(P)$, hence $\gcd(M'(P))$ divides $\gcd(M(P))$. □

Example 3.3.25. In Example 3.3.16, we can apply Lemma 3.3.22 (ii) to express $|\det(P_{A_{02}})|$ as a $\mathbb{Z}$-linear combination of the other maximal minors. Hence we have

$$M'(P) = M(P) \backslash \{|\det(P_{A_{02}})|\} = \{12, 8, 4\}.$$

3.3.3 The construction of $\hat{P}$

The aim of this section is to provide an explicit matrix representation of the map $\hat{P}$ from Construction 2.1.5 for the case of a (toric ambient of a) $\mathbb{C}^*$-surface. We will later use this to describe its maximal minors in Section 3.3.4.

Construction 3.3.26. Let Σ be the fan of an ambient toric variety of a $\mathbb{C}^*$-surface, as defined in Construction 3.1.4. Consider the lattices $F_\sigma := \mathbb{Z}^{n_\sigma}$ and $N_\sigma := \operatorname{lin}_{N_\mathbb{Q}}(\sigma) \cap N$ from Construction 2.1.4. We have $n_{\sigma^+} = n_{\sigma^-} = r+1$ and $n_{\tau_{ij}} = 2$. Moreover, we have $N_{\sigma^+} = N_{\sigma^-} = N = \mathbb{Z}^{r+1}$ and $N_{\tau_{ij}} = \mathbb{Z}\, e_i + \mathbb{Z}\, u \subseteq N$. We will work with the identifications

$$\begin{array}{lcllcl} F_{\sigma^+} & \cong & \mathbb{Z}\, f^+_{01} \oplus \cdots \oplus \mathbb{Z}\, f^+_{r1}, & N_{\sigma^+} & \cong & \mathbb{Z}\, e_1^+ \oplus \cdots \oplus \mathbb{Z}\, e_r^+ \oplus \mathbb{Z}\, u^+, \\ F_{\tau_{ij}} & \cong & \mathbb{Z}\, f^-_{ij} \oplus \mathbb{Z}\, f^+_{ij+1}, & N_{\tau_{ij}} & \cong & \mathbb{Z}\, e_{ij} \oplus \mathbb{Z}\, u_{ij}, \\ F_{\sigma^-} & \cong & \mathbb{Z}\, f^-_{0n_0} \oplus \cdots \oplus \mathbb{Z}\, f^-_{rn_r}, & N_{\sigma^-} & \cong & \mathbb{Z}\, e_1^- \oplus \cdots \oplus \mathbb{Z}\, e_r^- \oplus \mathbb{Z}\, u^-. \end{array}$$

Then according to the case $\mathfrak{c}$, a lattice basis of $\mathbf{N}$ is given by

$$\begin{array}{lcccl} \text{(ee)} & \{e_1^+, \ldots, e_r^+, u^+\} & \cup & S & \cup \quad \{e_1^-, \ldots, e_r^-, u^-\}, \\ \text{(pe)} & \{e_{i0}, u_{i0} \; ; \; i = 0, \ldots, r\} & \cup & S & \cup \quad \{e_1^-, \ldots, e_r^-, u^-\}, \\ \text{(ep)} & \{e_1^+, \ldots, e_r^+, u^+\} & \cup & S & \cup \quad \{e_{in_i}, u_{in_i} \; ; \; i = 0, \ldots, r\}, \\ \text{(pp)} & \{e_{i0}, u_{i0} \; ; \; i = 0, \ldots, r\} & \cup & S & \cup \quad \{e_{in_i}, u_{in_i} \; ; \; i = 0, \ldots, r\}, \end{array}$$

where $S := \{e_{ij}, u_{ij}\ ;\ i = 0, \ldots, r,\ j = 1, \ldots, n_i - 1\}$. A lattice basis of $\mathbf{F}$ is given by

$$\begin{array}{lllll}
\text{(ee)} & \{f_{ij}^-\ ;\ i = 0, \ldots, r,\ j = 1, \ldots, n_i\} & \cup & \{f_{ij}^+\ ;\ i = 0, \ldots, r,\ j = 1, \ldots, n_i\}, \\
\text{(pe)} & \{f_{ij}^-\ ;\ i = 0, \ldots, r,\ j = 0, \ldots, n_i\} & \cup & \{f_{ij}^+\ ;\ i = 0, \ldots, r,\ j = 1, \ldots, n_i\}, \\
\text{(ep)} & \{f_{ij}^-\ ;\ i = 0, \ldots, r,\ j = 1, \ldots, n_i\} & \cup & \{f_{ij}^+\ ;\ i = 0, \ldots, r,\ j = 1, \ldots, n_i + 1\}, \\
\text{(pp)} & \{f_{ij}^-\ ;\ i = 0, \ldots, r,\ j = 0, \ldots, n_i\} & \cup & \{f_{ij}^+\ ;\ i = 0, \ldots, r,\ j = 1, \ldots, n_i + 1\}.
\end{array}$$

In particular, we have $\operatorname{rank}(\mathbf{F}) = \operatorname{rank}(\mathbf{N}) = 2n + m(r+1)$. With respect to these bases, the maps α and β from Construction 2.1.5 are

$$\begin{aligned}
&\alpha\colon \mathbf{N} \to N, \quad e_{ij}, e_i^+, e_i^- \mapsto e_i, \qquad u_{ij}, u^+, u^- \mapsto u, \\
&\beta\colon \mathbf{F} \to F, \quad f_{ij}^+, f_{ij}^- \mapsto f_{ij}.
\end{aligned}$$

Setting $e_0^+ := -(e_1^+ + \cdots + e_r^+)$ and $e_0^- := -(e_1^- + \cdots + e_r^-)$, the maps $P_{\sigma^+}, P_{\sigma^-}$ and $P_{\tau_{ij}}$ are then given as

$$\begin{aligned}
&P_{\sigma^+}\colon F_{\sigma^+} \to N_{\sigma^-}, \quad f_{i1}^+ \mapsto l_{i1} e_i^+ + d_{i1} u^+, \\
&P_{\sigma^-}\colon F_{\sigma^-} \to N_{\sigma^-}, \quad f_{in_i}^- \mapsto l_{in_i} e_i^- + d_{in_i} u^-, \\
&P_{\tau_{ij}}\colon F_{\tau_{ij}} \to N_{\tau_{ij}}, \quad f_{ij}^- \mapsto l_{ij} e_{ij} + d_{ij} u_{ij} \\
&\qquad\qquad\qquad\qquad\quad f_{ij+1}^+ \mapsto l_{ij+1} e_{ij} + d_{ij+1} u_{ij}.
\end{aligned}$$

Remark 3.3.27. Clearly, the map α in Construction 3.3.26 is surjective in all cases. Hence Corollary 2.1.7 implies that the Picard group of a projective rational $\mathbb{C}^*$-surface is torsion-free.

Example 3.3.28. With P as in Example 3.3.16, we have $\{e_1^+, e_2^+, u^+, e_{01}, u_{01}, e_1^-, e_2^-, u^-\}$ as a basis for $\mathbf{N}$ and $\{f_{01}^+, f_{11}^+, f_{21}^+, f_{01}^-, f_{02}^+, f_{02}^-, f_{11}^-, f_{21}^-\}$ as a basis for $\mathbf{F}$. The matrix representations of α and β are

$$\alpha = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}, \qquad \beta = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Construction 3.3.29. We continue in the setting of Construction 3.3.26. We will give explicit descriptions of the maps γ, δ and $\hat{P}$ from Construction 2.1.5. According to the case $\mathfrak{c}$, let us set for all $i = 0, \ldots, r$

$$n_i' := \begin{cases} n_i - 1, & \text{(ee)}, \\ n_i, & \text{(pe)}, \text{(ep)}, \text{(pp)}, \end{cases}$$

$$\hat{\mathcal{N}}_i := \{\hat{e}_{ij}, \hat{u}_{ij}\ ;\ j = 1, \ldots, n_i'\}, \qquad \hat{\mathcal{F}}_i := \{\hat{f}_{ij}\ ;\ j = 1, \ldots n_i\}.$$

Now define the sets of symbols

$$\tilde{n} := \begin{cases} \{\tilde{e}_1, \dots, \tilde{e}_r, \tilde{u}\}, & \text{(ee)} \\ \emptyset, & \text{(pe)} \\ \emptyset, & \text{(ep)} \\ \{\tilde{u}_1, \dots, \tilde{u}_r, \tilde{e}\}, & \text{(pp)} \end{cases}, \qquad \bar{\mathcal{F}} := \begin{cases} \emptyset, & \text{(ee)} \\ \{\hat{f}_i^- \ ; \ i = 1, \dots, r\}, & \text{(pe)} \\ \{\hat{f}_i^+ \ ; \ i = 1, \dots, r\}, & \text{(ep)} \\ \{\hat{f}_i^+, \hat{f}_i^- \ ; \ i = 1, \dots, r\}, & \text{(pp)} \end{cases},$$

$$\hat{n} := \hat{n}_0 \cup \ldots \cup \hat{n}_\iota \cup \tilde{n}, \qquad \hat{\mathcal{F}} := \hat{\mathcal{F}}_0 \cup \ldots \cup \hat{\mathcal{F}}_\iota \cup \bar{\mathcal{F}}.$$

Let $\hat{N}$ and $\hat{F}$ be the free lattices over $\hat{n}$ and $\hat{\mathcal{F}}$ respectively. In particular, we have $\operatorname{rank}(\hat{N}) = 2n+(m-1)(r+1)$ and $\operatorname{rank}(\hat{F}) = n+mr$. According to the case $\mathfrak{c}$, we define a map $\gamma\colon \hat{N} \to \mathbf{N}$ as follows:

$$\begin{aligned}
\text{(ee)} \quad \hat{e}_{ij} &\mapsto \begin{cases} e_i^+ - e_{i1}, & j = 1 \\ e_{ij-1} - e_{ij}, & 2 \le j \le n_i - 1 \end{cases} \\
\hat{u}_{ij} &\mapsto \begin{cases} u_i^+ - u_{i1}, & j = 1 \\ u_{ij-1} - u_{ij}, & 2 \le j \le n_i - 1 \end{cases} \\
\tilde{e}_i &\mapsto e_i^+ - e_i^- \\
\tilde{u} &\mapsto u^+ - u^-, \\
\text{(pe)} \quad \hat{e}_{ij} &\mapsto \begin{cases} e_{ij-1} - e_{ij}, & 1 \le j \le n_i - 1 \\ e_{in_i-1} - e_i^-, & j = n_i \end{cases} \\
\hat{u}_{ij} &\mapsto \begin{cases} u_{ij-1} - u_{ij}, & 1 \le j \le n_i - 1 \\ u_{in_i-1} - u^-, & j = n_i \end{cases} \\
\text{(ep)} \quad \hat{e}_{ij} &\mapsto \begin{cases} e_i^+ - e_{i1}, & j = 1, \\ e_{ij-1} - e_{ij}, & 2 \le j \le n_i \end{cases} \\
\hat{u}_{ij} &\mapsto \begin{cases} u^+ - u_{i1}, & j = 1, \\ u_{ij-1} - u_{ij}, & 2 \le j \le n_i \end{cases} \\
\text{(pp)} \quad \hat{e}_{ij} &\mapsto e_{ij-1} - e_{ij}, \\
\hat{u}_{ij} &\mapsto u_{ij-1} - u_{ij}, \\
\tilde{u}_i &\mapsto u_{in_i} - u_{i-1,n_{i-1}} \\
\tilde{e} &\mapsto (e_{00} + e_{10} + \cdots + e_{r0}).
\end{aligned}$$

Then γ is a kernel of $\alpha\colon \mathbf{N} \to N$. Next, we define a map $\delta\colon \hat{F} \to \mathbf{F}$ as follows:

$$
\begin{array}{lll}
\text{(ee)} & \hat{f}_{ij} & \mapsto \quad f^+_{ij} - f^-_{ij} \\
\\
\text{(pe)} & \hat{f}_{ij} & \mapsto \quad f^+_{ij} - f^-_{ij} \\
 & \hat{f}^-_i & \mapsto \quad f^-_{i-1,0} - f^-_{i,0} \\
\\
\text{(ep)} & \hat{f}_{ij} & \mapsto \quad f^+_{ij} - f^-_{ij} \\
 & \hat{f}^+_i & \mapsto \quad f^+_{i-1,n_{i-1}+1} - f^+_{i,n_i+1} \\
\\
\text{(pp)} & \hat{f}_{ij} & \mapsto \quad f^+_{ij} - f^-_{ij} \\
 & \hat{f}^-_i & \mapsto \quad f^-_{i-1,0} - f^-_{i,0} \\
 & \hat{f}^+_i & \mapsto \quad f^+_{i-1,n_{i-1}+1} - f^+_{i,n_i+1}.
\end{array}
$$

Then δ is a kernel of $\beta\colon\ \mathbf{F} \to F$. If $\mathfrak{c} = (\text{ee})$, set $\tilde{e}_0 := -(\tilde{e}_1 + \cdots + \tilde{e}_r)$. Then the induced map $\hat{P}\colon \hat{F} \to \hat{N}$ is given as follows.

$$
\text{(ee)} \qquad \hat{f}_{ij} \quad \mapsto \quad \begin{cases} l_{ij}\hat{e}_{ij} + d_{ij}\hat{u}_{ij}, & 1 \le j \le n_i - 1 \\ l_{in_i}\left(\tilde{e}_i - \sum_{k=0}^{n_i-1} \hat{e}_{ik}\right) + d_{in_i}\left(\tilde{u} - \sum_{k=1}^{n_i-1} \hat{u}_{ik}\right), & j = n_i, \end{cases}
$$

$$
\begin{array}{lll}
\text{(pe)} & \hat{f}_{ij} & \mapsto \quad l_{ij}\hat{e}_{ij} + d_{ij}\hat{u}_{ij} \\
 & \hat{f}^-_i & \mapsto \quad \left(\sum_{k=1}^{n_{i-1}} \hat{u}_{i-1,k}\right) - \left(\sum_{k=1}^{n_i} \hat{u}_{ik}\right) \\
\\
\text{(ep)} & \hat{f}_{ij} & \mapsto \quad l_{ij}\hat{e}_{ij} + d_{ij}\hat{u}_{ij} \\
 & \hat{f}^+_i & \mapsto \quad \left(\sum_{k=1}^{n_{i-1}} \hat{u}_{i-1,k}\right) - \left(\sum_{k=1}^{n_i} \hat{u}_{ik}\right) \\
\\
\text{(pp)} & \hat{f}_{ij} & \mapsto \quad l_{ij}\hat{e}_{ij} + d_{ij}\hat{u}_{ij} \\
 & \hat{f}^-_i & \mapsto \quad \left(\sum_{k=1}^{n_{i-1}} \hat{u}_{i-1,k}\right) - \left(\sum_{k=1}^{n_i} \hat{u}_{ik}\right) - \tilde{u}_i \\
 & \hat{f}^+_i & \mapsto \quad \tilde{u}_i.
\end{array}
$$

Proof. In all cases, we can verify that $\alpha \circ \gamma = 0$ and $\beta \circ \delta = 0$. Moreover, γ and δ are both injective and we have

$$
\begin{aligned}
\operatorname{rank}(\hat{N}) &= 2n - (m-1)(r+1) = \operatorname{rank}(\mathbf{N}) - \operatorname{rank}(N), \\
\operatorname{rank}(\hat{F}) &= n + mr = \operatorname{rank}(\mathbf{F}) - \operatorname{rank}(F).
\end{aligned}
$$

This shows that γ and δ are kernels of α and β respectively. Furthermore, direct computations verify that $\gamma \circ \hat{P} = \mathbf{P} \circ \delta$ holds in all cases. This shows that $\hat{P}$ is the induced map between the kernels. □

Example 3.3.30. Continuing Example 3.3.28, we get $\hat{n} = \{\hat{e}_{01}, \hat{u}_{01}, \tilde{e}_1, \tilde{e}_2, \tilde{u}\}$ and $\hat{\mathcal{F}} = \{\hat{f}_{01}, \hat{f}_{02}, \hat{f}_{11}, \hat{f}_{21}\}$. We obtain the matrix representations

$$\gamma = \begin{bmatrix} -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix}, \qquad \delta = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix},$$

$$P = \begin{bmatrix} l_{01} & -l_{02} & 0 & 0 \\ d_{01} & -d_{02} & 0 & 0 \\ 0 & -l_{02} & l_{11} & 0 \\ 0 & -l_{02} & 0 & l_{21} \\ 0 & d_{02} & d_{11} & d_{21} \end{bmatrix}.$$

Remark 3.3.31. Let us give the general matrix representations of $\hat{P}$ from Construction 3.3.29 in each case. Compare this also to the toric version in Remark 2.3.4. Let $\hat{F}_i$ and $\hat{N}_i$ be the free lattices over $\hat{\mathcal{F}}_i$ and $\hat{n}_i$ respectively. Furthermore, let $\tilde{N}$ and $\overline{F}$ be the free lattice over $\tilde{n}$ and $\bar{\mathcal{F}}$ respectively.

We begin with the case (ee). Then the involved lattices are

$$\begin{aligned} \hat{N} &= \hat{N}_0 \oplus \cdots \oplus \hat{N}_r \oplus \tilde{N}, & \hat{F} &= \hat{F}_0 \oplus \cdots \oplus \hat{F}_r, \\ \hat{N}_i &= \bigoplus_{j=1}^{n_i - 1} \mathbb{Z}\,\hat{e}_{ij} \oplus \mathbb{Z}\,\hat{u}_{ij}, & \hat{F}_i &= \bigoplus_{j=1}^{n_i} \mathbb{Z}\,\hat{f}_{ij}, \\ \tilde{N} &= \mathbb{Z}\,\tilde{e}_1 \oplus \cdots \oplus \mathbb{Z}\,\tilde{e}_r \oplus \mathbb{Z}\,\tilde{u}. \end{aligned}$$

Consider the lattice maps

$$\hat{P}_i \colon \hat{F}_i \to \hat{N}_i, \qquad \hat{f}_{ij} \mapsto \begin{cases} l_{ij}\hat{e}_{ij} + d_{ij}\hat{u}_{ij}, & 1 \le j \le n_i - 1 \\ -l_{in_i}\left(\sum_{k=1}^{n_i-1} \hat{e}_{ik}\right) - d_{in_i}\left(\sum_{k=1}^{n_i-1} \hat{u}_{ik}\right), & j = n_i, \end{cases}$$

$$\tilde{P}_i \colon \hat{F}_i \to \tilde{N}, \qquad \hat{f}_{ij} \mapsto \begin{cases} 0, & 1 \le j \le n_i - 1 \\ l_{in_i}\tilde{e}_i + d_{in_i}\tilde{u}, & j = n_i. \end{cases}.$$

Then we have $\hat{P}(\hat{f}_{ij}) = \hat{P}_i(\hat{f}_{ij}) + \tilde{P}_i(\hat{f}_{ij})$. We obtain the matrix representations:

$$\tilde{P}_0 = \begin{bmatrix} 0 & \dots & 0 & -l_{0n_0} \\ 0 & \dots & 0 & -l_{0n_0} \\ \vdots & & \vdots & \vdots \\ 0 & \dots & 0 & -l_{0n_0} \\ 0 & \dots & 0 & d_{0n_0} \end{bmatrix}, \qquad \tilde{P}_i = \begin{bmatrix} 0 & \dots & 0 & 0 \\ \vdots & & \vdots & \vdots \\ 0 & \dots & 0 & l_{in_i} \\ \vdots & & \vdots & \vdots \\ 0 & \dots & 0 & 0 \\ 0 & \dots & 0 & d_{in_i} \end{bmatrix} \quad (i \geq 1),$$

$$\hat{P}_i = \begin{bmatrix} l_{i1} & 0 & \dots & 0 & -l_{in_i} \\ d_{i1} & 0 & \dots & 0 & -d_{in_i} \\ 0 & l_{i2} & & 0 & -l_{in_i} \\ 0 & d_{i2} & & 0 & -d_{in_i} \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & l_{in_i-1} & -l_{in_i} \\ 0 & 0 & \dots & d_{in_i-1} & -d_{in_i} \end{bmatrix}, \qquad \hat{P} = \begin{bmatrix} \hat{P}_0 & 0 & \dots & 0 \\ 0 & \hat{P}_1 & \dots & 0 \\ \vdots & & \ddots & \\ 0 & 0 & & \hat{P}_r \\ \tilde{P}_0 & \tilde{P}_1 & \dots & \tilde{P}_r \end{bmatrix}.$$

Next, we treat the case (pe). Here, the lattices are

$$\begin{aligned} \hat{N} &= \hat{N}_0 \oplus \cdots \oplus \hat{N}_r, & \hat{F} &= \hat{F}_0 \oplus \cdots \oplus \hat{F}_r \oplus \overline{F}, \\ \hat{N}_i &= \bigoplus_{j=1}^{n_i} \mathbb{Z}\,\hat{e}_{ij} \oplus \mathbb{Z}\,\hat{u}_{ij}, & \hat{F}_i &= \bigoplus_{j=1}^{n_i} \mathbb{Z}\,\hat{f}_{ij}, \\ & & \overline{F} &= \bigoplus_{i=1}^{r} \mathbb{Z}\,\hat{f}_i^-. \end{aligned}$$

We consider the lattice maps

$$\begin{aligned} &\hat{P}_i \colon \hat{F}_i \to \hat{N}_i, \quad \hat{f}_{ij} \mapsto l_{ij}\hat{e}_{ij} + d_{ij}\hat{u}_{ij}, && \text{for} \quad i = 0, \dots, r, \\ &\overline{P}_0 \colon \overline{F} \to \hat{N}_0, \quad \hat{f}_k^- \mapsto \begin{cases} -(\hat{u}_{01} + \cdots + \hat{u}_{0n_0}), & k = r, \\ \hat{u}_{01} + \cdots + \hat{u}_{0n_0}, & k = 1, \\ 0, & k \notin \{1, r\}, \end{cases} \\ &\overline{P}_i \colon \overline{F} \to \hat{N}_i, \quad \hat{f}_k^- \mapsto \begin{cases} -(\hat{u}_{i1} + \cdots + \hat{u}_{in_i}), & k = i, \\ \hat{u}_{i1} + \cdots + \hat{u}_{in_i}, & k = i+1, \\ 0, & k \notin \{i, i+1\}, \end{cases} && \text{for} \quad i = 1, \dots, r. \end{aligned}$$

Then we have $\hat{P}(\hat{f}_{ij}) = \hat{P}_i(\hat{f}_{ij})$ and $\hat{P}(\hat{f}_i^-) = \sum_{k=0}^{r} \overline{P}_k(\hat{f}_i^-)$. Thus we have the matrix repre-

sentations:

$$\overline{P}_0 = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & -1 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & -1 \end{bmatrix}, \qquad \overline{P}_i = \begin{bmatrix} 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \dots & -1 & 1 & \dots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \dots & -1 & 1 & \dots & 0 \end{bmatrix} \quad (i \geq 1),$$

$$\hat{P}_i = \begin{bmatrix} l_{i1} & 0 & \dots & 0 \\ d_{i1} & 0 & \dots & 0 \\ 0 & l_{i2} & & 0 \\ 0 & d_{i2} & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & l_{in_i} \\ 0 & 0 & \dots & d_{in_i} \end{bmatrix}, \qquad \hat{P} = \begin{bmatrix} \hat{P}_0 & 0 & \dots & 0 & \overline{P}_0 \\ 0 & \hat{P}_1 & \dots & 0 & \overline{P}_1 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & \hat{P}_r & \overline{P}_r \end{bmatrix}.$$

The case (ep) is analogous. Lastly, for (pp), the involved lattices are

$$\begin{aligned} \hat{N} &= \hat{N}_0 \oplus \cdots \oplus \hat{N}_r \oplus \tilde{N}, & \hat{F} &= \hat{F}_0 \oplus \cdots \oplus \hat{F}_r \oplus \overline{F}, \\ \hat{N}_i &= \bigoplus_{j=1}^{n_i} \mathbb{Z}\,\hat{e}_{ij} \oplus \mathbb{Z}\,\hat{u}_{ij}, & \hat{F}_i &= \bigoplus_{j=1}^{n_i} \mathbb{Z}\,\hat{f}_{ij}, \\ \tilde{N} &= \mathbb{Z}\,\hat{u}_1 \oplus \cdots \oplus \mathbb{Z}\,\hat{u}_r \oplus \mathbb{Z}\,\tilde{e}, & \overline{F} &= \bigoplus_{i=1}^{r} \mathbb{Z}\,\hat{f}_i^- \oplus \bigoplus_{i=1}^{r} \mathbb{Z}\,\hat{f}_i^+. \end{aligned}$$

Let us define $\hat{P}_i$ and $\overline{P}_i$ for $i = 0, \dots, r$ in exactly the same way as for (pe), and additionally set

$$\overline{\tilde{P}} \colon \overline{F} \to \tilde{N}, \qquad \hat{f}_i \mapsto -\tilde{u}_i, \quad \hat{f}_i^+ \mapsto \tilde{u}_i.$$

Then we have $\hat{P}(\hat{f}_{ij}) = \hat{P}_i(\hat{f}_{ij})$ and $\hat{P}(\hat{f}_i^{\pm}) = \overline{\tilde{P}}(\hat{f}_i^{\pm}) + \sum_{k=0}^{r} \overline{P}_k(\hat{f}_i^{\pm})$. Thus we obtain matrix representations:

$$\overline{\tilde{P}} = \begin{bmatrix} -1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 \\ 0 & -1 & \dots & 0 & 0 & 1 & \dots & 0 \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & -1 & 0 & 0 & \dots & 1 \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \end{bmatrix}, \qquad \hat{P} = \begin{bmatrix} \hat{P}_0 & 0 & \dots & 0 & \overline{P}_0 \\ 0 & \hat{P}_1 & \dots & 0 & \overline{P}_1 \\ \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \hat{P}_r & \overline{P}_r \\ 0 & 0 & \dots & 0 & \overline{\tilde{P}} \end{bmatrix}.$$

3.3.4 Maximal minors of $\hat{P}$

We begin by examining the set $M(\hat{P})$ of maximal minors from the matrix representation of $\hat{P}$ discussed earlier. Through a series of reduction steps, we can simplify $M(\hat{P})$ to a smaller set

that shares the same greatest common divisor. This allows us to make comparisons with $M'(P)$. The cases (pe), (ep), and (pp) are relatively straightforward; see Proposition 3.3.38. In contrast, the (ee) case requires more attention and is covered in the second half of this section; see Proposition 3.3.48.

Definition 3.3.32. Let a subset $A \subseteq \hat{n}$ with $|A| = |\hat{\mathcal{F}}|$ be given. Then we have a sublattice $\hat{N}_A := \bigoplus_{x\in A} \mathbb{Z}\cdot x \subseteq \hat{N}$ and an induced map $\hat{P}_A \colon \hat{F} \to \hat{N}_A$ as in the commutative diagram

$$\begin{array}{ccc} \hat{F} & \xrightarrow{\hat{P}} & \hat{N} \\ & \searrow^{\hat{P}_A} & \uparrow \\ & & \hat{N}_A. \end{array}$$

We call $|\det(\hat{P}_A)| \in \mathbb{Z}$ the *maximal minor of $\hat{P}$ associated to A*. The set of all maximal minors of $\hat{P}$ is defined as

$$M(\hat{P}) := \{|\det(\hat{P}_A)| \,;\, A \subseteq \hat{n},\ |\mathcal{A}| = |\hat{\mathcal{F}}|\}.$$

Construction 3.3.33. Let $A \subseteq \hat{n}$ with $|A| = |\hat{\mathcal{F}}|$. We define

$$\begin{aligned} \hat{n}_{\mathcal{A}}^{\text{sing}} &:= \{\hat{e}_{ij} \,;\, \hat{e}_{ij} \in A \text{ and } \hat{u}_{ij} \notin A\} \cup \{\hat{u}_{ij} \,;\, \hat{e}_{ij} \notin A \text{ and } \hat{u}_{ij} \in A\} \subseteq A, \\ \hat{\mathcal{F}}_{\mathcal{A}}^{\text{sing}} &:= \{\hat{f}_{ij} \,;\, \hat{e}_{ij} \in A \text{ and } \hat{u}_{ij} \notin A\} \cup \{\hat{f}_{ij} \,;\, \hat{e}_{ij} \notin A \text{ and } \hat{u}_{ij} \in A\} \subseteq \hat{\mathcal{F}}, \end{aligned}$$

$$\hat{n}_{\mathcal{A}}^{\text{red}} := \mathcal{A} \setminus \hat{n}_{\mathcal{A}}^{\text{sing}}, \qquad \hat{\mathcal{F}}_{\mathcal{A}}^{\text{red}} := \hat{\mathcal{F}} \setminus \hat{\mathcal{F}}_{\mathcal{A}}^{\text{sing}}.$$

Note that we have $|\hat{n}_{\mathcal{A}}^{\text{red}}| = |\hat{\mathcal{F}}_{\mathcal{A}}^{\text{red}}|$. Let $\hat{N}_A^{\text{red}}$ and $\hat{F}_A^{\text{red}}$ be the free lattices over $\hat{n}_{\mathcal{A}}^{\text{red}}$ and $\hat{\mathcal{F}}_{\mathcal{A}}^{\text{red}}$ respectively. We obtain an induced map $\hat{P}_A^{\text{red}}$ as in the commutative diagram

$$\begin{array}{ccc} \hat{F} & \xrightarrow{\hat{P}_A} & \hat{N}_A \\ \uparrow & & \uparrow \\ \hat{F}_A^{\text{red}} & \xrightarrow[\hat{P}_A^{\text{red}}]{} & \hat{N}_A^{\text{red}}. \end{array}$$

We call $|\det(\hat{P}_A^{\text{red}})|$ the *reduced minor of $\hat{P}$ associated to A*. We define the set of reduced minors of $\hat{P}$ as

$$M^{\text{red}}(\hat{P}) := \{|\det(\hat{P}_A^{\text{red}})| \,;\, A \subseteq \hat{n},\ |\mathcal{A}| = |\hat{\mathcal{F}}|\}.$$

Example 3.3.34. The matrix $\hat{P}$ from Example 3.3.30 has five maximal minors and four reduced minors:

$$\begin{aligned} M(\hat{P}) &= \{d_{01}|\hat{\mu}|, \quad l_{01}|\hat{\mu}|, \quad d_{11}l_{21}|\hat{\nu}(0,1)|, \quad l_{11}d_{21}|\hat{\nu}(0,1)|, \quad l_{11}l_{21}|\hat{\nu}(0,1)|\} \\ M^{\text{red}}(\hat{P}) &= \{|\hat{\mu}|, \quad d_{11}l_{21}|\hat{\nu}(0,1)|, \quad l_{11}d_{21}|\hat{\nu}(0,1)|, \quad l_{11}l_{21}|\hat{\nu}(0,1)|\}. \end{aligned}$$

Proposition 3.3.35. *Let $A \subseteq \hat{\mathcal{N}}$ with $|A| = |\hat{\mathcal{F}}|$. Then we have*

$$\det(\hat{P}_A) = \det(\hat{P}_A^{\mathrm{red}}) \prod_{\hat{f}_{ij} \in \hat{\mathcal{F}}_{\mathcal{A}}^{\mathrm{sing}}} x_{ij}, \quad \textit{where} \quad x_{ij} := \begin{cases} l_{ij}, & \hat{e}_{ij} \in A, \\ d_{ij}, & \hat{u}_{ij} \in A. \end{cases}$$

Moreover, $\gcd(M(\hat{P})) = \gcd(M^{\mathrm{red}}(\hat{P}))$ *holds.*

Proof. If $\hat{f}_{ij} \in \hat{\mathcal{F}}_{\mathcal{A}}^{\mathrm{sing}}$, we have

$$\hat{P}_A(\hat{f}_{ij}) = \begin{cases} l_{ij}\hat{e}_{ij}, & \hat{e}_{ij} \in A, \\ d_{ij}\hat{u}_{ij}, & \hat{u}_{ij} \in A. \end{cases}$$

In other words, the matrix representation of $\hat{P}_A$ has a column with the single entry x_{ij} and zeroes elsewhere. Doing cofactor expansion by all these columns amounts to passing from $\det(\hat{P}_A)$ to $\det(\hat{P}_A^{\mathrm{red}})$. This shows the first claim. The second one then follows from $\gcd(l_{ij}, d_{ij}) = 1$. □

Definition 3.3.36. Set $\mathcal{L} := \{(i,j)\,;\ i = 0, \ldots, r,\ j = 1, \ldots, n_i'\}$. For $A \subseteq \hat{\mathcal{N}}$ with $|A| = |\hat{\mathcal{F}}|$, we define

$$L(A) \ := \ \{(i,j)\,;\ \hat{e}_{ij} \in A \text{ and } \hat{u}_{ij} \in A\} \ \subseteq \ \mathcal{L}.$$

Lemma 3.3.37. *Let $A \subseteq \hat{\mathcal{N}}$ with $|A| = |\hat{\mathcal{F}}|$.*

(i) *If $\hat{e}_{ij} \notin A$ and $\hat{u}_{ij} \notin A$ for some $i = 0, \ldots, r$ and $j = 1, \ldots, n_i'$, we have* $\det(\hat{P}_A) = 0$.

(ii) *If $(i, j_0), (i, j_1) \in L(A)$ for some $i = 0, \ldots, r$ and $1 \leq j_0 < j_1 \leq n_i'$, we have* $\det(\hat{P}_A) = 0$.

