Dissertation
zur
Erlagung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

A Toolbox to Compute the Cohomology of Arithmetic Groups in Case of the Group $\mathrm{Sp}_2(\mathbb{Z})$

vorgelegt von
Diplom-Mathematiker
Jens Frederik Bernhard Putzka
aus Bonn-Duisdorf

Bonn, May 2012

Dipl. Math. Jens Putzka
Mathematisches Institut der Universität Bonn
Max-Planck-Institut für Mathematik Bonn
Hausdorffcenter for Mathematics Bonn

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

Betreuer:
1. Gutachter: Prof. Dr. Jens Franke,
Mathematisches Institut der Universität Bonn
2. Gutachter: Prof. Dr. Günter Harder,
Mathematisches Institut der Universität Bonn, Max-Planck Institut für Mathematik Bonn

Tag der Promotion: 30. Oktober 2012
Erscheinungsjahr: 2013

Summaries

English Summary

It is the aim of this thesis to present a toolbox of methods which can be used to compute the group cohomology $H^q(\mathbf{Sp}_2(\mathbb{Z}), \mathbb{M}_\lambda)$ of the Siegel modular group $\mathbf{Sp}_2(\mathbb{Z})$ for a given integer $q \geq 0$ and a highest weight module $\mathbb{M}_\lambda$ with respect to a weight λ. Many tools we introduce in this thesis can be applied with minor changes also to groups different from the symplectic group. By some spectral sequence argument we get under some week assumptions an isomorphism

$$H^q(\mathbf{Sp}_2(\mathbb{Z}), \mathbb{M}_\lambda) \cong H^q\left(\mathfrak{S}_2/\mathbf{Sp}_2(\mathbb{Z}), \widetilde{\mathbb{M}_\lambda}\right)$$

between group cohomology of $\mathbf{Sp}_2(\mathbb{Z})$ and sheaf cohomology of the sheaf $\widetilde{\mathbb{M}_\lambda}$ over $\mathfrak{S}_2/\mathbf{Sp}_2(\mathbb{Z})$, which is associated to the module $\mathbb{M}_\lambda$. Here $\mathfrak{S}_2$ denotes the Siegel upper half-space. The cohomology groups behave very differently for different choices of λ. In particular, those cases are of special interest in number theory, e.g. for Harder's conjecture on eigenvalues of Hecke operators, which have non-vanishing so called cusp cohomology. This happens for the first time for the highest weight module of weight $\lambda = (7, 4)$, which has rank 1820 over the integers, which is not really small.

There are three main branches in this thesis:

Topological Model: It is known by a result of Mark McConnell and Robert MacPherson from the late eighties that there is a Γ'-equivariant deformation retract $\mathbb{W}$ of $\mathfrak{S}_2$ for a neat arithmetic subgroup $\Gamma' \subset \mathbf{Sp}_2(\mathbb{Z})$, which has the structure of a regular cell complex, and descends to a 4-dimensional retract $\mathbb{W}/\Gamma'$ of the 6-dimensional space $\mathfrak{S}_2/\Gamma'$, which has again a cell decomposition. Our first main result is the generalization of this model to the torsion case. To do this we have to replace cells by new objects called orbicells, which are the orbifold equivalent to cells in manifolds, and have very similar properties. We obtain a retract $\mathbb{W}/\Gamma'$ of $\mathfrak{S}_2/\Gamma'$ for any subgroup $\Gamma' \subseteq \mathbf{Sp}_2(\mathbb{Z})$ of finite index. Each of these retracts has an orbicell decomposition, which derives from the cell decomposition of $\mathbb{W}$. We implemented a computer program in SAGE which computes various things related to these decompositions, e.g. closures, stabilizers, and neighbours of cells. We illustrate up to some level – drawing in 4-D is a little bit ambitious – how the building blocks look like.

Highest Weight Modules: It is essential for an efficient computation of the cohomology to be able to perform various actions with and on highest weight modules. Most things are quite simple if a basis is known. However it turned out that the computation of the action of a group element on the module is not that easy. Therefore, our second main result is an algorithm to compute this action. This involves the decomposition of a given group element into generic generators coming from some roots via the morphism between Lie algebras and Lie groups. To obtain the needed decomposition we introduce a structured Gaussian elimination which preserves the symplectic structure.

Compute Cohomology: There is the notion of constructible sheaves due to Alexander Grothendieck and his collaborators, which describes a category of sheaves which are locally constant restricted to objects which cover (in Zariski-sense) a variety. We were able to generalize this category to a category of sheaves on the obtained orbicell decomposition. This construction is quite beautiful, and works in a much more general context. Therefore, we present parts of it not only for $\mathbb{W}$ or $\mathbb{W}/\Gamma'$, but for an abstract space with a suitable decomposition and an action of a group. We use this language to get an abstract description of the cohomology groups we are looking for, which later could be used to compute the cohomology.

Finally we discuss the application of this construction to the orbicell decomposition of the deformation retract $\mathbb{W}/\Gamma$ of the Siegel modular variety $\mathfrak{S}_2/\Gamma$ and indicate what has to be done to complete the computation of the cohomology of the symplectic group.

Deutsche Zusammenfassung

Das Ziel dieser Doktorarbeit ist es, einen Werkzeugkausten bereitzustellen, der dazu verwendet werden kann die Kohomologiegruppen $H^q(\mathbf{Sp}_2(\mathbb{Z}), \mathbb{M}_\lambda)$ der Siegelschen Modulgruppe $\mathbf{Sp}_2(\mathbb{Z})$ für eine gegebene ganze Zahl $q \geq 0$ und einen Höchstgewichtsmodul $\mathbb{M}_\lambda$ zu einem Höchstgewicht λ zu berechnen. Viele Hilfsmittel, die wir in dieser Arbeit einführen, können mit geringfügigen Änderungen auch für die Berechnung der Kohomologie anderer Gruppen eingesetzt werden. Unter einigen schwachen Annahmen erhalten wir mittels geeigneter Spektralsequenzen einen Isomorphismus

$$H^q(\mathbf{Sp}_2(\mathbb{Z}), \mathbb{M}_\lambda) \cong H^q\Big(\mathfrak{S}_2/\mathbf{Sp}_2(\mathbb{Z}), \widetilde{\mathbb{M}_\lambda}\Big)$$

zwischen der Gruppenkohomologie der Gruppe $\mathbf{Sp}_2(\mathbb{Z})$ und der Garbenkohomologie der zum Modul $\mathbb{M}_\lambda$ assoziierten Garbe $\widetilde{\mathbb{M}_\lambda}$ auf $\mathfrak{S}_2/\mathbf{Sp}_2(\mathbb{Z})$. Dabei bezeichnet $\mathfrak{S}_2$ den Siegelschen oberen Halbraum. Die Kohomologiegruppen verhalten sich abhängig vom Gewicht λ recht unterschiedlich. Insbesondere diejenigen Gewichte sind von besonderem zahlentheoretischem Interesse, etwa für Harders Vermutung über die Eigenwerte von Heckeoperatoren, die nicht verschwindende Spitzenkohomologie haben. Der kleinste Höchstgewichtsmodul, für den dies eintritt, ist der Modul zum Gewicht $\lambda = (7, 4)$, der schon Rang 1820 über den ganzen Zahl hat.

Diese Doktorarbeit besteht aus drei Hauptzweigen:

Topologisches Modell: Aus den späten 80er Jahren gibt es ein Ergebnis von Mark McConnell und Robert MacPherson, das besagt, dass es für jede arithmetische Untergruppe $\Gamma' \subset \mathbf{Sp}_2(\mathbb{Z})$, die zusätzlich *neat* ist, einen Γ'-äquivarianten Deformationsretrakt $\mathbb{W}$ von $\mathfrak{S}_2$

gibt, der die Struktur eines regulären Zellenkomplexes besitzt. Aus diesem Retrakt erhält man einen 4-dimensionalen Retrakt $\mathbb{W}/\Gamma'$ des 6-dimensionalen Raumes $\mathfrak{S}_2/\Gamma'$, der wiederum eine Zellenzerlegung besitzt. Unser erstes Hauptresultat verallgemeinert dieses Modell, so dass es sich auch für Grupppen mit Torsion anwenden lässt. Dazu mussten wir Zellen durch ihr Orbifoldäquivalent, Orbizellen, die ähnliche Eigenschaften wie Zellen haben, ersetzen. Schließlich erhalten wir einen Retrakt $\mathbb{W}/\Gamma'$ von $\mathfrak{S}_2/\Gamma'$ für jede Untergruppe $\Gamma' \subseteq \mathbf{Sp}_2(\mathbb{Z})$ von endlichem Index. Jeder dieser Retrakte besitzt eine Orbizellzerlegung, die man aus der Zellenzerlegung von $\mathbb{W}$ erhält. Wir haben in SAGE ein Computerprogramm geschrieben, das verschieden Dinge zu diesen Zerlegungen, wie etwa Abschlüsse, Stabilisatoren oder Nachbarn von Zellen, ausrechnen kann. Wir können auch, soweit es geht – Zeichnen in 4-D ist doch eine gewisse Herausforderung –, die einzelnen Bausteine graphisch darstellen.

Kohomologieberechnung: Es is unerlässlich für eine effektive Berechnung der Kohomologie, verschiedene Operationen auf und mit Höchstgewichtsmoduln durchführen zu können. Das meiste davon ist einfach, sofern man eine Basis des Moduls kennt. Es stellte sich jedoch heraus, dass die Berechnung der Gruppenwirkung nicht ganz so einfach ist. Somit stellt ein Algorithmus zur Berechnung der Gruppenwirkung das zweite Hauptergebnis dieser Arbeit dar. Zur Berechnung der Gruppenwirkung muss man ein gegebenes Gruppenelement in generische Erzeuger zerlegen, die man aus der Theorie von Liealgebren und deren Beziehung zu Liegruppen erhält. Zur Berechnung der benötigten Zerlegung führen wir dann einen strukturierten Gaußalgorithmus ein, der die symplektischen Strukturen respektiert.

Kohomologieberechnung: Konstruktible Garben wurden ursprünglich durch Alexander Grothendieck und die Gruppe von Mathematiker um ihn eingeführt. Die Garben dieser Kategorie zeichnen sich dadurch aus, dass sie eingeschränkt auf gewisse Objekte, die die Varietät (im Zariski-Sinn) überdecken, lokal konstant sind. Wir verallgmeinern diese Kategorie zu einer Kategorie von Garben, die man aus einer gegeben (Orbi-) Zellzerlegung erhält. Diese Konstruktion funktioniert auch in einem weitaus allgmeinerem Kontext, so dass wir deren Darstellung nicht auf die Räume $\mathbb{W}$ or $\mathbb{W}/\Gamma'$ beschränken, sondern für einen allgemeinen Raum mit einer geeigneten Zerlegung und Gruppenwirkung darstellen. Wir verwenden diese Sprache, um eine abstrakte Beschreibung der Kohomologiegruppen, die wir berechnen wollen, zu erhalten, die sich für eine spätere praktische Berechnung der Kohomologie bestens eignet.

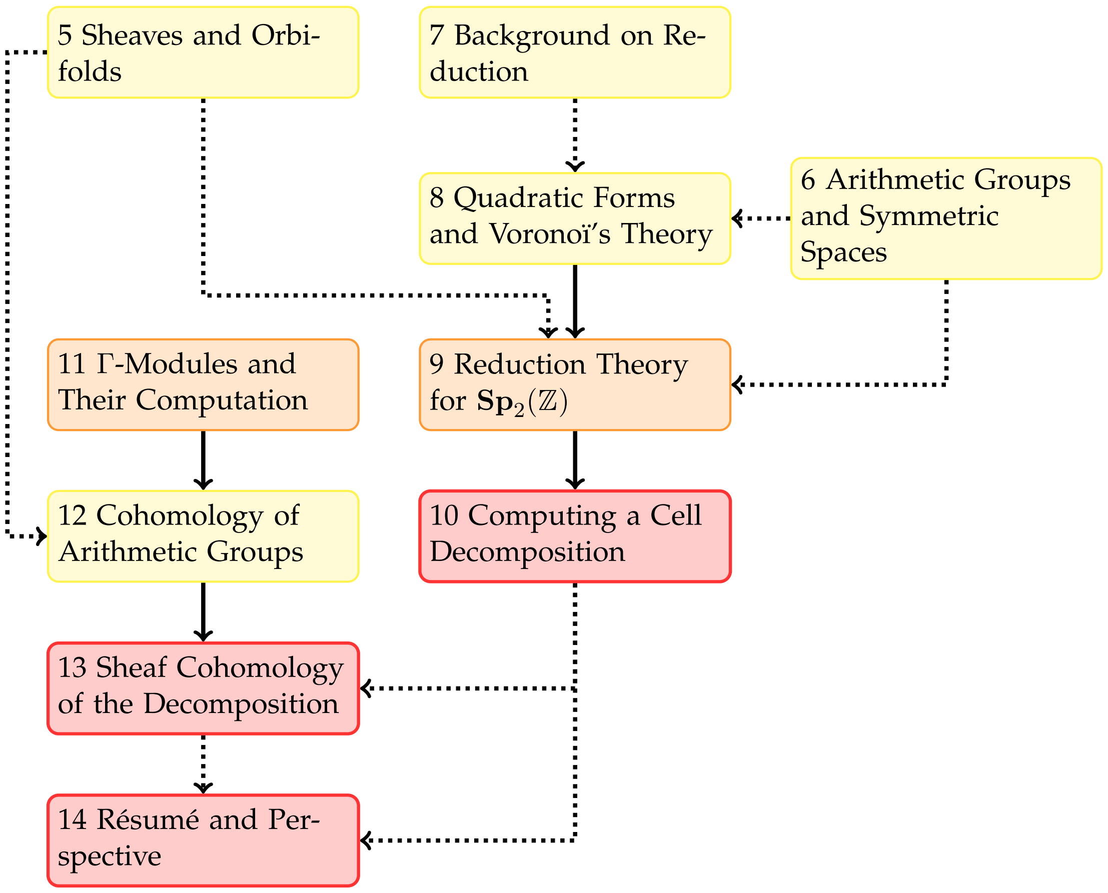
Zu guter Letzt diskutieren wir noch, wie sich diese Konstruktion auf die von uns eingeführte Orbizellzerlegung des Deformationsretrakts $\mathbb{W}/\Gamma$ der Siegelschen Modulvarietät $\mathfrak{S}_2/\Gamma$ anwenden lässt, und skizzieren, was noch notwendig ist, um die Berechnung der Kohomologie der symplektischen Gruppe zu vervollständigen.

Contents

Hints for Reading

In this short section we give the reader some guidance through this thesis. We try to visualize the structure and dependencies in Figure 0.1.

Figure 0.1.: *Structure and dependencies in this thesis. Yellow boxes indicate that the most of the contents of that chapter is known. Orange boxed parts are mixtures of my own work and known results to equal parts. The red boxed parts are completely my own work. The Black arrows indicate strong dependencies, whereas the dotted arrows show some weak dependency.*


5 Sheaves and Orbifolds
7 Background on Reduction
8 Quadratic Forms and Voronoï's Theory
6 Arithmetic Groups and Symmetric Spaces
11 Γ-Modules and Their Computation
9 Reduction Theory for $\mathbf{Sp}_2(\mathbb{Z})$
12 Cohomology of Arithmetic Groups
10 Computing a Cell Decomposition
13 Sheaf Cohomology of the Decomposition
14 Résumé and Perspective

There are two main branches, Chapters 11 – 13, which are dedicated to Γ-modules, the computation of sheaf cohomology and some related topics, and Chapters 7 – 10, which cover to the construction of the cell decomposition. These two branches are up to some level independent. Therefore, one can read one of them more or less independently of the other. The auxiliary chapters, which are not part of these two branches, might be skipped by an advanced reader except for parts of Chapter 5, where we define orbicells, which are essential for the further understanding, in particular for Chapters 9 and 13.

Preface

1. Introduction

1.1. Extended Summary

This thesis essentially consists of three parts. First, we collect some more ore less commonly known results we will use later on. The second part is dedicated to the question how to obtain the topological model and to get some additional information on the structure out of it. In the third and last part we will handle questions related to cohomology and their computation.

Background

In Chapter 5 we introduce the necessary definitions related to sheaf cohomology. After that we give some summary on orbifolds, before we introduce cells, which we assume to be commonly known, and orbicells, which are our orbifold counterpart to cells in a manifold and which are in some way new objects.

The main objects in Chapter 6 are algebraic groups and their associated symmetric spaces. Therein we discuss first arithmetic groups and Γ-modules in Section 6.1 and symmetric and locally symmetric spaces in Section 6.2 in a more general context, before we start to focus on the symplectic group and Siegel modular varieties, some locally symmetric spaces associated to the symplectic group, which are the main objects in this thesis. Here we have to discuss different ways to define "the" symplectic group $\mathbf{Sp}_2(\mathbb{Z})$ and present some basic properties of this group (see Section 6.3.1). The definition of the Siegel upper half-spaces $\mathfrak{S}_g$ as well as a collection of basic facts on them and their quotients by arithmetic subgroups of the symplectic group are given in Section 6.3.2. Then we discuss which integers can occur as orders of elements in $\mathbf{Sp}_g(\mathbb{Z})$ (see Section 6.3.3). We close this section with a short survey on the connection between the structure of points in the moduli space of principally polarized abelian varieties with a prescribed automorphism group and fixed point sets in the Siegel upper half-space under the symplectic group in Siegel modular varieties (see Section 6.3.4). We continue in Section 6.4 with a summary on the basic notions related to Lie algebras and highest weight modules associated to them. We assume that the reader is familiar with this topic, but since there are different naming and enumerating conventions, we decided to include a part on that into this thesis. The knowledge of this topic is essentiall to compute bases of Γ-modules and the action of Lie group elements on them later in Section 11.2. Finally, in Section 6.5, we state a second description of highest weight modules for the symplectic group.