Proof. For (i), we have $\hat{P}_A(\hat{f}_{ij}) = 0$, hence $\det(\hat{P}_A) = 0$. We show (ii). Consider first the case (ee). Then we have $n_i' = n_i - 1$. Set $\hat{\mathcal{N}}_{\mathcal{A},i} := \mathcal{A} \cap \hat{\mathcal{N}}_i$ and $\hat{N}_{A,i} := \hat{N}_A \cap \hat{N}_i$. By (i), we may assume that $\hat{e}_{ij} \in \hat{\mathcal{N}}_{\mathcal{A},i}$ or $\hat{u}_{ij} \in \hat{\mathcal{N}}_{\mathcal{A},i}$ holds for all $1 \leq j \leq n_i - 1$. Since $(i, j_0), (i, j_1) \in L(A)$, we thus have $|\hat{\mathcal{N}}_{\mathcal{A},i}| > n_i$. Consider the map $\hat{P}_{A,i} \colon \hat{N}_{A,i} \to \hat{F}_i$. In the matrix representation from Remark 3.3.31, we have

$$\hat{P}_A = \begin{bmatrix} * & 0 & 0 \\ * & \boxed{\hat{P}_{A,i} \quad 0} & 0 \\ * & * & * \end{bmatrix},$$

where the outlined box is a square $|\hat{\mathcal{N}}_{\mathcal{A},i}| \times |\hat{\mathcal{N}}_{\mathcal{A},i}|$-matrix. Since the determinant of the outlined box vanishes, also $\det(\hat{P}_A) = 0$.

Now let $\mathfrak{c} \in \{(\mathrm{pe}), (\mathrm{ep}), (\mathrm{pp})\}$. Then $n_i' = n_i$ and we define the set

$$\bar{\mathcal{N}}_{\mathcal{A}} \ := \ \{\hat{u}_{ij}\,;\ (i,j) \in \mathcal{L}(\mathcal{A})\} \ \subseteq \ \hat{\mathcal{N}}_{\mathcal{A}}^{\mathrm{red}}.$$

Writing $\bar{N}_A$ for the free lattice over $\bar{n}_{\mathcal{A}}$ and $\bar{F}$ for the free lattice over $\bar{\mathcal{F}}$, we obtain an induced map $\bar{P}_A\colon \bar{F} \to \bar{N}_A$ as in the commutative diagram

$$\begin{array}{ccc} \hat{F}_A^{\mathrm{red}} & \xrightarrow{\hat{P}_A^{\mathrm{red}}} & \hat{N}_A^{\mathrm{red}} \\ \uparrow & & \uparrow \\ \bar{F} & \xrightarrow[\bar{P}_A]{} & \bar{N}_A. \end{array}$$

Note that if $(i,j) \in L(A)$, we have $(\hat{P}_A^{\mathrm{red}})^*(\hat{e}_{ij}^*) = l_{ij}\hat{f}_{ij}^*$. That means, $\hat{P}_A^{\mathrm{red}}$ contains a row with a single entry l_{ij} and zeroes elsewhere. Doing cofactor expansion, we arrive at

$$\det(\hat{P}_A^{\mathrm{red}}) = \det(\bar{P}_A) \prod_{(i,j)\in L(A)} l_{ij}.$$

But since $(i,j_0),(i,j_1) \in L(A)$, we have $\bar{P}_A^*(\hat{u}_{ij_0}^*) = \bar{P}_A^*(\hat{u}_{ij_1}^*)$, i.e. $\bar{P}_A$ contains two equal rows. Hence we have $\det(\hat{P}_A) = \det(\hat{P}_A^{\mathrm{red}}) = \det(\bar{P}_A) = 0$. □

Proposition 3.3.38. *Assume* $\mathfrak{c} \in \{(\mathrm{pe}),(\mathrm{ep}),(\mathrm{pp})\}$. *Let* $A \subseteq \hat{n}$ *with* $|A| = |\hat{\mathcal{F}}|$ *such that* $\det(\hat{P}_A) \neq 0$. *Then we have* $|L(A)| = r$. *Furthermore, there exists an* $i_1 = 0,\dots,r$ *and* $j_i = 1,\dots,n_i$ *for all* $i \neq i_1$ *such that*

$$|\det(\hat{P}_A^{\mathrm{red}})| = \prod_{i\neq i_1} l_{ij_i}.$$

In particular, we have $\gcd(M^{\mathrm{red}}(\hat{P})) = \gcd(M'(P))$.

Proof. By Lemma 3.3.37 (i), we have $\hat{e}_{ij} \in A$ or $\hat{u}_{ij} \in A$ for all i and j. By Lemma 3.3.37 (ii), for each i there is at most one j with $\hat{e}_{ij} \in A$ and $\hat{u}_{ij} \in A$. Writing $\pi_1\colon \mathbb{Z}\times\mathbb{Z} \to \mathbb{Z}$ for the projection onto the first coordinate, this implies that $|\pi_1(L(A))| = |L(A)|$. Together, we obtain

$$|A \cap \hat{n}_i| = \begin{cases} n_i, & i \notin \pi_1(L(A)), \\ n_i + 1, & i \in \pi_1(L(A)). \end{cases}$$

Thus, we have $|A| = n + |L(A)| + |A\cap\tilde{n}|$. Recall that for the cases (pe) and (ep), we have $\tilde{n} = \emptyset$ and $|A| = |\hat{\mathcal{F}}| = n + \mathfrak{r}$, hence $|L(A)| = r$. For (pp), we must have $\tilde{u}_i \in A$ for all $i = 1,\dots,r$, since otherwise $\hat{P}_A(\hat{f}_i^+) = 0$. Hence $|A\cap\tilde{n}| = \mathfrak{r}$. Since in this case, $|A| = n+2r$, we also arrive at $|L(A)| = r$. This implies that we have $L(A) = \{(i,j_i)\ ;\ i \neq i_1\}$ for some $i_1 = 0,\dots,r$ and $j_i = 1,\dots,n_i$. Following the proof of Lemma 3.3.37 (ii), we proceed to do cofactor expansion and see that $|\det(\bar{P}_A)| = 1$. This proves the claim. The supplement follows directly from the Definition of $M'(P)$. □

The preceding proposition settles the discussion of maximal minors of $\hat{P}$ for the cases (pe), (ep) and (pp). For the remainder of this section, we assume $\mathfrak{c} = (\mathrm{ee})$.

Lemma 3.3.39. *Let $A \subseteq \hat{\mathcal{N}}$ with $|A| = |\hat{\mathcal{F}}| = n$ and $\det(\hat{P}_A) \neq 0$. Then we have*

$$|L(A)| = (r+1) - |A \cap \tilde{\mathcal{N}}|.$$

In particular, $0 \leq |L(A)| \leq r+1$.

Proof. Let $\pi_1 \colon \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ be the projection onto the first coordinate. Lemma 3.3.37 (ii) implies $|\pi_1(L(A))| = |L(A)|$. Furthermore, we have

$$|A \cap \hat{\mathcal{N}}_i| = \begin{cases} n_i - 1, & i \notin \pi_1(L(A)) \\ n_i, & i \in \pi_1(L(A)) \end{cases}.$$

Since, $|A| = n$, this implies the claim. □

Definition 3.3.40. For $k = 0, \ldots, r+1$, we define

$$M_k^{\mathrm{red}}(\hat{P}) := \{|\det(\hat{P}_A^{\mathrm{red}})| \,;\, |L(A)| = k\}.$$

Remark 3.3.41. Let $A \subseteq \hat{\mathcal{F}}$ with $|A| = n$ and $L(A) = \emptyset$. Lemma 3.3.39 implies that $A \cap \tilde{\mathcal{N}} = \tilde{\mathcal{N}}$. We obtain $\det(\hat{P}_A^{\mathrm{red}}) = \hat{\mu}$. Hence $M_0^{\mathrm{red}}(\hat{P}) = \{|\hat{\mu}|\}$.

Proposition 3.3.42. *Let $A \subseteq \hat{\mathcal{N}}$ with $|A| = n$ such that $|\hat{P}_A^{\mathrm{red}}| \neq 0$. Write $\pi_1 \colon \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ for the projection onto the first coordinate.*

(i) *Assume $\tilde{u} \notin A$. Then for all $i = 1, \ldots, r$, we have $\tilde{e}_i \in A$ or $i \in \pi_1(L(A))$. In this case,*

$$|\det(\hat{P}_A^{\mathrm{red}})| = \left(\prod_{(i,j) \in L(A)} |\hat{\nu}(i,j)| \right) \left(\prod_{\substack{0 \leq i \leq r \\ i \notin \pi_1(L(A))}} l_{in_i} \right).$$

(ii) *Assume $\tilde{u} \in A$. Then there exists at most one $i_1 = 1, \ldots, r$ with $\tilde{e}_{i_1} \notin A$ and $i_1 \notin \pi_1(L(A))$. If there exists such an i_1, we have*

$$|\det(\hat{P}_A^{\mathrm{red}})| = \left| d_{i_1 n_{i_1}} \left(\prod_{(i,j) \in L(A)} \hat{\nu}(i,j) \right) \left(\prod_{\substack{0 \leq i \leq r \\ i \notin \pi_1(L(A)) \cup \{i_1\}}} l_{in_i} \right) \right|.$$

If there exists no such i_1, we have

$$|\det(\hat{P}_A^{\mathrm{red}})| = \left| \sum_{\substack{0 \leq i' \leq r \\ i' \notin \pi_1(L(A))}} \pm d_{i' n_{i'}} \left(\prod_{(i,j) \in L(A)} \hat{\nu}(i,j) \right) \left(\prod_{\substack{0 \leq i \leq r \\ i \notin \pi_1(L(A)) \cup \{i'\}}} l_{in_i} \right) \right|.$$

Proof. For part (i), assume that there is some $i = 1, \ldots, r$ such that $\tilde{e}_i \notin A$ and $i \notin \pi_1(L(A))$. This implies $\hat{e}_{ij} \notin \hat{\mathcal{N}}_{\mathcal{A}}^{\mathrm{red}}$ and $\hat{u}_{ij} \notin \hat{\mathcal{N}}_{\mathcal{A}}^{\mathrm{red}}$ for all $j = 1, \ldots, n_i - 1$. Since also $\tilde{u} \notin A$, we obtain $\hat{P}_A^{\mathrm{red}}(\hat{f}_{in_i}) = 0$. Hence $\det(\hat{P}_A^{\mathrm{red}}) = 0$, a contradiction. It follows that if $i \notin \pi_1(L(A))$, either $i = 0$ or $\tilde{e}_i \in A$. The formula for $\det(\hat{P}_A^{\mathrm{red}})$ now follows from cofactor expansion.

For part (ii), assume that there exist $1 \leq i_0 < i_1 \leq r$ such that $\tilde{e}_{i_0}, \tilde{e}_{i_1} \notin A$ and $i_0, i_1 \notin \pi_1(L(A))$. We obtain $\hat{P}_A^{\mathrm{red}}(\hat{f}_{i_0 n_{i_0}}) = d_{i_0 n_{i_0}} \tilde{u}$ and $\hat{P}_A^{\mathrm{red}}(\hat{f}_{i_1 n_{i_1}}) = d_{i_1 n_{i_1}} \tilde{u}$. Hence $\det(\hat{P}_A^{\mathrm{red}}) = 0$, a contradiction. The formulas for $\det(\hat{P}_A^{\mathrm{red}})$ again follow from cofactor expansion. □

Definition 3.3.43. Let $\pi_1 \colon \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ be the projection onto the first coordinate. We call a subset $L \subseteq \mathcal{L}$ *valid*, if $|\pi_1(L)| = |L|$. For $k = 1, \ldots, r$, we define

$$M_k'(\hat{P}) := \left\{ \left(\prod_{(i,j) \in L} |\hat{\nu}(i,j)| \right) \left(\prod_{\substack{0 \leq i \leq r \\ i \notin \pi_1(L) \cup \{i_1\}}} l_{in_i} \right) \; ; \; \begin{array}{l} L \subseteq \mathcal{L} \text{ valid}, \; |L| = k, \\ 0 \leq i_1 \leq r, \; i_1 \notin \pi_1(L), \end{array} \right\}.$$

Example 3.3.44. Continuing Example 3.3.34, we have

$$M_0^{\mathrm{red}}(\hat{P}) = \{|\hat{\mu}|\}, \quad M_1^{\mathrm{red}}(\hat{P}) = \{d_{11} l_{21} |\hat{\nu}(0,1)|, \; l_{11} d_{21} |\hat{\nu}(0,1)|, \; l_{11} l_{21} |\hat{\nu}(0,1)|\}$$

$$M_1'(\hat{P}) = \{l_{11} |\hat{\nu}(0,1)|, \; l_{21} |\hat{\nu}(0,1)|\}, \qquad M_2'(\hat{P}) = M_2^{\mathrm{red}}(\hat{P}) = M_3^{\mathrm{red}}(\hat{P}) = \emptyset.$$

Proposition 3.3.45. *We have*

(i) $\gcd(M_k^{\mathrm{red}}(\hat{P})) = \gcd(M_k'(\hat{P}))$ *for all* $k = 1, \ldots, r$,

(ii) $\gcd(M_{r+1}^{\mathrm{red}}(\hat{P}) \cup M_r'(\hat{P})) = \gcd(M_r'(\hat{P}))$,

(iii) $\gcd\left(\bigcup_{k=1}^{r+1} M_k^{\mathrm{red}}(\hat{P})\right) = \gcd\left(\bigcup_{k=1}^{r} M_k'(\hat{P})\right)$.

Proof. We show (i). Proposition 3.3.42 implies that every element of $M_k^{\mathrm{red}}(\hat{P})$ is a $\mathbb{Z}$-linear combination of elements of $M_k'(\hat{P})$. This shows that $\gcd(M_k'(\hat{P}))$ divides $\gcd(M_k^{\mathrm{red}}(\hat{P}))$. For the converse, it suffices to show that $\gcd(M_k^{\mathrm{red}}(\hat{P})) \mid x$ holds for all $x \in M_k'(\hat{P})$. So, let

$$x = \left(\prod_{(i,j) \in L} \nu(i,j) \right) \left(\prod_{\substack{0 \leq i \leq r \\ i \notin \pi_1(L) \cup \{i_1\}}} l_{in_i} \right) \in M_k'(\hat{P})$$

be arbitrary, where $L \subseteq \mathcal{L}$ is a valid subset with $|L| = k$ and $0 \leq i_1 \leq r$ with $i_1 \notin \pi_1(L)$.

Case 1: $i_1 \neq 0$ *and* $0 \in \pi_1(L)$. Choose a subset $A \subseteq \hat{\mathcal{N}}$ with $|A| = n$ such that $L(A) = L$ and

$$A \cap \tilde{\mathcal{N}} = \{\tilde{e}_i \; ; \; 1 \leq i \leq r, \; i \notin \pi_1(L)\}.$$

Since $0 \in \pi_1(L)$, we have $|A \cap \tilde{\mathcal{N}}| = r - (|L| - 1) = (r+1) - |L|$. This means we can choose A such that $\det(\hat{P}_A^{\mathrm{red}}) \neq 0$. Now set

$$A' := (A \backslash \{\tilde{e}_{i_1}\}) \cup \{\tilde{u}\}.$$

Proposition 3.3.42 implies that $\det(\hat{P}_A^{\mathrm{red}}) = l_{i_1 n_{i_1}} x$ and $\det(\hat{P}_{A'}^{\mathrm{red}}) = d_{i_1 n_{i_1}} x$. Hence we have

$$\gcd(M_k^{\mathrm{red}}(\hat{P})) \mid \gcd(l_{i_1 n_{i_1}} x, d_{i_1 n_{i_1}} x) = x.$$

Case 2: $i_1 \neq 0$ *and* $0 \notin \pi_1(L)$. Since $k \geq 1$, we find some $i_0 \in \pi_1(L)$. Now choose $A \subseteq \hat{\mathcal{N}}$ with $|A| = n$ such that $L(A) = L$ and

$$A \cap \tilde{\mathcal{N}} = \{\tilde{e}_i \,;\, 1 \leq i \leq r,\ i \notin \pi_1(L)\} \cup \{\tilde{e}_{i_0}\}.$$

Again, we have $|A \cap \tilde{\mathcal{N}}| = (r+1) - |L|$, hence we can pick A such that $\det(\hat{P}_A^{\mathrm{red}}) \neq 0$. Proceeding in the same way as in Case 1, we arrive at $\gcd(M_k^{\mathrm{red}}(\hat{P})) \mid x$.

Case 3: $i_1 = 0$. As in Case 2, we can pick some $i_0 \in \pi_1(L)$ as well as a subset $A \subseteq \hat{\mathcal{N}}$ with $|A| = n$ such that $L(A) = L$ and

$$A \cap \tilde{\mathcal{N}} = \{\tilde{e}_i \,;\, 1 \leq i \leq r,\ i \notin \pi_1(L)\} \cup \{\tilde{e}_{i_0}\}.$$

For all $1 \leq i \leq r$ with $i \notin \pi_1(L)$, set $A_i := (A \backslash \{\tilde{e}_i\}) \cup \{\tilde{u}\}$. Then Proposition 3.3.42 implies that $\det(P_A^{\mathrm{red}}) = l_{0n_0} x$ and

$$\det(P_{A_{i_0}}^{\mathrm{red}}) = d_{0n_0} x + \sum_{\substack{1 \leq i' \leq r \\ i' \notin \pi_1(L)}} \pm \det(P_{A_{i'}}^{\mathrm{red}}).$$

This implies $\gcd(M_k^{\mathrm{red}}(\hat{P})) \mid \gcd(l_{0n_0} x, d_{0n_0} x) = x$.

Part (ii) follows from the fact that every element of $M_{r+1}^{\mathrm{red}}(\hat{P})$ is an integer multiple of an element of $M_r'(\hat{P})$. Part (iii) is a consequence of (i) and (ii). □

Definition 3.3.46. For $k = 1, \ldots, r$, we define the set

$$M_k''(\hat{P}) := \left\{ \left(\prod_{(i,j) \in L} |\hat{\nu}(i,j)| \right) \left(\prod_{\substack{0 \leq i \leq r \\ i \notin \pi_1(L) \cup \{i_1\}}} l_{ij_i} \right) \;;\; \begin{array}{c} L \subseteq \mathcal{L} \text{ valid},\ |L| = k \\ 0 \leq i_1 \leq r,\ i_1 \notin \pi_1(L) \\ 1 \leq j_i \leq n_i \text{ for all } i \end{array} \right\}.$$

Lemma 3.3.47. *Let* $i = 0, \ldots, r$ *and* $j_i = 1, \ldots, n_i - 1$. *Then we have*

$$\gcd(l_{in_i}, \hat{\nu}(i,j)) \mid l_{ij}.$$

Proof. By definition we have $\hat{\nu}(i,j) = l_{in_i} d_{ij} - l_{ij} d_{in_i}$. Since l_{in_i} and d_{in_i} are coprime, we find $x, y \in \mathbb{Z}$ such that $x l_{in_i} + y d_{in_i} = 1$. Then we have

$$(x l_{ij} + y d_{in_i}) l_{in_i} - y \hat{\nu}(i,j) = l_{ij}(x l_{in_i} + y d_{in_i}) = l_{ij}.$$

This implies the claim. □

Proposition 3.3.48. *We have*

(i) $\gcd(M'_k(\hat{P}) \cup M''_{k+1}(\hat{P})) = \gcd(M''_k(\hat{P}))$ *for all* $k = 1, \ldots, r-1$,

(ii) $\gcd\left(\bigcup_{k=1}^{r} M'_k(\hat{P})\right) = \gcd(M''_1(\hat{P}))$.

Proof. We show (i). Since $M'_k(\hat{P}) \subseteq M''_k(\hat{P})$ and elements of $M''_{k+1}(\hat{P})$ are $\mathbb{Z}$-linear combinations of elements of $M''_k(\hat{P})$, we have $\gcd(M''_k(\hat{P})) \mid \gcd(M'_k(\hat{P}) \cup M''_{k+1}(\hat{P}))$. For the converse, let

$$x = \left(\prod_{(i,j)\in L} \hat{\nu}(i,j)\right)\left(\prod_{\substack{0\le i\le r \\ i\notin \pi_1(L)\cup\{i'\}}} l_{ij_i}\right) \in M''_k(\hat{P}),$$

where $L \subseteq \mathscr{L}$ is a valid subset with $|L| = k$ and $0 \le i' \le r$ with $i' \notin \pi_1(L)$ and $1 \le j_i \le n_i$ for all $i \notin \pi_1(L) \cup \{i'\}$. Let us write

$$\{i_1, \ldots, i_{r-k}\} := \{0, \ldots, r\} \backslash (\pi_1(L) \cup \{i'\}).$$

We define numbers

$$\begin{aligned} x_0 &:= l_{i_1 n_{i_1}} l_{i_2 n_{i_2}} \ldots l_{i_{r-k} n_{i_{r-k}}} \\ x_1 &:= l_{i_1 j_{i_1}} l_{i_2 n_{i_2}} \ldots l_{i_{r-k} n_{i_{r-k}}} \\ &\vdots \\ x_{r-k} &:= l_{i_1 j_{i_1}} l_{i_2 j_{i_2}} \ldots l_{i_{r-k} j_{i_{r-k}}} \end{aligned}$$

as well as

$$\begin{aligned} y_1 &:= \hat{\nu}(i_1, j_{i_1}) l_{i_2 n_{i_2}} l_{i_3 n_{i_3}} \ldots l_{i_{r-k} n_{i_{r-k}}} \\ y_2 &:= l_{i_1 j_{i_1}} \hat{\nu}(i_2, j_{i_2}) l_{i_3 n_{i_3}} \ldots l_{i_{r-k} n_{i_{r-k}}} \\ &\vdots \\ y_{r-k} &:= l_{i_1 j_{i_1}} \ldots l_{i_{r-k-1} j_{i_{r-k-1}}} \hat{\nu}(i_{r-k}, j_{i_{r-k}}). \end{aligned}$$

Then Lemma 3.3.47 implies $\gcd(x_{m-1}, y_m) \mid x_m$ for all $m = 1, \ldots, r-k$. In particular, we obtain $\gcd(x_0, y_1, \ldots, y_{r-k}) \mid x_{r-k}$. Now set $c := \prod_{(i,j)\in L} \hat{\nu}(i,j)$. Then we have $c x_{r-k} = x$ as well as $x_0 c \in M'_k(\hat{P})$ and $y_m c \in M''_{k+1}(\hat{P})$ for all $m = 1, \ldots r-k$. Together, we have

$$\gcd(M'_k(\hat{P}) \cup M''_{k+1}(\hat{P})) \mid \gcd(x_0 c, y_1 c, \ldots, y_{r-k} c) \mid x_{r-k} c = x.$$

Part (ii) follows from repeated application of (i), together with the fact that $M'_r(\hat{P}) = M''_r(\hat{P})$. □

Proposition 3.3.49. *We have* $\gcd(M(\hat{P})) = \gcd(M(P))$.

Proof. By Construction 3.3.33, we have $\gcd(M(\hat{P})) = \gcd(M^{\mathrm{red}}(\hat{P}))$. On the other hand, Proposition 3.3.24 says that $\gcd(M(P)) = \gcd(M'(P))$. In the cases (pe), (ep) and (pp), Proposition 3.3.38 gives the result. In the case (ee), combining Remark 3.3.41, Proposition 3.3.45 (iii) and Proposition 3.3.48 (ii), we get

$$\begin{aligned}\gcd(M^{\mathrm{red}}(\hat{P})) &= \gcd\left(\{|\hat{\mu}|\} \cup \bigcup_{k=1}^{r+1} M_k^{\mathrm{red}}(\hat{P})\right)\\ &= \gcd\left(\{|\hat{\mu}|\} \cup \bigcup_{k=1}^{r} M_k'(\hat{P})\right)\\ &= \gcd\left(\{|\hat{\mu}|\} \cup M_1''(\hat{P})\right).\end{aligned}$$

By definition, we have $\{|\hat{\mu}|\} \cup M_1''(\hat{P}) = M'(P)$, hence we arrive at the claim. □

3.3.5 Completing the proof and examples

In this section, we finally complete the proof of the Picard index formula from Theorem 3.3.13, which implies Theorem 7. We then give two examples where the formula fails: The first one is a toric threefold, the second one is the D_8-singular log del Pezzo surface of Picard number one.

Proof of Theorem 3.3.13. By applying Proposition 2.1.9 to the toric ambient $Z(\mathfrak{c}, l, d)$, we see that it is enough to show the equality $\mu = |\hat{K}|$, where $\hat{K}$ is the cokernel of the map $\hat{P}$ described in Subsection 3.3.3. By definition, μ is the greatest common divisor of all maximal minors of P. On the other hand, $|\hat{K}|$ is the greatest common divisor of all maximal minors of $\hat{P}$. Hence Proposition 3.3.49 implies the claim. □

Example 3.3.50. Consider once more the defining triple from Example 3.3.16. The multiplicity is $\mu = 4$, while the local multiplicities are

$$\mu^+ = 20, \qquad \mu^- = 12, \qquad \mu_{01} = 1.$$

By Theorem 3.3.13, we obtain $\iota_{\mathrm{Pic}}(X) = 60$.

The following example shows that Theorem 7 does not hold for higher dimensional toric varieties:

Example 3.3.51. Consider the three-dimensional weighted projective space $Z = \mathbb{P}(2, 2, 3, 5)$. Note that the weights are well-formed, i.e. any three weights have no common factor. The Picard group is given by

$$\mathrm{Pic}(Z) = 2\,\mathbb{Z} \cap 2\,\mathbb{Z} \cap 3\,\mathbb{Z} \cap 5\,\mathbb{Z} = 30\,\mathbb{Z} \subseteq \mathbb{Z} \cong \mathrm{Cl}(Z).$$

Hence we have $\iota_{\mathrm{Pic}}(Z) = 30$. On the other hand, the product of the orders of the local class groups is 60.

We now give an example of a singular del Pezzo surface which does not admit a $\mathbb{C}^*$-action, where Theorem 7 fails. It is the D_8-singular log del Pezzo surface of Picard number one. Using the description of its Cox ring [30, Theorem 4.1], we construct the surface via its canonical ambient toric variety, see also [2, Sections 3.2 and 3.3]. This allows us to compute its class group, local class groups and Picard index.

Example 3.3.52. Consider the integral matrix

$$P := \begin{bmatrix} v_1 & v_2 & v_3 & v_4 \end{bmatrix} := \begin{bmatrix} 1 & 0 & 1 & -3 \\ 0 & 1 & 1 & -2 \\ 0 & 0 & 2 & -2 \end{bmatrix}.$$

Let $Z = Z_\Sigma$ be the toric variety whose fan Σ has the following maximal cones:

$$\sigma_{12} := \operatorname{cone}(v_1, v_2), \qquad \sigma_{23} := \operatorname{cone}(v_2, v_3), \qquad \sigma_{24} := \operatorname{cone}(v_2, v_4),$$

$$\sigma_{134} := \operatorname{cone}(v_1, v_3, v_4).$$

Let $p\colon \hat{Z} \to Z$ be Cox's quotient presentation of Z, where $\hat{Z} \subseteq \bar{Z} := \mathbb{C}^4$. Consider the polynomial

$$f := T_1^2 - T_2 T_3 T_4^2 + T_3^4 + T_4^4.$$

We obtain a commutative diagram

$$\begin{array}{rcccc} V(f) & =: & \bar{X} & \hookrightarrow & \bar{Z} & = & \mathbb{C}^4 \\ & & \cup\mathrm{l} & & \cup\mathrm{l} & & \\ \bar{X} \cap \hat{Z} & =: & \hat{X} & \hookrightarrow & \hat{Z} & & \\ & & \downarrow p & & \downarrow p & & \\ p(\hat{X}) & =: & X & \xhookrightarrow{\iota} & Z. & & \end{array}$$

Here, $\iota\colon X \to Z$ is the canonical toric embedding in the sense of [2, Section 3.2.5]. This implies that X has non-empty intersection with the toric orbits $\mathbb{T}^3 \cdot z_\sigma$ for $\sigma \in \Sigma_{\max}$. We obtain a decomposition into pairwise disjoint pieces

$$X = \bigcup_{\sigma \in \Sigma_{\max}} X(\sigma), \qquad \text{where } X(\sigma) := X \cap \mathbb{T}^3 \cdot z_\sigma.$$

By [2, Proposition 3.3.1.5], we have $\operatorname{Cl}(X, x) \cong \operatorname{Cl}(Z, z_\sigma)$ for $x \in X(\sigma)$. Note that σ_{12}, σ_{23} and σ_{24} are regular and $|\operatorname{Cl}(Z, z_{\sigma_{134}})| = |\det(v_1, v_3, v_4)| = 2$. This shows that

$$\prod_{x \in X} |\operatorname{Cl}(X, x)| = \prod_{\sigma \in \Sigma_{\max}} |\operatorname{Cl}(z, z_\sigma)| = 2.$$

We turn to the Picard index. In the notation of Construction 2.1.5, we have

$$N_{\sigma_{ij}} = \mathbb{Z}\, v_i + \mathbb{Z}\, v_j, \qquad N_{\sigma_{134}} = N = \mathbb{Z}^3.$$

Under the lattice bases $\{v_i, v_j\}$ of $N_{\sigma_{ij}}$, we can view $P_{\sigma_{ij}}$ as the identity matrix and $P_{\sigma_{134}} = \begin{bmatrix} v_1 & v_3 & v_4 \end{bmatrix}$. Computing matrix representations of the maps involved in Construction 2.1.5, we obtain

$$\alpha = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & -3 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & -2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & -2 & 0 & 0 & 1 \end{bmatrix}, \quad \beta = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix},$$

$$\gamma = \begin{bmatrix} -1 & 0 & 3 & -1 & 0 & 0 \\ -1 & -1 & 2 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ -2 & 0 & 2 & 0 & 0 & 0 \end{bmatrix}, \quad \delta = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & 0 \end{bmatrix},$$

$$\hat{P} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 3 & 1 & 0 & -1 \\ 0 & 2 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

We see that $\hat{P}^\top$, is surjective, hence its cokernel $\hat{K}$ is trivial. Using Proposition 2.1.9, we conclude $\iota_{\mathrm{Pic}}(X) = \iota_{\mathrm{Pic}}(Z) = 2$. On the other hand, we have

$$\mathrm{Cl}(X) \cong \mathrm{Cl}(Z) \cong \mathbb{Z}^3 / \operatorname{im}(P^\top) \cong \mathbb{Z} \times \mathbb{Z}/2\,\mathbb{Z}\,.$$

This shows that the formula from Theorem 7 does not hold for X.

3.4 Classifications by Picard index

In this section, we provide algorithms to classify $\mathbb{C}^*$-surfaces of Picard number one by Picard index. As an application, we classify the non-toric $\mathbb{C}^*$-surfaces of Picard number one having at most log terminal singularities up to Picard index 10 000, see Classification 3.4.7. Implementations are available in `CStarSurfaces.jl`, see Section B.4. Furthermore, we prove that there are no non-toric log del Pezzo $\mathbb{C}^*$-surfaces of Picard number one whose Picard index is a Mersenne prime number greater than seven, see Theorem 3.4.12

Parts of this section have been published in shortened form as [56, Section 8].

Let $(\mathfrak{c}, l, d)$ be a defining triple. Recall that $l = (l_0, \ldots, l_r)$ and $d = (d_0, \ldots, d_r)$, where $l_i \in \mathbb{Z}^{n_i}_{\geq 0}$ and $d_i \in \mathbb{Z}^{n_i}$. Furthermore, we write $n = n_0 + \cdots + n_r$ and $m \in \{0, 1, 2\}$ for the number of parabolic fixed point curves, see Definition 3.1.2.

Proposition 3.4.1. *Let $(\mathfrak{c}, l, d)$ be an irredundant defining triple of Picard number one with $r > 1$. Then $(\mathfrak{c}, l, d)$ is equivalent to a defining triple of one of the following forms:*

(i) $\mathfrak{c} = (\mathrm{ee})$, $(n_0, n_1, \ldots, n_r) = (2, 1, \ldots, 1)$ *and* $0 < d_i < l_i$ *for* $i = 1, \ldots, r$.

(ii) $\mathfrak{c} = (\mathrm{pe})$, $(n_0, \ldots, n_r) = (1, \ldots, 1)$ *and* $0 < d_i < l_i$ *for* $i = 1, \ldots, r$.

Proof. The Picard number is the rank of the class group, which is $\rho = n + m - (r + 1)$. In particular, $\rho = 1$ gives us $n + m = r + 2$. Since $n = n_0 + \cdots + n_r \geq r + 1$, we must have $m \leq 1$. Consider first the case $m = 0$. Then $n_i = 1$ for all except one i, hence a permutation achieves $(n_0, n_1, \ldots, n_r) = (2, 1, \ldots, 1)$. Next, if $m = 1$, we clearly have $n_i = 1$ for all i and after an inversion, we achieve also $\mathfrak{c} = (\mathrm{pe})$. In both cases, an addition achieves $0 \leq d_i < l_i$, see also Definition 3.2.24 (iii). However, since we assume that $(\mathfrak{c}, l, d)$ is irredundant, we must have $l_i > 1$ for $i = 1, \ldots, r$. Thus by $\gcd(l_i, d_i) = 1$, we have $d_i > 0$. □

Proposition 3.4.2. *Let* $((\mathrm{ee}), l, d)$ *be a defining triple of Picard number one as in Proposition 3.4.1 (i). Then the generator matrix* P *and the free part of the grading matrix* Q_0 *are*

$$P = \begin{bmatrix} -l_{01} & -l_{02} & l_1 & & 0 \\ \vdots & \vdots & & \ddots & \\ -l_{01} & -l_{02} & 0 & & l_r \\ d_{01} & d_{02} & d_1 & \ldots & d_r \end{bmatrix}, \qquad Q_0 = \begin{bmatrix} w_{01} & w_{02} & w_1 & \ldots & w_r \end{bmatrix},$$

where $w_{01}, w_{02}, w_i \geq 0$. *There are two elliptic fixed points* $x^{\pm}$ *and one hyperbolic fixed point* x_{01}. *Their multiplicities satisfy the following:*

(i) $\mu^+ = d_{01} l_1 \cdots l_r + l_{01} \sum_{i=1}^r d_i \prod_{j \neq i} l_j = \mu w_{02}$,

(ii) $\mu^- = -d_{02} l_1 \cdots l_r - l_{02} \sum_{i=1}^r d_i \prod_{j \neq i} l_j = \mu w_{01}$,

(iii) $\mu_{01} = l_{02} d_{01} - l_{01} d_{02}$,

(iv) $l_{01} \mu^- + l_{02} \mu^+ = l_1 \cdots l_r \mu_{01}$

In particular, the Picard index is $\iota = \mu w_{01} w_{02} \mu_{01}$. *Moreover, we have*

(v) $l_{01} w_{01} + l_{02} w_{02} = l_1 w_1 = \cdots = l_r w_r$,

(vi) $d_{01} w_{01} + d_{02} w_{02} + d_1 w_1 + \ldots d_r w_r = 0$,

(vii) $\mu w_i = \mu_{01} \prod_{j \neq i} l_j$ *for* $i = 1, \ldots, r$.