On Quadratic Forms and Cells

Chapter 7 is meant as an informal introduction to different notions of reduction theories and in addition contains a short survey on the already known results on this topic for different kinds of groups.

In Chapter 8 we proceed with an outline of Voronoï's work on reduction theory of quadratic forms, which is related to the groups $\mathbf{GL}_d(\mathbb{Z})$ or $\mathbf{SL}_d(\mathbb{Z})$ respectively. We concentrate on the so called Voronoï I reduction. (He give two different constructions.) To formulate his results we have to have a closer look at the space of positive definite quadratic forms of dimension d and the action of $\mathbf{GL}_d(\mathbb{Z})$ on it.

We start Chapter 9 with an extended summary of the results of McConnell and MacPherson from [MM93, MM89] for neat subgroups of $\mathbf{Sp}_2(\mathbb{Z})$ in Section 9.1, which is the starting point of our own extension of their results to the torsion case in Section 9.2, which is our first main result (Theorem 9.2.11). This shows that in $\mathbb{W}$ there are up to equivalence under $\mathbf{Sp}_2(\mathbb{Z})$ only eleven classes of cells. This is an equivalent of [MM93; Theorem 8.8] stated in context of orbifolds with an orbicell decomposition.

In Chapter 10 we discuss questions related to the computation of several derived objects from cells or orbicells respectively. We present a method how to compute closures (see Section 10.4.2), stabilizers (see Section 10.5) and neighbours (see Section 10.6) of a given (orbi-)cell in a (orbi-)cell decomposition in a simple and fast way with one single idea. This is the second main result of this thesis. The speed-up of the computation bases on two simple observations. The first idea is that we can represent (orbi-)cells by collections of column vectors together with an equivalence relation, on which the group acts essentially by matrix multiplication with the inverse of the group element from the left. Therefore, we find that the set of columns representing top-dimensional cells are up to to some equivalence the column vectors of matrices in $\mathbf{Sp}_2(\mathbb{Z})$. We consider two sets as equivalent if they are equal up to their order and multiplication by ± 1. The second idea is that cells in the boundary of a given cell are represented by a collection of vectors which contains the collection of vectors which represents the inner cell. There is, up to equivalence under the group, only one top-dimensional cell, i.e. a cell of dimension four, which is represented by the four unit column vectors in $\mathbb{Z}^4$. Our algorithm now uses – with some variations – these two ingredients in the following way: Generate matrices from a collection of vectors by permuting the columns and giving them some signs, check for each matrix whether it is symplectic or not and proceed with further checks and operations depending on the type of object to be computed. We implemented some simple ideas to reduce the runtime, for example by reducing the number of candidate matrices which have to be checked. We close this chapter with a presentation of the obtained results. In particular, we give a description how the closures, neighbours, and stabilizers of each type of cells in $\mathbb{W}$ and how their images in the quotient look like (see Section 10.7).

Cohomology

Chapter 11 is dedicated to the construction and computation of highest weight modules with an action of the symplectic group, i.e. a Lie group. We start with their abstract construction in

Section 11.1. Here we focus on their relation to highest weight modules for Lie algebra representations. This is used in Section 11.2 to state a basis, which is called Poincaré-Birkhoff-Witt basis, of that module. It turns out that it is necessary for the description of a matrix representation of the group action on a highest weight module to decompose the given group element into a product of images of generic generators of the Lie algebra. We give an explicit construction for this decomposition, which is in some way a structured version of the Gaussian elimination algorithm (see Section 11.2.3). At last we give some examples including some discussion of the runtime and explain how to compute invariants of highest weight modules under some finitely generated groups (see Sections 11.2.4 to 11.2.6).

We start Chapter 12 with the introduction of orbilocal systems in Section 12.1, which some authors also call local systems, and proceed in Section 12.2 with the definition of the object of interest in this thesis, i.e. the cohomology of arithmetic groups. Later in Section 12.3 we collect some vanishing results for the cohomology of arithmetic groups which can be used to check results for their correctness or later in praxi to avoid unnecessary computations of cases where the cohomology is known to be equal to zero. Beside these results also we find the vanishing result in relation to the virtual cohomological dimension, which was the starting point for McConnell and MacPherson to look for the retract we are using in Part II. In the following Section 12.4 we show how our program can compute global sections and analyse its runtime.

In Chapter 13 we introduce a partially new concept of constructible sheaves[1] on a topological space $\mathcal{X}$ with a suitable decomposition, extend it in a natural way to the new concept of orbiconstructible sheaves on quotients of $\mathcal{X}$ and construct some suitable resolutions using these category of sheaves. Those resolutions can be used to compute the cohomology we are looking for. Both definitions depend up to some level on a cell decomposition of the space or an orbicell decomposition of its quotient. We can apply this construction in a slightly more general context than needed to obtain only the cohomology of the symplectic group. We start in Section 13.1 with the definition of a constructible sheaf, which is roughly spoken a sheaf which is constant restricted to each cell of some cellular decomposition. In addition, if we have a group action which is compatible with the cell decomposition we can under certain conditions descend to an orbiconstructible sheaf on the quotient (see Section 13.2). After the analysis of some interesting properties we are then able to write down several exact sequences which can be used to construct resolutions as well for a constant sheaf on the covering space as for the associated sheaf on its quotient (see Section 13.3). This is the third main result of this thesis. At the end of Chapter 13, in Section 13.4, we discuss the application of the previously introduced methods to the retract $\mathbb{W}$ and $\mathbb{W}/\Gamma$. First, we explain how to compute the cohomology groups we are looking for in case of this example. For this, we focus on the resolution from Theorem 13.3.16 in this special case (see Theorem 13.4.3). Here one has to discuss different problems which occur during the implementation of the necessary operations on (orbi-)constructible sheaves. Then we continue with the specialization and discuss the case where the underlying coefficient module of the cohomology group is given by a highest weight module (see Section 13.4.2). We end up Chapter 13 with a short section on the further steps one needs to complete the computation of the cohomology of the symplectic group.

[1] An analogue concept was originally introduced by Alexander Grothendieck and his collaborators in the early sixties.

In the last chapter, i.e. Chapter 14, we summarize the problems which we solved in this thesis and indicate which problems are still open.

1.2. History of this Thesis

The starting point of this thesis was in some sense the retirement of one of my academic teachers and, which I did not know at that time, of one of my advisors of this thesis project. On February 7th 2003 Professor Günter Harder gave, on occasion of his retirement, a colloquium talk in Bonn with the title *A Congruence between Siegel and Elliptic Modular Forms* [Har03]. In this talk he presented some ideas he was thinking about for many years, which lead to the original motivation of this thesis.

In this talk Harder presented a conjectural connection between the eigenvalues of Hecke operators $\mathbf{T}'_p$ and $\mathbf{T}_p$ on the space of cusp forms on one side for the Siegel modular group $\mathbf{Sp}_2(Z)$ and on the other side for the elliptic modular group $\mathbf{SL}_2(\mathbb{Z})$ in terms of some congruence conditions for the eigenvalues modulo a large[2] prime ℓ. Such a congruence should exist if ℓ does not divide some critical values of some L-functions related to modular cusp forms of given weights. In some way this conjecture is a generalization of the classical result of Srinivasa Ramanujan [Ram16] published in 1916 that

$$\tau(p) \equiv p^{11} + 1 \mod 691 \tag{1.1}$$

holds for all primes p, where $\tau(p)$ is the p-th coefficient of the q-expansion of the famous Δ-function

$$\Delta(z) = q - 24q^2 + 252q^3 - 1472q^4 + 4830q^5 \mp \ldots = \sum_{n=1}^{\infty} \tau(n) \cdot q^n. \tag{1.2}$$

Whereas Ramanujan formulated his result as a property of the q-expansion of the Δ-function, which can be turned into a relation for Hecke eigenvalues since the Hecke eigenvalues are the same as the Fourier coefficients of Δ, Harder's conjecture is a conjecture on Hecke eigenvalues and there seems to be no analogous result for the corresponding Fourier series.

Harder showed – using some computational results of Gerard van der Geer from Amsterdam – that the conjecture holds for $2 \leq p \leq 11$.

In June 2004 there was a summer school on Modular Forms at Nordfjordeid in Norway. As one part van der Geer gave there lectures on Siegel Modular Form in which he presented some additional evidence[3] for Harder's conjectures and gave a more precise formulation of this conjecture [Ran08]. Due to this publication Harder's conjecture became much more popular in the number theory community.

At the time when I though about the possible subjects for a thesis, I found by accident the book by William Stein [Ste07] on computational aspects of modular forms with a quite interesting appendix by Paul Gunnells on higher rank computations [Gun07]. In his contribution

[2]Here large does not mean that ℓ has to have hundreds of digits as in cryptography. It is enough that ℓ is larger than 20.

[3]He and his collaborators computed the eigenvalues up to $p = 37$.