Proof. For parts (i)-(iii), see Remark 3.3.9 and Proposition 3.3.11. Part (iv) is an immediate consequence of (i)-(iii). The formula for the Picard index then follows with Theorem 3.3.13. Part (v) and (vi) follow from the fact that Q_0 annihilates the rows of P. For part (vii), apply Lemma 3.3.5 to $I := \{f_{01}, f_{02}, f_1, \ldots, f_r\} \backslash \{f_i\}$. □

We are ready to classify $\mathbb{C}^*$-surfaces of type (ee) with Picard number one by their Picard index, using the relations from Proposition 3.4.2. To do this, we must fix a specific subset of the entries $(l_{01}, l_{02}, l_1, \ldots, l_r)$. More precisely, we need to fix either l_{01} and l_{02}, or all of the l_i, in order to obtain bounds for the remaining entries. The following two algorithms describe the classification procedure for each case, respectively.

Algorithm 3.4.3 (See B.4.6)**.** Let $\iota, l_{01}, l_{02} \in \mathbb{Z}_{\geq 1}$ be given. We can classify all defining triples as in Proposition 3.4.1 (i) with Picard index ι and fixed (l_{01}, l_{02}) as follows. First, go through all decompositions $\iota = \mu w_{01} w_{02} \mu_{01}$ into four positive integers. By Proposition 3.4.2 (v), we have

$$l_{01}w_{01} + l_{02}w_{02} = l_1 w_1 = \cdots = l_r w_r.$$

Since the left-hand side is known, we can go through all its divisors to obtain all possibilities of l_i and w_i, for all $i = 1, \ldots, r$. At this point, we may check that $Q_0 = \begin{bmatrix} w_{01} & w_{02} & w_1 & \ldots & w_r \end{bmatrix}$ is almost free, i.e. that any $r+1$ of the weights are coprime. If this is not satisfied, we can discard this choice of $(l_1, \ldots, l_r)$ and continue with the next one. We then go through all $0 < d_i < l_i$ with $\gcd(d_i, l_i) = 1$. Rearranging Equation 3.4.2 (ii) yields

$$d_{01} = \frac{\mu w_{02} - l_{01} \sum_{i=1}^{r} d_i \prod_{j \neq i} l_j}{l_1 \cdots l_r}.$$

We check that the right-hand side is integral and that $\gcd(d_{01}, l_{01}) = 1$. Next, we use Equation 3.4.2 (iii) to obtain

$$d_{02} = \frac{l_{02}d_{01} - \mu_{01}}{l_{01}}.$$

Again, we check that the right-hand side is integral and $\gcd(d_{02}, l_{02}) = 1$. Lastly, we check that Equation 3.4.2 (vi) holds. If so, we add the resulting defining triple $((\text{ee}), l, d)$ to our list of results, where we avoid duplicate entries by using the normal form from Section 3.2.5.

Algorithm 3.4.4 (See B.4.7)**.** Let $\iota \in \mathbb{Z}_{\geq 1}$ and $l_1, \ldots, l_r \in \mathbb{Z}_{\geq 2}$ be given. We can classify all defining triples as in Proposition 3.4.2 (ii) with Picard index ι and fixed $(l_1, \ldots, l_r)$ as follows. First, go through all decompositions $\iota = \mu w_{01} w_{02} \mu_{01}$ into four positive integers. By Proposition 3.4.2 (iv), we have

$$l_{01}\mu^- \leq l_{01}\mu^- + l_{02}\mu^+ = l_1 \cdots l_r \mu_{01}.$$

Therefore, we obtain the upper bound $l_{01} \leq \frac{l_1 \cdots l_r \mu_{01}}{\mu w_{01}}$, hence we can go through all possible values of l_{01}. Using again Proposition 3.4.2 (iv), we have

$$l_{02} = \frac{l_1 \cdots l_r \mu_{01} - l_{01} \mu w_{01}}{\mu w_{02}}.$$

We check that the right-hand side is integral and if not, discard this choice of l_{01} and continue with the next one. Using Proposition 3.4.2 (v), we can set for all $i = 1, \ldots, r$:

$$w_i = \frac{l_{01}w_{01} + l_{02}w_{02}}{l_i},$$

where again, we check that the right-hand is integral. At this point, we may also check that $Q_0 = \begin{bmatrix} w_{01} & w_{02} & w_1 & \ldots & w_r \end{bmatrix}$ is almost free, i.e. that any $r+1$ of the weights are coprime.

Now that l_{01} and l_{02} have been fixed, we can proceed in the same way as in Algorithm 3.4.3. That is, we define

$$d_{01} = \frac{\mu w_{02} - l_{01} \sum_{i=1}^{r} d_i \prod_{j \neq i} l_j}{l_1 \cdots l_r}, \qquad d_{02} = \frac{l_{02} d_{01} - \mu_{01}}{l_{01}},$$

where we again check that they are integral and $\gcd(l_{01}, d_{01}) = \gcd(l_{02}, d_{02}) = 1$. Lastly, we check that Equation 3.4.2 (vi) holds. If so, we add the resulting defining triple $((\mathrm{ee}), l, d)$ to our list of results, where we avoid duplicate entries by using the normal form from Section 3.2.5.

Proposition 3.4.5. *Let $((\mathrm{pe}), l, d)$ be a defining triple of Picard number one as in Proposition 3.4.1 (ii). Then the generator matrix P and the free part of the grading matrix Q_0 are*

$$P = \begin{bmatrix} -l_0 & l_1 & & 0 & 0 \\ \vdots & & \ddots & & \\ -l_0 & 0 & & l_r & 0 \\ d_0 & d_1 & \dots & d_r & 1 \end{bmatrix}, \qquad Q_0 = \begin{bmatrix} w_0 & w_1 & \dots & w_r & w^+ \end{bmatrix},$$

where $w^+, w_i \geq 0$. There are parabolic fixed points $x_0^+, \dots, x_r^+$, no hyperbolic fixed points and one elliptic fixed point x^-. Their multiplicities are given as follows:

(i) $\mu_i^+ = l_i$,

(ii) $\mu^- = -d_0 l_1 \cdots l_r + -l_0 \sum_{i=1}^{r} d_i \prod_{j \neq i} l_j = \mu w^+$.

In particular, the Picard index is $\iota = w^+ l_0 \cdots l_r$. Moreover, we have

(iii) $l_0 w_0 = l_1 w_1 = \cdots = l_r w_r$,

(iv) $d_0 w_0 + d_1 w_1 + \dots d_r w_r + w^+ = 0$,

(v) $\mu w_i = \prod_{j \neq i}^{r} l_j$ *for* $i = 0, \dots, r$.

Proof. For parts (i) and (ii), see Remark 3.3.9 and Proposition 3.3.11. The formula for the Picard index then follows with Theorem 3.3.13. Parts (iii) and (iv) follow from the fact that Q_0 annihilates the rows of P. For part (v), apply Lemma 3.3.5 to $I := \{f^+, f_0, \dots, f_r\} \backslash \{f_i\}$. □

Algorithm 3.4.6 (See B.4.8)**.** Let $\iota \in \mathbb{Z}_{\geq 1}$ be given. We can classify all defining triples as in Proposition 3.4.1 (ii) as follows. First, go through all decompositions of $\iota = w^+ l_0 \dots l_r$ into $r+2$ positive integers, where $l_i \geq 2$. By Proposition 3.4.5 (v), the multiplicity μ divides $l_1 \cdots \not{l_i} \cdots l_r$ for all i. Hence we may go through all divisors μ of $\gcd(l_1 \cdots \not{l_i} \cdots l_r; i = 1, \dots, r)$ and set

$$w_i = \frac{l_0 \cdots l_r}{\mu l_i}.$$

At this point, we may check that $Q_0 = \begin{bmatrix} w_0 & \dots & w_r & w^+ \end{bmatrix}$ is almost free, i.e. any $r+1$ of the weights are coprime. Next, for $i = 1, \dots, r$, we go through all $0 < d_i < l_i$ such that $\gcd(l_i, d_i) = 1$. Solving Equation 3.4.5 for d_0, we obtain

$$d_0 = -\frac{\mu w^+ + l_0 \sum_{i=1}^{r} d_i \prod_{j \neq i} l_j}{l_1 \cdots l_r}.$$

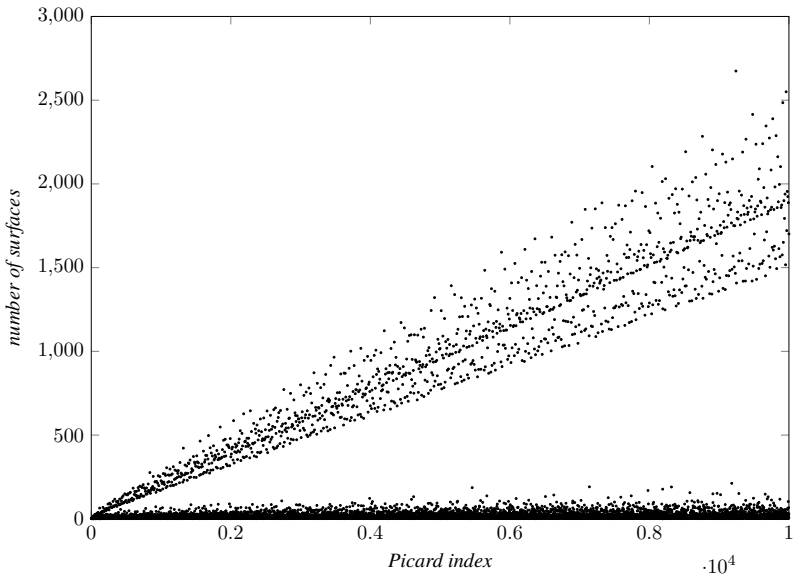
We check that the right-hand side is integral, that $\gcd(l_0, d_0) = 1$ holds and that Equation 3.4.5 (iv) is satisfied. If so, we add the resulting defining triple $((\mathrm{pe}), l, d)$ to our list of results, where we avoid duplicate entries by using the normal form from Section 3.2.5.

Table 3.4: Classification cases for log del Pezzo $\mathbb{C}^*$-surfaces of Picard number one.

Type	Possible forms of $l = (l_0, \dots, l_r)$
eAeA	$((1,1), \geq 2, \geq 2)$
eAeD	$((1, \geq 2), 2, 2)$, $((1,2), \geq 3, 2)$
eDeD	$((\geq 2, \geq 2), 2, 2)$, $((2,2), \geq 3, 2)$, $((1,1), \geq 2, 2, 2)$
eAeE	$((1,3),3,2)$, $((1,4),3,2)$, $((1,5),3,2)$, $((1,3),4,2)$, $((1,3),5,2)$, $((1,2),3,3)$, $((1,2),4,3)$, $((1,2),5,3)$
eDeE	$((2,3),3,2)$, $((2,3),4,2)$, $((2,3),5,2)$, $((2,4),3,2)$, $((2,5),3,2)$
eEeE	$((2,2),3,3)$, $((2,2),4,3)$, $((2,2),5,3)$, $((3,3),3,2)$, $((3,4),3,2)$, $((3,5),3,2)$, $((4,4),3,2)$, $((4,5),3,2)$, $((5,5),3,2)$, $((3,3),4,2)$, $((3,3),5,2)$, $((1,1),3,3,2)$, $((1,1),4,3,2)$, $((1,1),5,3,2)$
eDp	$(\geq 2, 2, 2)$
eEp	$(5,3,2)$, $(4,3,2)$, $(3,3,2)$

Algorithms 3.4.3, 3.4.4 and 3.4.6 allow us to classify all non-toric $\mathbb{C}^*$-surfaces of Picard number one by Picard index, provided we fix the isotropy group orders of at least some of their fixed points. In particular, we can use this to classify those with at most log terminal singularities. For a $\mathbb{C}^*$-surface associated to a defining triple $(\mathfrak{c}, l, d)$, [27, Corollary 8.12] tells us that the elliptic fixed points x^+ and x^- are log terminal if and only if $(l_{01}, \dots, l_{r1})$ and $(l_{0n_0}, \dots, l_{rn_r})$ are platonic tuples respectively. Considering a defining triple $((\mathrm{ee}), l, d)$ as in Proposition 3.4.1, irredundancy means $l_i \geq 2$ for $i = 1, \dots, r$. If $x^\pm$ are both log terminal, this implies $r \leq 3$. Hence we obtain a finite list of possible shapes of $l = (l_0, \dots, l_r)$, which we provide in Table 3.4. We sort them according to the singularity type of the elliptic fixed points $x^\pm$. Note that the same case distinction was used in [26, Proposition 5.9], where a classification by Gorenstein index was done. For the purposes of classifying by Picard index, we note that in each case, either both l_{01} and l_{02}, or all of the l_i are fixed. Hence Algorithms 3.4.3, 3.4.4 and 3.4.6 suffice to perform the classification, see also B.4.10 for an implementation. We obtained the following result:

Classification 3.4.7. *There are* 1 347 433 *families of non-toric, log del Pezzo* $\mathbb{C}^*$*-surfaces of Picard number one and Picard index at most* 10 000*. The numbers of families for given Picard index develop as follows:*



Remark 3.4.8. Recall that the toric log del Pezzo surfaces of Picard number one correspond to LDP triangles, which we classified up to Picard index 1 000 000 in Classification 2.4.2. Comparing the numbers of surfaces per Picard index in the different cases of our classification, we observe that the non-toric $\mathbb{C}^*$-surfaces quickly outnumber the toric ones. The fastest growing case seems to be eDeD, more specifically when $((l_{01}, l_{02}), l_1, l_2)$ is of the form $((\geq 2, \geq 2), 2, 2)$. This case is is responsible for the cone shape in the upper part of the plot in Classification 3.4.7. The following table gives an impression of the proportion of the different cases:

ι_{Pic}	toric	eAeA	eAeD	eAeE	eDeD	eDeE	eEeE	eDp	eEp
≤ 10	14	5	4	10	1	0	0	1	0
≤ 100	243	260	129	39	117	4	15	28	5
$\leq 1\,000$	4 205	7 425	2 209	206	11 622	32	103	521	51
$\leq 10\,000$	68 053	157 482	31 561	1 011	1 148 587	197	569	7 520	506

In Classification 3.4.7, we notice that there are no examples of $\mathbb{C}^*$-surfaces with Picard index equal to 31, 127, or 8 191. These are exactly the Mersenne prime numbers smaller than 10 000, excluding 3 and 7. Moreover, we find that except for a finite list of exceptions with small Picard index, all $\mathbb{C}^*$-surfaces whose Picard index is a prime number are quasismooth, i.e. are of type eAeA in Table 3.4. In the remainder of this section, we prove that these two observations hold generally. Compare this also to Proposition 2.4.3, where we treated the toric case.

Lemma 3.4.9. *Let* $((\text{ee}), l, d)$ *be a defining triple of Picard number one as in Proposition 3.4.1 and let* $\iota \in \mathbb{Z}_{\geq 1}$ *be the Picard index. Then the following holds*

(i) $\gcd(l_{01}, l_{02}) \mid \iota$,

(ii) $\gcd\left(\prod_{j\neq i} l_j \;;\; i=1,\dots,r\right) \mid \iota$,

(iii) *If* $\gcd(l_{01},l_i)=1$ *for some* $i=1,\dots,r$, *then* $\gcd(l_{02},l_i) \mid \iota$,

(iv) *If* $\gcd(l_{02},l_i)=1$ *for some* $i=1,\dots,r$, *then* $\gcd(l_{01},l_i) \mid \iota$.

Proof. Recall from Proposition 3.4.2 that the Picard index is $\iota = \mu w_{01} w_{02} \mu_{01}$, where μ is the multiplicity, $Q = \begin{bmatrix} w_{01} & w_{02} & w_1 & \dots & w_r \end{bmatrix}$ is the free part of the grading matrix and $\mu_{01} = l_{02}d_{01} - l_{01}d_{02}$ is the multiplicity of the hyperbolic fixed point. In particular, $\gcd(l_{01},l_{02})$ divides μ_{01}, which divides ι, hence (i) holds. For part (ii), recall that the multiplicity μ equals the greatest common divisor of all maximal minors of P, which are $\mu^{\pm}$ and μw_i for $i=1,\dots,r$. Using the formulas from Proposition 3.4.2 (i), (ii) and (vii), we see that $\gcd\left(\prod_{j\neq i} l_j \;;\; i=1,\dots,r\right)$ is a divisor of all of them, hence it divides μ and we get (ii). For part (iii), let $\alpha := \gcd(l_{02},l_i)$ and write $l_i = \alpha x$ and $l_{02} = \alpha y$. With Proposition 3.4.2 (v), we get

$$l_{01}w_{01} + \alpha y w_{02} = l_{01}w_{01} + l_{02}w_{02} = l_i w_i = \alpha x w_i.$$

Hence α must divide $l_{01}w_{01}$. Since we also assume $\gcd(l_{01},l_i)=1$, we get $\alpha \mid w_{01} \mid \iota$. Part (iv) is proved analogously. □

Proposition 3.4.10. *Let X be a non-toric log del Pezzo $\mathbb{C}^*$-surface of Picard number one with Picard index a prime number greater than 29. Then X is quasismooth, i.e. its defining triple satisfies $l_{01} = l_{02} = 1$.*

Proof. Let X arise from a defining triple $(\mathfrak{c}, l, d)$ as in Proposition 3.4.1. Log terminality implies that l must be among the cases listed in Table 3.4. If $\mathfrak{c} = (\mathrm{pe})$, Proposition 3.4.5 implies $l_i \mid \iota$ for all $i=1,\dots,r$. In particular, we can rule out the cases eDp and eEp. Furthermore, with Lemma 3.4.9, we can rule out most other cases by showing that either 2, 3, or 5 divides the Picard index. The only remaining cases where this does not work are the following:

$$\begin{aligned} &\text{eAeA}\colon && ((1,1),\geq 2,\geq 2)\\ &\text{eAeE}\colon && ((1,5),3,2),\ ((1,3),5,2),\ ((1,2),5,3)\\ &\text{eEeE}\colon && ((1,1),5,3,2). \end{aligned}$$

By Proposition 3.4.2, the Picard index is $\iota = \mu w_{01} w_{02} \mu_{01}$. Since it is prime, one factor must equal ι while the others are one. Assume first that $w_{01} = w_{02} = 1$. Then by 3.4.2 (v), we get $l_{01} + l_{02} = l_i w_i$. Out of the possible shapes for l listed above, this is only possible for eAeA, or if $l = ((1,5),3,2)$, where in the latter case we get $(w_1,w_2) = (2,3)$. But then by 3.4.2 (vii), we get $\mu = \mu_{01}$, a contradiction.

Now assume $\{w_{01},w_{02}\} = \{1,\iota\}$. Then we must have $\mu = \mu_{01} = 1$. In particular, by 3.4.2 (vii), we have $w_i = \prod_{j\neq i} l_j$ for $i=1,\dots,r$. Let us go through each case listed above individually.

If $l = ((1,5),3,2)$, we obtain $(w_1,w_2) = (2,3)$. Hence $w_{01} + 5w_{02} = 6$, which implies $w_{01} = w_{02} = 1$, a contradiction.

If $l = ((1,3),5,2)$, we get $(w_1, w_2) = (2,5)$. Hence $w_{01} + 3w_{02} = 10$. Since either w_{01} or w_{02} must be one and the other a prime, the possible options are $(w_{01}, w_{02}) = (1,3)$ and $(w_{01}, w_{02}) = (7,1)$.

If $l = ((1,2),5,3)$, we get $(w_1, w_2) = (3,5)$. Hence $w_{01} + 2w_{02} = 15$. Since either w_{01} or w_{02} must be one and the other a prime, the possible options are $(w_{01}, w_{02}) = (1,7)$ and $(w_{01}, w_{02}) = (13,1)$.

If $l = ((1,1),5,3,2)$, we get $(w_1, w_2, w_3) = (6,10,15)$. Hence $w_{01} + w_{02} = 30$. Since either w_{01} or w_{02} must be one and the other a prime, the possible options are $(w_{01}, w_{02}) = (1,29)$ or $(w_{01}, w_{02}) = (29,1)$.

In each non-quasismooth case, we have seen that if the Picard index is prime, it can be at most 29. In particular, if the Picard index is a prime greater than 29, the surface must be quasismooth. □

Using our classification, we can explicitly enumerate the finitely many non-quasismooth surfaces whose Picard index is a prime:

Corollary 3.4.11. *Let X be a non-toric log del Pezzo $\mathbb{C}^*$-surface of Picard number one with Picard index a prime. Then either X is quasismooth, or it is isomorphic to a $\mathbb{C}^*$-surface with generator matrix among the following list:*

ι_{Pic}	*Generator matrices*
2	$\begin{bmatrix}-1&-4&3&0\\-1&-4&0&2\\-1&-5&2&1\end{bmatrix}$ $\begin{bmatrix}-1&-3&4&0\\-1&-3&0&2\\-1&-4&3&1\end{bmatrix}$
3	$\begin{bmatrix}-1&-3&3&0\\-1&-3&0&2\\-1&-4&2&1\end{bmatrix}$ $\begin{bmatrix}-1&-3&5&0\\-1&-3&0&2\\-1&-4&4&1\end{bmatrix}$ $\begin{bmatrix}-1&-2&3&0\\-1&-2&0&3\\-1&-3&2&2\end{bmatrix}$
7	$\begin{bmatrix}-1&-3&5&0\\-1&-3&0&2\\-1&-4&3&1\end{bmatrix}$ $\begin{bmatrix}-1&-2&5&0\\-1&-2&0&3\\-1&-3&4&2\end{bmatrix}$
13	$\begin{bmatrix}-1&-2&5&0\\-1&-2&0&3\\-1&-3&2&2\end{bmatrix}$
29	$\begin{bmatrix}-1&-1&5&0&0\\-1&-1&0&3&0\\-1&-1&0&0&2\\-1&-2&4&2&1\end{bmatrix}$

Theorem 3.4.12. *There are no non-toric log del Pezzo $\mathbb{C}^*$-surfaces of Picard number one whose Picard index is a Mersenne prime number greater than seven.*

Proof. Assume X is such a surface. The next Mersenne prime after 7 is 31, hence Proposition 3.4.10 implies that X must be quasismooth. That is, X arises from a defining triple $((\mathrm{ee}), l, d)$ with $l_0 = (l_{01}, l_{02}) = (1,1)$. By Proposition 3.4.2, its Picard index is $\iota = \mu w_{01} w_{02} \mu_{01}$. Since we assume it to be a Mersenne prime, we either have $w_{01} = w_{02} = 1$, or $\{w_{01}, w_{02}\} = \{1, 2^n - 1\}$

for some $n \geq 2$. In any case, we have $w_{01} + w_{02} = 2^n$ for some $n \geq 1$. By Proposition 3.4.2 (v), we obtain $2^n = l_1 w_1 = l_2 w_2$. Since $l_1, l_2 \geq 2$, Lemma 3.4.9 (ii) implies $2 \mid \gcd(l_1, l_2) \mid \iota$, a contradiction. □

APPENDIX A

RationalPolygons.jl

This appendix serves as a reference for `RationalPolygons.jl`, a Julia package for computations with rational convex polygons. `RationalPolygons.jl` does not make use of any external computer algebra system, but implements all necessary algorithms, including two-dimensional euclidean geometry, from scratch in pure Julia. This allows for quite good performance, with computations involving billions of polygons being feasible on a personal computer.

Let us give an impression of `RationalPolygons.jl` by means of an example session. After loading in the package, we can create a polygon by calling the `convex_hull` function on a set of rational points:

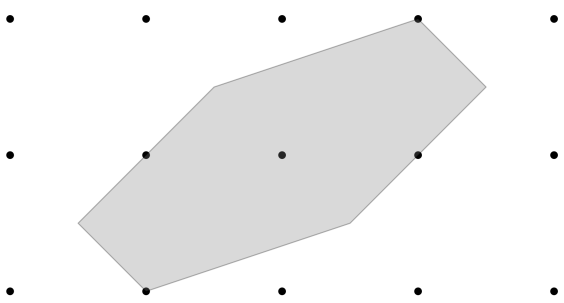
```
julia> using RationalPolygons

julia> P = convex_hull(RationalPoint{Int}[(-3//2,-1//2), (-1,-1), (1//2,-1//2),
↪  (3//2,1//2), (1,1), (-1//2,1//2)])
Rational polygon of rationality 2 with 6 vertices.
```

Using Julia's plotting library, we can visualize the polygon:

```
julia> using Plots

julia> plot(P);
```



Next, we compute some basic properties:

```
julia> number_of_interior_lattice_points(P)
1

julia> number_of_boundary_lattice_points(P)
4

julia> euclidean_area(P)
3//1

julia> ehrhart_quasipolynomial(P)
2×3 Matrix{Int64}:
 24  16  0
 24  16  8

julia> affine_automorphism_group(P)
D2

julia> gorenstein_index(P)
4
```

Lastly, we verify that the polygon is equivalent to its own dual.

```
julia> dual(P)
Rational polygon of rationality 2 with 6 vertices.

julia> are_affine_equivalent(P, dual(P))
true
```

The rest of this appendix is organized as follows: In Section A.1, we go over some basic functions for two-dimensional euclidean geometry over the rationals, such as computing the convex hull and intersecting lines. Section A.2 covers the type of rational polygons as well as basic properties and normal forms. In Section A.3, we discuss computation of subpolygons, following the approach from Section 1.2.2. Finally, Section A.4 covers implementations of the classification algorithms from Chapter 1, Sections 2.4 and 2.5, as well as various classification algorithms by other authors.

This documentation is generated directly from the docstrings of the package's source code. An online version is available on its GitHub page [55]. Every documented item includes a clickable `source` link that directs to the corresponding line in the source code where it is defined. The content here refers to version `v1.2.0-thesis` of `RationalPolygons.jl`. All example sessions have been tested and verified to work with this version.

A.1 2D Geometry

We provide basic functionality for two-dimensional geometry over the rational numbers. This includes distances and angles, Graham scan for computing the convex hull, intersection of lines, and affine halfplanes.

A.1.1 Points

In `RationalPolygons.jl`, we represent points as static vectors of length two. As the name suggests, these are *statically sized*, which leads to improved performance and memory management for many common operations. We provide three type aliases `LatticePoint`, `RationalPoint`, and `Point`, the latter being the union of the previous two. Note that everything is defined for an arbitrary subtype `T <: Integer`. This allows using both machine integers, like `Int64`, and unbounded integer types, like `BigInt`.

Julia Type A.1.1. source

```
LatticePoint{T <: Integer}
```

A lattice point in two-dimensional space. This is an alias for `SVector{2, T}`.

Julia Type A.1.2. source

```
RationalPoint{T<:Integer}
```

A rational point in two-dimensional space. This is an alias for `SVector{2, Rational{T}}`.

Julia Type A.1.3. source

```
Point{T<:Integer}
```

The union of `LatticePoint` and `RationalPoint`.

Julia Method A.1.4. source

```
is_k_rational(k :: T, p :: Point{T}) where {T <: Integer}
```

Checks whether a point p is k-rational, i.e. its coordinates have denominator at most k.

Julia Function A.1.5. source

```
is_integral(p :: Point{T}) where {T <: Integer}
```

Checks whether a point p is integral.

Julia Method A.1.6. source

```
denominator(p :: Point)
```

The smallest integer r such that $r \cdot p$ is integral.

Example:

```
julia> denominator(RationalPoint(1//2,1//3))
6
```

Julia Method A.1.7. source

```
is_primitive(p :: Point)
```

Checks whether a point is primitive, i.e. is integral and its coordinates are coprime.

Julia Method A.1.8. source

```
multiplicity(p :: Point)
```

The unique rational number x such that $x \cdot p$ is primitive and integral.

Example:

```
julia> multiplicity(RationalPoint(4//3,2//3))
3//2
```

Julia Method A.1.9. source

```
primitivize(p :: Point)
```

Return the unique primitive lattice point on the ray spanned by p.

Example:

```
julia> primitivize(RationalPoint(4//3,2//3))
2-element SVector{2, Int64} with indices SOneTo(2):
 2
 1
```

Julia Function A.1.10. source

```
norm(p :: Point{T})
```

Return the square of the euclidean norm of p.

Example:

```
julia> norm(RationalPoint(4//3,2//3))
20//9
```

Julia Function A.1.11. source

```
distance(p :: Point{T}, q :: Point{T})
```

Return the square of the euclidean distance between p and q.

Julia Method A.1.12. source

```
pseudo_angle(p :: Point{T}) where {T <: Integer}
```

Returns a value in the half-open interval $(-2, 2]$. A pseudo-angle allows comparing vectors by angle, while being faster to compute than the euclidean angle.

Example:

```
julia> pseudo_angle(LatticePoint(1,1))
1//2
```

A.1.2 Graham scan

The Graham scan is a planar convex hull algorithm developed by Ronald Graham [23]. With an asymptotic running time of $O(n \cdot \log(n))$, it is a lot quicker than algorithms that work in arbitrary dimension.

Julia Function A.1.13. source

```
graham_scan!(points :: Vector{<:Point{T}}) where {T <: Integer}
```

Perform a Graham scan on the given points, removing all points that are not vertices of their convex hull and ordering them counterclockwise.

Example:

```
julia> points = LatticePoint{Int}[(0,0), (1,0), (1,1), (0,1), (-1,1), (0,-1), (-1,-1),
↪  (0,-1)];

julia> graham_scan!(points)
5-element Vector{SVector{2, Int64}}:
 [-1, -1]
 [0, -1]
 [1, 0]
 [1, 1]
 [-1, 1]
```

Julia Function A.1.14. source

```
graham_scan(points :: Vector{<:Point{T}}) where {T <: Integer}
```

A non-modifying version of `graham_scan!`.

A.1.3 Lines

We provide basic functionality for lines in the two-dimensional rational plane. A line is encoded as a simple struct consisting of a base point and a direction vector. Two lines can be intersected to produce a value of type `IntersectionBehaviour`. This is an abstract type with three concrete subtypes `NoIntersection`, `LinesAreEqual` and `IntersectInPoint`, capturing the three intersection scenarios.

Julia Type A.1.15. source

```
Line{T <: Integer}
```

A line in two-dimensional rational space. It has two fields: `base_point :: RationalPoint{T}` and `direction_vector :: RationalPoint{T}`.

Julia Method A.1.16. source

```
base_point(L :: Line{T}) where {T <: Integer}
```

Return a point on the line L.

Julia Method A.1.17. source

```
direction_vector(L :: Line{T}) where {T <: Integer}
```

Return the direction vector of L.

Julia Function A.1.18. source

```
line_through_points(A :: Point{T}, B :: Point{T}) where {T <: Integer}
```

Return the line going through the points A and B.

Example:

```
julia> line_through_points(Point(1,0),Point(2,2))
Line with base point Rational{Int64}[1, 0] and direction vector Rational{Int64}[1, 2]
```

Julia Function A.1.19. source

```
horizontal_line(y :: Union{T, Rational{T}}) where {T <: Integer}
```

Return the horizontal line at y.

Julia Function A.1.20. source

```
vertical_line(x :: Union{T, Rational{T}}) where {T <: Integer}
```

Return the vertical line at x.

Julia Method A.1.21. source

```
Base.in(x :: Point{T}, L :: Line{T}) where {T <: Integer}
```

Check whether a point x lies on a line L.

Example:

```
julia> RationalPoint(3//2,1) ∈ line_through_points(Point(1,0),Point(2,2))
true
```

Julia Method A.1.22. source

```
normal_vector(L :: Line{T}) where {T <: Integer}
```

Return a primitive vector orthogonal to the direction vector of L.

Example:

```
julia> normal_vector(line_through_points(Point(1,0),Point(2,2)))
2-element SVector{2, Int64} with indices SOneTo(2):
 -2
  1
```

Julia Type A.1.23. source

```
IntersectionBehaviour{T <: Integer}
```

An abstract supertype for possible intersection behaviours of two lines. There are the three subtypes `IntersectInPoint`, `NoIntersection`, and `LinesAreEqual`.

Julia Type A.1.24. source

```
IntersectInPoint{T <: Integer}
```

The intersection behaviour of two lines intersecting in a unique point. This struct has a single field p, which is the intersection point.

Julia Type A.1.25. source

```
NoIntersection{T <: Integer}
```

The intersection behaviour of parallel lines.

Julia Type A.1.26. source

```
LinesAreEqual{T <: Integer}
```

The intersection behaviour of two equal lines.

Julia Function A.1.27. source

```
intersection_behaviour(L1 :: Line{T}, L2 :: Line{T}) where {T <: Integer}
```

Given two lines in two-dimensional rational space, return the intersection behaviour of the two lines: Possible values are `LinesAreEqual()`, `NoIntersection()` and `IntersectInPoint(p)` where p is the unique intersection point.

Julia Function A.1.28. source

```
intersection_point(L1 :: Line{T}, L2 :: Line{T}) where {T <: RationalUnion}
```

Return the intersection point of two lines in two-dimensional rational space. Throws an error if the lines do not intersect uniquely.

A.1.4 Affine halfplanes

An affine halfplane is encoded as a struct consisting of a normal vector $v \in \mathbb{Q}^2$ and a translation $b \in \mathbb{Q}^2$. It represents the set of points $\{x \in \mathbb{Q}^2 | \langle v, x \rangle \geq b\}$.

Julia Type A.1.29. source

```
AffineHalfplane{T <: Integer}
```

An affine halfplane in two-dimensional rational space. It has two fields `normal_vector :: RationalPoint{T}` and `translation :: Rational{T}`.

Julia Function A.1.30. source

```
affine_halfplane(nv :: Point{T}, b :: Union{T, Rational{T}}) where {T <: Integer}
```

Return the affine halfplane given by the equation `nv[1] * x[1] + nv[2] * x[2] ≥ b`.

Example:

```
julia> affine_halfplane(Point(-2,1),1)
Affine halfplane given by -2//1 x + 1//1 y ≥ 1//1
```

source

```
affine_halfplane(L :: Line{T}) where {T <: Integer}
```

Return the affine halfplane associated to a line. The halfplane is understood to consist of those points to the left of the line L, looking in the direction given by `direction_vector(L)`.

source

```
affine_halfplane(p :: Point{T}, q :: Point{T}) where {T <: Integer}
```

Return the affine halfplane associated to the line going through the points p and q.

source

```
affine_halfplane(P :: RationalPolygon, i :: Int)
```

Return the i-th describing halfplane of P.

Julia Method A.1.31. source

```
normal_vector(H :: AffineHalfplane{T}) where {T <: Integer}
```

Return the normal vector of H.

Julia Function A.1.32. source

```
translation(H :: AffineHalfplane{T}) where {T <: Integer}
```

Return the affine translation of H.

Julia Method A.1.33. source

```
Base.in(x :: Point{T}, H :: AffineHalfplane{T}) where {T <: Integer}
```

Check whether a point x lies in the halfplane H.

Example:

```
julia> Point(0,1) ∈ affine_halfplane(Point(-2,1),1)
true
```

Julia Method A.1.34. source

```
contains_in_interior(x :: Point{T}, H :: AffineHalfplane{T}) where {T <: Integer}
```

Check whether a point x lies in the interior of H.

Example:

```
julia> contains_in_interior(Point(0,1), affine_halfplane(Point(-2,1),1))
false
```

Julia Method A.1.35. source

```
Base.issubset(H1 :: AffineHalfplane{T}, H2 :: AffineHalfplane{T}) where {T <: Integer}
```

Check whether H_1 is a subset of H_2.

Julia Method A.1.36. source

```
line(H :: AffineHalfplane{T}) where {T <: Integer}
```

Return the line associated to H.

A.2 Polygons

In `RationalPolygons.jl`, we represent a polygon $P \subseteq \mathbb{R}^2$ by two pieces of data: An integral matrix $V \in \mathbb{Z}^{2\times N}$, called the *vertex matrix* and an integer $k \in \mathbb{Z}$, called the *rationality*. The associated polygon has as vertices the columns of V divided by k. To represent V, we use static

matrices, which are faster than Julia's internal matrices for many common operations. However, this implies that the type `RationalPolygon` depends on the number of vertices N as a type parameter, which affects Julia's dispatch mechanism at runtime. As long as the number of distinct values of N occurring during a computation remains relatively small, this should not cause performance issues.

There are two ways in which this encoding of rational polygons is not unique: First, scaling V and k by the same factor does not change the polygon, e.g. (V, k) describes the same polygon as $(2V, 2k)$. Even though they are mathematically the same polygon, `RationalPolygon.jl` views them as different objects: once as a k-rational polygon and once as a $2k$-rational polygon. The second way in which this encoding is not unique is that we can change the order of the columns. While we require them to be sorted counterclockwise, we may use any vertex as the first column.

Julia Type A.2.1. source

```
RationalPolygon{T<:Integer,N,M}
```

The type of rational polygons in two-dimensional space. `T` is the type of integers to be used. N is the number of vertices of the polygon and M equals $2 \cdot N$. It has the following fields:

- `rationality :: T`: The rationality of the polygon, e.g. 1 for lattice polygons, 2 for half-integral polygons etc.
- `vertex_matrix :: SMatrix{2,N,T,M}`: An integral $2 \times N$ matrix. The vertices of the polygon are understood to be the columns of this matrix divided by `rationality`.
- `number_of_vertices :: Int`: The number of vertices of the polygon. This is redundant information, since the number of vertices is already available as the type parameter N. However, getting the number of vertices of a polygon through the type parameter means lots of work for Julia's dispatch algorithm. Therefore, we found it to improve performance to put it as a variable into the struct as well.
- `is_unimodular_normal_form :: Bool`: A flag to remember that the polygon is already in unimodular normal form. This is used internally to avoid redundant computations of the normal form.
- `is_affine_normal_form :: Bool`: A flag variable to remember that the polygon is already in affine normal form. This is used internally to avoid redundant computations of the normal form.

The rest of this section is organized as follows. In Subsection A.2.1, we describe different methods of constructing a rational polygon. Subsection A.2.2 is about basic properties. In Subsection A.2.3, we discuss various ways of counting lattice points as well as Ehrhart theory. Subsection A.2.4 is about properties of LDP polygons that correspond to meaningful invariants of their associated toric del Pezzo surfaces. In Subsection A.2.5, we discuss unimodular and affine unimodular normal forms. Finally, Subsection A.2.6 is about lattice width and related concepts.

A.2.1 Constructors

To construct a rational polygon, one can either use a type constructor or one of the functions `convex_hull` and `intersect_halfplanes`. We describe the type constructors first.

Julia Method A.2.2. source

```
RationalPolygon(vertex_matrix :: SMatrix{2,N,T,M}, rationality :: T) where {N, M, T <:
↪  Integer}
RationalPolygon(scaled_vertices :: Vector{LatticePoint{T}}, rationality :: T)
RationalPolygon(vertices :: Vector{RationalPoint{T}})
RationalPolygon(vertices :: Vector{RationalPoint{T}}, rationality :: T)
```

Construct a rational polygon from a given set of vertices and possibly prescribed rationality. If `vertex_matrix` or `scaled_vertices` is provided, these are understood to be the integral vertices of the scaled polygon `rationality(P) * P`.

These constructors don't perform any consistency checks on the input. In particular, the user must be sure that the given points are truly vertices of the polygon and that they are ordered counter-clockwise. If this is not known ahead of construction, `convex_hull` should be used instead.

All constructors accept the optional keyword arguments `is_unimodular_normal_form` and `is_affine_normal_form`, which are set to `false` by default. If they are set to true, this means the user is certain that the given polygon is already in the respective normal form. This information will be used to prevent additional computations of the normal form and thus speed up equivalence checking.

Example:

The standard lattice triangle.

```
julia> P = RationalPolygon(LatticePoint{Int}[(0,0),(1,0),(0,1)], 1)
Rational polygon of rationality 1 with 3 vertices.
```

Example:

A half-integral polygon.

```
julia> P = RationalPolygon(RationalPoint{Int}[(1,0),(0,1//2),(-1,0),(0,-1//2)])
Rational polygon of rationality 2 with 4 vertices.
```

Julia Function A.2.3. source

```
convex_hull(points :: Vector{LatticePoint{T}}, k :: T = one(T)) where {T <: Integer}
```

Return the k-rational polygon given by the convex hull of p/k, where `p ∈ points`.

source

```
convex_hull(points :: Vector{RationalPoint{T}}) where {T <: Integer}
```

Return the convex hull of a given set of rational points. The rationality will be inferred from the input.

Example:

```
julia> convex_hull(RationalPoint{Int}[(1,0),(0,1//2),(0,-1//3)])
Rational polygon of rationality 6 with 3 vertices.
```

source

```
convex_hull(points :: Vector{RationalPoint{T}}, k :: T) where {T <: Integer}
```

Return the convex hull of a given set of rational points, viewed as a k-rational polygon.

Example:

```
julia> convex_hull(RationalPoint{Int}[(1,0),(0,1//2),(0,-1//3)], 12)
Rational polygon of rationality 12 with 3 vertices.
```

Julia Function A.2.4.

source

```
intersect_halfplanes(halfplanes :: Vector{AffineHalfplane{T}}; rationality ::
↪  Union{Missing,T} = missing) where {T <: Integer}
```

Return the polygon described by the intersection of the given affine halfplanes. Throws an error if the intersection is unbounded. If `rationality` is not provided, it will be inferred from the input.

Julia Function A.2.5.

source

```
empty_polygon(rationality :: T) where {T <: Integer}
```

Return the empty polygon of given rationality.

source

```
empty_polygon(::Type{T}) where {T <: Integer}
```

Return the empty polygon of integer type `T`. The rationality is understood to be one, i.e. it is a lattice polygon.

A.2.2 Basic Properties

We provide basic properties and checks for rational polygons. Note that indices corresponding to vertices are always considered cyclic, i.e. the $N+1$-th vertex of a polygon with N vertices equals its first vertex.

Julia Function A.2.6. source

```
number_of_vertices(P :: RationalPolygon)
```

Return the number of vertices of a polygon.

Julia Method A.2.7. source

```
rationality(P :: RationalPolygon)
```

Return the rationality of P. Note that this does not need to be the *denominator* of P, which is the smallest positive integer k such that kP is a lattice polygon: The standard lattice triangle may also be viewed as a half-integral polygon, in which case the rationality would be two, but the denominator is one.

Julia Method A.2.8. source

```
Base.denominator(P :: RationalPolygon{T,N}) where {N,T <: Integer}
```

Return the smallest positive integer k such that kP is a lattice polygon.

Julia Function A.2.9. source

```
vertex_matrix(P :: RationalPolygon)
```

Return the $2 \times N$ integral matrix containing the vertices of `rationality(P) * P` as its columns.

Example:

```
julia> P = convex_hull(RationalPoint{Int}[(1,0),(0,1//2),(0,-1//3)])
Rational polygon of rationality 6 with 3 vertices.

julia> vertex_matrix(P)
2×3 SMatrix{2, 3, Int64, 6} with indices SOneTo(2)×SOneTo(3):
  0  6  0
 -2  0  3
```

Julia Function A.2.10. source

```
scaled_vertex(P :: RationalPolygon, i :: Int)
```

Return the i-th integral vertex of the polygon `rationality(P) * P`. The index i is regarded as a cyclic index, e.g. the $N+1$-th vertex is equal to the first vertex.

Julia Function A.2.11. source

```
vertex(P :: RationalPolygon, i :: Int)
```

Return the i-th vertex of the polygon P. The index i is regarded as a cyclic index, e.g. the $N+1$-th vertex is equal to the first vertex.

Julia Function A.2.12. source

```
vertices(P :: RationalPolygon)
```

Return the vertices of P.

Julia Method A.2.13. source

```
affine_halfplane(P :: RationalPolygon, i :: Int)
```

Return the i-th describing halfplane of P.

Julia Method A.2.14. source

```
affine_halfplanes(P :: RationalPolygon)
```

Return the describing halfplanes of P.

Julia Method A.2.15. source

```
Base.in(x :: Point{T}, P :: RationalPolygon{T}) where {T <: Integer}
```

Check whether a point x is contained in the polygon P.

Example:

```
julia> P = convex_hull(LatticePoint{Int}[(1,0),(0,1),(-1,-1)])
Rational polygon of rationality 1 with 3 vertices.

julia> Point(0,0) ∈ P
true

julia> Point(1//2, 1//2) ∈ P
true
```

Julia Method A.2.16. source

```
contains_in_interior(x :: Point{T}, P :: RationalPolygon{T}) where {T <: Integer}
```

Check whether a point x is contained in the interior of P.

Example:

```
julia> P = convex_hull(LatticePoint{Int}[(1,0),(0,1),(-1,-1)])
Rational polygon of rationality 1 with 3 vertices.

julia> contains_in_interior(Point(0,0), P)
true

julia> contains_in_interior(Point(1//2,1//2), P)
false
```

Julia Function A.2.17. source

```
dim(P :: RationalPolygon)
```

Return the dimension of P. For empty polygons, this returns -1. Otherwise, it returns 0, 1 or 2.

Julia Function A.2.18. source

```
normalized_area(P :: RationalPolygon)
```

Return the normalized area of a k-rational polygon, i.e. $2k^2$ times its euclidean area. The result is always an integer, counting the number of standard k-rational triangles contained in P.

Julia Function A.2.19. source

```
euclidean_area(P :: RationalPolygon)
```

Return the euclidean area of a rational polygon.

Julia Function A.2.20. source

```
is_maximal(P :: RationalPolygon)
```

Check whether a rational polygon is maximal among all polygons sharing the same rationality and number of interior lattice points.

Example:

```
julia> P = convex_hull(LatticePoint{Int}[(0,0),(3,0),(0,3)])
Rational polygon of rationality 1 with 3 vertices.

julia> is_maximal(P)
true

julia> Q = convex_hull(LatticePoint{Int}[(0,0),(2,0),(2,1),(0,3)])
Rational polygon of rationality 1 with 4 vertices.

julia> is_maximal(Q)
false
```

Julia Function A.2.21. source

```
dual(P :: RationalPolygon{T}) where {T <: Integer}
```

Return the dual of a polygon P. Throws an error if P does not contain the origin in its interior.

Example:

```
julia> P = convex_hull(LatticePoint{Int}[(-1,-1),(1,0),(0,1)])
Rational polygon of rationality 1 with 3 vertices.

julia> vertex_matrix(dual(P))
2×3 SMatrix{2, 3, Int64, 6} with indices SOneTo(2)×SOneTo(3):
 -2   1  1
  1  -2  1
```

A.2.3 Ehrhart Theory

Recall that the number of lattice points in integral multiples of a k-rational polygon P is a quasipolynomial, called its *Ehrhart quasipolynomial*:

$$\mathrm{ehr}_P(t) = |tP \cap \mathbb{Z}^2| = At^2 + a(t)t + b(t), \qquad t \in \mathbb{Z}.$$

Here, A is the euclidean area of P and $a, b\colon \mathbb{Z} \to \mathbb{Q}$ are k-periodic functions. These can be computed by

$$a(t) = -(2t+k)\cdot A + \frac{\mathrm{ehr}_P(t+k) - \mathrm{ehr}_P(t)}{k},$$

$$b(t) = (t^2+tk)\cdot A + \frac{(t+k)\mathrm{ehr}_P(t) - t\mathrm{ehr}_P(t+k)}{k}.$$

Setting $\tilde{A} := 2k^2A$, $\tilde{a} := 2k^2a$ and $\tilde{b} := 2k^2b$, we get integer-valued functions $\tilde{a}$ and $\tilde{b}$, which we call the *normalized Ehrhart coefficients*. In `RationalPolygons.jl`, we encode the Ehrhart quasipolynomial by the $3 \times k$-integral matrix of its normalized Ehrhart coefficients:

$$\begin{bmatrix} \tilde{A} & \tilde{a}(1) & \tilde{b}(1) \\ \tilde{A} & \tilde{a}(2) & \tilde{b}(2) \\ \vdots & \vdots & \vdots \\ \tilde{A} & \tilde{a}(k) & \tilde{b}(k) \end{bmatrix}$$

If P is integral, we have $k = 1$ and its Ehrhart quasipolynomial is a regular polynomial of degree two. In general, the periods of a and b are divisors of k.

`RationalPolygons.jl` comes with many methods for counting the (interior, boundary) lattice points of a rational polygon as well as computing its Ehrhart quasipolynomial and its periods.

Counting lattice points

Julia Function A.2.22. source

```
generic_lattice_points(P :: RationalPolygon{T,N}, k :: T)
```

Return the lattice points in the k-fold of the rational polygon P. If the keyword argument `interior = true` is passed, only the lattice points in the relative interior are computed. If `only_count = true` is passed, only the total number of lattice points is returned.

Julia Function A.2.23. source

```
boundary_k_rational_points(P :: RationalPolygon{T,N}, k :: T) where {N,T <: Integer}
```

Return all k-rational points on the boundary of P.

Julia Function A.2.24. source

```
number_of_boundary_k_rational_points(P :: RationalPolygon{T}, k :: T) where {T <:
↪  Integer}
```

Return the number of k-rational points on the boundary of P.

Julia Function A.2.25. source

```
boundary_lattice_points(P :: RationalPolygon{T}) where {T <: Integer}
```

Return all lattice points on the boundary of P.

Julia Function A.2.26. source

```
number_of_boundary_lattice_points(P :: RationalPolygon)
```

Return the number of lattice points on the boundary of P.

Julia Function A.2.27. source

```
interior_k_rational_points(P :: RationalPolygon{T}, k :: T) where {T <: Integer}
```

Return all k-rational points in the interior of P.

Julia Function A.2.28. source

```
number_of_interior_k_rational_points(k :: T, P :: RationalPolygon{T}) where {T <:
↪  Integer}
```

Return the number of k-rational points in the interior of P.

Julia Function A.2.29. source

```
interior_lattice_points(P :: RationalPolygon{T}) where {T <: Integer}
```

Return all lattice points in the interior of P.

Julia Function A.2.30. source

```
number_of_interior_lattice_points(P :: RationalPolygon{T}) where {T <: Integer}
```

Return the number of lattice points in the interior of P.

Julia Function A.2.31. source

```
k_rational_points(k :: T, P :: RationalPolygon{T}) where {T <: Integer}
```

Return all k-rational points in P.

Julia Function A.2.32. source

```
number_of_k_rational_points(P :: RationalPolygon{T}, k :: T) where {T <: Integer}
```

Return the number of k-rational points in P.

Julia Function A.2.33. source

```
lattice_points(P :: RationalPolygon{T}) where {T <: Integer}
```

Return all lattice points in P.

Julia Function A.2.34. source

```
number_of_lattice_points(P :: RationalPolygon{T}) where {T <: Integer}
```

Return the number of lattice points in P.

Julia Function A.2.35. source

```
k_rational_hull(P :: RationalPolygon{T}, k :: T) where {T <: Integer}
```

Return the convex hull of all k-rational points contained in P. If `primitive = true` is passed, only the primitive k-rational points are taken.

Julia Function A.2.36. source

```
interior_k_rational_hull(P :: RationalPolygon{T}, k :: T) where {T <: Integer}
```

Return the convex hull of all interior k-rational points of P. If `primitive = true` is passed, only the primitive k-rational points are taken.

Julia Function A.2.37. source

```
integer_hull(P :: RationalPolygon{T}) where {T <: Integer}
```

Return the convex hull of all lattice points in the interior of P. If `primitive = true` is passed, only the primitive interior lattice points are taken.

Julia Function A.2.38. source

```
interior_integer_hull(P :: RationalPolygon{T}) where {T <: Integer}
```

Return the convex hull of all interior lattice points of P. If `primitive = true` is passed, only the primitive lattice points are taken.

Ehrhart quasipolynomial

Julia Function A.2.39. source

```
is_periodic(v :: Vector, k :: Int)
```

Check if a vector v is k-periodic, i.e. `v[i] == v[mod(i+k,1:n)]` for all `i`, where `n = length(v)`.

Julia Function A.2.40. source

```
period(v :: Vector)
```

Return the smallest positive integer k such that v is k-periodic.

Julia Function A.2.41. source

```
ehrhart_quasipolynomial_with_periods(P :: RationalPolygon{T}) where {T <: Integer}
```

Return a $k \times 3$-matrix of normalized coefficients of the Ehrhart quasipolynomial of a k-rational polygon P, together with a vector of its three periods.

Julia Function A.2.42. source

```
ehrhart_quasipolynomial(P :: RationalPolygon)
```

Return a $k \times 3$-matrix of normalized coefficients of the Ehrhart quasipolynomial of a k-rational polygon P.

Julia Function A.2.43. source

```
ehrhart_quasipolynomial_periods(P :: RationalPolygon)
```

Return the periods of the Ehrhart quasipolynomial of a k-rational polygon P.

Julia Function A.2.44. source

```
ehrhart_quasipolynomial_period(P :: RationalPolygon)
```

Return the period of the Ehrhart quasipolynomial of a k-rational polygon P.

Julia Function A.2.45. source

```
is_quasiintegral(P :: RationalPolygon)
```

Check whether a rational polygon is quasiintegral, i.e. has an Ehrhart polynomial.

A.2.4 LDP polygons and toric surfaces

An LDP polygon is a lattice polygon with primitive vertices containing the origin in its interior. LDP polygons correspond to toric log del Pezzo surfaces. In this package, we call more generally a k-rational polygon *LDP*, if it contains the origin in its interior and its k-fold multiple has primitive vertices. With this definition, the k-rational LDP polygons with exactly one interior lattice point correspond to the toric del Pezzo surfaces with at most $\frac{1}{k}$-log canonical singularities. Here, we list some properties of LDP polygons that correspond to meaningful invariants of the associated toric del Pezzo surface. For more background on LDP polygons, see Section 2.2.

Julia Function A.2.46. source

```
contains_origin_in_interior(P :: RationalPolygon)
```

Check whether P contains the origin in its interior.

Julia Method A.2.47. source

```
is_primitive(P :: RationalPolygon)
```

Check whether all vertices of the scaled lattice polygon `rationality(P) * P` are primitive.

Julia Function A.2.48. source

```
is_ldp(P :: RationalPolygon)
```

Check whether P is LDP, i.e. is primitive and contains the origin in its interior.

Julia Method A.2.49. source

```
multiplicity(P :: RationalPolygon{T}, i :: Int) where {T <: Integer}
```

The index of the sublattice spanned by the i-th and $i+1$-th scaled vertex of P (i.e. the determinant of those two vertices). For LDP polygons, this equals the order of the local class group associated with the toric fixed point associated to the i-th and $i+1$-th ray. See Definition 2.2.1 and Proposition 2.2.10.

Julia Method A.2.50. source

```
multiplicity(P :: RationalPolygon{T,N}) where {N,T <: Integer}
```

The order of the sublattice spanned by the scaled vertices of P. For LDP polygons, this equals the order of the torsion part of the divisor class group of the associated toric surface. See Definition 2.2.1 and Proposition 2.2.5.

Julia Function A.2.51. source

```
grading_matrix_free_part(P :: RationalPolygon)
```

Return the free part of the grading matrix associated to P.

Example:

```
julia> P = convex_hull(LatticePoint{Int}[(1,0), (2,5), (-4,-5), (-1,-5)])
Rational polygon of rationality 1 with 4 vertices.
```

```
julia> grading_matrix_free_part(P)
2×4 SMatrix{2, 4, Int64, 8} with indices SOneTo(2)×SOneTo(4):
 1  -1  3  0
 1   0  2  1
```

Julia Function A.2.52. source

```
grading_matrix_torsion_part(P :: RationalPolygon)
```

Return the torsion part of the grading matrix associated to P. If P has N vertices and has multiplicity μ, the result is a static vector of length N whose entries are all between 0 and $\mu - 1$.

Example:

```
julia> P = convex_hull(LatticePoint{Int}[(1,0), (2,5), (-4,-5), (-1,-5)])
Rational polygon of rationality 1 with 4 vertices.

julia> multiplicity(P)
5

julia> grading_matrix_torsion_part(P)
4-element SVector{4, Int64} with indices SOneTo(4):
 4
 0
 1
 0
```

Julia Function A.2.53. source

```
grading_matrix(P :: RationalPolygon{T,N}) where {N, T <: Integer}
```

Return a tuple (Q_0, Q_1) where Q_0 is the free part and Q_1 the torsion part of the grading matrix associated to P.

Julia Function A.2.54. source

```
is_smooth(P :: RationalPolygon, i :: Int)
```

Check whether the cone spanned by the i-th and $i+1$-th vertex of P is regular, i.e. generates the entire lattice. For LDP polygons, this means that the toric fixed point associated to the i-th and $i+1$-th ray is smooth.

source

```
is_smooth(P :: RationalPolygon)
```

Check whether all cones of the face fan of P are regular. For LDP polygons, this means that the associated toric surface is smooth.

Julia Function A.2.55. source

```
picard_index(P :: RationalPolygon{T,N}) where {N,T <: Integer}
```

The product of all local multiplicities of P divided by the global multiplicity. For LDP polygons, this equals the index of the Picard group inside the divisor class group of the associated toric surface, see Theorem 2.2.11

Julia Function A.2.56. source

```
gorenstein_index(P :: RationalPolygon{T}, i :: Int)
```

The multiplicity of P divided by `gcd(w[2] - v[2], v[1] - w[1])`, where v and w are the i-th and $i+1$-th scaled vertices of P respectively. For LDP polygons, this equals the local Gorenstein at the toric fixed point associated to the i-th and $i+1$-th ray of P, see Proposition 2.2.14

source

```
gorenstein_index(P :: RationalPolygon{T}) where {T <: Integer}
```

The least common multiple of the local Gorenstein indices of P. For LDP polygons, this equals the Gorenstein index of the associated toric surface.

Julia Function A.2.57. source

```
gorenstein_coefficients(P :: RationalPolygon{T,N})
```

Return the Gorenstein coefficients of an LDP polygon with N vertices. This is an integral matrix $A = (a_{ij}) \in \mathbb{Z}^{N\times(N-2)}$ such that $\iota w = \sum_{j=1}^{N-2} a_{ij} w_{j+i-1}$, where ι is the Gorenstein index, w_i are the columns of the free part of the grading matrix, and $w = w_1 + \cdots + w_N$ is the class of the anticanonical divisor. See Construction 2.5.2.

Example:

```
julia> P = convex_hull(LatticePoint{Int}[(1,0), (2,5), (-4,-5), (-1,-5)])
Rational polygon of rationality 1 with 4 vertices.

julia> gorenstein_coefficients(P)
4×2 SMatrix{4, 2, Int64, 8} with indices SOneTo(4)×SOneTo(2):
 20   5
 15  10
  5  10
  5  15
```

Julia Function A.2.58. source

```
gorenstein_matrix(P :: RationalPolygon{T,N}) where {T <: Integer, N}
```

Return the Gorenstein matrix of an LDP polygon. See Definition 2.5.7.

Example:

```
julia> P = convex_hull(LatticePoint{Int}[(1,0), (2,5), (-4,-5), (-1,-5)])
Rational polygon of rationality 1 with 4 vertices.

julia> gorenstein_matrix(P)
4×4 SMatrix{4, 4, Int64, 16} with indices SOneTo(4)×SOneTo(4):
 -15    0   5   5
   5  -10  -5   5
   5    5   0  -5
 -10    5   5   0
```

Julia Function A.2.59. source

```
discrepancies(v1 :: LatticePoint{T}, v2 :: LatticePoint{T}) where {T <: Integer}
```

The discrepancies of the two-dimensional cone spanned by two given integral primitive vectors. See Definition 2.2.18 and Proposition 2.2.19.

Julia Function A.2.60. source

```
log_canonicity(P :: RationalPolygon)
```

Given a k-rational polygon P, return the maximal rational number $0 < \varepsilon \leq 1$ such that $\varepsilon * P$ contains only one k-rational point in its interior (the origin). For an LDP polygon, this equals the maximal rational number $0 < \varepsilon \leq 1$ such that the associated toric surface is ε-log canonical. See Definition 2.2.16.

Julia Function A.2.61. source

```
toric_prime_divisor_self_intersection(P :: RationalPolygon, i :: Int)
```

Writing u, v and w for the $i-1$-th, i-th and $i+1$-th scaled vertex of P respectively, return $\frac{det(w,u)}{det(u,v)det(v,w)}$. For LDP polygons, this equals the self intersection number of the i-th toric prime divisor. See Definition 2.2.22.

Julia Function A.2.62. source

```
toric_prime_divisor_adjacent_intersection(P :: RationalPolygon, i :: Int)
```

Writing v and w for the i-th and $i+1$-th scaled vertex of P, return $\frac{1}{\det(v,w)}$. For LDP polygons, this equals the intersection number between the i-th and $i+1$-th toric prime divisors. See Definition 2.2.22.

Julia Method A.2.63. source

```
degree(P :: RationalPolygon)
```

For LDP polygons, return the self intersection number of an anticanonical divisor of the associated toric surface. See Definition 2.2.22.

Example:

The projective plane has degree 9.

```
julia> P = convex_hull(LatticePoint{Int}[(1,0),(0,1),(-1,-1)])
Rational polygon of rationality 1 with 3 vertices.

julia> degree(P)
9//1
```

A.2.5 Normal forms and automorphism groups

Two k-rational polygons are called *(affine) unimodular* equivalent if they can be transformed into each other by an (affine) unimodular transformation. The purpose of a normal form is to provide a unique representative for every equivalence class, i.e. two polygons should be (affine) unimodular equivalent to each other if and only if their (affine) unimodular normal forms coincide.

For details about the normal form used in `RationalPolygons.jl`, see Section 1.2.1.

Julia Function A.2.64. source

```
unimodular_normal_form(P :: RationalPolygon{T,N}) where {N,T <: Integer}
```

Return a unimodular normal form of a rational polygon. Two rational polygons have the same unimodular normal form if and only if the can be transformed into each other by applying a unimodular transformation.

Julia Function A.2.65. source

```
are_unimodular_equivalent(P :: RationalPolygon, Q :: RationalPolygon)
```

Check whether two rational polygons are equivalent by a unimodular transformation.

Julia Function A.2.66. source

```
affine_normal_form(P :: RationalPolygon{T,N}) where {N,T <: Integer}
```

Return an affine normal form of a rational polygon. Two rational polygons have the same affine normal form if and only if the can be transformed into each other by applying an affine unimodular transformation.

Julia Function A.2.67. source

```
are_affine_equivalent(P :: RationalPolygon, Q :: RationalPolygon)
```

Check whether two rational polygons are equivalent by an affine unimodular transformation.

Julia Type A.2.68. source

```
PolygonAutomorphismGroup
```

A struct holding information about the automorphism group of a rational polygon. It has two fields `is_cyclic :: Bool` and `n :: Int`.

Julia Function A.2.69. source

```
CyclicGroup(n :: Int)
```

Return the cyclic group of order n, as a `PolygonAutomorphismGroup`.

Julia Function A.2.70. source

```
DihedralGroup(n :: Int)
```

Return the dihedral group of order $2n$, as a `PolygonAutomorphismGroup`.

Julia Function A.2.71. source

```
is_cyclic(G :: PolygonAutomorphismGroup)
```

Check whether G is a cyclic group.

Julia Function A.2.72. source

```
order(G :: PolygonAutomorphismGroup)
```

Return the order of the group G.

Julia Function A.2.73. source

```
unimodular_automorphism_group(P :: RationalPolygon)
```

Return the automorphism group of P with respect to unimodular transformations.

Example:

```
julia> P = convex_hull(LatticePoint{Int}[(1,0),(0,1),(-1,1),(-1,0),(0,-1),(1,-1)])
Rational polygon of rationality 1 with 6 vertices.

julia> unimodular_automorphism_group(P)
D6
```

Julia Function A.2.74. source

```
affine_automorphism_group(P :: RationalPolygon{T,N}) where {N,T <: Integer}
```

Return the automorphism group of P with respect to affine unimodular transformations.

A.2.6 Lattice width

We provide functions to compute the lattice width as well as all direction vectors in which the lattice width is attained. Furthermore, we implement the concept of *lattice width data* following Bohnert [14], which captures information about the slicing lengths of a polygon with respect to given direction vectors.

Julia Function A.2.75. source

```
width(P :: RationalPolygon{T}, w :: Point{T}) where {T <: Integer}
```

Return the lattice width of P in direction w.

Example:

```
julia> P = convex_hull(LatticePoint{Int}[(1,1),(1,-2),(-4,2),(-2,2)],2)
Rational polygon of rationality 2 with 4 vertices.

julia> width(P, Point(0,1))
2//1

julia> width(P, Point(1,0))
5//2
```

source

```
width(P :: RationalPolygon)
```

Return the lattice width of P.

Example:

```
julia> P = convex_hull(LatticePoint{Int}[(1,1),(1,-2),(-4,2),(-2,2)],2)
Rational polygon of rationality 2 with 4 vertices.

julia> width(P)
2//1
```

Julia Function A.2.76.

source

```
all_direction_vectors_with_width_less_than(P :: RationalPolygon{T}, c :: Rational{T})
↪  where {T <: Integer}
```

Return all direction vectors in which the width of P is less than or equal to a given constant.

Example:

```
julia> P = convex_hull(LatticePoint{Int}[(1,1),(1,-2),(-4,2),(-2,2)],2)
Rational polygon of rationality 2 with 4 vertices.

julia> all_direction_vectors_with_width_less_than(P, 3//1)
4-element Vector{SVector{2, Int64}}:
 [1, 0]
 [0, 1]
 [1, 1]
 [1, 2]
```

Julia Function A.2.77.

source

```
width_direction_vectors(P :: RationalPolygon)
```

Return the lattice width direction vectors of P, i.e. those directions that realize the lattice width of P.