Paul Gunnells also gives an overview on open and/or only partially answered questions in this area, i.e. the practical computation of cohomology groups in case of higher rank, and I remembered the lecture of Harder some four and a half years ago. After some informal discussion with Jens Franke and Günter Harder we agreed that there should be enough substance to do my thesis in this area. Since there are several results for the cohomology of $\mathbf{SL}_n(\mathbb{Z})$ and related subgroups, but only a few on symplectic groups, we decided to focus on this. Furthermore, an efficient computation of the cohomology (as module under the Hecke algebra) would be a way to check – as a goal in the far future – the conjecture of Harder.

As a first task we wanted to compute the cohomology for $\mathbf{Sp}_2(\mathbb{Z})$, before we start to compute Hecke operators on these spaces and to check Harder's conjecture. Unfortunately we underestimated the complexity of this task completely. Very optimistic we wrote a proposal for a project on these tasks and applied for some funding by the Hausdorff Center for Mathematics. We got that project for two years and I started with my first steps in this direction.

At the beginning of the project we had essentially two ideas how to attack the problem, which seemed to be promising. One was to use the identification of the Siegel modular variety with the moduli space of principally polarized abelian varieties of dimension three as well as the structure of the automorphism groups of these abelian varieties. The other idea was to use a result of Mark McConnell and Robert MacPherson, who introduced a deformation retract $\mathbb{W}/\Gamma'$ for quotients of the Siegel upper half-space $\mathfrak{S}_2$ by neat/torsion free subgroups $\Gamma' \subset \mathbf{Sp}_2(\mathbb{Z})$. The group $\mathbf{Sp}_2(\mathbb{Z})$ has torsion, thus one could use a standard technique where one covers the quotient by the torsion group by the quotient for the torsion free group. The cohomology groups of the original quotient are then the invariants under the deck transformation group of the covering. I decided to try the method which uses the retracts first. The main problem with that method is that the number of cells increases if we replace one space by a covering of it, because the number of cells scales with the cardinality of the deck transformation group[4]. Thus, complexity is quite bad. Therefore, after a (too) short reflection on this, I decided to give the other method a chance. To say it in advance, it was a mistake to change the method at that early stage.

We started to follow the second alternative and found out after a while that there are several problems which one could not solve as easily as we thought. By accident I met Paul Gunnells in Oberwolfach and he pointed out how one should be able to use McConnells and MacPhersons results in my situation without the scaling problem mentioned above. He believed that an analogous result should be true also in the torsion case. We found out that he was completely right and we generalized in the following the original construction, which is now one main result in this thesis. For this we had to face several, most times rather technical, problems related to the question how to compute the needed things efficiently. We solved or even avoided almost all of them, but this took much more time than expected such that we could not reach the original goal of this thesis completely.

[4]The deck transformation group related to the principal congruence subgroup $\Gamma(p) \subset \mathbf{Sp}_2(\mathbb{Z})$ is isomorphic to $\mathbf{Sp}_2(\mathbb{F}_p)$ and thus also in the smallest torsion free cases, i.e. $p = 3$, quite huge ($\#\mathbf{Sp}_2(\mathbb{F}_3) = 51840$). We refer to page 59 for a more detailed discussion of this.

1.3. History of the Problem

At the same time there are a lot of results and only a few of them related to the topics of this thesis. We give an overview what this means. This thesis is dedicated to the question how to get a method to compute the cohomology of the symplectic group $\mathbf{Sp}_g(\mathbb{Z})$ for $g = 2$ with coefficients coming from highest weight modules. For this we use a construction bases on a suitable deformation retract.

Hence, there are three aspects which we have a closer look at:

- Harder's conjecture (motivation for this thesis)
- Cohomology of the symplectic group with different coefficients
- Cohomology computed by deformation retracts

Harder's Conjecture Since it was the central motivation for this thesis, we give an overview on publications related to Harder's conjecture, which we discussed in the previous section. First, there is the original source, i.e. Günter Harder's colloquium [Har03] which was published in [Ran08] later on. In the same volume Gerard van der Geer presents some additional considerations related to Harder's conjecture (see [vdG08]) and gave it a more precise form. In some way, Harder raised similar questions already earlier (see e.g. [Har93]).

During the last years there were some efforts by van der Geer and some of his collaborators to get computational evidence for the conjecture (see [BFvdG08, FvdG04a, FvdG04b]). Although there is up to now no proof of the conjecture, some researchers also try to generalize the conjecture to other groups, where similar congruences can occur (see e.g. [Ibu08, Dum11, BFvdG11]). The most recent publication related to Harder's conjecture is the preprint [GRS12] published in March 2012, where the authors give some further verifications for the conjecture. The known examples let us believe that the conjecture might be true.

Cohomology We do not want to discuss here the development of the cohomology of arithmetic groups in general. We focus on the results related to the symplectic group $\mathbf{Sp}_2(\mathbb{Z})$. For some results on cohomology of arithmetic groups in general and of the symplectic group in particular one might have a look into the works of Joachim Schwermer [Sch83, Sch94, Sch86, LS09, SL01, Sch10]. He also studies the question under which conditions the cohomology or parts of it vanish. This information allows us a deeper look into inner structures of the cohomology.

There are several results for the cohomology $H^q(\mathbf{Sp}_2(\mathbb{Z}), \mathbf{R})$ of the symplectic group $\mathbf{Sp}_2(\mathbb{Z})$ (and some subgroups of this group) with coefficients in a commutative ring $\mathbf{R}$. Most of the results are part of a sequence of papers of Ronnie Lee and Steven H. Weintraub and their collaborators. The starting point of this sequence is [LW85] from 1985, which considers the group cohomology of $\mathbf{Sp}_2(\mathbb{Z})$ and some subgroups, where the coefficients for the cohomology come from a field $\mathbb{F}$. As an example they compute the rank of the cohomology of the principal congruence subgroup $\Gamma(2) \subset \mathbf{Sp}_2(\mathbb{Z})$ of level 2. They proved that

$$\operatorname{rank}_{\mathbb{F}} H^q(\Gamma(2), \mathbb{F}) = \begin{cases} 1 & q = 0 \\ 0 & q = 1 \\ 1 & q = 2 \\ 14 & q = 3 \\ 16 & q = 4 \end{cases}$$

for any field $\mathbb{F}$ with $\operatorname{char}(\mathbb{F}) > 2$. In a similar way Rainer Weissauer gave in [Wei88] an explicit formula for the rank over $\mathbb{C}$ of the global sections $H^0(\mathfrak{S}_2/\Gamma(N), \underline{\mathbb{C}})$ for principal congruence subgroups $\Gamma(N)$ of level N.

The main result in this sequence of papers is contained in the two papers [BL92, BL94b] of Alan Brownstein and Ronnie Lee from 1992 and 1994. In these papers the authors compute the cohomology groups $H^q(\mathbf{Sp}_2(\mathbb{Z}), \mathbb{Z})$ explicitly. They start their computation with the 2-primary part of the cohomology in [BL92] and continue with the odd torsion cases in [BL94b]. It is enough to compute the 2-, 3- and 5-primary parts, because there are no elements in $\mathbf{Sp}_2(\mathbb{Z})$ which have an order divisible by a prime different from 2, 3 and 5 (comp. Section 6.3.3). These local informations together with the rank over $\mathbb{R}$ or $\mathbb{C}$ give a complete description of $H^q(\mathbf{Sp}_2(\mathbb{Z}), \mathbb{Z})$.

These results are followed in 1998 by the computation of the cohomology of $\Gamma(4) \subset \mathbf{Sp}_2(\mathbb{Z})$ by J. William Hoffman and Steven H. Weintraub in [HW98], using techniques of Takayuki Oda and Joachim Schwermer presented in [OS90] for the cohomology of torsion free arithmetic subgroups $\Gamma' \subset \mathbf{Sp}_2(\mathbb{Q})$. Some of the results up to that point are summarized in [LW98], which also contains the computation of Hodge numbers.

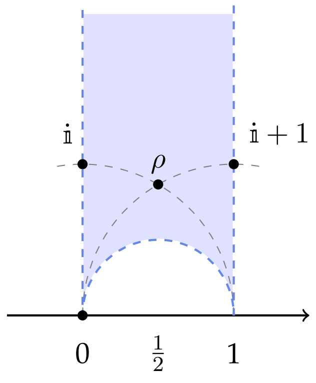
In 2000 Gunnells published in [Gun00] an additional approach using symplectic modular symbols, which were already used e.g. in case of the cohomology of $\mathbf{SL}_2(\mathbb{Z})$ and its subgroups, and obtains some results for the cohomology of the symplectic group in certain degrees.

The most recent contribution in this direction were published in 2001 and 2003 by Hoffman and Weintraub, who computed the cohomology of the moduli space of principally polarized abelian varieties with N-level structure (comp. Section 6.3.4) [HW01], which is essentially the same as the cohomology of $\Gamma(N)$, and of its boundary [HW03] with respect to a smooth toroidal compactification (see e.g. [Nam80]).