Example:

```
julia> P = convex_hull(LatticePoint{Int}[(1,1),(1,-2),(-4,2),(-2,2)],2)
Rational polygon of rationality 2 with 4 vertices.

julia> width_direction_vectors(P)
2-element Vector{SVector{2, Int64}}:
 [0, 1]
 [1, 1]
```

Julia Function A.2.78. source

```
adjust_to_width_direction(P :: RationalPolygon{T}, w :: Point{T}) where {T <: Integer}
```

Apply an affine unimodular transformation to P that transforms the given width direction vector to $(1, 0)$, see Lemma 2.10 of [14].

Julia Function A.2.79. source

```
number_of_interior_integral_lines(P :: RationalPolygon{T}, w :: Point{T}) where {T <:
↪  Integer}
```

Return the number of interior integral lines of P with respect to a given direction vector w.

Example:

```
julia> P = convex_hull(RationalPoint{Int}[(1//3,-1),(4//3,2),(2//3,2),(-4//3,-1)])
Rational polygon of rationality 3 with 4 vertices.

julia> number_of_interior_integral_lines(P, LatticePoint(1,0))
3

julia> number_of_interior_integral_lines(P, LatticePoint(0,1))
2
```

Julia Function A.2.80. source

```
minimal_number_of_interior_integral_lines(P :: RationalPolygon{T}) where {T <:
↪  Integer}
```

Return the minimal number of interior integral lines of P.

Example:

```
julia> P = convex_hull(RationalPoint{Int}[(1//3,-1),(4//3,2),(2//3,2),(-4//3,-1)])
Rational polygon of rationality 3 with 4 vertices.

julia> minimal_number_of_interior_integral_lines(P)
2
```

Julia Function A.2.81. source

```
is_realizable_in_interval(P :: RationalPolygon{T}, h :: T) where {T <: Integer}
```

Check whether P is realizable in $\mathbb{R} \times [0, h]$. This is true if and only if the minimal number of interior integral lines is less than or equal to $h - 1$.

Example:

```
julia> P = convex_hull(RationalPoint{Int}[(1//3,-1),(4//3,2),(2//3,2),(-4//3,-1)])
Rational polygon of rationality 3 with 4 vertices.

julia> is_realizable_in_interval(P,2)
false

julia> is_realizable_in_interval(P,3)
true
```

Julia Type A.2.82.

source

```
LatticeWidthData{T <: Integer}
```

A struct capturing information about the lattice width of a rational polygon with respect to a lattice width direction vector, see Definition 2.11 of [14]. It has two fields:

- `interval_of_nonzero_vertical_slice_length :: Tuple{Rational{T},Rational{T}}`,
- `interval_of_longest_vertical_slice_length :: Tuple{Rational{T},Rational{T}}`.

Julia Function A.2.83.

source

```
lattice_width_data(P :: RationalPolygon{T}, w :: Point{T}) where {T <: Integer}
```

Compute the lattice width data of a rational polygon with respect to a given width direction vector, see Definition 2.11 of [14]. This function returns a value of type `LatticeWidthData`.

Example:

```
julia> P = convex_hull(LatticePoint{Int}[(1,1),(1,-2),(-4,2),(-2,2)],2)
Rational polygon of rationality 2 with 4 vertices.

julia> ws = width_direction_vectors(P)
2-element Vector{SVector{2, Int64}}:
 [0, 1]
 [1, 1]

julia> lattice_width_data(P,ws[1])
LatticeWidthData{Int64}((0//1, 2//1), (3//2, 3//2))

julia> lattice_width_data(P,ws[2])
LatticeWidthData{Int64}((0//1, 2//1), (1//2, 1//2))
```

Julia Function A.2.84. source

```
number_of_interior_integral_vertical_lines(P :: RationalPolygon, w :: Point{T}) where
↪  {T <: Integer}
```

Return the number of interior integral vertical lines of P with respect to a given lattice width direction vector, see Definition 2.11 of [14].

Julia Function A.2.85. source

```
position_of_longest_vertical_slice_length(P :: RationalPolygon, w :: Point{T}) where
↪  {T <: Integer}
```

Return the position of the longest vertical slicing length of P with respect to a given lattice width direction vector, see Definition 2.11 of [14].

Julia Function A.2.86. source

```
lattice_width_datas(P :: RationalPolygon{T}) where {T <: Integer}
```

Return the lattice width data for all lattice width direction vectors of P.

Julia Function A.2.87. source

```
numbers_of_interior_integral_vertical_lines(P :: RationalPolygon{T}) where {T <:
↪  Integer}
```

Return the number of interior integral vertical lines for all width direction vectors of P.

Julia Function A.2.88. source

```
positions_of_longest_vertical_slice_length(P :: RationalPolygon{T}) where {T <:
↪  Integer}
```

Return the positions of the longest vertical slicing lengths for all width direction vectors of P.

A.2.7 IO

`RationalPolygons.jl` provides two ways to read and write polygons from files. The first is text-based. Polygons can be read from files containing one polygon per line like this:

```
[[2, 0], [1, 3], [-1, 0], [-3, -4]]
[[1, 0], [2, 6], [-4, -9]]
[[1, 0], [3, 5], [0, 1], [-5, -8]]
....
```

This text-based format has the advantage of being easy to understand and use. However, storing polygons as ASCII strings is not very space-efficient, as they contain many redundant control characters. Hence we provide another way to store polygons in binary and compressed form, which uses the HDF5 file format. This is more suitable for large datasets. For an example session, see `write_polygon_dataset`.

Julia Function A.2.89. source

```
parse_rational_polygons(k :: T, files :: AbstractVector{String}) where {T <: Integer}
parse_rational_polygons(k :: T, file :: String) where {T <: Integer}
```

Parse a list of files containing the vertices of a k-rational polygon. The files must have one polygon per line containing its vertices, as in the following example:

```
[[2, 0], [1, 3], [-1, 0], [-3, -4]]
[[1, 0], [2, 6], [-4, -9]]
[[1, 0], [3, 5], [0, 1], [-5, -8]]
....
```

Julia Function A.2.90. source

```
write_rational_polygons(Ps :: Vector{<:RationalPolygon{T}}, filepath :: String) where
↪  {T <: Integer}
```

Write a list of polygons to a text file. Each polygon will be written as one line containing its vertices, as in the following example:

```
[[2, 0], [1, 3], [-1, 0], [-3, -4]]
[[1, 0], [2, 6], [-4, -9]]
[[1, 0], [3, 5], [0, 1], [-5, -8]]
....
```

Julia Function A.2.91. source

```
create_polygon_dataset(f :: Union{HDF5.File, HDF5.Group}, path :: String, N :: Int)
```

Create an HDF5 dataset named `path` for storing rational polygons with N vertices. The dataset will have an HDF5 compound datatype with $2N$ integers, which are the entries of the vertex matrices stored in a column major layout. This function takes three keyword arguments:

- `T :: Type{<:Integer}`: The integer type to be used, e.g. `Int64`, `Int32`, ... The default is `Int`.
- `chunk_size :: Int`: Chunk size for HDF5. The default is `1000`.

- `deflate :: Int`: Deflate parameter for HDF5. The default is `3`.

Julia Function A.2.92. source

```
write_polygon_dataset(f :: Union{HDF5.File, HDF5.Group}, path :: String, Ps ::
↪  Vector{RationalPolygon{T,N,M}}) where {N,M,T <: Integer}
```

Write the polygons `Ps` to an HDF5 dataset named `path`. Creates the dataset if it does not exist already. If it does exist, the data will be appended to it.

Example:

Write the reflexive lattice triangles to an HDF5 file:

```
julia> using RationalPolygons, StaticArrays, HDF5

julia> Vs = SMatrix{2,3}[[1 0 -2 ; 0 1 -1], [1 1 -3 ; 0 2 -4], [1 0 -1 ; 0 1 -1],[1 0
↪  -3 ; 0 1 -2], [1 1 -2 ; 0 3 -3]];

julia> Ps = [RationalPolygon(V,1) for V ∈ Vs];

julia> f = h5open("/tmp/reflexive_triangles.h5", "cw")
🗂️ HDF5.File: (read-write) /tmp/reflexive_triangles.h5

julia> write_polygon_dataset(f, "my_triangles", Ps)

julia> close(f)
```

The resulting HDF5 file can be inspected using a command line tool such as `h5dump`:

```
$ h5dump /tmp/reflexive_triangles.h5
HDF5 "/tmp/reflexive_triangles.h5" {
GROUP "/" {
   DATASET "my_triangles" {
      DATATYPE  H5T_COMPOUND {
         H5T_STD_I64LE "1";
         H5T_STD_I64LE "2";
         H5T_STD_I64LE "3";
         H5T_STD_I64LE "4";
         H5T_STD_I64LE "5";
         H5T_STD_I64LE "6";
      }
      DATASPACE  SIMPLE { ( 5 ) / ( H5S_UNLIMITED ) }
      DATA {
      (0): {
            1,
            0,
            0,
            1,
```

```
            -2,
            -1
        },
    (1): {
            1,
            0,
            1,
            2,
            -3,
            -4
        },
    (2): {
            1,
            0,
            0,
            1,
            -1,
            -1
        },
    (3): {
            1,
            0,
            0,
            1,
            -3,
            -2
        },
    (4): {
            1,
            0,
            1,
            3,
            -2,
            -3
        }
    }
  }
}
}
```

We can read them back into `RationalPolygons.jl` at any time using `read_polygon_dataset`:

```
julia> using RationalPolygons, HDF5

julia> f = h5open("/tmp/reflexive_triangles.h5", "r")
🗂️ HDF5.File: (read-only) /tmp/reflexive_triangles.h5
└─ 🔢 my_triangles
```

```
julia> Ps = read_polygon_dataset(1, f, "my_triangles")
5-element Vector{RationalPolygon{Int64, 3, 6}}:
 Rational polygon of rationality 1 with 3 vertices.
 Rational polygon of rationality 1 with 3 vertices.
 Rational polygon of rationality 1 with 3 vertices.
 Rational polygon of rationality 1 with 3 vertices.
 Rational polygon of rationality 1 with 3 vertices.
```

Julia Function A.2.93. source

```
read_polygon_dataset(k :: T, f :: Union{HDF5.File, HDF5.Group}, path :: String, I...)
↪  where {T <: Integer}
```

Read from an HDF5 dataset containing k-rational polygons.

A.3 Subpolygons

A polygon P is called a *subpolygon* of Q, if $\varphi(P) \subseteq Q$ for some affine unimodular transformation φ. Given a k-rational polygon P, we can find all k-rational subpolygons of P using the algorithm from Subsection 1.2.2. The idea is to successively remove vertices of P by computing Hilbert bases.

A.3.1 Hilbert bases

We follow [20] to compute Hilbert bases of two-dimensional cones using Hirzebruch-Jung continued fractions.

Julia Function A.3.1. source

```
cls_cone_normal_form(A :: Matrix2{T}) where {T <: Integer}
```

Bring a two-dimensional cone into normal form in the sense of [20]. The result is a triple (d, k, M), where d and k are the parameters of the cone and M is a 2×2 integral matrix such that

$$M \cdot \begin{bmatrix} 0 & d \\ 1 & -k \end{bmatrix} = A.$$

Example:

```
julia> A = Matrix2(2, 1, -3, -5)
2×2 SMatrix{2, 2, Int64, 4} with indices SOneTo(2)×SOneTo(2):
 2  -3
```

```
 1  -5

julia> d, k, M = cls_cone_normal_form(A)
(7, 5, [1 2; 0 1])

julia> M * [0 d ; 1 -k] == A
true
```

Julia Function A.3.2. source

```
hirzebruch_jung(x :: T, y :: T) where {T <: Integer}
```

Return the Hirzebruch-Jung continued fraction associated to x/y.

Example:

See Example 10.2.4 of [20].

```
julia> hirzebruch_jung(7,5)
3-element Vector{Int64}:
 2
 2
 3
```

Julia Function A.3.3. source

```
hilbert_basis(A :: Matrix2{T}) where {T <: Integer}
```

Return the Hilbert basis of a two-dimensional cone spanned by the columns of A, which must be primitive.

Example:

See Example 10.2.4 of [20].

```
julia> A = Matrix2(0,1,7,-5)
2×2 SMatrix{2, 2, Int64, 4} with indices SOneTo(2)×SOneTo(2):
 0   7
 1  -5

julia> hilbert_basis(A)
5-element Vector{SVector{2, Int64}}:
 [0, 1]
 [1, 0]
 [2, -1]
 [3, -2]
 [7, -5]
```

source

```
hilbert_basis(v1 :: LatticePoint{T}, v2 :: LatticePoint{T}) where {T <: Integer}
```

Return the Hilbert basis of a two-dimensional cone spanned by given integral primitive vectors v_1 and v_2.

Julia Function A.3.4. source

```
remove_vertex(P :: RationalPolygon{T}, i :: Int) where {T <: Integer}
```

Return a pair `(Q, keeps_genus)`, where Q is the convex hull of all `k`-rational points of P except the i-th vertex and `keeps_genus :: Bool` is true if and only if Q has the same number of interior lattice points as P. If the argument `primitive = true` is passed, the convex hull is taken of all *primitive* k-rational points of P except the i-th vertex.

Example:

Compare Example 10.2.7 of [20].

```
julia> P = convex_hull(LatticePoint{Int}[(0,0),(7,-5),(7,1),(0,1)])
Rational polygon of rationality 1 with 4 vertices.

julia> Q,_ = remove_vertex(P,1)
(Rational polygon of rationality 1 with 4 vertices., false)

julia> vertices(Q)
4-element Vector{SVector{2, Rational{Int64}}}:
 [0, 1]
 [3, -2]
 [7, -5]
 [7, 1]
```

A.3.2 Computing subpolygons

Subpolygons can be either computed in memory or on disk using HDF5. The latter is useful for computations where the amount of polygons will grow very large (e.g. several billions), which would overload available memory. Note that even in the latter case, some polygons will have to be kept in memory in order to do equivalence checks with them later. In fact, to minimize memory usage further, we will only keep the *hashes* of those polygons and compare those. We use `128`-bit hashes here, which makes the probability of a hash collision negligible even for several billions of polygons.

Julia Type A.3.5. source

```
SubpolygonStorage{T <: Integer}
```

An abstract supertype of storage options for computing subpolygons. There are two subtypes `InMemorySubpolygonStorage` and `HDFSubpolygonStorage`. The former keeps all subpolygons in memory, the latter delegates their storage to the disk using the HDF5 format. Both implement `subpolygons_single_step`.

Julia Type A.3.6. source

```
struct InMemorySubpolygonStoragePreferences{T <: Integer}
```

A struct holding preferences for `InMemorySubpolygonStorage`. There are the following fields:

- `primitive :: Bool`: Whether only subpolygons with primitive vertices should be computed. The default is `false`.
- `use_affine_normal_form :: Bool`: Whether to use `affine_normal_form` or `unimodular_normal_form`. The default is `true`, which means affine normal form.
- `only_equal_number_of_interior_lattice_points :: Bool`: Whether only subpolygons having the same number of interior lattice points as the starting polygons should be computed. The default is `false`.
- `exclude_very_thin_polygons`: Whether polygons that can be realized in $\mathbb{R} \times [0, 1]$ should be excluded. This is only relevant for polygons with no interior lattice points. The default is `false`.

Julia Type A.3.7. source

```
mutable struct InMemorySubpolygonStorage{T <: Integer} <: SubpolygonStorage{T}
```

A struct holding results of a subpolygon computation. It has the following fields:

- `preferences :: InMemorySubpolygonStoragePreferences{T}`
- `polygons :: Dict{T,Set{RationalPolygon{T}}}`
- `last_completed_area :: T`
- `total_count :: Int`

Julia Type A.3.8. source

```
struct HDFSubpolygonStoragePreferences{T <: Integer}
```

A struct holding preferences for `HDFSubpolygonStorage`. There are the following fields:

- `rationality :: T`: The rationality of the polygons. The default is `one(T)`.
- `primitive :: Bool`: Whether only subpolygons with primitive vertices should be computed. The default is `false`.
- `use_affine_normal_form :: Bool`: Whether to use `affine_normal_form` or `unimodular_normal_form`. The default is `true`, which means affine normal form.
- `only_equal_number_of_interior_lattice_points :: Bool`: Whether only subpolygons having the same number of interior lattice points as the starting polygons should be computed. The default is `false`.
- `exclude_very_thin_polygons`: Whether polygons that can be realized in $\mathbb{R} \times [0, 1]$ should be excluded. This is only relevant for polygons with no interior lattice points. The default is `false`.
- `block_size :: Int`: How many polygons should be read into memory at once during the shaving process. Defaults to 10^6.
- `maximum_number_of_vertices :: Int`: An upper bound for the maximal number of vertices to be expected in the computation. This has to be set since every HDF5 file generated will have a dataset `numbers_of_polygons` storing the number of polygons for each number of vertices and the size of this dataset needs to be set beforehand. Defaults to 100, which should be more than enough for any feasible computation.
- `swmr :: Bool`: Whether to use single-writer-multiple-reader mode for HDF5. Defaults to `true`.

Julia Type A.3.9. source

```
mutable struct HDFSubpolygonStorage{T <: Integer} <: SubpolygonStorage{T}
```

A struct for managing the result of a subpolygon computation using the HDF5 file format. It has the following fields:

- `preferences :: HDFSubpolygonStoragePreferences{T}`
- `file_path :: String`: The file path of the HDF5 file to be created.
- `group_path :: String`: Path to a group within the HDF5 file, if it already exists. Defaults to `"/"`, i.e. the root group.
- `hash_sets :: Dict{T,Set{UInt128}}`: A dictionary of hashes of polygons already encountered. This is used for comparison with new polygons to ensure the result contains each polygon exactly once. We hold these hashes in memory to avoid needing to read in polygons that have been written out in the past, which saves a lot of time.

- `last_completed_area :: T`: The last area that has been completed. This counts down from the maximum area of the starting polygons to one.
- `total_count :: Int`

Julia Function A.3.10. source

```
initialize_subpolygon_storage(st :: HDFSubpolygonStorage{T}, Ps ::
↪  Vector{<:RationalPolygon{T}}) where {T <: Integer}
```

Initialize a newly created `HDFSubpolygonStorage` with a given set of starting polygons. To restart an interrupted computation, see also `restore_hdf_subpolygon_storage_status`.

Julia Function A.3.11. source

```
subpolygons_single_step(st :: InMemorySubpolygonStorage{T}; logging :: Bool = false)
↪  where {T <: Integer}
```

Perform a single step of a subpolygon computation in memory.

source

```
subpolygons_single_step(st :: HDFSubpolygonStorage{T}; logging :: Bool = false) where
↪  {T <: Integer}
```

Perform a single step in a subpolygon computation using the HDF5 file format.

Julia Function A.3.12. source

```
subpolygons(st :: SubpolygonStorage{T}; logging :: Bool = false) where {T <: Integer}
```

Compute all subpolygons with the given storage option.

source

```
subpolygons(Ps :: Vector{<:RationalPolygon{T}}) where {T <: Integer}
subpolygons(P :: RationalPolygon{T}) where {T <: Integer}
```

Compute all subpolygons of a rational polygon or list of rational polygons. The computation is done in memory. For storage on disk, see `HDFSubpolygonStorage`. This function takes the following keyword arguments:

- `primitive :: Bool`: Whether only subpolygons with primitive vertices should be returned. The default is `false`.
- `use_affine_normal_form :: Bool`: Whether to use affine or unimodular normal form. The default is `true`, which means affine normal form.

- `only_equal_number_of_interior_lattice_points :: Bool`: Whether only subpolygons that share the same number of interior lattice points with the starting polygons should be returned.
- `exclude_very_thin_polygons`: Whether polygons that can be realized in $\mathbb{R} \times [0, 1]$ should be excluded. This is only relevant for polygons with no interior lattice points. The default is `false`.
- `logging :: Bool`: Whether to display logging messages about the computation progress.

Example:

There are 148 subpolygons of the square of side length three, up to affine equivalence. The maximal number of vertices of those is eight, attained by exactly one polygon.

```
julia> Ps = subpolygons(convex_hull(LatticePoint{Int}[(0,0),(3,0),(0,3),(3,3)]));

julia> length(Ps)
148

julia> maximum(number_of_vertices.(Ps))
8
```

Julia Function A.3.13. source

```
restore_hdf_subpolygon_storage_status(st :: HDFSubpolygonStorage{T}) where {T <:
↪  Integer}
```

Restore a subpolygon computation's status that was interrupted from an HDF5 file. All polygons will be read in, hashed and saved into `st.hash_sets`. Moreover, `st.last_completed_area` and `st.total_count` will be properly set. After calling this function, the computation can be resumed by calling `subpolygons(st)` again.

A.4 Classifications

We provide implementations for the classification algorithms of rational polygons from Chapter 1, see Sections A.4.1 - A.4.4. Additionally, we implement the classification of LDP triangles by Picard index from Section 2.4, see A.4.5, and the classification of LDP quadrangles by Gorenstein index from Section 2.5, see A.4.6. Moreover, several other classifications of polygons are implemented. This includes the classification of lattice polygons by number of lattice points by Koelman [38] in Section A.4.7, lattice polygons by number of interior lattice points by Castryck [19] in Section A.4.8, lattice polygons contained in a square by Brown and Kasprzyk [18] in Section A.4.9, LDP polygons by Gorenstein index from Kasprzyk, Kreuzer and Nill [36] in Section A.4.10, LDP triangles by Gorenstein index from Bäuerle in Section A.4.11 [11] and LDP triangles with integral degree by Hausen and Király [31] in Section A.4.12.

A.4.1 Maximal rational polygons contained in $\mathbb{R} \times [-1, 1]$

We provide an implementation of Algorithm 1.3.4.

Julia Function A.4.1. source

```
classify_maximal_polygons_m1p1(k :: T, i :: Int) where {T <: Integer}
```

Return all maximal k-rational polygons with i interior lattice points that can be realized in $\mathbb{R} \times [-1, 1]$.

Example:

Compute the numbers of maximal k-rational polygons in $\mathbb{R} \times [-1, 1]$ for $k \leq 5$ and $i \leq 10$ interior lattice points.

```
julia> [length(classify_maximal_polygons_m1p1(k,i)) for k = 1 : 5, i = 0 : 10]
5×11 Matrix{Int64}:
  1    2    4    5    6    7    8    9   10   11   12
  4    9   13   18   22   26   30   34   38   42   46
 12   26   41   54   68   81   94  107  120  133  146
 24   57   86  117  145  174  203  231  259  288  316
 54  132  209  280  353  422  493  562  631  701  771
```

A.4.2 Maximal rational polygons with no interior lattice points

We provide an implementation of Algorithm 1.4.4.

Julia Function A.4.2. source

```
classify_maximal_lattice_free_polygons_m1p2_squares(k :: T) where {T <: Integer}
```

Return all k-maximal polygons with no interior lattice points that are contained in $A \cup \mathbb{R} \times [-1, 2] \cup B$ where A is the square with vertices $(0, 1), (1, 1), (1, 2), (0, 2)$ and B is the square with vertices $(0, 0), (0, -1), (1, -1), (1, 0)$.

Julia Function A.4.3. source

```
classify_maximal_lattice_free_polygons_m1p2_trapezoids(k :: T) where {T <: Integer}
```

Return all k-maximal polygons with no interior lattice points that are contained in $A \cup \mathbb{R} \times [-1, 2] \cup B$ where A is the trapezoid with vertices $(-1, 2), (0, 1), (1, 1), (1, 2)$ and B is the trapezoid with vertices $(0, 0), (0, -1), (2, -1), (1, 0)$, excluding the polygons from `classify_maximal_lattice_free_polygons_m1p2_squares`.

Julia Function A.4.4. source

```
classify_maximal_lattice_free_polygons_m1p2(k :: T) where {T <: Integer}
```

Return all k-maximal polygons with no interior lattice points contained in $\mathbb{R} \times [-1, 2]$. This is simply the union of `classify_maximal_lattice_free_polygons_m1p2_squares` and `classify_maximal_lattice_free_polygons_m1p2_trapezoids`.

Julia Function A.4.5. source

```
classify_maximal_lattice_free_polygons(k :: T ; logging = false) where {T <: Integer}
```

Return all k-maximal polygons with no interior lattice points.

Example:

We compute the numbers of k-maximal polygons with no interior lattice points for $1 \leq k \leq 6$. See also Classification 1.4.5.

```
julia> length.(classify_maximal_lattice_free_polygons.(1:6))
6-element Vector{Int64}:
   1
   4
  14
  39
 134
 299
```

A.4.3 Rational polygons with one interior lattice point

We provide an implementation of Algorithm 1.5.4.

Julia Function A.4.6. source

```
classify_maximal_polygons_genus_one_m1p1(k :: T) where {T <: Integer}
```

Return all maximal k-rational polygons with exactly one interior lattice point that can be realized in $\mathbb{R} \times [-1, 1]$. If `primitive = true` is passed, then only primitive polygons (i.e. LDP polygons) are returned.

Julia Function A.4.7. source

```
classify_maximal_polygons_genus_one_m1p2(k :: T) where {T <: Integer}
```

Return all maximal k-rational polygons with exactly one interior lattice point that can be realized in $\mathbb{R} \times [-1, 2]$. If `primitive = true` is passed, then only primitive polygons (i.e. LDP polygons) are returned.

Julia Function A.4.8. source

```
classify_maximal_polygons_genus_one_m2p2(k :: T, q :: Int) where {T <: Integer}
```

Return all maximal k-rational polygons with exactly one interior lattice point that can be realized in $\mathbb{R} \times [-2, 2]$ and that have non-empty intersection with the q-th classification box, where $1 \leq q \leq 3$. If `primitive = true` is passed, then only primitive polygons (i.e. LDP polygons) are returned.

Julia Function A.4.9. source

```
classify_maximal_polygons_genus_one(k :: T) where {T <: Integer}
```

Return all k-maximal polygons with exactly one interior lattice point. If `primitive = true` is passed, then only primitive polygons (i.e. LDP polygons) are returned.

Example:

We compute the numbers of polygons for $k \leq 3$. See also Classification 1.5.5.

```
julia> length.(classify_maximal_polygons_genus_one.(1:3))
3-element Vector{Int64}:
  3
 10
 39
```

Julia Function A.4.10. source

```
classify_polygons_genus_one(k :: T) where {T <: Integer}
```

Compute all k-rational polygons with exactly one interior lattice point. The following keyword arguments are supported:

- `primitive :: Bool`. If set to true, only primitive polygons (i.e. LDP polygons) are returned.
- `logging :: Bool`. Controls whether to display logging messages showing the current progress.

Example:

We reproduce the classification of all 5145 half-integral polygons with exactly one interior lattice point. It works by first computing all maximal polygons with `classify_maximal_polygons_genus_one` and then generating all their subpolygons.

```
julia> classify_polygons_genus_one(2; logging=true);
[ Info: Found 9 maximal polygons in QQ x [-1,1]. New: 9, total: 9
[ Info: Found 2 maximal polygons in QQ x [-1,2]. New: 1, total: 10
[ Info: Found 0 maximal polygons in QQ x [-2,2], box 1. New: 0, total: 10
[ Info: Found 3 maximal polygons in QQ x [-2,2], box 2. New: 0, total: 10
[ Info: Found 3 maximal polygons in QQ x [-2,2], box 3. New: 0, total: 10
[ Info: [a = 36]. Polygons to peel: 2.
[ Info: [a = 36]. Peeling complete. New polygons: 2. Running total: 12
[ Info: [a = 35]. Polygons to peel: 2.
[ Info: [a = 35]. Peeling complete. New polygons: 5. Running total: 17
[ Info: [a = 34]. Polygons to peel: 5.
[ Info: [a = 34]. Peeling complete. New polygons: 10. Running total: 27
[ Info: [a = 33]. Polygons to peel: 9.
[ Info: [a = 33]. Peeling complete. New polygons: 19. Running total: 46
[ Info: [a = 32]. Polygons to peel: 23.
[ Info: [a = 32]. Peeling complete. New polygons: 45. Running total: 91
[ Info: [a = 31]. Polygons to peel: 31.
[ Info: [a = 31]. Peeling complete. New polygons: 69. Running total: 160
[ Info: [a = 30]. Polygons to peel: 60.
[ Info: [a = 30]. Peeling complete. New polygons: 115. Running total: 275
[ Info: [a = 29]. Polygons to peel: 84.
[ Info: [a = 29]. Peeling complete. New polygons: 170. Running total: 445
[ Info: [a = 28]. Polygons to peel: 137.
[ Info: [a = 28]. Peeling complete. New polygons: 239. Running total: 684
[ Info: [a = 27]. Polygons to peel: 171.
[ Info: [a = 27]. Peeling complete. New polygons: 285. Running total: 969
[ Info: [a = 26]. Polygons to peel: 240.
[ Info: [a = 26]. Peeling complete. New polygons: 364. Running total: 1333
[ Info: [a = 25]. Polygons to peel: 286.
[ Info: [a = 25]. Peeling complete. New polygons: 440. Running total: 1773
[ Info: [a = 24]. Polygons to peel: 356.
[ Info: [a = 24]. Peeling complete. New polygons: 466. Running total: 2239
[ Info: [a = 23]. Polygons to peel: 351.
[ Info: [a = 23]. Peeling complete. New polygons: 433. Running total: 2672
[ Info: [a = 22]. Polygons to peel: 396.
[ Info: [a = 22]. Peeling complete. New polygons: 439. Running total: 3111
[ Info: [a = 21]. Polygons to peel: 391.
[ Info: [a = 21]. Peeling complete. New polygons: 394. Running total: 3505
[ Info: [a = 20]. Polygons to peel: 390.
[ Info: [a = 20]. Peeling complete. New polygons: 338. Running total: 3843
[ Info: [a = 19]. Polygons to peel: 349.
[ Info: [a = 19]. Peeling complete. New polygons: 269. Running total: 4112
[ Info: [a = 18]. Polygons to peel: 357.
[ Info: [a = 18]. Peeling complete. New polygons: 249. Running total: 4361
[ Info: [a = 17]. Polygons to peel: 301.
[ Info: [a = 17]. Peeling complete. New polygons: 215. Running total: 4576
[ Info: [a = 16]. Polygons to peel: 292.
[ Info: [a = 16]. Peeling complete. New polygons: 178. Running total: 4754
[ Info: [a = 15]. Polygons to peel: 233.
```

```
[ Info: [a = 15]. Peeling complete. New polygons: 121. Running total: 4875
[ Info: [a = 14]. Polygons to peel: 191.
[ Info: [a = 14]. Peeling complete. New polygons: 91. Running total: 4966
[ Info: [a = 13]. Polygons to peel: 140.
[ Info: [a = 13]. Peeling complete. New polygons: 71. Running total: 5037
[ Info: [a = 12]. Polygons to peel: 121.
[ Info: [a = 12]. Peeling complete. New polygons: 40. Running total: 5077
[ Info: [a = 11]. Polygons to peel: 67.
[ Info: [a = 11]. Peeling complete. New polygons: 21. Running total: 5098
[ Info: [a = 10]. Polygons to peel: 56.
[ Info: [a = 10]. Peeling complete. New polygons: 14. Running total: 5112
[ Info: [a = 9]. Polygons to peel: 38.
[ Info: [a = 9]. Peeling complete. New polygons: 13. Running total: 5125
[ Info: [a = 8]. Polygons to peel: 31.
[ Info: [a = 8]. Peeling complete. New polygons: 13. Running total: 5138
[ Info: [a = 7]. Polygons to peel: 14.
[ Info: [a = 7]. Peeling complete. New polygons: 3. Running total: 5141
[ Info: [a = 6]. Polygons to peel: 13.
[ Info: [a = 6]. Peeling complete. New polygons: 3. Running total: 5144
[ Info: [a = 5]. Polygons to peel: 4.
[ Info: [a = 5]. Peeling complete. New polygons: 1. Running total: 5145
[ Info: [a = 4]. Polygons to peel: 3.
[ Info: [a = 4]. Peeling complete. New polygons: 0. Running total: 5145
[ Info: [a = 3]. Polygons to peel: 1.
[ Info: [a = 3]. Peeling complete. New polygons: 0. Running total: 5145
[ Info: [a = 2]. Polygons to peel: 0.
[ Info: [a = 2]. Peeling complete. New polygons: 0. Running total: 5145
[ Info: [a = 1]. Polygons to peel: 0.
[ Info: [a = 1]. Peeling complete. New polygons: 0. Running total: 5145
```

A.4.4 Almost k-hollow LDP polygons

We can instruct `classify_polygons_genus_one` to only output k-rational polygons with primitive vertices. These are exactly the almost k-hollow LDP polygons and they correspond to $\frac{1}{k}$-log canonical toric del Pezzo surfaces. In particular, we can reproduce the classification of the 48032 almost 3-hollow LDP polygons ($\frac{1}{3}$-log canonical toric del Pezzo surfaces) from Theorem 4.11 of [27]. See also Classification 1.5.6 for the numbers up to $k = 6$.

```
julia> Pss = [classify_polygons_genus_one(k; primitive=true) for k = 1 : 3];

julia> numbers_of_polygons = length.(Pss)
3-element Vector{Int64}:
    16
   505
 48032
```

```
julia> max_vertices = [maximum(number_of_vertices.(Pss[k])) for k = 1 : 3]
3-element Vector{Int64}:
  6
  8
 12

julia> max_volumes = [k^2 * maximum(euclidean_area.(Pss[k])) for k = 1 : 3]
3-element Vector{Rational{Int64}}:
 9//2
 17
 47
```

A.4.5 LDP triangles by Picard index

We provide an implementation of the classification of LDP triangles (fake weighted projective planes) by Picard index from Section 2.4.

Julia Type A.4.11. source

```
abstract type PicardIndexStorage{T <: Integer} end
```

Abstract supertype of `InMemoryPicardIndexStorage` and `HDFPicardIndexStorage`.

Julia Type A.4.12. source

```
mutable struct InMemoryPicardIndexStorage{T <: Integer} <: PicardIndexStorage{T}
```

A struct holding classification results of LDP triangles by Picard index.

Julia Type A.4.13. source

```
struct HDFPicardIndexStoragePreferences{T <: Integer}
```

A struct holding preferences for the classification of LDP triangles by Picard index using the HDF5 file format. It has the following fields:

- `swmr :: Bool`: Whether to use single-writer-multiple-reader mode for HDF5. Defaults to `true`.
- `step_size :: Int`: The step size for multithreaded classification in terms of the Picard index. Defaults to 10^4.
- `maximum_picard_index :: Int`: The maximum Picard index to be classified. Defaults to 10^7.

Julia Type A.4.14. source

```
mutable struct HDFPicardIndexStorage{T <: Integer} <: PicardIndexStorage{T}
```

A struct for managing classification results of LDP triangles by Picard index using the HDF5 file format. It has the following fields:

- `preferences :: HDFPicardIndexStoragePreferences{T}`
- `file_path :: String`: The path of the HDF file to be generated.
- `last_completed_picard_index :: Int`: The last completed step of the classification. Initially, this will be 0.
- `total_count :: Int`: The total number of triangles found so far.

Julia Function A.4.15. source

```
quadruple_decompositions(p :: T) where {T <: Integer}
```

Return all quadruples (μ, w_1, w_2, w_3) such that $p = \mu^2 w_1 w_2 w_3$ and $w_1 \leq w_2 \leq w_3$.

Example:

```
julia> quadruple_decompositions(24)
4-element Vector{NTuple{4, Int64}}:
 (1, 1, 1, 24)
 (1, 1, 3, 8)
 (2, 1, 1, 6)
 (2, 1, 2, 3)
```

Julia Function A.4.16. source

```
classify_lattice_triangles_by_picard_index(p :: T) where {T <: Integer}
```

Return all LDP triangles with Picard index p.

Example:

There are two LDP triangles with Picard index six:

```
julia> classify_lattice_triangles_by_picard_index(6)
Set{RationalPolygon{Int64, 3, 6}} with 2 elements:
  Rational polygon of rationality 1 with 3 vertices.
  Rational polygon of rationality 1 with 3 vertices.
```

source

```
classify_lattice_triangles_by_picard_index(st :: InMemoryPicardIndexStorage{T},
↪  max_picard_index :: T) where {T <: Integer}
```

Perform the classification of LDP triangles up to `max_picard_index`, storing the results in memory.

Example:

Reproduce the classification up to Picard index 10000, see also Theorem 1.2 of [56].

```
julia> st = InMemoryPicardIndexStorage{Int}()
InMemoryPicardIndexStorage{Int64}(Vector{RationalPolygon{Int64, 3, 6}}[], 0)

julia> classify_lattice_triangles_by_picard_index(st, 10000);

julia> sum(length.(st.polygons))
68053
```

source

```
classify_lattice_triangles_by_picard_index(st :: HDFPicardIndexStorage{T},
↪  max_picard_index :: T; logging :: Bool = false) where {T <: Integer}
```

Perform the classification of LDP triangles up to `max_picard_index`, storing the results in an HDF5 file.

A.4.6 LDP quadrangles by Gorenstein index

We provide an implementation of the classification of LDP quadrangles by Gorenstein index described in Subsection 2.5.4.

Julia Function A.4.17. source

```
modified_unit_fraction_solutions(r :: T, s :: T, c :: T, d :: T) where {T <: Integer}
modified_unit_fraction_solutions(q :: Rational{T}, c :: T, d :: T) where {T <:
↪  Integer}
```

Return all integral solutions $(x, y) \in \mathbb{Z}^2_{\geq 1}$ to the equation $q = \frac{r}{s} = \frac{1}{x} + \frac{1}{y} - \frac{c}{dy}$ (see Lemma 2.5.26). Returns an error if there are infinitely many solutions, which happens if and only if $c = d$.

Example:

```
julia> modified_unit_fraction_solutions(1//5, 2, 5)
4-element Vector{Tuple{Int64, Int64}}:
 (6, 18)
 (8, 8)
```

```
(20, 4)
(10, 6)
```

Julia Function A.4.18. source

```
gorenstein_coefficients_to_grading_matrix_minors(ι :: T, A :: SMatrix{4,2,T,8}) where
↪  {T <: Integer}
```

Compute all minors of the Fano ι-Gorenstein matrix Q associated with the Gorenstein coefficients A.

Julia Function A.4.19. source

```
classify_gorenstein_coefficients(ι :: T, ::Val{t})
classify_gorenstein_coefficients(ι :: T)
```

Classify Gorenstein coefficients associated to Fano ι-Gorenstein matrices. Optionally takes in a `Val{t}` argument, where `t` can be `1`, `2`, `3` or `4`. In this case, only the Gorenstein coefficients of that type are classified, see Definition 2.5.24.

Julia Function A.4.20. source

```
gorenstein_coefficients_normal_form(A :: SMatrix{4,2,T,8}) where {T <: Integer}
```

Bring Gorenstein coefficients $A \in \mathbb{Z}^{2\times 4}$ into normal form, see Definition 2.5.6.

Julia Function A.4.21. source

```
classify_quadrilaterals_by_gorenstein_index(ι :: T, As :: Set{SMatrix{4,2,T}}) where
↪  {T <: Integer}
```

Return all LDP quadrilaterals with Gorenstein index ι whose Gorenstein coefficients are among the given set `As`.

source

```
classify_quadrilaterals_by_gorenstein_index(ι :: Integer)
```

Return all LDP quadrilaterals with Gorenstein index ι.

Example:

There are 73725 distinct LDP quadrilaterals with Gorenstein index at most 50.

```
julia> Pss = classify_quadrilaterals_by_gorenstein_index.(1:50);

julia> sum(length.(Pss))
73725
```

A.4.7 Lattice polygons by number of lattice points

Koelman [38] gave an algorithm to classify lattice polygons with a given number of lattice points and ran it up to 42 lattice points, Table 4.4.3 of [38]. We have implemented their algorithm here, which successfully reproduces their numbers. See also Section 1.6 and A371917 on OEIS [43] for the numbers up to 112 lattice points.

Julia Function A.4.22. source

```
height_one_points(P :: RationalPolygon)
```

Given a lattice polygon P, return all lattice points that have lattice height one with respect to some edge of P. Equivalently, return the set of boundary lattice points of `move_out_edges(P)`.

Julia Function A.4.23. source

```
single_point_extensions(Ps :: Vector{<:RationalPolygon{T}}) where {T <: Integer}
```

Return all lattice polygons that can be obtained by adding a single height one point to a polygon of `Ps`, up to affine unimodular equivalence.

Julia Type A.4.24. source

```
abstract type KoelmanStorage{T <: Integer} end
```

Abstract supertype of `InMemoryKoelmanStorage` and `HDFKoelmanStorage`. Both implement `classify_next_number_of_lattice_points`, which performs a single step in Koelman's classification of lattice polygons.

Julia Type A.4.25. source

```
mutable struct InMemoryKoelmanStorage{T <: Integer} <: KoelmanStorage{T}
```

A struct holding classification results of Koelman's classification of lattice polygons by number of lattice points. It has two fields `polygons :: Vector{Vector{RationalPolygon{T}}}` and `total_count :: Int`.

Julia Type A.4.26. source

```
struct HDFKoelmanStoragePreferences{T <: Integer}
```

A struct holding preferences for Koelman's classification using the HDF5 file format. It has the following fields:

- `swmr :: Bool`: Whether to use single-writer-multiple-reader mode for HDF5. Defaults to `true`.
- `maximum_number_of_vertices :: Int`: An upper bound for the maximal number of vertices to be expected in the classification. This has to be set since every HDF5 file generated will have a dataset `numbers_of_polygons` storing the number of polygons for each number of vertices and the size of this dataset needs to be set beforehand. Defaults to 100, which should be more than enough for any feasible computation.
- `block_size :: Int`: How many polygons should be read into memory at once during the extension process. Defaults to 10^6.

Julia Type A.4.27. source

```
mutable struct HDFKoelmanStorage{T <: Integer} <: KoelmanStorage{T}
```

A struct for managing classification results of Koelman's classification of lattice polygons using the HDF5 file format. It has the following fields:

- `preferences :: HDFKoelmanStoragePreferences{T}`
- `directory_path :: String`: The directory where the HDF5 files will be generated.
- `last_completed_number_of_lattice_points :: Int`: The last completed step of the classification. Initially, this will be 3.
- `total_count :: Int`: The total number of polygons found so far.

Julia Function A.4.28. source

```
classify_next_number_of_lattice_points(st :: InMemoryKoelmanStorage{T}) where {T <:
↪  Integer}
```

Perform a single step in Koelman's classification of lattice polygons using in-memory storage.

Example:

Perform a single step of Koelmans classification using `Int64`. The result tells us that there are three lattice polygons with exactly four lattice points.

```
julia> st = InMemoryKoelmanStorage{Int64}();

julia> classify_next_number_of_lattice_points(st)
3
```

source

```
classify_next_number_of_lattice_points(st :: HDFKoelmanStorage{T}; logging :: Bool =
↪  false) where {T <: Integer}
```

Perform a single step in Koelman's classification of lattice polygons using on-disk storage with HDF5.

Julia Function A.4.29. source

```
classify_polygons_by_number_of_lattice_points(st :: KoelmanStorage{T},
↪  max_number_of_lattice_points :: Int; logging :: Bool = false) where {T <: Integer}
```

Run Koelman's classification of lattice polygons by number of lattice points, up to `max_number_of_lattice_points`.

Example:

Reproduce Koelman's original classification in memory, see Table 4.4.3 of [38] or A371917 on OEIS [43]. This should not take longer than a few minutes on modern hardware.

```
julia> st = InMemoryKoelmanStorage{Int}();

julia> classify_polygons_by_number_of_lattice_points(st, 42; logging=true);
[ Info: [l = 4]. New polygons: 3. Total: 4
[ Info: [l = 5]. New polygons: 6. Total: 10
[ Info: [l = 6]. New polygons: 13. Total: 23
[ Info: [l = 7]. New polygons: 21. Total: 44
[ Info: [l = 8]. New polygons: 41. Total: 85
[ Info: [l = 9]. New polygons: 67. Total: 152
[ Info: [l = 10]. New polygons: 111. Total: 263
[ Info: [l = 11]. New polygons: 175. Total: 438
[ Info: [l = 12]. New polygons: 286. Total: 724
[ Info: [l = 13]. New polygons: 419. Total: 1143
[ Info: [l = 14]. New polygons: 643. Total: 1786
[ Info: [l = 15]. New polygons: 938. Total: 2724
[ Info: [l = 16]. New polygons: 1370. Total: 4094
[ Info: [l = 17]. New polygons: 1939. Total: 6033
[ Info: [l = 18]. New polygons: 2779. Total: 8812
[ Info: [l = 19]. New polygons: 3819. Total: 12631
[ Info: [l = 20]. New polygons: 5293. Total: 17924
[ Info: [l = 21]. New polygons: 7191. Total: 25115
[ Info: [l = 22]. New polygons: 9752. Total: 34867
```

```
[ Info: [l = 23]. New polygons: 12991. Total: 47858
[ Info: [l = 24]. New polygons: 17321. Total: 65179
[ Info: [l = 25]. New polygons: 22641. Total: 87820
[ Info: [l = 26]. New polygons: 29687. Total: 117507
[ Info: [l = 27]. New polygons: 38533. Total: 156040
[ Info: [l = 28]. New polygons: 49796. Total: 205836
[ Info: [l = 29]. New polygons: 63621. Total: 269457
[ Info: [l = 30]. New polygons: 81300. Total: 350757
[ Info: [l = 31]. New polygons: 102807. Total: 453564
[ Info: [l = 32]. New polygons: 129787. Total: 583351
[ Info: [l = 33]. New polygons: 162833. Total: 746184
[ Info: [l = 34]. New polygons: 203642. Total: 949826
[ Info: [l = 35]. New polygons: 252898. Total: 1202724
[ Info: [l = 36]. New polygons: 313666. Total: 1516390
[ Info: [l = 37]. New polygons: 386601. Total: 1902991
[ Info: [l = 38]. New polygons: 475540. Total: 2378531
[ Info: [l = 39]. New polygons: 582216. Total: 2960747
[ Info: [l = 40]. New polygons: 710688. Total: 3671435
[ Info: [l = 41]. New polygons: 863552. Total: 4534987
[ Info: [l = 42]. New polygons: 1048176. Total: 5583163
```

Example:

Reproduce Koelman's classification and store the output to HDF5 files.

```
julia> st = HDFKoelmanStorage{Int}("/tmp");

julia> classify_polygons_by_number_of_lattice_points(st, 42);
```

The result will be HDF5 files named `11.h5`, `12.h5` etc., each containing one dataset of polygons for every number of vertices as well as a dataset `numbers_of_polygons` holding their numbers.

```
julia> using HDF5

julia> f = h5open("/tmp/test/l42.h5", "r")
🗂️ HDF5.File: (read-only) /tmp/test/l42.h5
├─ 🔢 n10
├─ 🔢 n11
├─ 🔢 n12
├─ 🔢 n3
├─ 🔢 n4
├─ 🔢 n5
├─ 🔢 n6
├─ 🔢 n7
├─ 🔢 n8
├─ 🔢 n9
└─ 🔢 numbers_of_polygons
```

```
julia> A = read_dataset(f, "numbers_of_polygons");

julia> sum(A) # The number of lattice polygons with 42 lattice points
1048176

julia> Ps = read_polygon_dataset(1, f, "n5"); # All pentagons with 42 lattice points.

julia> all(P -> number_of_lattice_points(P) == 42, Ps)
true
```

A.4.8 Lattice polygons by number of interior lattice points

Castryck [19] gave an algorithm to classify lattice polygons by number of interior lattice points i and ran it up to $i = 30$. Our implementation here successfully reproduces their numbers from Table 1 of [19], see also A322343 on OEIS [43].

Julia Function A.4.30. source

```
move_out_edges(P :: RationalPolygon)
```

Given a k-rational polygon P, return the polygon $P^{(-1)}$ obtained by moving out all edges by $\frac{1}{k}$. See Subsection 1.2.3.

Julia Function A.4.31. source

```
classify_maximal_lattice_polygons_with_collinear_interior_points(i :: Int, T ::
↪  Type{<:Integer} = Int)
```

Return all maximal lattice polygons with i collinear interior lattice points.

Julia Function A.4.32. source

```
classify_maximal_lattice_polygons_with_two_dimensional_empty_fine_interior(i :: Int, T
↪  :: Type{<:Integer} = Int)
```

Return all maximal lattice polygons with i interior lattice points, where the convex hull of these points is a two-dimensional lattice polygon without interior lattice points.

Julia Type A.4.33. source

```
abstract type CastryckStorage{T <: Integer} end
```

Abstract supertype of `InMemoryCastryckStorage` and `HDFCastryckStorage`. Both implement `classify_next_genus`, which performs a single step in Castryck's classification of lattice polygons.

Julia Type A.4.34. source

```
mutable struct InMemoryCastryckStorage{T <: Integer} <: CastryckStorage{T}
```

A struct holding classification results of Castryck's classification of lattice polygons by number of interior lattice points. It has the following fields:

- `maximum_genus :: Int`: An upper bound for the maximum number of lattice points that the classification should run to. Defaults to 100.
- `maximal_polygons :: Vector{Vector{RationalPolygon{T}}}`,
- `all_polygons :: Vector{Vector{RationalPolygon{T}}}`,
- `total_count :: Int`: The total number of polygons found so far.

Julia Type A.4.35. source

```
struct HDFCastryckStoragePreferences{T <: Integer}
```

A struct holding preferences for Castryck's classification using the HDF5 file format. It has the following fields:

- `swmr :: Bool`: Whether to use single-writer-multiple-reader mode for HDF5. Defaults to `true`.
- `maximum_genus :: Int`: An upper bound for the maximal number of interior lattice points to which the classification should be run. Defaults to 100.
- `maximum_number_of_vertices :: Int`: An upper bound for the maximal number of vertices to be expected in the classification. This has to be set since every HDF5 file generated will have a dataset `numbers_of_polygons` storing the number of polygons for each number of vertices and the size of this dataset needs to be set beforehand. Defaults to 100, which should be more than enough for any feasible computation.
- `block_size :: Int`: How many polygons should be read into memory at once during the computation of subpolygons and the moving-out process. Defaults to 10^6.

Julia Type A.4.36. source

```
mutable struct HDFCastryckStorage{T <: Integer} <: CastryckStorage{T}
```

A struct for managing classification results of Castryck's classification of lattice polygons using the HDF5 file format. It has the following fields:

- `preferences :: HDFCastryckStoragePreferences{T}`

- `directory_path :: String`: The directory where the HDF5 files will be generated.
- `last_completed_genus :: Int`: The last completed step of the classification. Initially, this will be 0.
- `total_count :: Int`: the total number of polygons found so far.

Julia Function A.4.37. source

```
classify_next_genus(st :: InMemoryCastryckStorage{T}; logging :: Bool = false) where
↪  {T <: Integer}
```

Perform a single step in Castryck's classification of lattice polygons by number of interior lattice points, using in-memory storage. Returns a tuple where the first entry is the number of lattice polygons obtained and the second number is the number of maximal lattice polygons.

Example:

Perform two steps in Castryck's classification. The result tells us that there are 45 lattice polygons with exactly two interior lattice points, four of which are maximal.

```
julia> st = InMemoryCastryckStorage{Int}();

julia> classify_next_genus(st)
(16, 3)

julia> classify_next_genus(st)
(45, 4)
```

source

```
classify_next_genus(st :: HDFCastryckStorage{T}; logging :: Bool = false) where {T <:
↪  Integer}
```

Perform a single step in Castryck's classification of lattice polygons using on-disk storage with HDF5.

Julia Function A.4.38. source

```
classify_lattice_polygons_by_genus(st :: CastryckStorage{T}, max_genus :: Int; logging
↪  :: Bool = false) where {T <: Integer}
```

Run Castryck's classification of lattice polygons by number of interior lattice points, up to `max_genus`.

Example:

Reproduce Castryck's classification in memory, see Table 1 of [19] or A322343 on OEIS [43]. This should not take longer than a few minutes on modern hardware.

```
julia> st = InMemoryCastryckStorage{Int}();

julia> classify_lattice_polygons_by_genus(st, 30; logging=true)
[ Info: [i = 1]. Got 3 maximal polygons.
[ Info: [i = 1]. Subpolygons complete. Num of polygons: 16
[ Info: [i = 1]. Moving out complete. New maximal polygons: 16
[ Info: [i = 2]. Got 4 maximal polygons.
[ Info: [i = 2]. Subpolygons complete. Num of polygons: 45
[ Info: [i = 2]. Moving out complete. New maximal polygons: 22
[ Info: [i = 3]. Got 6 maximal polygons.
[ Info: [i = 3]. Subpolygons complete. Num of polygons: 120
[ Info: [i = 3]. Moving out complete. New maximal polygons: 63
[ Info: [i = 4]. Got 9 maximal polygons.
[ Info: [i = 4]. Subpolygons complete. Num of polygons: 211
[ Info: [i = 4]. Moving out complete. New maximal polygons: 78
[ Info: [i = 5]. Got 11 maximal polygons.
[ Info: [i = 5]. Subpolygons complete. Num of polygons: 403
[ Info: [i = 5]. Moving out complete. New maximal polygons: 122
[ Info: [i = 6]. Got 13 maximal polygons.
[ Info: [i = 6]. Subpolygons complete. Num of polygons: 714
[ Info: [i = 6]. Moving out complete. New maximal polygons: 192
[ Info: [i = 7]. Got 16 maximal polygons.
[ Info: [i = 7]. Subpolygons complete. Num of polygons: 1023
[ Info: [i = 7]. Moving out complete. New maximal polygons: 239
[ Info: [i = 8]. Got 21 maximal polygons.
[ Info: [i = 8]. Subpolygons complete. Num of polygons: 1830
[ Info: [i = 8]. Moving out complete. New maximal polygons: 316
[ Info: [i = 9]. Got 27 maximal polygons.
[ Info: [i = 9]. Subpolygons complete. Num of polygons: 2700
[ Info: [i = 9]. Moving out complete. New maximal polygons: 508
[ Info: [i = 10]. Got 33 maximal polygons.
[ Info: [i = 10]. Subpolygons complete. Num of polygons: 3659
[ Info: [i = 10]. Moving out complete. New maximal polygons: 509
[ Info: [i = 11]. Got 38 maximal polygons.
[ Info: [i = 11]. Subpolygons complete. Num of polygons: 6125
[ Info: [i = 11]. Moving out complete. New maximal polygons: 700
[ Info: [i = 12]. Got 51 maximal polygons.
[ Info: [i = 12]. Subpolygons complete. Num of polygons: 8101
[ Info: [i = 12]. Moving out complete. New maximal polygons: 1044
[ Info: [i = 13]. Got 61 maximal polygons.
[ Info: [i = 13]. Subpolygons complete. Num of polygons: 11027
[ Info: [i = 13]. Moving out complete. New maximal polygons: 1113
[ Info: [i = 14]. Got 76 maximal polygons.
[ Info: [i = 14]. Subpolygons complete. Num of polygons: 17280
[ Info: [i = 14]. Moving out complete. New maximal polygons: 1429
[ Info: [i = 15]. Got 86 maximal polygons.
[ Info: [i = 15]. Subpolygons complete. Num of polygons: 21499
[ Info: [i = 15]. Moving out complete. New maximal polygons: 2052
[ Info: [i = 16]. Got 113 maximal polygons.
```

```
[ Info: [i = 16]. Subpolygons complete. Num of polygons: 28689
[ Info: [i = 16]. Moving out complete. New maximal polygons: 1962
[ Info: [i = 17]. Got 129 maximal polygons.
[ Info: [i = 17]. Subpolygons complete. Num of polygons: 43012
[ Info: [i = 17]. Moving out complete. New maximal polygons: 2651
[ Info: [i = 18]. Got 166 maximal polygons.
[ Info: [i = 18]. Subpolygons complete. Num of polygons: 52736
[ Info: [i = 18]. Moving out complete. New maximal polygons: 3543
[ Info: [i = 19]. Got 200 maximal polygons.
[ Info: [i = 19]. Subpolygons complete. Num of polygons: 68557
[ Info: [i = 19]. Moving out complete. New maximal polygons: 3638
[ Info: [i = 20]. Got 240 maximal polygons.
[ Info: [i = 20]. Subpolygons complete. Num of polygons: 97733
[ Info: [i = 20]. Moving out complete. New maximal polygons: 4594
[ Info: [i = 21]. Got 281 maximal polygons.
[ Info: [i = 21]. Subpolygons complete. Num of polygons: 117776
[ Info: [i = 21]. Moving out complete. New maximal polygons: 5996
[ Info: [i = 22]. Got 352 maximal polygons.
[ Info: [i = 22]. Subpolygons complete. Num of polygons: 152344
[ Info: [i = 22]. Moving out complete. New maximal polygons: 6364
[ Info: [i = 23]. Got 403 maximal polygons.
[ Info: [i = 23]. Subpolygons complete. Num of polygons: 209409
[ Info: [i = 23]. Moving out complete. New maximal polygons: 7922
[ Info: [i = 24]. Got 506 maximal polygons.
[ Info: [i = 24]. Subpolygons complete. Num of polygons: 248983
[ Info: [i = 24]. Moving out complete. New maximal polygons: 9692
[ Info: [i = 25]. Got 584 maximal polygons.
[ Info: [i = 25]. Subpolygons complete. Num of polygons: 319957
[ Info: [i = 25]. Moving out complete. New maximal polygons: 10208
[ Info: [i = 26]. Got 708 maximal polygons.
[ Info: [i = 26]. Subpolygons complete. Num of polygons: 420714
[ Info: [i = 26]. Moving out complete. New maximal polygons: 12727
[ Info: [i = 27]. Got 821 maximal polygons.
[ Info: [i = 27]. Subpolygons complete. Num of polygons: 497676
[ Info: [i = 27]. Moving out complete. New maximal polygons: 15431
[ Info: [i = 28]. Got 995 maximal polygons.
[ Info: [i = 28]. Subpolygons complete. Num of polygons: 641229
[ Info: [i = 28]. Moving out complete. New maximal polygons: 15918
[ Info: [i = 29]. Got 1121 maximal polygons.
[ Info: [i = 29]. Subpolygons complete. Num of polygons: 813814
[ Info: [i = 29]. Moving out complete. New maximal polygons: 20354
[ Info: [i = 30]. Got 1352 maximal polygons.
[ Info: [i = 30]. Subpolygons complete. Num of polygons: 957001
[ Info: [i = 30]. Moving out complete. New maximal polygons: 23873
```

Example:

Reproduce Castryck's classification and store the output to HDF5 files

```
julia> st = HDFCastryckStorage{Int}("/tmp");

julia> classify_lattice_polygons_by_genus(st, 30);
```

This will create two directories `all` and `maximal` in the target directory and populate them with HDF5 files `i1.h5`, `i2.h5`, etc., containing the polygons. The files in `maximal` contain datasets ordered by number of vertices. The files in `all` contain groups ordered by area, which contain datasets ordered by number of vertices. Every HDF5 file additionally contains a dataset `numbers_of_polygons`.

```
julia> using HDF5

julia> f = h5open("/tmp/all/i30.h5", "r");

julia> Ps = read_polygon_dataset(1, f, "a69/n8"); # Read in all octagons of genus 30
↪  with normalized volume 69

julia> all(P -> number_of_interior_lattice_points(P) == 30, Ps)
true

julia> A = read_dataset(f, "numbers_of_polygons"); # The total number of lattice
↪  polygons with 30 interior lattice points

julia> sum(A) # The total number of lattice polygons with 30 interior lattice points.
957001
```

A.4.9 Lattice polygons contained in a square

Brown and Kasprzyk [18] considered lattice polygons that are contained in a square of fixed side length and classified them up to side length seven. Their numbers (Table 1 of [18], see also A374975) can be reproduced with `RationalPolygons.jl` by computing subpolygons of the square. See also Classification 1.2.13 for the numbers up to side length 11.

```
julia> square(m) = convex_hull(LatticePoint{Int}[(0,0),(m,0),(0,m),(m,m)])
square (generic function with 1 method)

julia> Pss = [subpolygons(square(m); use_affine_normal_form = true,
↪  only_equal_number_of_interior_lattice_points = false) for m = 1 : 7];

julia> numbers_of_polygons = [length(Pss[m]) - length(Pss[m-1]) for m = 2 : 7]
6-element Vector{Int64}:
    15
   131
  1369
 13842
```

```
  129185
 1104895

julia> max_vertices = [maximum(number_of_vertices.(Pss[m])) for m = 1 : 7]
7-element Vector{Int64}:
  4
  6
  8
  9
 10
 12
 13

julia> [length(filter(P -> number_of_vertices(P) == max_vertices[m], Pss[m])) for m =
↪  1 : 7]
7-element Vector{Int64}:
  1
  1
  1
  1
 15
  2
  3
```

A.4.10 LDP polygons by Gorenstein index

Kasprzyk, Kreuzer and Nill [36] gave an algorithm to classify LDP polygons by Gorenstein index and ran the classification up to index 17. `RationalPolygons.jl` implements a version of their algorithm, which successfully reproduces their numbers (see Theorem 1.2 of [36]). See also A145581 on OEIS [43] for the numbers up to index 32.

Julia Type A.4.39. source

```
PartialLDP{T<:Integer,N,M}
```

A struct used in the classification of polygons by Gorenstein index. It contains three fields:

- `P :: RationalPolygon{T,N,M}`: The polygon constructed so far.
- `initial_local_index : T`: The local index of the special facet.
- `ymin : T`: Lower bound for the y-value of the next vertex to be chosen, see Algorithm 6.3 of [36].

Julia Function A.4.40. source

```
initial_special_facets(index :: T, initial_local_index :: T) where {T <: Integer}
```

Return all possible initial special facet to start the classification as in step (1) of Algorithm 6.3 of [36].

Julia Function A.4.41. source

```
choose_next_vertex(ldps :: Vector{<:PartialLDP{T,N}}, index :: T) where {N, T <:
↪  Integer}
```

Perform a single step in the classification of LDP polygons by Gorenstein index as in step (2) of Algorithm 6.3 of [36].

Julia Function A.4.42. source

```
classify_lattice_polygons_by_gorenstein_index(index :: T; logging :: Bool = false)
↪  where {T <: Integer}
```

Return all LDP polygons with given Gorenstein index, using the Algorithm described in [36].

Example:

There are 91 LDP polygons of Gorenstein index four: 13 triangles, 48 quadrilaterals, 29 pentagons and one hexagon.

```
julia> Pss = classify_lattice_polygons_by_gorenstein_index(4);

julia> length.(Pss)
4-element Vector{Int64}:
 13
 48
 29
  1

julia> sum(ans)
91
```

A.4.11 LDP triangles by Gorenstein index

Bäuerle [11] gave an algorithm to classify Fano simplices by dimension and Gorenstein index. `RationalPolygons.jl` implements a version of his algorithm (specialized to the two-dimensional case), which reproduces his numbers successfully. See also Subsection 2.5.3 for a description of the Algorithm and the classification up to Gorenstein index 45 000.

Julia Function A.4.43. source

```
unit_fraction_partitions_length_three(ι :: T) where {T <: Integer}
```

Return all triples (a, b, c) such that $\frac{1}{\iota} = \frac{1}{a} + \frac{1}{b} + \frac{1}{c}$ and $a \leq b \leq c$. See also A004194 on OEIS [43].

Example:

```
julia> unit_fraction_partitions_length_three(2)
10-element Vector{Tuple{Int64, Int64, Int64}}:
 (3, 7, 42)
 (3, 8, 24)
 (3, 9, 18)
 (3, 10, 15)
 (3, 12, 12)
 (4, 5, 20)
 (4, 6, 12)
 (4, 8, 8)
 (5, 5, 10)
 (6, 6, 6)
```

Julia Type A.4.44. source

```
abstract type BaeuerleStorage{T <: Integer} end
```

Abstract supertype of `InMemoryBaeuerleStorage` and `HDFBaeuerleStorage`.

Julia Type A.4.45. source

```
mutable struct InMemoryBaeuerleStorage{T <: Integer} <: BaeuerleStorage{T}
```

A struct holding classification results of Bäuerle's classification of LDP triangles by Gorenstein index.

Julia Type A.4.46. source

```
struct HDFBaeuerleStoragePreferences{T <: Integer}
```

A struct holding preferences for Bäuerle's classification using the HDF5 file format. It has the following fields:

- `swmr :: Bool`: Whether to use single-writer-multiple-reader mode for HDF5. Defaults to `true`.
- `step_size :: Int`: The step size for multithreaded classification in terms of the Gorenstein index. Defaults to 100.

- `maximum_gorenstein_index :: Int`: The maximum Gorenstein index to be classified. Defaults to 10^5.

Julia Type A.4.47. source

```
mutable struct HDFBaeuerleStorage{T <: Integer} <: BaeuerleStorage{T}
```

A struct for managing classification results of Bäuerle's classification of LDP triangles using the HDF5 file format. It has the following fields:

- `preferences :: HDFBaeuerleStoragePreferences{T}`
- `file_path :: String`: The path of the HDF file to be generated.
- `last_completed_gorenstein_index :: Int`: The last completed step of the classification. Initially, this will be 0.
- `total_count :: Int`: The total number of polygons found so far.

Julia Function A.4.48. source

```
classify_lattice_triangles_by_gorenstein_index(ι :: T) where {T <: Integer}
```

Return all LDP triangles with Gorenstein index ι.

Example:

There are five LDP triangles with Gorenstein index one:

```
julia> classify_lattice_triangles_by_gorenstein_index(1)
Set{RationalPolygon{Int64, 3}} with 5 elements:
  Rational polygon of rationality 1 with 3 vertices.
  Rational polygon of rationality 1 with 3 vertices.
  Rational polygon of rationality 1 with 3 vertices.
  Rational polygon of rationality 1 with 3 vertices.
  Rational polygon of rationality 1 with 3 vertices.
```

source

```
classify_lattice_triangles_by_gorenstein_index(st :: InMemoryBaeuerleStorage{T},
↪  max_gorenstein_index :: T) where {T <: Integer}
```

Perform Bäuerle's classification of LDP triangles up to `max_gorenstein_index`, storing the results in memory.

Example:

Reproduce Bäuerle's original classification up to Gorenstein index 1000, see Theorem 1.4 of [11].

```
julia> st = InMemoryBaeuerleStorage{Int}()
InMemoryBaeuerleStorage{Int64}(Vector{RationalPolygon{Int64, 3, 6}}[], 0)

julia> classify_lattice_triangles_by_gorenstein_index(st, 1000);

julia> sum(length.(st.polygons))
2992229
```

source

```
classify_lattice_triangles_by_gorenstein_index(st :: HDFBaeuerleStorage{T},
↪  max_gorenstein_index :: T; logging :: Bool = false) where {T <: Integer}
```

Perform Bäuerle's classification of LDP triangles up to `max_gorenstein_index`, storing the results in an HDF5 file

A.4.12 LDP triangles with integral degree

Hausen and Király [31] classified fake weighted projective planes having integral degree (=canonical self intersection). In terms of polygons, these can be described as LDP triangles such that twice the euclidean area of its dual is an integer (see also Proposition 2.2.23). The attained values of this integer (which is the degree of the associated fake weighted projective plane) are the integers $\{1, 2, 3, 4, 5, 6, 8, 9\}$. In total, there are 24 infinite series of these triangles, where each of them is parameterized by the solution set of a squared Markov type equation (see Theorem 1.1 of [31]). These solution sets can be described as infinite binary trees with a unique root. `RationalPolygons.jl` uses this description to implement a classification algorithm for LDP triangles with integral degree. To make this classification finite, one has to provide a maximal depth to which the solution trees are traversed.