These are in some sense the main results. There are some more for special subgroups in the literature, such that one can state that we have a quite good understanding of the cohomology of the symplectic group as long as the coefficients are $\mathbb{Z}$ or a field. The problem is that we need the cohomology for non-trivial coefficient systems, but in this case there are no explicit computational results known. There are only some vanishing results, e.g. by Schwermer et al. (see above).

Cohomology Computed by Deformation Retracts: Georgy F. Voronoï (1868–1908) introduced in a sequence of posthumously published papers [Vor08b, Vor08a, Vor09] a reduction theory for quadratic forms, which can be used to get some fundamental domains for the

Figure 1.1.: *A fundamental domain for the group* $\mathbf{SL}_2(\mathbb{Z})$. *For the classical one see Figure 7.2a*



groups $\mathbf{GL}_d(\mathbb{Z})$ for some integer d (comp. the discussion in Section 7.1). His construction gives a model which has the same dimension as the original space. For our purpose we want to have a minimal model, which carries all information on the cohomology of the space, but has lower dimension than the original space (see Chapter 7). Such spaces are called deformation retracts (see page 37).

Since this question is not a new one there are some results on the construction of deformation retracts for several groups. First, there are results in the case of groups of type $\mathbf{SL}_d(\mathbb{Z})$. In 1971 Jean-Pierre Serre introduced the so called **Serre tree** for the group $\mathbf{SL}_2(\mathbb{Z})$ acting on the upper half-plane [Ser71]. This is the classical example for most things related to the cohomology of arithmetic groups and thus the most discussed case in the whole area. Here one contracts the fundamental domain (see Section 7.1) shown in Figure 1.1 (or one of its translates, resp.) to the union of three (geodesic) edges connecting the central fixed point $\rho = \exp(\pi\mathbb{i}/3)$ with the three images of $\mathbb{i}$ under $\mathbf{SL}_2(\mathbb{Z})$ lying on the boundary of the shaded area in Figure 1.1, i.e. with $\mathbb{i}$, $\mathbb{i}+1$ and $-(\mathbb{i}-1)^{-1} = \frac{1}{2}(\mathbb{i}+1)$. If we now translate this component by $\mathbf{SL}_2(\mathbb{Z})$ we get an object which is in this case a tree, i.e. a graph without loops (see Figure 1.2). The group $\mathbf{PSL}_2(\mathbb{Z})$ acts on this tree by permuting the edges and vertices. The elements in $\mathbf{PSL}_2(\mathbb{Z})$ of finite order, i.e. the stabilizers of $\mathbb{i}$ or ρ (or one of its translates), have order two and three. In the picture one can see their action. The elements of order two flip two vertices which are connected by one edge. The element of order 3 rotates the free neighbours of a vertex.

For $d = 3$ there is the Soulé Cube found by Christophe Soulé [Sou78] and, independently, by Ronnie Lee and Robert Henry Szczarba [LS78]. The examples of the cases $d = 2$ and $d = 3$ are both handled in [Ste07].

The most important known example are the recent results for the group $\mathbf{SL}_4(\mathbb{Z})$ by Avner Ash, Mark McConnell, and Paul Gunnells, see e.g. [Ash80, AGM02, AGM08, AGM10, AGM11, Gun09] and for $\mathbf{GL}_n(\mathbb{Z})$ by McConnell in [AM97]. They obtain their results on reduction with a method which is very similar to the one we use (comp. Chapters 8 and 9). Further, there are results for $\mathbf{GL}_d(\mathbb{Z})$ and $\mathbf{SL}_d(\mathbb{Z})$ for $d = 5, 6$ by Philippe Elbaz-Vincent, Herbert Gangl and Christophe Soulé [EVGS02] which one should mention. They use a slightly different approach to get their reduced model of the quotient and handle in the following parts the K-theory of $\mathbb{Z}$ which is strongly related to the cohomology with trivial coefficients.

Figure 1.2.: *Serre tree for the group $\mathbf{SL}_2(\mathbb{Z})$. The gray lines indicate the boundary of the copies of a fundamental domain of the form shown in Figure 7.2b and the blue vertices and edges are the related spine or to be precise a finite part of it. This picture was partially computed with* SAGE *and then enhanced with Adobe® Illustrator®.*

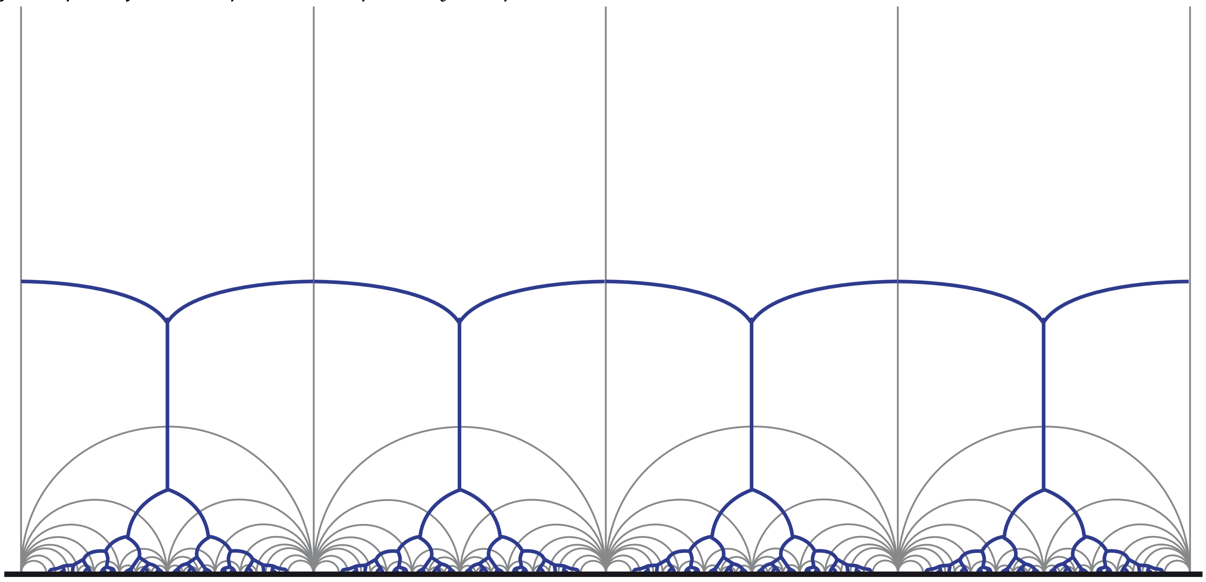
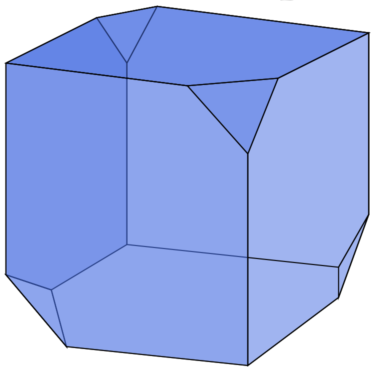


Figure 1.3.: *Soulé Cube. This picture was drawn with LaTeX and* tikz *after the picture in [Gun07].*



For $\mathbf{SL}_d(\mathbb{Z})$ with $d = 3, 4, 5, 6$ you will find in the mentioned references beside the reduced models for the groups also results on the cohomology. However all authors only handle trivial coefficients, i.e. $\mathbb{Z}$ or $\mathbb{R}$, which make considerable difference to our work since we are looking in particular for application to non-trivial coefficient systems coming from irreducible representations (comp. Chapters 12 and 13).

Dan Yasaki proved in his PhD-thesis a general result how to construct a spine for $\mathbb{Q}$-rank one groups. He then applied his method to some example groups like the groups $\mathbf{SU}(2, 1, \mathscr{O}_k)$ where $\mathscr{O}_k$ is the ring of integers of some algebraic number field k [Yas05, Yas07b, Yas07a, Yas10]. Thus, the question is completely answered if the group has rank one. The group $\mathbf{Sp}_2(\mathbb{Z})$, we are interested in, has rank two.

There are much less known results for higher rank groups. An overview on the problems and results in the higher rank case can be found in the appendix of [Ste07] by Paul Gunnells [Gun07]. More or less the only known examples are the groups $\mathbf{SL}_d(\mathbb{Z})$ and $\mathbf{GL}_d(\mathbb{Z})$ for some $d > 2$ (for $d = 2$ they are of rank 1) mentioned above and some result MacPherson and McConnell [MM93, MM89] for symplectic groups. After a very short survey how to attack symplectic groups in general the two authors focus on the case of neat subgroups of $\mathbf{Sp}_2(\mathbb{Z})$ and prove their main result only in that case. As far as I know there is no result known neither for symplectic groups with torsion and a non-trivial coefficient system nor for symplectic groups of higher rank. We discuss Section 9.1 the method of MacPherson and McConnell more in detail. Furthermore, we are going to extend their method to a slightly more general context such that we can apply this method to the Sieg modular group $\mathbf{Sp}_2(\mathbb{Z})$ and its finite index subgroups (see Section 9.2). This extension is one main goal of this thesis.