As with our other classifications, all functions come with a parameter `T <: Integer`, which is the integer type to be used. Since the entries of the solution triples of squared Markov type equations grow very quickly with increasing depth, it is recommended to use `BigInt` here instead of fixed-size integer types (`Int64` overflows already for `depth = 4`).

Julia Method A.4.49. source

```
degree(w :: SVector{3})
```

Return the degree of a fake weighted projective plane with given fake weight vector w. This is given by the formula $(w_1 + w_2 + w_3)/(w_1 w_2 w_3)$, see for instance Proposition 3.7 of [31].

Julia Function A.4.50. source

```
fake_weight_vector(P :: RationalPolygon{T,3}) where {T <: Integer}
```

Return the fake weight vector of the fake weighted projective plane associated to the LDP triangle P.

Julia Function A.4.51. source

```
n_step_mutations(us :: Set{SVector{3,T}}, depth :: Int) where {T <: Integer}
```

Given solution triples `us` of a squared Markov type equation, return all solution triples that can be obtained from `us` by applying at most `depth` many mutations. This function returns a vector of length `depth+1`, containing for each level i from 0 to `depth` the set of solution triples after exactly i mutation steps.

Example:

Get all solutions to the squared Markov type equation $9xyz = (x + y + z)^2$ by starting with the initial solution $(1, 1, 1)$ and applying at most four mutations. Compare with $T(9)$ of Remark 2.5 of [31].

```
julia> us = Set([SVector{3,Int}(1,1,1)])
Set{SVector{3, Int64}} with 1 element:
  [1, 1, 1]

julia> n_step_mutations(us, 4)
5-element Vector{Set{SVector{3, Int64}}}:
 Set([[1, 1, 1]])
 Set([[1, 1, 4]])
 Set([[1, 4, 25]])
 Set([[4, 25, 841], [1, 25, 169]])
 Set([[25, 169, 37636], [25, 841, 187489], [1, 169, 1156], [4, 841, 28561]])
```

Julia Function A.4.52. source

```
initial_triple(:: Val{A}, T :: Type{<:Integer} = BigInt)
```

Return the unique initial triple of the squared Markov type equation $Axyz = (x + y + z)^2$. Allowed values of A are 9, 8, 6 and 5.

Example:

The unique initial triples for all allowed values of A, see Theorem 2.2 of [31].

```
julia> [initial_triple(Val(A)) for A in [9,8,6,5]]
4-element Vector{SVector{3, BigInt}}:
 [1, 1, 1]
 [1, 1, 2]
```

```
[1, 2, 3]
[1, 4, 5]
```

Julia Function A.4.53. source

```
adjust_triple(::Val{A}, u :: SVector{3,T}) where {T <: Integer}
adjust_triple(u :: SVector{3,T}) where {T <: Integer}
```

Reorder a solution triple of a squared Markov type equation to make it adjusted, according to Definition 3.14 of [31]. Allowed values of A are 9, 8, 6 and 5. If A is not given, it is determined by calculating the degree of u.

Example:

The solution triple $(50, 9, 3481)$ for $A = 8$ is not adjusted, since the even entry is not last.

```
julia> adjust_triple(Val(8), SVector{3,Int}(50,9,3481))
3-element SVector{3, Int64} with indices SOneTo(3):
    9
 3481
   50
```

Julia Function A.4.54. source

```
classify_squared_markov_type_equation_solutions(::Val{A}, depth :: Int, T ::
↪  Type{<:Integer} = BigInt) where {A}
```

Return all solutions to the squared Markov type equation $Axyz = (x + y + z)^2$ by starting with the initial solution and applying at most `depth` many mutations. Allowed values of A are 9, 8, 6 and 5.

Example:

All solution triples to $8xyz = (x + y + z)^2$ with depth at most three.

```
julia> classify_squared_markov_type_equation_solutions(Val(8),3)
4-element Vector{Set{SVector{3, BigInt}}}:
 Set([[1, 1, 2]])
 Set([[1, 2, 9]])
 Set([[2, 9, 121], [1, 9, 50]])
 Set([[9, 50, 3481], [2, 121, 1681], [1, 50, 289], [9, 121, 8450]])
```

Julia Function A.4.55. source

```
fake_weight_vectors_to_triangles(us :: Set{SVector{3,T}}, μ :: T) where {T <: Integer}
```

Given a set of triples `us` sharing the same integral degree A and an integer μ, return all LDP triangles having fake weight vector $\mu \cdot u$. Allowed values of A are 9, 8, 6 and 5. Moreover, μ must be a divisor of A. The resulting triangles will have degree A/μ and multiplicity μ.

Julia Function A.4.56. source

```
classify_lattice_triangles_integral_degree(::Val{K}, ::Val{μ}, depth :: Int, T ::
↪  Type{<:Integer} = BigInt) where {K, μ}
```

Return all LDP triangles (= fake weighted projective planes) with integral degree K and class group torsion order μ, up to a given depth in the Markov tree. Essentially, this returns the triangles of the series (K-μ-*) according to the notation of Theorem 1.1 of [31]. Allowed values of K are 1, 2, 3, 4, 5, 6, 8 and 9. Allowed values of μ are all integers such that $\mu \cdot K$ is 5, 6, 8 or 9.

Example:

Compute all LDP triangles of degree one and class group torsion order nine up to Markov depth five. These consist of the three series (1-9-2), (1-9-5) and (1-9-8) from Theorem 1.1 of [31]. Note that for depths zero and one (which correspond to solution triples $(1, 1, 1)$ and $(1, 1, 4)$ in the Markov tree), the series overlap, hence there are only one resp. two triangles in this case, which is also mentioned in the Theorem. After that, the number of triangles doubles with each additional step.

```
julia> Pss = classify_lattice_triangles_integral_degree(Val(1), Val(9), 5);

julia> length.(Pss)
6-element Vector{Int64}:
  1
  2
  3
  6
 12
 24
```

source

```
classify_lattice_triangles_integral_degree(::Val{K}, depth :: Int, T ::
↪  Type{<:Integer} = BigInt) where {K}
```

Return all LDP triangles (= fake weighted projective planes) with integral degree K, up to a given depth in the Markov tree. Essentially, this returns the triangles of the series (K - * - *) according to the notation of Theorem 1.1 of [31]. Allowed values of K are 1, 2, 3, 4, 5, 6, 8 and 9.

Example:

Print the numbers of LDP triangles with integral degree K, for all possible values of K, up to depth 10.

```
julia> [length.(classify_lattice_triangles_integral_degree(Val(K), 10)) for K in
↪  [1,2,3,4,5,6,8,9]]
8-element Vector{Vector{Int64}}:
 [10, 18, 35, 70, 140, 280, 560, 1120, 2240, 4480, 8960]
 [4, 6, 12, 24, 48, 96, 192, 384, 768, 1536, 3072]
 [2, 3, 5, 10, 20, 40, 80, 160, 320, 640, 1280]
 [1, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512]
 [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]
 [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]
 [1, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512]
 [1, 1, 1, 2, 4, 8, 16, 32, 64, 128, 256]
```

source

```
classify_lattice_triangles_integral_degree(depth :: Int, T :: Type{<:Integer} =
↪  BigInt)
```

Return all LDP triangles (= fake weighted projective planes) with integral degree, up to a given depth in the Markov tree. This returns the union of all 24 series classified in Theorem 1.1 of [31].

APPENDIX B

CStarSurfaces.jl

This appendix serves as a reference for `CStarSurfaces.jl`, a Julia package for computations with rational projective $\mathbb{C}^*$-surfaces using their combinatorial description. It is built on top of `RationalPolygons.jl`, which is described in Appendix A.

Let us give an impression of `CStarSurfaces.jl` by means of an example session. After loading in the package, we can create a `CStarSurface` by providing a defining triple as in Construction 3.1.1.

```
julia> using CStarSurfaces

julia> X = cstar_surface(PE, [[1,1],[4,2],[2]], [[-1,-2],[3,-3],[1]])
C*-surface of case PE with l = ((1,1),(4,2),2) and d = ((-1,-2),(3,-3),1)
```

Let us see its generator matrix and grading matrix:

```
julia> generator_matrix(X)
3×6 StaticArraysCore.SMatrix{3, 6, Int64, 18} with indices SOneTo(3)×SOneTo(6):
 -1  -1  4   2  0  0
 -1  -1  0   0  2  0
 -1  -2  3  -3  1  1

julia> grading_matrix_free_part(X)
3×6 StaticArraysCore.SMatrix{3, 6, Int64, 18} with indices SOneTo(3)×SOneTo(6):
 9  -9  -1  2  0  0
 6  -4   0  1  1  0
 8  -8  -1  2  0  1

julia> grading_matrix_torsion_part(X)
1×6 StaticArraysCore.SMatrix{1, 6, Int64, 6} with indices SOneTo(1)×SOneTo(6):
 0  0  1  1  0  0
```

Next, we compute basic global properties, like the class group, Gorenstein index, Picard index, log canonicity and degree:

```
julia> class_group(X)
Z^3 x Z/2

julia> gorenstein_index(X)
```

```
18

julia> picard_index(X)
864

julia> log_canonicity(X)
1//3

julia> degree(X)
14//9
```

Let us now consider the fixed points of the $\mathbb{C}^*$-surface. In this case, there is the elliptic fixed point x^-, the hyperbolic fixed points x_{01} and x_{11} and the parabolic fixed points x_0^+, x_1^+ and x_2^+:

```
julia> xs = fixed_points(X)
6-element Vector{FixedPoint}:
 x⁻
 x⁺₀
 x⁺₁
 x⁺₂
 x₀₁
 x₁₁
```

We can arrange their local properties into a pretty table as follows:

```
julia> using PrettyTables

julia> fs = [(X,x) -> x, class_group, gorenstein_index, log_canonicity, is_smooth,
↪  is_canonical];

julia> header = ["Point", "Class group", "Gorenstein index", "Log canonicity",
↪  "smooth?", "canonical?"];

julia> pretty_table([f(X,x) for x in xs, f in fs]; header)
┌─────────┬─────────────┬──────────────────┬──────────────────┬──────────┬──────────────┐
│   Point │ Class group │ Gorenstein index │ Log canonicity │ smooth? │ canonical? │
├─────────┼─────────────┼──────────────────┼──────────────────┼──────────┼──────────────┤
│      x⁻ │        Z/12 │                3 │            1//3 │    false │        false │
│    x⁺₀ │         Z/1 │                1 │            1//0 │     true │         true │
│    x⁺₁ │         Z/4 │                2 │            1//2 │    false │        false │
│    x⁺₂ │         Z/2 │                1 │            1//1 │    false │         true │
│    x₀₁ │         Z/1 │                1 │            1//0 │     true │         true │
│    x₁₁ │        Z/18 │                9 │            1//3 │    false │        false │
└─────────┴─────────────┴──────────────────┴──────────────────┴──────────┴──────────────┘
```

Here, a log canonicity value of `1//0` (infinity by Julia's convention) is used for smooth points.

This documentation is generated directly from the docstrings of the package's source code. An online version is available on its GitHub page [52]. Every documented item includes a clickable `source` link that directs to the corresponding line in the source code where it is defined. The content here refers to version `v1.0.0-thesis` of `CStarSurfaces.jl`. All example sessions have been tested and verified to work with this version.

B.1 Basic types

We briefly go over two basic types used in this package: `CStarSurfaceCase` and `AbelianGroup`.

B.1.1 Cases

Julia Type B.1.1. source

```
CStarSurfaceCase
```

An enum type with four values: `EE`, `PE`, `EP`, `PP`.

Julia Function B.1.2. source

```
invert_case(C :: CStarSurfaceCase, invert :: Bool = true)
```

Invert a case, see Definition 3.2.8. It sends `EE` to `EE`, `PE` to `EP`, `EP` to `PE` and `PP` to `PP`. If `invert` is set to false, it just returns the given case.

Julia Method B.1.3. source

```
has_elliptic_fixed_point_plus(C :: CStarSurfaceCase)
```

Return true if `C` is `EE` or `EP`, false otherwise.

Julia Method B.1.4. source

```
has_elliptic_fixed_point_minus(C :: CStarSurfaceCase)
```

Return true if `C` is `EE` or `PE`, false otherwise.

Julia Method B.1.5. source

```
has_parabolic_fixed_point_curve_plus(C :: CStarSurfaceCase)
```

Return true if `C` is `PP` or `PE`, false otherwise.

Julia Method B.1.6. source

```
has_parabolic_fixed_point_curve_minus(C :: CStarSurfaceCase)
```

Return true if `C` is `PP` or `EP`, false otherwise.

B.1.2 Abelian groups

Julia Type B.1.7. source

```
AbelianGroup{T <: Integer}
```

A struct representing a finitely generated abelian group. It consists of two fields:

- `rank :: T`: The rank of the group.
- `elementary_divisors :: T`: The elementary divisors, i.e. the orders of the torsion part.

Example:

Abelian groups are displayed using standard mathematical notation:

```
julia> AbelianGroup(3, [2,4])
Z^3 x Z/2 x Z/4
```

Julia Function B.1.8. source

```
rank(G :: AbelianGroup)
```

The rank of an abelian group.

Julia Function B.1.9. source

```
elementary_divisors(A :: AbstractMatrix{T}) where {T <: Integer}
```

Return the elementary divisors of A, i.e. the nonzero diagonal entries in the Smith normal form.

source

```
elementary_divisors(G :: AbelianGroup)
```

The elementary divisors of the abelian group, i.e. the orders of the torsion part.

Julia Function B.1.10. source

```
torsion_order(G :: AbelianGroup)
```

The order of the torsion part of the abelian group. This equals the product of its elementary divisors.

Julia Function B.1.11. source

```
torsion_part(G :: AbelianGroup)
```

The torsion part of an abelian group.

Julia Function B.1.12. source

```
cokernel(A :: AbstractMatrix)
```

Return the cokernel of a matrix as an abelian group by computing its Smith normal form.

B.2 $\mathbb{C}^*$-surfaces

This section documents the main functionality around $\mathbb{C}^*$-surfaces and their global properties. In Subsection B.2.1, we introduce the type `CStarSurface`, which is modeled by defining triples. Subsection B.2.2 goes over functions computing global invariants. Finally, Subsection B.2.3 is about the normal form.

B.2.1 The `CStarSurface` type

Recall from Section 3.1 that a $\mathbb{C}^*$-surface can be described by a defining triple $(\mathfrak{c}, l, d)$ together with a coefficient matrix $A \in \mathbb{C}^{2\times n}$. In `CStarSurfaces.jl`, we model only the defining triple and ignore the coefficient matrix. This is because almost all numerical invariants (Gorenstein index, Picard index, log canonicity etc.) only depend on the defining triple anyway, so there isn't much use in distinguishing between $\mathbb{C}^*$-surfaces having equivalent defining triples, but different coefficient matrices.

Since Julia's indexing of vectors is one-based, the convention for numbering the blocks used by `CStarSurfaces.jl` differs from the one used throughout Chapter 3. In particular, we write R for the number of blocks, which equals $r + 1$ in the notation of Construction 3.1.1. For some user-facing functions, like `l`, `d` and `block_sizes`, we add an offset to match the convention used in Chapter 3.

Like the polygon type in `RationalPolygons.jl`, we use statically sized matrices for the defining data of a $\mathbb{C}^*$-surface. This means the type will depend both on the number of blocks and the total number of rays $n = n_0 + \cdots + n_r$ as a type parameter.

Julia Type B.2.1. source

```
CStarSurface{T<:Integer,C,N,M,R}
```

The type of a $\mathbb{C}^*$ surface. It has the following type parameters:

- `T <: Integer`. The integer type to be used, e.g. `Int64` or `BigInt`.
- `C :: CStarSurfaceCase`. One of `EE`, `EP`, `PE` and `PP`. This describes the fixed point set of the $\mathbb{C}^*$-surface, i.e. the existence of elliptic fixed points and parabolic fixed point curves.
- `N :: Int`. The number of rays in the toric ambient variety. This equals $n_0 + \cdots + n_r$ in the notation of Section 3.1.
- `M :: Int`. This always equals $2N$.
- `R :: Int`. The number of arms of the $\mathbb{C}^*$-surface. This equals $r + 1$ in the notation of Section 3.1.

The type itself has two fields:

- `vertex_matrix :: SMatrix{2,N,T,M}`. This contains the main part of the data: A $2 \times N$ integral matrix encoding the entries l_{ij} and d_{ij} of the rays of the toric ambient. In the notation of Section 3.1, the vertex matrix has the form
$$\begin{bmatrix} l_{01} & \dots & l_{0n_0} & \dots & l_{r1} & \dots & l_{rn_r} \\ d_{01} & \dots & d_{0n_0} & \dots & d_{r1} & \dots & d_{rn_r} \end{bmatrix}.$$
- `block_sizes :: SVector{R, T}`. This determines which rays belong to which arm in the toric ambient variety. In the notation of Section 3.1, it is the tuple $(n_0, \dots, n_r)$.

Julia Function B.2.2. source

```
cstar_surface(C :: CStarSurfaceCase, l :: Vector{Vector{T}}, d :: Vector{Vector{T}})
↪  where {T <: Integer}
```

Construct a $\mathbb{C}^*$-surface from a defining triple. This function does not check whether the input data truly is a defining triple in the sense of Construction 3.1.1. In particular, it is the responsibility of the user to ensure that $\gcd(l_{ij}, d_{ij}) = 1$, that $\frac{d_{i1}}{l_{i1}} > \cdots > \frac{d_{in_i}}{l_{in_i}}$ and that the columns of the generator matrix generate $\mathbb{Q}^{r+1}$ as a convex cone.

Example:

```
julia> cstar_surface(EE, [[1,4],[3],[2]], [[-1,-5],[2],[1]])
C*-surface of case EE with l = ((1,4),3,2) and d = ((-1,-5),2,1)
```

Julia Method B.2.3. source

```
vertex_matrix(X :: CStarSurface)
```

The vertex matrix of a $\mathbb{C}^*$-surface. This encodes the rays in the toric ambient variety.

Example:

```
julia> X = cstar_surface(EE, [[1,4],[3],[2]], [[-1,-5],[2],[1]])
C*-surface of case EE with l = ((1,4),3,2) and d = ((-1,-5),2,1)

julia> vertex_matrix(X)
2×4 SMatrix{2, 4, Int64, 8} with indices SOneTo(2)×SOneTo(4):
  1   4  3  2
 -1  -5  2  1
```

Julia Function B.2.4. source

```
l(X :: CStarSurface, i :: Int, j :: Int)
```

Return the entry l_{ij} of the defining triple. By convention, the indexation of the blocks starts with zero, i.e. i goes from 0 to `number_of_blocks(X)-1`. The indexation of the rays in each individual block starts with one.

Julia Function B.2.5. source

```
d(X :: CStarSurface, i :: Int, j :: Int)
```

Return the entry d_{ij} of the defining triple. By convention, the indexation of the blocks starts with zero, i.e. i goes from 0 to `number_of_blocks(X)-1`. The indexation of the rays in each individual block starts with one.

Julia Function B.2.6. source

```
number_of_blocks(X :: CStarSurface)
```

The number of blocks (arms) of a $\mathbb{C}^*$-surface.

Julia Function B.2.7. source

```
block_sizes(X :: CStarSurface)
```

The sizes of the individual blocks of a $\mathbb{C}^*$-surface.

source

```
block_sizes(X :: CStarSurface, i :: Int)
```

The size of the i-th block of a $\mathbb{C}^*$-surface.

Julia Function B.2.8. source

```
case(X :: CStarSurface)
```

The case of the $\mathbb{C}^*$-surface, as a `CStarSurfaceCase`.

Julia Method B.2.9. source

```
has_elliptic_fixed_point_plus(X :: CStarSurface)
```

Whether the $\mathbb{C}^*$-surface has an elliptic fixed point x^+ as the source. This means the case is either `EE` or `EP`.

Julia Method B.2.10. source

```
has_elliptic_fixed_point_minus(X :: CStarSurface)
```

Whether the $\mathbb{C}^*$-surface has an elliptic fixed point x^- as the sink. This means the case is either `EE` or `PE`.

Julia Method B.2.11. source

```
has_parabolic_fixed_point_curve_plus(X :: CStarSurface)
```

Whether the $\mathbb{C}^*$-surface has a curve of parabolic fixed points as the source. This means the case is either `PE` or `PP`.

Julia Method B.2.12. source

```
has_parabolic_fixed_point_curve_minus(X :: CStarSurface)
```

Whether the $\mathbb{C}^*$-surface has a curve of parabolic fixed points as the sink. This means the case is either `EP` or `PP`.

Julia Function B.2.13. source

```
ray(X :: CStarSurface, i :: Int, j :: Int)
```

Return the j-th ray of the i-th block of the $\mathbb{C}^*$-surface. By convention, the indexation of the blocks starts with zero, i.e. i goes from 0 to `number_of_blocks(X)-1`. The indexation of the rays in each individual block starts with one.

This returns the ray as a vector with two entries. See also `embedded_ray` for the embedding into R-dimensional space, which is the actual ray of the ambient toric variety.

Example:

```
julia> X = cstar_surface(EE, [[1,4],[3],[2]], [[-1,-5],[2],[1]])
C*-surface of case EE with l = ((1,4),3,2) and d = ((-1,-5),2,1)

julia> ray(X,0,2)
2-element SVector{2, Int64} with indices SOneTo(2):
  4
 -5
```

Julia Function B.2.14. source

```
embedded_ray(X :: CStarSurface, i :: Int, j :: Int)
```

The j-th ray of the i-th block of the toric ambient variety. This is the embedded ray into R-dimensional space, where R is the number of blocks.

Example:

```
julia> X = cstar_surface(EE, [[1,4],[3],[2]], [[-1,-5],[2],[1]])
C*-surface of case EE with l = ((1,4),3,2) and d = ((-1,-5),2,1)

julia> embedded_ray(X,0,2)
3-element SVector{3, Int64} with indices SOneTo(3):
 -4
 -4
 -5
```

Julia Function B.2.15. source

```
generator_matrix(X :: CStarSurface)
```

The generator matrix of the ambient toric variety of X. The columns of this matrix are the primitive ray generators of the fan of the ambient toric variety. See Construction 3.1.1.

Example:

```
julia> X = cstar_surface(EE, [[1,4],[3],[2]], [[-1,-5],[2],[1]])
C*-surface of case EE with l = ((1,4),3,2) and d = ((-1,-5),2,1)

julia> generator_matrix(X)
3×4 SMatrix{3, 4, Int64, 12} with indices SOneTo(3)×SOneTo(4):
 -1  -4  3  0
 -1  -4  0  2
 -1  -5  2  1
```

Julia Function B.2.16. source

```
slope(X :: CStarSurface, i :: Int, j :: Int)
```

The slope of the j-th ray of the i-th block, i.e. d_{ij}/l_{ij}.

Julia Function B.2.17. source

```
sum_of_maximal_slopes(X :: CStarSurface)
```

The sum of the maximal slopes over all blocks.

Julia Function B.2.18. source

```
sum_of_minimal_slopes(X :: CStarSurface)
```

The sum of the minimal slopes over all blocks.

Julia Function B.2.19. source

```
l_plus(X :: CStarSurface)
```

The rational number $\ell^+ := \frac{1}{l_{01}} + \cdots + \frac{1}{l_{r1}} - r + 1$, see Definition 7.4 of [27]

Julia Function B.2.20. source

```
l_minus(X :: CStarSurface)
```

The rational number $\ell^- := \frac{1}{l_{0n_0}} + \cdots + \frac{1}{l_{rn_r}} - r + 1$, see Definition 7.4 of [27].

B.2.2 Global properties

We describe functions that compute global properties of $\mathbb{C}^*$-surfaces. For the `class_group`, `multiplicity` and `picard_index`, we refer to Section 3.3.1 and [27] for more details. For other properties, like `gorenstein_index`, `is_quasismooth`, `is_log_terminal`, `log_canonicity`, and `degree`, we refer to [27].

Julia Method B.2.21. source

```
class_group(X :: CStarSurface)
```

Return the divisor class group of X. See Definition 3.3.1.

Example:

```
julia> X = cstar_surface(PE, [[1,1],[3],[3]], [[-1,-3],[2],[2]])
C*-surface of case PE with l = ((1,1),3,3) and d = ((-1,-3),2,2)

julia> class_group(X)
Z^2 x Z/3
```

Julia Method B.2.22. source

```
multiplicity(X :: CStarSurface)
```

Return the order of the torsion part of the class group of X.

Julia Method B.2.23. source

```
grading_matrix_free_part(X :: CStarSurface)
```

Return the free part of the grading matrix associated to X. See Definition 3.3.1.

Example:

```
julia> X = cstar_surface(PE, [[1,1],[3],[3]], [[-1,-3],[2],[2]])
C*-surface of case PE with l = ((1,1),3,3) and d = ((-1,-3),2,2)

julia> grading_matrix_free_part(X)
2×5 SMatrix{2, 5, Int64, 10} with indices SOneTo(2)×SOneTo(5):
  5  1   2   2  0
 -3  0  -1  -1  1
```

Julia Method B.2.24. source

```
grading_matrix_torsion_part(X :: CStarSurface)
```

Return the torsion part of the grading matrix associated to X. See Definition 3.3.1.

Example:

```
julia> X = cstar_surface(PE, [[1,1],[3],[3]], [[-1,-3],[2],[2]])
C*-surface of case PE with l = ((1,1),3,3) and d = ((-1,-3),2,2)

julia> grading_matrix_torsion_part(X)
1×5 SMatrix{1, 5, Int64, 5} with indices SOneTo(1)×SOneTo(5):
 0  0  1  2  0
```

Julia Method B.2.25. source

```
grading_matrix(X :: CStarSurface)
```

Return a tuple (Q_0, Q_1) where Q_0 is the free part and Q_1 is the torsion part of the grading matrix associated to X. See Definition 3.3.1.

Julia Method B.2.26. source

```
picard_index(X :: CStarSurface)
```

Return the index of the Picard group inside the divisor class group of X. This equals the product of all local multiplicities divided by the global multiplicity, see Theorem 3.3.13.

Example:

```
julia> X = cstar_surface(PE, [[1,1],[3],[3]], [[-1,-3],[2],[2]])
C*-surface of case PE with l = ((1,1),3,3) and d = ((-1,-3),2,2)

julia> picard_index(X)
90
```

Julia Method B.2.27. source

```
gorenstein_index(X :: CStarSurface)
```

Return the Gorenstein index of X. This is the least common multiple of the local Gorenstein indices, see also gorenstein_index.

Example:

```
julia> X = cstar_surface(PE, [[1,1],[3],[3]], [[-1,-3],[2],[2]])
C*-surface of case PE with l = ((1,1),3,3) and d = ((-1,-3),2,2)

julia> gorenstein_index(X)
15
```

Julia Method B.2.28. source

```
is_quasismooth(X :: CStarSurface)
```

Check whether X is quasismooth, i.e. its characteristic space is smooth. See Summary 8.1 of [27] for a description in terms of the defining data.

Example:

```
julia> X = cstar_surface(PE, [[1,1],[3],[3]], [[-1,-3],[2],[2]])
C*-surface of case PE with l = ((1,1),3,3) and d = ((-1,-3),2,2)

julia> is_quasismooth(X)
true
```

Julia Method B.2.29. source

```
is_factorial(X :: CStarSurface)
```

Check whether X is locally factorial, i.e. all local class groups are trivial.

Example:

```
julia> X = cstar_surface(PE, [[1,1],[3],[3]], [[-1,-3],[2],[2]])
C*-surface of case PE with l = ((1,1),3,3) and d = ((-1,-3),2,2)

julia> is_factorial(X)
false
```

Julia Method B.2.30. source

```
is_smooth(X :: CStarSurface)
```

Check whether X is smooth. This is equivalent to being factorial and quasismooth.

Julia Method B.2.31. source

```
log_canonicity(X :: CStarSurface)
```

Return the maximal $\varepsilon > 0$ such that the surface has ε-log canonical singularities. For smooth surfaces, this returns `1//0`, which is infinity by Julia's convention.

Example:

```
julia> X = cstar_surface(PE, [[1,1],[3],[3]], [[-1,-3],[2],[2]])
C*-surface of case PE with l = ((1,1),3,3) and d = ((-1,-3),2,2)

julia> log_canonicity(X)
2//5
```

Julia Method B.2.32. source

```
is_log_canonical(X :: CStarSurface, ε :: Real = 0)
```

Check whether X is ε-log canonical.

Julia Method B.2.33. source

```
is_log_terminal(X :: CStarSurface, ε :: Real = 0)
```

Check whether X is ε-log terminal.

Julia Method B.2.34. source

```
degree(X :: CStarSurface)
```

The self-intersection number of an anticanonical divisor of X. See Proposition 7.9 of [27] for a formula in terms of the defining data.

B.2.3 Admissible operations and the normal form

We provide an implementation of the normal form for defining triples from Section 3.2.

Julia Function B.2.35. source

```
mfrak_plus(X :: CStarSurface)
```

The sum of the rounded down slopes $\lfloor m_{i1} \rfloor$, see Definition 3.2.20.

Julia Function B.2.36. source

```
mfrak_minus(X :: CStarSurface)
```

Minus the sum of the rounded up slopes $\lceil m_{in_i} \rceil$, see Definition 3.2.20.

Julia Function B.2.37. source

```
beta_plus(X :: CStarSurface)
```

The nested vector with entries $m_{ij} - \lfloor m_{i1} \rfloor$, see Definition 3.2.20. It is not yet sorted.

Julia Function B.2.38. source

```
beta_minus(X :: CStarSurface)
```

The nested vector with entries $\lceil m_{in_i} \rceil - m_{ij}$, see Definition 3.2.20. It is not yet sorted.

Julia Function B.2.39. source

```
are_equivalent(X :: CStarSurface{T}, Y :: CStarSurface{T}) where {T <: Integer}
```

Check whether two $\mathbb{C}^*$-surfaces are isomorphic, i.e. whether the defining triples are equivalent. This is done by comparing $(\mathfrak{m}^+, \beta^+)$ and $(\mathfrak{m}^-, \beta^-)$, see Lemma 3.2.21.

Julia Function B.2.40. source

```
orientation(X :: CStarSurface)
```

The orientation of a $\mathbb{C}^*$-surface, see Definition 3.2.22. This is either -1, 0, or 1.

Julia Function B.2.41. source

```
is_normal_form(X :: CStarSurface)
```

Check whether a (defining triple of a) $\mathbb{C}^*$-surface is in normal form. This holds if the following three conditions are satisfied, see Definition 3.2.24.

- The `orientation` is non-negative,
- `beta_plus` is sorted lexicographically,
- We have $0 \leq d_{i1} < l_{i1}$ for all $i = 1, \ldots, r$.

Julia Type B.2.42. source

```
AdmissibleOperation{T<:Integer,R}
```

An admissible operation of a $\mathbb{C}^*$-surface. It consists of three fields:

- `invert :: Bool`: Whether the operation contains an inversion,
- `perm :: SVector{R,Int}`: The permutation to apply to the blocks,
- `c :: SVector{R,T}`: The coefficients to apply in the addition.

See also Definition 3.2.8 and Lemma 3.2.9. Note that both `perm` and `c` are static vectors of length equal to the number of blocks. In particular, we require that $c_1 = -(c_2 + \cdots + c_r)$ (in contrast to Section 3.2.5, the indexation here is one-based).

Admissible operations can be applied to $\mathbb{C}^*$-surfaces using standard Julia call syntax. See for instance the example at `inversion`.

Julia Function B.2.43. source

```
admissible_operation(invert :: Bool, perm :: SVector{R,Int}, c :: SVector{S,T}) where
↪  {R,S,T<:Integer}
```

Construct an admissible operation from an inversion, a permutation and an addition. Here, we must have $S = R - 1$.

Julia Function B.2.44. source

```
inversion(R :: Int, T :: Type{<:Integer} = Int)
```

Return an inversion as an admissible operation, see Definition 3.2.8. It takes the number of blocks of the $\mathbb{C}^*$-surface and optionally an integer type as input.

Example:

```
julia> X = cstar_surface(PE, [[1,1],[2],[4]], [[-2,-3],[1],[3]])
C*-surface of case PE with l = ((1,1),2,4) and d = ((-2,-3),1,3)

julia> α = inversion(3)
AdmissibleOperation{Int64, 3}(true, [1, 2, 3], [0, 0, 0])

julia> generator_matrix(α(X))
3×5 SMatrix{3, 5, Int64, 15} with indices SOneTo(3)×SOneTo(5):
 -1  -1   2   0   0
 -1  -1   0   4   0
  3   2  -1  -3  -1
```

Julia Function B.2.45. source

```
permutation(perm :: SVector{R,Int}, T :: Type{<:Integer} = Int) where {R}
```

Return a permutation as an admissible operation, see Definition 3.2.8.

Example:

```
julia> X = cstar_surface(PE, [[1,1],[2],[4]], [[-2,-3],[1],[3]])
C*-surface of case PE with l = ((1,1),2,4) and d = ((-2,-3),1,3)

julia> α = permutation(@SVector [2,3,1])
AdmissibleOperation{Int64, 3}(false, [2, 3, 1], [0, 0, 0])

julia> generator_matrix(α(X))
3×5 SMatrix{3, 5, Int64, 15} with indices SOneTo(3)×SOneTo(5):
 -2  4   0   0  0
 -2  0   1   1  0
  1  3  -2  -3  1
```

Julia Function B.2.46. source

```
addition(c :: SVector{R,T}) where {R, T <: Integer}
```

Return an addition as an admissible operation, see Definition 3.2.8.