The reason why there are up to now no results on topological reductions for $\mathbf{Sp}_2(\mathbb{Z})$ might be that the quotient will – if one uses the usual approaches – not be a cell complex. We will remove this problem via a kind of redefinition of the problem. We will simply replace the types of objects we are looking for. Instead of looking for a cell complex we will look for an orbicell complex (which is in some sense the orbifold equivalent to a cell complex) which is compatible with the underlying group structure (comp. Section 5.3). This orbicell decomposition can be a classical cell decomposition, but does not have to. We have to emphasize that it is not enough to modify the definitions such that they fit the situation. One has to ensure in addition that the new model is one which is powerful enough to do its duty, i.e. it allows to compute the cohomology from it. We will show that this is possible. However this needs some additional work.

2. Used Computer Resources

2.1. Used Software

For my work I used several software packages:

- Python™ 2.6 (`http://www.python.org`)
- SAGE 4.6.2 (`http://www.sagemath.org`)
- GAP 4.4.12 with GAP `SmallGroups` Libray and the `QuaGroup`-package (`http://www.gap-system.org`)
- Mathematica® 8.0 by Wolfram Research Inc. (`http://www.wolfram.com`)
- This document was generated with LaTeX, AMS-LaTeX, SAGETeX, and KOMA-Script.

Most software – except Mathematica® – which I used is free software and in most cases also under GNU licence. In the following, I omit all references to different kinds of licences and trademarks.

Usually I used the open source program SAGE, which is written in Python, and which either can be used as stand-alone computer algebra system or as Python library. Python is an object-oriented programming language which allows for a high degree of abstractness which is quite useful in several cases. I decided to use SAGE since there are several libraries available which seemed to be well suited for my kind of problems. Furthermore, there is the possibility to wrap code of other systems like Pari or GAP into SAGE-programs such that one can benefit from the advantages of different worlds. SAGE uses this ability also for its own algorithms. For example, the implementation of matrix groups, which we use to handle stabilizers of cells, uses for a lot of its routines essentially the realization of matrix groups in GAP. For my project in particular the indirect use of GAP was quite useful to handle Lie-algebras and highest weight modules as parts of a SAGE program (see e.g. Section 11.2).

Note that Python and therefore also SAGE starts indexing with zero. Therefore, there might be some index shifts in parts which describe source code compared to some usual notations in mathematics, which I also use in some place inside the text of this thesis.

The SAGE source code of the programs described in this thesis can be found completely on the CD included in the printed copy of this thesis or can be obtained from the author. It is partially included in the Appendix B. All in all, the program has a length of about 4800 lines (counted only the parts written by myself).

2.2. Used Hardware

To test my program I used the following computer systems at home and in the Max-Planck-Institut für Mathematik:

theia@MPIM A SUN ray server with eight 2.7GHz Quad-Core AMD Opteron$^{\text{TM}}$8384 processors with Debian 5.0, Linux kernel 2.6.26 with a maximum of 320GB DDR2 666MHz memory, an L2 cache of 512KB per Core, and an L3 cache of 6MB per CPU. For the tests in this thesis I used only a single core on one processor per run, but took several runs in parallel.

harmonia@MPIM A SUN ray server with 24 AMD 2.1GHz Opteron$^{\text{TM}}$6172 processor with 12 cores each, Debian Release 6.0.1 (squeeze), Linux kernel 2.6.32-5-amd64, 64GB DDR2 RAM 666MHz, and 512kB L2 cache per core.

NOETHER@HOME An iMac 2.66GHz Intel Core Duo with Mac OS X 10.6.8, 4GB 800MHz DDR2 SDRAM, and 6MB L2 Cache.

Here I want to thank the IT support at the Max-Planck-Institut für Mathematik for continuous support, their kind and fast help if I needed some software to be installed or updated and the possibility to use the computer facilities.

3. Notation

For an index of notations and symbols used in this thesis we refer to the list of symbols beginning on page 248. Here we introduce some general notations we are use during this thesis.

We denote by $\mathbb{N}$ the positive integers and by $\mathbb{N}_0$ the non-negative integers. The ring of integers is $\mathbb{Z}$. Further, we denote by $\mathbb{Q}$ the field of rational numbers, by $\mathbb{R}$ the real numbers, by $\mathbb{R}^+ = \{x \in \mathbb{R} : x > 0\}$ the positive real numbers, and by $\mathbb{C}$ the complex numbers.

We use $\mathbb{i} = \sqrt{-1}$ for the imaginary unit, and sometimes $\pi = 3.1415\ldots$ for the number π and not a canonical projection, which occurs much more often than the number.

We sometimes omit zeros in matrices to make the entries more readable. The set of $d \times d$ matrices with entries from a ring $\mathbf{R}$ is denoted by $\mathrm{Mat}_d(\mathbf{R})$ and analogously for non-square matrices if we use two indices.

Usually we denote column vectors by bold face letters like $\mathbf{x}, \mathbf{y}, \mathbf{z}$. By $\mathbf{x}^t$ and A^t we denote the transpose of a column vector $\mathbf{x}$ and a matrix A. The standard basis vectors, where at the i-th position is a 1 and 0 at all other positions, we call $\mathbf{e}_i$.

We use $\subset$ to indicate an arbitrary subset. If we want to indicate a proper subset we use $\subsetneq$. In some cases we will in addition, if we want to emphasize that equality can hold, use $\subseteq$ for a subset which can also be equal. To indicate the boundary and the closure of a topological space $\mathcal{X}$ we use $\partial\mathcal{X}$ and $\overline{\mathcal{X}}$.

To avoid misunderstandings with quotients, which occur quite often, we use $G - H$ to express the elements of G without the elements of H and not the notion $G \backslash H$ which is also used in the literature. We use $\#$ or $|\_|$ to indicate the number of elements in a set or the order of a group which is the same.

We use the following short form

$$a^{\otimes n} := \underbrace{a \otimes \ldots \otimes a}_{n \text{ times}}.$$

We write types of cells or their images in the quotient with small caps like CRYSTAL. Source code and commands in programming languages are given in type writer face, like `Command`.

4. Acknowledgements

There are essentially two ways to write an acknowledgement for a thesis. One can keep it short or try to – as I will do – mention at least all those who supported, encouraged and motivated me during the last years.

First of all, I would like to say thank you to my two advisors Prof. Jens Franke and Prof. Günter Harder, who agreed to supervise this thesis and supported me in different ways during the last years. In particular, I have to thank Prof. Harder for many long discussions in his office.

For her patience, her words of encouragement, and the many fruitful (long) discussions on many evenings during the last years I am very grateful to my wife Anna Engels-Putzka. I am sure I was sometimes a little bit demanding. For their continuous support during my studies at university and before in school I have also to be thankful to my parents Manfred and Ulrike Putzka.

This thesis was supported by many institutions and organizations directly and indirectly. Therefore, I have to acknowledge the financial support by the Hausdorff Center for Mathematics (HCM) in Bonn, the Mathematical Institute of the University of Bonn, the International Max-Planck-Research School (IMPRS) on Moduli Spaces. For hospitality I have to thank the Max-Planck-Institut für Mathematik in Bonn. Furthermore, I have to thank the Bonn International Graduate School in Mathematics (BIGS), the Minerva Foundation, the Mathematisches Forschungsinstitut Oberwolfach and its 'Oberwolfach Leibnitz Junior Fellow' program for covering the costs of several visits of conferences in and outside of Germany. Here I have to emphasize the support of Karen Bingel and Sabine George of BIGS.

I was very happy for the continuous help from the whole staff of the Max-Planck-Institut für Mathematik. In particular, I be thankful to the different members in the computer support team under Alexander Weisse, the librarian Anke Völzman, Peter Winter and Cerolein Wels.

During the years I had many interesting discussions on mathematics, this thesis, and many more with many people around. Among these I have to thank in particular Christian Kaiser from the MPI in Bonn, who manages the IMPRS program there and whose door was always open for me to enter and ask many - sometime maybe also silly – questions, and Paul E. Gunnells from the University of Massachusetts in Amherst, because of a long inspiring discussion in Oberwolfach, which led me to *the* right way to handle the problems in Chapters 9. I have to thank my friends Nico Möser and Antje Kiesel because they were two brave volunteers (beside my wife) who read this thesis and gave some hints how to make things better.

Further, I want to thank some more people who contributed by hints, suggestions, or simply by waiting until I stop to talk: This are in alphabetical order: Jonas Bergström, Soumya Bhattacharya, Matthias Kretschmer, Kiarash Kuchaki, Christoph Lienau, Hanna Liese, Daniel Loebenberger, Oliver Lorscheid, Anton Mellit, Pieter Moree, Martin Raum, Marco Streng, and Christian Weiß.

Part I.

Background

5. Sheaves and Orbifolds

Quae legeris, memento!

(Marcus Porcius Cato)

We assume that the reader is familiar with the basic topological notions, in particular topological spaces, categories, functors, and sheaves. So it is not the aim of this section to be a self contained introduction to basic topological notions. We will give the most important definitions used in this thesis and some references for further reading. For general topological definitions we refer to the classical books of William Massey [Mas78, Mas91], or the more recent book of Alan Hatcher [Hat02]. For a slightly more computational approach there is the book by James Munkres [Mun84]. A basic introduction to categories and functors can be found – if necessary – in the book of Saunders MacLane [Mac98].

We assume that all topological spaces in this thesis are Hausdorff.

5.1. Sheaf Cohomology

There are several good references for sheaves in general and sheaf cohomology in particular. The general theory can be found in the book on sheaf theory of Glen Bredon [Bre97] or that of Birger Iversen on cohomology of sheaves [Ive86]. A good introduction more focused on questions related to the cohomology of arithmetic groups is the book of Günter Harder [Har08c] and his lecture notes [Har05, Har08a], which can be found on his website.

We briefly recall the definition of sheaf cohomology. Let for the following part $\mathcal{X}$ be a nice, connected topological space and let $\mathcal{M}$ be a sheaf on $\mathcal{X}$. All sheaves we consider in this thesis are sheaves of abelian groups (or later sheaves of $\mathbf{R}$-modules for some commutative ring with identity $\mathbf{R}$). The global section functor

$$\begin{aligned} H^0(\mathcal{X}, \_) : \mathbf{Sheaves}_{\mathcal{X}} &\to \mathbf{Ab} \\ \mathcal{M} &\mapsto H^0(\mathcal{X}, \mathcal{M}) = \mathcal{M}(\mathcal{X}) \end{aligned} \tag{5.1}$$

from the category of sheaves to the category of abelian groups is a left exact functor (see [Har08c; p. 51]). To avoid to much Γs later on we use $H^0(\mathcal{X}, \_)$ instead of $\Gamma_{\mathcal{X}}$ since Γ will be an arithmetic group and occurs in plenty variations. Since $H^0(\mathcal{X}, \_)$ is left exact we can form the right derived functors $\mathrm{R}^q(H^0(\mathcal{X}, \_))$ of $H^0(\mathcal{X}, \_)$.

Definition 5.1.1 (Cohomology): *Let $\mathcal{M}$ be a sheaf (of abelian groups) over a topological space $\mathcal{X}$. Then we define for $q \geq 0$ by*

$$H^q(\mathcal{X}, \mathcal{M}) = \mathrm{R}^q\big(H^0(\mathcal{X}, \mathcal{M})\big)$$

the q-th cohomology group of $\mathcal{X}$ with values in the sheaf $\mathcal{M}$. $H^q(\mathcal{X}, \_)$ is functor from the category of sheaves on $\mathcal{X}$ $\mathbf{Sheaves}_{\mathcal{X}}$ to the category of abelian groups $\mathbf{Ab}$.

We have always that $H^0(\mathcal{X};\mathcal{M}) = \mathcal{M}(\mathcal{M})$ which explains the notion $H^0(\mathcal{X};\_)$ for the global section functor in (5.1). Remember the definition of an acyclic sheaf.

Definition 5.1.2: *Let $\mathcal{M}$ be a sheaf on $\mathcal{X}$. Then $\mathcal{M}$ is called an* **acyclic** *if*

$$H^q(\mathcal{X},\mathcal{M}) = \begin{cases} \mathcal{M}(\mathcal{X}) & q = 0 \\ 0 & q > 0. \end{cases}$$

To compute the cohomology groups we use so called acyclic resolutions.

Definition 5.1.3: *An* **acyclic resolution** *of a sheaf $\mathcal{M}$ over $\mathcal{X}$ is an exact sequence*

$$\begin{tikzcd} 0 \arrow[r] & \mathcal{M} \arrow[r, hook, "\iota"] & \mathcal{I}^0 \arrow[r, "t_0"] & \mathcal{I}^1 \arrow[r, "t_1"] & \mathcal{I}^2 \arrow[r, "t_2"] & \mathcal{I}^3 \arrow[r, "t_3"] & \dots \end{tikzcd}$$

of sheaves where the $\mathcal{I}^k$ are acyclic sheaves.

The global section functor $H^0(\mathcal{X},\_)$ is only left exact (see [Har08c; p. 51]). If we have an acyclic resolution $0 \to \mathcal{M} \to \mathcal{I}^\bullet$ we can apply this functor to $0 \to \mathcal{I}^\bullet$ and we get

$$\begin{tikzcd} 0 \arrow[r] & \mathcal{I}^0(\mathcal{X}) \arrow[r, hook] & \mathcal{I}^1(\mathcal{X}) \arrow[r] & \mathcal{I}^2(\mathcal{X}) \arrow[r] & \mathcal{I}^3(\mathcal{X}) \arrow[r] & \dots . \end{tikzcd} \tag{5.2}$$

Let us denote this sequence by $H^0(\mathcal{X},\mathcal{I}^\bullet)$. We can now apply the known theory on that acyclic resolution (see e.g. [Bre97; II.4], or [Har08c; 2.3.1]) and get the following theorem.

Theorem 5.1.4: *Let $0 \to \mathcal{M} \to \mathcal{I}^\bullet$ be an acyclic resolution for the sheaf $\mathcal{M}$ over the topological space $\mathcal{X}$, then*

$$H^q(H^0(\mathcal{X},\mathcal{M})) = H^q(\mathcal{X},\mathcal{M}).$$

We know that if we have an exact sequence of sheaves on $\mathcal{X}$

$$\begin{tikzcd} 0 \arrow[r] & \mathcal{M}_1 \arrow[r] & \mathcal{M} \arrow[r] & \mathcal{M}_2 \arrow[r] & 0 \end{tikzcd} \tag{5.3}$$

then there is an exact sequence in the cohomology

$$\begin{tikzcd} 0 \arrow[r] & \mathcal{M}_1(\mathcal{X}) \arrow[r] & \mathcal{M}(\mathcal{X}) \arrow[r] & \mathcal{M}_2(\mathcal{X}) \arrow[dll, rounded corners] \\ & H^1(\mathcal{X};\mathcal{M}_1) \arrow[r] & H^1(\mathcal{X};\mathcal{M}) \arrow[r] & H^1(\mathcal{X};\mathcal{M}_2) \arrow[dll, rounded corners] \\ & H^2(\mathcal{X};\mathcal{M}_1) \arrow[r] & H^2(\mathcal{X};\mathcal{M}) \arrow[r] & \dots . \end{tikzcd} \tag{5.4}$$

This follows from the functorial properties of the cohomology (comp. [Har08c; p. 51]).

Definition 5.1.5: *Let $\mathcal{M}$ be a sheaf of abelian groups on $\mathcal{X}$, then the* **support of the sheaf** *$\mathcal{M}$ is given by*

$$\mathbf{supp}\,(\mathcal{M}) = \overline{\{x \in \mathcal{X} : \mathcal{M}_x \neq 0\}} \tag{5.5}$$

and the support of a section $s \in \mathcal{M}(U)$ for an open set $U \subset \mathcal{X}$ is given by

$$\mathbf{supp}\,(s) = \overline{\{x \in U : s_x \neq 0\}}, \tag{5.6}$$

where $\overline{\quad}$ indicates the closure of the set.

5.2. From Manifolds to Orbifolds

The basic objects in differential geometry are manifolds, i.e. topological spaces which are locally homeomorphic to open sets in $\mathbb{R}^k$ for some integer k together with some compatibility conditions for the homeomorphisms. The basic geometric objects in this thesis are quotients of symmetric spaces, i.e. manifolds, by a discrete properly discontinuously acting group of isometries $\mathbf{G}$ (see Section 6.2). If $\mathbf{G}$ is not torsion free the quotient need not be again a manifold. It is only a so called orbifold. In the literature there are several, sometimes equal, sometimes slightly different, definitions for objects called orbifold, thus we will give a definition of what we call an orbifold. Let in the following $\mathbf{G}$ be a discrete group of isometries of a smooth manifold $\mathbf{M}$ which acts properly discontinuously on $\mathbf{M}$.

Definition 5.2.1: *A topological space $\mathscr{O}$ is called* **orbifold** *if there is a manifold $\mathbf{M}$ with an atlas $\{\phi_i : U_i \to V_i \subset \mathbb{R}^k\}_{i \in \mathcal{I}}$ and a family of finite subgroups $\mathbf{H}_i \subset \mathbf{G}$ such that*

a) The sets U_i are invariant under $\mathbf{H}_i$.

b) The continuous maps ϕ_i are $\mathbf{H}_i$-equivariant.

c) The sets $U_i/\mathbf{H}_i$ cover $\mathscr{O}$ and fulfil the classical conditions for coordinate charts of manifolds, except that they need not to be open.

An orbifold is called smooth if the underlying manifold $\mathbf{M}$ is smooth.