Example:

```
julia> X = cstar_surface(PE, [[1,1],[2],[4]], [[-2,-3],[1],[3]])
C*-surface of case PE with l = ((1,1),2,4) and d = ((-2,-3),1,3)

julia> α = addition(@SVector [2,-1])
AdmissibleOperation{Int64, 3}(false, [1, 2, 3], [-1, 2, -1])

julia> generator_matrix(α(X))
3×5 SMatrix{3, 5, Int64, 15} with indices SOneTo(3)×SOneTo(5):
 -1  -1  2   0  0
 -1  -1  0   4  0
 -3  -4  5  -1  1
```

Julia Function B.2.47. source

```
normal_form_with_operation(X :: CStarSurface)
```

Return a pair (Y, ψ), where ψ is an admissible operation turning X into normal form and $Y = \psi(X)$, see Proposition 3.2.25

Example:

```
julia> X = cstar_surface(EP, [[2],[1,1],[4]], [[-5],[2,1],[9]])
C*-surface of case EP with l = (2,(1,1),4) and d = (-5,(2,1),9)

julia> Y, α = normal_form_with_operation(X)
(C*-surface of case PE with l = ((1,1),2,4) and d = ((-2,-3),1,3) ,
↪  AdmissibleOperation{Int64, 3}(true, [2, 1, 3], [-1, -2, 3]))

julia> α(X) == Y
true
```

Julia Function B.2.48. source

```
normal_form(X :: CStarSurface)
```

Return the normal form of a defining triple of a $\mathbb{C}^*$-surface, see Proposition 3.2.25.

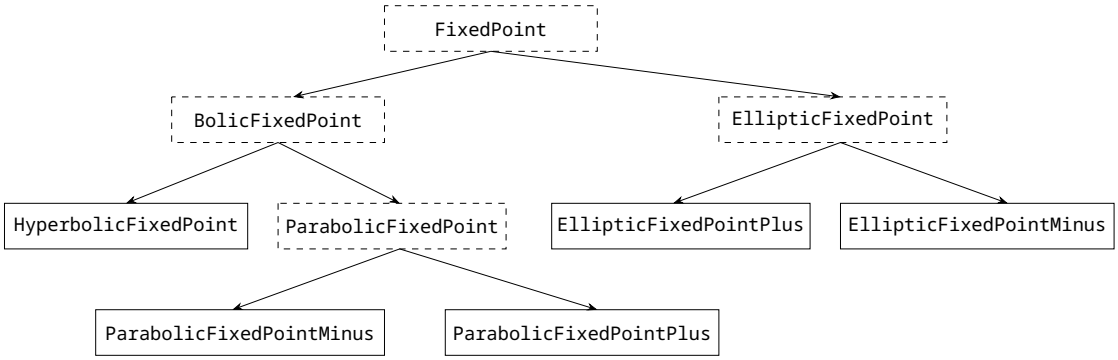
B.3 Fixed points and local properties

Recall that there are three kinds of fixed points of $\mathbb{C}^*$-surfaces: elliptic, hyperbolic and parabolic. In `CStarSurfaces.jl`, we model fixed points by a type hierarchy with an abstract type `FixedPoint` at the top. This allows us to implement local properties like `gorenstein_index` and `log_canonicity` separately for each type of fixed point using Julia's dispatch mechanism. We found that sometimes, hyperbolic and parabolic fixed point can be treated uniformly, while elliptic fixed points need to be treated differently. Hence we will refer to hyperbolic and parabolic fixed points collectively as *bolic* fixed points.

In Subsection B.3.1, we go over the type hierarchy of fixed points. In Subsection B.3.2, we discuss local properties.

B.3.1 Fixed points

The following graph summarizes the different fixed point types and their subtype relations. Abstract types have dashed boundaries.



Julia Type B.3.1. source

```
FixedPoint
```

Abstract supertype of a fixed point on a $\mathbb{C}^*$-surface. Here, a fixed point is understood to be a formal symbol associated to the defining triple, see Definition 3.3.6.

This type has the two subtypes `EllipticFixedPoint` and `BolicFixedPoint`.

Julia Type B.3.2. source

```
EllipticFixedPoint <: FixedPoint
```

Abstract supertype of elliptic fixed points of a $\mathbb{C}^*$-surface. It has two subtypes `EllipticFixedPointPlus` and `EllipticFixedPointMinus`.

Julia Type B.3.3. source

```
EllipticFixedPointPlus <: EllipticFixedPoint
```

The elliptic fixed point x^+. This struct has no fields.

Julia Type B.3.4. source

```
EllipticFixedPointMinus <: EllipticFixedPoint
```

The elliptic fixed point x^-. This struct has no fields.

Julia Type B.3.5. source

```
BolicFixedPoint <: FixedPoint
```

Abstract supertype of `ParabolicFixedPoint` and `HyperbolicFixedPoint`.

Julia Type B.3.6. source

```
HyperbolicFixedPoint <: BolicFixedPoint
```

A struct describing a hyperbolic fixed point x_{ij} of a $\mathbb{C}^*$-surface. It has two fields `i :: Int` and `j :: Int`, which are the indices of the block and the ray inside the block that this fixed point is associated to.

Julia Type B.3.7. source

```
ParabolicFixedPoint <: BolicFixedPoint
```

Abstract supertype of parabolic fixed points of a $\mathbb{C}^*$-surface. It has two subtypes `ParabolicFixedPointPlus` and `ParabolicFixedPointMinus`.

Julia Type B.3.8. source

```
ParabolicFixedPointPlus <: ParabolicFixedPoint
```

A parabolic fixed point x_i^+. It has one field `i :: Int`, which is the index of the arm of the fixed point.

Julia Type B.3.9. source

```
ParabolicFixedPointMinus <: ParabolicFixedPoint
```

A parabolic fixed point x_i^-. It has one field `i :: Int`, which is the index of the arm of the fixed point.

Julia Function B.3.10. source

```
elliptic_fixed_points(X :: CStarSurface)
```

Return all elliptic fixed points of X, see Definition 3.3.6.

Julia Function B.3.11. source

```
hyperbolic_fixed_points(X :: CStarSurface)
```

Return all hyperbolic fixed points of X, see Definition 3.3.6.

Julia Function B.3.12. source

```
parabolic_fixed_points(X :: CStarSurface)
```

Return all parabolic fixed points of X, see Definition 3.3.6.

Julia Function B.3.13. source

```
bolic_fixed_points(X :: CStarSurface)
```

Return all bolic (hyperbolic and parabolic) fixed point of X, see Definition 3.3.6.

Julia Function B.3.14. source

```
fixed_points(X :: CStarSurface)
```

Return all fixed points of X, see Definition 3.3.6.

Example:

```
julia> X = cstar_surface(PE, [[1,1],[1,1],[2]], [[-3,-4],[0,-1],[1]])
C*-surface of case PE with l = ((1,1),(1,1),2) and d = ((-3,-4),(0,-1),1)

julia> fixed_points(X)
6-element Vector{FixedPoint}:
 X⁻
 X⁺₀
 X⁺₁
 X⁺₂
 X₀₁
 X₁₁
```

B.3.2 Local properties

Julia Function B.3.15. source

```
toric_chart(X :: CStarSurface, x :: FixedPoint)
```

Return the generator matrix of the toric chart around a fixed point. This is the local generator matrix, as in Definition 3.3.7.

Example:

```
julia> X = cstar_surface(PE, [[1,1],[1,1],[2]], [[-3,-4],[0,-1],[1]])
C*-surface of case PE with l = ((1,1),(1,1),2) and d = ((-3,-4),(0,-1),1)

julia> toric_chart(X, EllipticFixedPointMinus())
3×3 SMatrix{3, 3, Int64, 9} with indices SOneTo(3)×SOneTo(3):
 -1   1  0
 -1   0  2
 -4  -1  1
```

Julia Function B.3.16. source

```
class_group(X :: CStarSurface, x :: FixedPoint)
```

Return the local class group at the point x.

Example:

```
julia> X = cstar_surface(PE, [[1,1],[1,1],[2]], [[-3,-4],[0,-1],[1]])
C*-surface of case PE with l = ((1,1),(1,1),2) and d = ((-3,-4),(0,-1),1)

julia> class_group(X, EllipticFixedPointMinus())
Z/9
```

Julia Function B.3.17. source

```
multiplicity(X :: CStarSurface, x :: FixedPoint)
```

Return the order of the local class group at the point x, see Definition 3.3.8.

Example:

```
julia> X = cstar_surface(PE, [[1,1],[1,1],[2]], [[-3,-4],[0,-1],[1]])
C*-surface of case PE with l = ((1,1),(1,1),2) and d = ((-3,-4),(0,-1),1)

julia> multiplicity(X, EllipticFixedPointMinus())
9
```

Julia Function B.3.18. source

```
is_factorial(X :: CStarSurface, x :: FixedPoint)
```

Check whether a fixed point is factorial, i.e. its multiplicity is one.

Example:

```
julia> X = cstar_surface(PE, [[1,1],[1,1],[2]], [[-3,-4],[0,-1],[1]])
C*-surface of case PE with l = ((1,1),(1,1),2) and d = ((-3,-4),(0,-1),1)

julia> filter(x -> is_factorial(X,x), fixed_points(X))
4-element Vector{FixedPoint}:
 x⁺₀
 x⁺₁
 x₀,₁
 x₁,₁
```

Julia Function B.3.19. source

```
is_quasismooth(X :: CStarSurface, x :: FixedPoint)
```

Check whether a fixed point is quasismooth. Bolic fixed points are always quasismooth. See Summary 8.1 of [27].

Example:

```
julia> X = cstar_surface(PE, [[1,1],[1,1],[2]], [[-3,-4],[0,-1],[1]])
C*-surface of case PE with l = ((1,1),(1,1),2) and d = ((-3,-4),(0,-1),1)

julia> all(x -> is_quasismooth(X,x), fixed_points(X))
true
```

Julia Function B.3.20. source

```
is_smooth(X :: CStarSurface, x :: FixedPoint)
```

Check whether a fixed point is smooth. This is equivalent to being factorial and quasismooth.

Example:

```
julia> X = cstar_surface(PE, [[1,1],[1,1],[2]], [[-3,-4],[0,-1],[1]])
C*-surface of case PE with l = ((1,1),(1,1),2) and d = ((-3,-4),(0,-1),1)

julia> filter(x -> is_smooth(X,x), fixed_points(X))
4-element Vector{FixedPoint}:
```

```
x⁺₀
x⁺₁
x₀₁
x₁₁
```

Julia Function B.3.21. source

```
gorenstein_index(X :: CStarSurface, x :: FixedPoint)
```

Return the local Gorenstein index at the point x. See Propositions 8.8 and 8.9 of [27] for their formulas in terms of defining data.

Example:

```
julia> X = cstar_surface(PE, [[1,1],[1,1],[2]], [[-3,-4],[0,-1],[1]])
C*-surface of case PE with l = ((1,1),(1,1),2) and d = ((-3,-4),(0,-1),1)

julia> gorenstein_index(X, EllipticFixedPointMinus())
3
```

Julia Function B.3.22. source

```
log_canonicity(X :: CStarSurface, x :: FixedPoint)
```

Return the maximal $\varepsilon > 0$ such that the singularity at x is ε-log canonical. By definition, this is set to be `1//0` (infinity) for smooth points.

Example:

```
julia> X = cstar_surface(PE, [[1,1],[1,1],[2]], [[-3,-4],[0,-1],[1]])
C*-surface of case PE with l = ((1,1),(1,1),2) and d = ((-3,-4),(0,-1),1)

julia> [log_canonicity(X,x) for x in fixed_points(X)]
6-element Vector{Rational{Int64}}:
 1//3
 1//0
 1//0
  1
 1//0
 1//0
```

Julia Function B.3.23. source

```
is_log_terminal(X :: CStarSurface, x :: FixedPoint, ε :: Real = 0)
```

Check whether a point on a $\mathbb{C}^*$-surface is ε-log terminal.

Julia Function B.3.24. source

```
is_log_canonical(X :: CStarSurface, x :: FixedPoint, ε :: Real = 0)
```

Check whether a point on a $\mathbb{C}^*$-surface is ε-log canonical.

Julia Function B.3.25. source

```
is_terminal(X :: CStarSurface, x :: FixedPoint)
```

Check whether a point on a $\mathbb{C}^*$-surface is terminal. This is equivalent to being smooth.

Example:

```
julia> X = cstar_surface(PE, [[1,1],[1,1],[2]], [[-3,-4],[0,-1],[1]])
C*-surface of case PE with l = ((1,1),(1,1),2) and d = ((-3,-4),(0,-1),1)

julia> filter(x -> is_terminal(X,x), fixed_points(X))
4-element Vector{FixedPoint}:
 x⁺₀
 x⁺₁
 x₀₁
 x₁₁
```

Julia Function B.3.26. source

```
is_canonical(X :: CStarSurface, x :: FixedPoint)
```

Check whether a point on a $\mathbb{C}^*$-surface is canonical.

Example:

```
julia> X = cstar_surface(PE, [[1,1],[1,1],[2]], [[-3,-4],[0,-1],[1]])
C*-surface of case PE with l = ((1,1),(1,1),2) and d = ((-3,-4),(0,-1),1)

julia> filter(x -> is_canonical(X,x), fixed_points(X))
5-element Vector{FixedPoint}:
 x⁺₀
 x⁺₁
 x⁺₂
 x₀₁
 x₁₁
```

B.4 Classifications

We provide implementations of the classification algorithms by Picard index from Section 3.4.

Julia Function B.4.1. source

```
decompositions(n :: T, len :: Int) where {T <: Integer}
```

Return all decompositions of n into a product of `len` positive integers.

Julia Function B.4.2. source

```
is_almost_free(xs :: AbstractVector)
```

Check whether a given n-element vector describes an almost free weight system, i.e. any $n-1$ of the element are coprime.

Julia Type B.4.3. source

```
VarBound{T <: Integer}
```

A simple struct with two fields `val :: T` and `eq :: Bool`. If `eq` is true, it is understood to be the condition that a variable equals `val`. If `eq` is false, it is the condition that a variable is at least `val`. This is used to encode the different subcases of the classification, see Table 3.4.

See also the constructors `eq` and `geq`.

Julia Function B.4.4. source

```
eq(x :: Integer)
```

A constructor of `VarBound` for an equality.

Julia Function B.4.5. source

```
geq(x :: Integer)
```

A constructor of `VarBound` for a lower bound.

Julia Function B.4.6. source

```
classify_ee_from_l0(p :: T, l01 :: T, l02 :: T, ls_bounds :: VarBound{T}...) where {T
↪  <: Integer}
```

An implementation of Algorithm 3.4.3. For the entries $l_1, \ldots, l_r$, one can provide either lower bounds or fixed values with `VarBound`.

Julia Function B.4.7. source

```
classify_ee_from_ls(p :: T, l01_bound :: VarBound{T}, l02_bound :: VarBound{T}, ls ::
↪  T...) where {T <: Integer}
```

An implementation of Algorithm 3.4.4. For the entries l_{01} and l_{02}, one can provide either lower bounds or fixed values with `VarBound`.

Julia Function B.4.8. source

```
classify_pe(p :: T, ls_bounds :: VarBound{T}...) where {T <: Integer}
```

An implementation of Algorithm 3.4.6. For the entries $l_0, \ldots, l_r$, one can provide either lower bounds or fixed values with `VarBound`.

Julia Type B.4.9. source

```
SingularityType
```

An enum type with values `eAeA`, `eAeD`, `eDeD`, `eAeE`, `eDeE`, `eEeE`, `eDp`, `eEp`. This encodes the possible singularity types in the classification of log del Pezzo $\mathbb{C}^*$-surfaces of Picard number one, see Table 3.4.

Julia Function B.4.10. source

```
classify_by_picard_index(p :: T, ::Val{<:SingularityType}) where {T <: Integer}
classify_by_picard_index(p :: T) where {T <: Integer}
```

Return all log del Pezzo $\mathbb{C}^*$-surfaces of Picard number one with Picard index p. If a `SingularityType` is provided, only surfaces of this singularity type are computed.

Example:

There are 7137 surfaces of Picard index at most 500. There are no surfaces of Picard index 31 or 127, see also Theorem 3.4.12.

```
julia> Xss = classify_by_picard_index.(1:500);

julia> sum(length.(Xss))
7137

julia> filter(i -> isempty(Xss[i]), 1 : 500)
2-element Vector{Int64}:
  31
 127
```

Acknowledgements

This dissertation would not have been possible without the support of many people, to whom I am deeply grateful. I would like to thank:

- My advisor, Prof. Dr. Jürgen Hausen, for his thoughtful guidance and advice, and his steady support throughout all phases of my studies,
- Prof. Dr. Ivo Radloff, for agreeing to be the second referee of this thesis,
- Everyone in the algebra group, for the kind and light-hearted atmosphere that made the past three years truly enjoyable,
- My family, for their encouragement and unconditional emotional support,
- My partner, Yuhan, for her love and care, and for always bringing out the best in me.

References

[1] Valery Alexeev. "Boundedness and K^2 for log surfaces". In: *Internat. J. Math.* 5.6 (1994), pp. 779–810. ISSN: 0129-167X,1793-6519. DOI: 10.1142/S0129167X94000395 (cited on page 1).

[2] Ivan Arzhantsev, Ulrich Derenthal, Jürgen Hausen, and Antonio Laface. *Cox Rings*. Vol. 144. Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge, 2015, pp. viii+530. ISBN: 978-1-107-02462-5 (cited on pages 1, 5, 49, 50, 55, 56, 88, 90, 91, 99, 100, 103, 105, 127).

[3] Ivan Arzhantsev, Jürgen Hausen, Elaine Herppich, and Alvaro Liendo. "The automorphism group of a variety with torus action of complexity one". In: *Mosc. Math. J.* 14.3 (2014), pp. 429–471, 641. ISSN: 1609-3321,1609-4514. DOI: 10.17323/1609-4514-2014-14-3-429-471 (cited on page 100).

[4] Gennadiy Averkov, Jan Krümpelmann, and Stefan Weltge. "Notions of maximality for integral lattice-free polyhedra: the case of dimension three". In: *Math. Oper. Res.* 42.4 (2017), pp. 1035–1062. ISSN: 0364-765X,1526-5471. DOI: 10.1287/moor.2016.0836 (cited on pages 10, 24, 26).

[5] Gennadiy Averkov, Christian Wagner, and Robert Weismantel. "Maximal lattice-free polyhedra: finiteness and an explicit description in dimension three". In: *Math. Oper. Res.* 36.4 (2011), pp. 721–742. ISSN: 0364-765X. DOI: 10.1287/moor.1110.0510 (cited on pages 3, 9, 10).

[6] Gabriele Balletti. "Enumeration of lattice polytopes by their volume". In: *Discrete Comput. Geom.* 65.4 (2021), pp. 1087–1122. ISSN: 0179-5376,1432-0444. DOI: 10.1007/s00454-020-00187-y (cited on page 10).

[7] Gabriele Balletti and Alexander Kasprzyk. *Three-dimensional lattice polytopes with two interior lattice points*. 2016. arXiv: 1612.08918 [math.CO] (cited on page 10).

[8] Victor Batyrev. "Canonical models of toric hypersurfaces". In: *Algebr. Geom.* 10.4 (2023), pp. 394–431. ISSN: 2313-1691,2214-2584 (cited on page 41).

[9] Victor Batyrev. "Higher-dimensional toric varieties with ample anticanonical class". Russian. PhD thesis. Moscow State University, 1985 (cited on page 10).

[10] Victor Batyrev. "The stringy Euler number of Calabi-Yau hypersurfaces in toric varieties and the Mavlyutov duality". In: *Pure Appl. Math. Q.* 13.1 (2017), pp. 1–47. ISSN: 1558-8599,1558-8602. DOI: 10.4310/PAMQ.2017.v13.n1.a1 (cited on page 41).

[11] Andreas Bäuerle. "Sharp volume and multiplicity bounds for Fano simplices". In: *Journal of Algebraic Combinatorics* 61.9 (2025). DOI: 10.1007/s10801-024-01366-3 (cited on pages 1, 66, 70, 74, 76, 178, 199, 201).

[12] Matthias Beck and Sinai Robins. *Computing the continuous discretely*. 2nd ed. Undergraduate Texts in Mathematics. Springer, New York, 2015, pp. xx+285. ISBN: 978-1-4939-2968-9. DOI: 10.1007/978-1-4939-2969-6 (cited on pages 15, 46).

[13] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. "Julia: A fresh approach to numerical computing". In: *SIAM Review* 59.1 (Sept. 2017), pp. 65–98. DOI: 10.1137/141000671 (cited on pages 7, 9).

[14] Martin Bohnert. *Area bounds for planar convex bodies containing a fixed number of interior integral points*. 2023. arXiv: 2305.11485 [math.MG] (cited on pages 12, 25, 28, 34, 38, 45, 164, 166–168).

[15] Martin Bohnert. *Lattice 3-polytopes of lattice width 2 and corresponding toric hypersurfaces*. 2025. arXiv: 2412.17545 [math.AG] (cited on pages 12, 41, 42, 46).

[16] Martin Bohnert and Justus Springer. *Classifying rational polygons with small denominator and few interior lattice points*. 2024. arXiv: 2410.17244 [math.CO] (cited on page 9).

[17] Martin Bohnert and Justus Springer. *Generalizations of Scott's inequality and Pick's formula to rational polygons*. 2024. arXiv: 2411.11187 [math.CO] (cited on page 9).

[18] Gavin Brown and Alexander Kasprzyk. "Small polygons and toric codes". In: *J. Symbolic Comput.* 51 (2013), pp. 55–62. ISSN: 0747-7171,1095-855X. DOI: 10.1016/j.jsc.2012.07.001 (cited on pages 17, 20, 178, 197).

[19] Wouter Castryck. "Moving out the edges of a lattice polygon". In: *Discrete Comput. Geom.* 47.3 (2012), pp. 496–518. ISSN: 0179-5376,1432-0444. DOI: 10.1007/s00454-011-9376-2 (cited on pages 3, 10, 12, 21, 22, 41, 178, 192, 194).

[20] David A. Cox, John B. Little, and Henry K. Schenck. *Toric varieties*. Vol. 124. Graduate Studies in Mathematics. American Mathematical Society, Providence, RI, 2011, pp. xxiv+841. ISBN: 978-0-8218-4819-7. DOI: 10.1090/gsm/124 (cited on pages 19, 49, 50, 53, 59, 60, 68, 172–174).

[21] Jonathan Fine. "Resolution and completion of algebraic varieties". Unpublished. PhD thesis. University of Warwick, June 1983. URL: http://webcat.warwick.ac.uk/record=b1464811~S1 (cited on page 41).

[22] William Fulton. *Introduction to toric varieties*. Vol. 131. Annals of Mathematics Studies. The William H. Roever Lectures in Geometry. Princeton University Press, Princeton, NJ, 1993, pp. xii+157. ISBN: 978-0-6910-0049-7. DOI: 10.1515/9781400882526 (cited on pages 49, 61).

[23] R.L. Graham. "An efficient algorith for determining the convex hull of a finite planar set". In: *Information Processing Letters* 1.4 (1972), pp. 132–133. ISSN: 0020-0190. URL: https://www.sciencedirect.com/science/article/pii/0020019072900452 (cited on pages 16, 141).

[24] Roland Grinis and Alexander Kasprzyk. *Normal forms of convex lattice polytopes*. 2013. arXiv: 1301.6641 [math.CO] (cited on page 15).

[25] Girtrude Hamm, Johannes Hofscheier, and Alexander Kasprzyk. *Classification and Ehrhart Theory of Denominator 2 Polygons*. 2024. arXiv: 2411.19183 [math.CO] (cited on page 46).

[26] Daniel Hättig, Beatrice Hafner, Jürgen Hausen, and Justus Springer. "Del Pezzo surfaces of Picard number one admitting a torus action". In: *Annali di Matematica Pura ed Applicata* (2025). DOI: 10.1007/s10231-025-01552-5 (cited on pages 66, 72, 132).

[27] Daniel Hättig, Jürgen Hausen, and Justus Springer. "Classifying log del Pezzo surfaces with torus action". In: *Revista Matemática Complutense* (2025). DOI: 10.1007/s13163-025-00526-8 (cited on pages 2, 10, 28, 30, 58, 92, 132, 183, 216, 218, 220, 228, 229).

[28] Jürgen Hausen and Elaine Herppich. "Factorially graded rings of complexity one". In: *Torsors, Étale Homotopy and Applications to Rational Points*. London Mathematical Society Lecture Note Series. Cambridge University Press, 2013, pp. 414–428. DOI: 10.1017/CBO9781139525350.011 (cited on pages 1, 84).

[29] Jürgen Hausen, Elaine Herppich, and Hendrik Süß. "Multigraded Factorial Rings and Fano Varieties with Torus Action". In: *Documenta Math.* 16.3 (2011), pp. 71–109. ISSN: 1431-0635. DOI: 10.4171/DM/327 (cited on page 5).

[30] Jürgen Hausen, Simon Keicher, and Antonio Laface. "Computing Cox rings". In: *Math. Comp.* 85.297 (2016), pp. 467–502. ISSN: 0025-5718,1088-6842. DOI: 10.1090/mcom/2989 (cited on page 127).

[31] Jürgen Hausen and Katharina Király. $\mathbb{K}^*$*-surfaces of Picard number one and integral degree*. 2024. arXiv: 2411.15079 [math.AG] (cited on pages 178, 202–206).

[32] Jürgen Hausen and Hendrik Süß. "The Cox ring of an algebraic variety with torus action". In: *Adv. Math.* 225.2 (2010), pp. 977–1012. ISSN: 0001-8708,1090-2082. DOI: 10.1016/j.aim.2010.03.010 (cited on pages 1, 84).

[33] Jürgen Hausen and Milena Wrobel. "Non-complete rational T-varieties of complexity one". In: *Math. Nachr.* 290.5-6 (2017), pp. 815–826. ISSN: 0025-584X,1522-2616. DOI: 10.1002/mana.201600009 (cited on pages 1, 84).

[34] Andrew John Herrmann. "Classification of Ehrhart quasi-polynomials of half-integral polygons". MA thesis. San Francisco State University, 2010. URL: https://matthbeck.github.io/teach/masters/andrewh.pdf (cited on page 46).

[35] Alexander Kasprzyk. "Canonical toric Fano threefolds". In: *Canad. J. Math.* 62.6 (2010), pp. 1293–1309. ISSN: 0008-414X,1496-4279. DOI: 10.4153/CJM-2010-070-3 (cited on page 10).

[36] Alexander Kasprzyk, Maximilian Kreuzer, and Benjamin Nill. "On the combinatorial classification of toric log del Pezzo surfaces". In: *LMS J. Comput. Math.* 13 (2010), pp. 33–46. ISSN: 1461-1570. DOI: 10.1112/S1461157008000387 (cited on pages 1, 30, 66, 178, 198, 199).

[37] A. G. Khovanskii. "Newton polytopes, curves on toric surfaces, and inversion of Weil's theorem". In: *Uspekhi Mat. Nauk* 52.6(318) (1997), pp. 113–142. ISSN: 0042-1316,2305-2872. DOI: 10.1070/RM1997v052n06ABEH002156 (cited on page 10).

[38] R. J. Koelman. "The number of moduli of families of curves on toric surfaces". PhD thesis. University of Nijmegen, 1991. URL: https://hdl.handle.net/2066/113957 (cited on pages 10, 21, 23, 24, 34, 35, 178, 188, 190).

[39] Maximilian Kreuzer and Harald Skarke. "Complete classification of reflexive polyhedra in four dimensions". In: *Adv. Theor. Math. Phys.* 4.6 (2000), pp. 1209–1230. ISSN: 1095-0761,1095-0753. DOI: 10.4310/ATMP.2000.v4.n6.a2 (cited on page 10).

[40] Maximilian Kreuzer and Harald Skarke. "PALP: a package for analysing lattice polytopes with applications to toric geometry". In: *Comput. Phys. Comm.* 157.1 (2004), pp. 87–106. ISSN: 0010-4655. DOI: 10.1016/S0010-4655(03)00491-0 (cited on page 15).

[41] Jeffrey C. Lagarias and Günter M. Ziegler. "Bounds for lattice polytopes containing a fixed number of interior points in a sublattice". In: *Canad. J. Math.* 43.5 (1991), pp. 1022–1035. ISSN: 0008-414X,1496-4279. DOI: 10.4153/CJM-1991-058-4 (cited on pages 3, 9).

[42] Morris Newman. *Integral matrices*. Vol. 45. Pure and Applied Mathematics. Academic Press, New York-London, 1972, pp. xvii+224. ISBN: 978-0-0808-7358-9 (cited on page 55).

[43] OEIS Foundation Inc. *The On-Line Encyclopedia of Integer Sequences*. Published electronically at http://oeis.org. 2025 (cited on pages 35, 75, 188, 190, 192, 194, 198, 200).

[44] Peter Orlik and Philip Wagreich. "Algebraic surfaces with k^*-action". In: *Acta Math.* 138.1-2 (1977), pp. 43–81. ISSN: 0001-5962 (cited on page 83).

[45] Peter Orlik and Philip Wagreich. "Isolated singularities of algebraic surfaces with $\mathbb{C}^*$-action". In: *Ann. of Math. (2)* 93 (1971), pp. 205–228. ISSN: 0003-486X (cited on page 83).

[46] Peter Orlik and Philip Wagreich. "Singularities of algebraic surfaces with $\mathbb{C}^*$-action". In: *Math. Ann.* 193 (1971), pp. 121–135. ISSN: 0025-5831 (cited on page 83).

[47] G. Pick. "Geometrisches zur Zahlenlehre". In: *Lotos - Zeitschrift für Naturwissenschaften* 47 (1899), pp. 311–319 (cited on pages 4, 10, 15).

[48] Stanley Rabinowitz. "A census of convex lattice polygons with at most one interior lattice point". In: *Ars Combin.* 28 (1989), pp. 83–96. ISSN: 0381-7032,2817-5204 (cited on page 10).

[49] Miles Reid. "Young person's guide to canonical singularities". In: *Algebraic geometry, Bowdoin, 1985 (Brunswick, Maine, 1985)*. Vol. 46, Part 1. Proc. Sympos. Pure Math. Amer. Math. Soc., Providence, RI, 1987, pp. 345–414. ISBN: 978-0-8218-1476-5. DOI: 10.1090/pspum/046.1/927963 (cited on page 41).

[50] Josef Schicho. "Simplification of surface parametrizations—a lattice polygon approach". In: vol. 36. 3-4. International Symposium on Symbolic and Algebraic Computation (ISSAC'2002) (Lille). 2003, pp. 535–554. DOI: 10.1016/S0747-7171(03)00094-4 (cited on page 10).

[51] P. R. Scott. "On convex lattice polygons". In: *Bull. Austral. Math. Soc.* 15.3 (1976), pp. 395–399. ISSN: 0004-9727. DOI: 10.1017/S0004972700022826 (cited on pages 4, 10).

[52] Justus Springer. *CStarSurfaces.jl*. Version 1.0.0-thesis. Oct. 2025. URL: https://github.com/justus-springer/CStarSurfaces.jl/tree/v1.0.0-thesis (cited on pages 7, 209).

[53] Justus Springer. *LDP quadrilaterals with Gorenstein index at most 500*. Version 1.0.0. Zenodo, Oct. 2025. DOI: 10.5281/zenodo.17287820 (cited on page 66).

[54] Justus Springer. *LDP triangles by Picard index at most 1000000*. Version 1.0.0. Zenodo, Oct. 2025. DOI: 10.5281/zenodo.17287717 (cited on page 65).

[55] Justus Springer. *RationalPolygons.jl*. Version 1.2.0-thesis. Oct. 2025. URL: https://github.com/justus-springer/RationalPolygons.jl/tree/v1.2.0-thesis (cited on pages 7, 9, 138).

[56] Justus Springer. "The Picard index of a surface with torus action". In: *Collect. Math.* 76.3 (2025), pp. 515–544. ISSN: 0010-0757,2038-4815. DOI: 10.1007/s13348-024-00443-x (cited on pages 51, 61, 64, 104, 128, 186).

[57] Justus Springer and Martin Bohnert. *3-rational polygons with few interior lattice points*. Version 1.0.0. Zenodo, Oct. 2024. DOI: 10.5281/zenodo.13960791 (cited on page 9).

[58] Justus Springer and Martin Bohnert. *Half-integral polygons with few interior lattice points*. Version 1.0.0. Zenodo, Oct. 2024. DOI: 10.5281/zenodo.13928298 (cited on pages 9, 24, 42).

[59] Justus Springer and Martin Bohnert. *Lattice polygons with at most 70 lattice points*. Version 1.1.0. Zenodo, Oct. 2024. DOI: 10.5281/zenodo.13959996 (cited on pages 9, 35).

[60] Justus Springer and Martin Bohnert. *Lattice subpolygons of a square with given sidelength*. Version 1.0.0. Zenodo, Sept. 2024. DOI: 10.5281/zenodo.13838476 (cited on pages 9, 20).

[61] Justus Springer and Martin Bohnert. *Rational polygons with exactly one interior lattice point*. Version 1.0.0. Zenodo, Sept. 2024. DOI: 10.5281/zenodo.13839216 (cited on pages 9, 30).

[62] Justus Springer and Martin Bohnert. *Rational polygons with no interior lattice points*. Version 1.0.0. Zenodo, Sept. 2024. DOI: 10.5281/zenodo.13838991 (cited on pages 9, 26).