This definition is equivalent to the following: An orbifold $\mathscr{O}$ is the quotient of a manifold by a finite group $\mathbf{G}_{\text{finite}}$ of isometries. The notion of an orbifold is a natural generalization of manifolds, because every manifold is an orbifold. It is easy to prove that $\mathscr{O} = \mathbf{M}/\mathbf{G}$ is a smooth orbifol, if $\mathbf{M}$ is a smooth manifold and $\mathbf{G}$ a discrete group of isometries as aboved. If the group $\mathbf{G}$ acts torsion free, it is well known that the quotient of a smooth manifold $\mathbf{M}$ by $\mathbf{G}$ is again a smooth manifold (see e.g. [Hel78]). Assume now that $\mathbf{G}$ acts with torsion, then there is a torsion free normal subgroup $\mathbf{G}' \subset \mathbf{G}$ and we have

$$\mathcal{X}/\mathbf{G} = \left(\mathcal{X}/\mathbf{G}'\right) / \left(\mathbf{G}'/\mathbf{G}\right) \tag{5.7}$$

where $\mathcal{X}/\mathbf{G}'$ is a smooth manifold, since $\mathbf{G}'$ is torsion free, and $\mathbf{G}'/\mathbf{G}$ is a finite group. This shows that $\mathcal{X}/\mathbf{G}$ is a smooth orbifold. This orbifold can – but does not have to – have in addition some quotient singularities.

5.3. Cells and Orbicells

Let $\mathcal{X}$ be a topological space. A subset $\sigma \subset \mathcal{X}$ is called a **(open) cell** if σ is homeomorphic to an open k-ball for some integer $k \geq 0$. All cells in this thesis are open (relative to the $\mathbb{R}^k$). If a cell σ is homeomorphic to a k-ball we call σ a **k-cell**. A **cell decomposition** or **cell complex** is a decomposition of $\mathcal{X}$ into a disjoint union of cells. A cell decomposition is called **locally finite** if all compact sets $K \subset \mathcal{X}$ intersect only finitely many cells in the decomposition. The closure of a cell is always in this thesis the closure in $\mathcal{X}$ (and not in some compactification of $\mathcal{X}$). A decomposition is **regular** if the closure of each cell is homeomorphic to a closed ball. If the decomposition is locally finite, the boundary of each cell is a union of finitely many lower dimensional cells.

The notion of a CW-complex (comp. [Mun84; §38]) is due to John Henry C. Whitehead and was created by him in 1949. A **CW-complex** is a Hausdorff space $\mathcal{X}$ together with a cell decomposition (into open cells) $\mathcal{X} = \bigsqcup_{i \in \mathcal{I}} \mathcal{X}_i$ with the following additional properties:

- For each cell $\mathcal{X}_i$ there is a homeomorphism $\iota_i : \mathcal{X}_i \hookrightarrow \mathcal{X}$ such that $\iota_i(\partial \mathcal{X}_i)$ is a finite union of cells in $\mathcal{X}$, each of dimension less than $\dim \mathcal{X}_i$.
- A subset of $\mathcal{X}$ is closed if and only if it meets the closure of a cell in a closed set whenever the intersection of closures is non-empty.

We also have to remember the definition of the barycentric subdivision of a cell decomposition (comp. [Mun84; §15]). The **barycentric subdivision** of $\mathcal{X} = \bigsqcup_{i \in \mathcal{I}} \mathcal{X}_i$ is an iterative procedure which we can apply to any cell complex. The easiest case is that of a simplicial complex. We start with the 0-cells in the decomposition. The subdivision of the 0-cells are the 0-cells itself. We proceed with the 1-cells. Here we add a 0-cell, i.e. a point, in the barycentre of the 1-cell, that is the middle of the 1-cell. So, in this step, we double the number of 1-cells, because we split the original 1-cell into two parts. In the next step we add a 0-cell in the barycentre of the 2-cells and connect the new 0-cells with all (also the newly created) 0-cells in the boundary of the cell it is put in. We can proceed in this way and add new 0-cells in the centres of all cells and connect them with lower dimensional cells to the cells in the boundary of the cell they are lying in. By $B(\mathcal{X})$ we denote the first barycentric subdivision of $\mathcal{X}$.

The advantage of such a refinement is that we can iteratively replace a maybe complicated cell decomposition with possibly very fancy cell shapes by finer decompositions into (usually) simpler cells. Maybe one has to repeat this procedure to get a decomposition with good properties. We only need the first barycentric subdivision (comp. Section 9.1.3).

We apply the notion of cells to the context of orbifolds. Let $\mathcal{O} = \mathcal{X}/\mathbf{G}$ be a smooth orbifold (comp. Section 5.2). A subset $\sigma \subset \mathcal{O}$ is called **orbicell** if there is a cell $\widetilde{\sigma} \subset \mathcal{X}$ such that

$$\sigma = \widetilde{\sigma}/\mathbf{G}_{\widetilde{\sigma}} \tag{5.8}$$

where $\mathbf{G}_{\widetilde{\sigma}} \subset \mathbf{G}$ is the stabilizer of $\widetilde{\sigma}$ in $\mathbf{G}$. If $\widetilde{\sigma}$ is a k-cell we call σ a **k-orbicell**. If $\mathbf{G}_{\widetilde{\sigma}}$ acts torsion free on $\widetilde{\sigma}$, the orbicell σ is also a cell in the usual sense. Thus, orbicells are a natural extension of the notion of cells to orbifolds. Nevertheless one can also define cells in the classical sense of orbifolds, but for our purpose orbicells are the better objects to consider. We use the analogous notions to the case of cells also in case of orbicells. Thus, we will deal with **orbicell decompositions**.

Let $\mathcal{X}$ be any topological space. Assume we have a decomposition $\mathcal{X} = \bigsqcup_i \mathcal{X}_i$, then we denote by

$$\mathbf{C}_k(\mathcal{X}) = \{\mathcal{X}_i : \dim \mathcal{X}_i = k\} \tag{5.9}$$

the set of k-cells or k-orbicells depending on whether $\mathcal{X}$ has a cell or an orbicell decomposition. Furthermore, we denote by

$$\mathbf{S}_k(\mathcal{X}) = \{\mathcal{X}_i : \dim \mathcal{X}_i \leq k\} \tag{5.10}$$

the **k-skeleton** of $\mathcal{X}$. If we use the term k-skeleton of a single (orbi-)cell we mean the k-skeleton of its closure in $\mathcal{X}$ (or $\mathcal{X}/\mathbf{G}$).

6. Arithmetic Groups and Symmetric Spaces

The mathematical sciences particularly exhibit order, symmetry and limitation; and these are the greatest forms of the beautiful.

(Aristoteles)

6.1. Arithmetic Groups

We assume that the reader is familiar with the general notion of algebraic groups, Lie groups and representation theory of groups. In the first section we establish some notations related to algebraic groups in general and arithmetic groups in particular before we focus on the case of $\mathbf{Sp}_2(\mathbb{Z})$. Arithmetic subgroups of classical groups and the related reduction theory, i.e. the questions related to finding fundamental domains for their group actions (see Section 7.1), were introduced by Carl Ludwig Siegel (1896–1981) around 1940 as a continuation of ideas which are much older for some special cases. Later the theory was fit in the framework of algebraic groups by Claude Chevalley (1909–1984), Armand Borel (1923–2003) and Jacques Tits (b.1930), and others. Arithmetic groups play a fundamental role not only in this thesis, but also in number theory in general and the theory of automorphic forms in particular. The most famous examples of such groups are $\mathbf{GL}_n(\mathbb{Z})$, $\mathbf{SL}_n(\mathbb{Z})$ and $\mathbf{Sp}_g(\mathbb{Z})$ and various subgroups of them. In this thesis we focus on the group $\mathbf{Sp}_2(\mathbb{Z})$ and related groups.

For this part we refer to the books of Jean-Pierre Serre [Ser73, Ser79], the book of Tony Albert Springer [Spr08], the paper of Armand Borel and Harish-Chandra [BHC61] or the survey article by Christophe Soulé [Sou07] for an introduction on algebraic groups. A quite good reference which focuses more to the case of Lie groups and their representations is the book of Daniel Bump [Bum04]. For our purpose it is enough to restrict ourselves to the case that the underlying algebraic groups are already defined over the rational numbers $\mathbb{Q}$. This is not a serious restriction, because the symplectic group we consider is already defined over the ring of integers, but the definition in case of non-fields is a little bit more fancy in detail.

In the second section we consider representations of Lie groups and introduce the basic notions related to Γ-modules and the associated sheaves which we use to describe the coefficients for the cohomology groups we want to compute. For details on Γ-modules and their relation to the cohomology of arithmetic groups we refer to [Har08c, Har75, Har08a] by Günter Harder.

6.1.1. Algebraic and Arithmetic Groups

Let $\mathscr{G}/\mathbb{Q}$ be an **algebraic group**, i.e. $\mathscr{G}$ is a functor from the category of $\mathbb{Q}$-algebras to the category of groups. We denote by $\mathbf{G} := \mathscr{G}(\mathbb{R})$ its $\mathbb{R}$-valued points of $\mathscr{G}$. For simplicity we consider $\mathscr{G}(k)$ as a subgroup of $\mathcal{G}\ell_d(k)$ for any field extension $k/\mathbb{Q}$ and some integer d, because we are interested in explicit computations and this point of view is better suited for that kind of considerations than the more abstract definition as a functor from the category of $\mathbb{Q}$-algebras to the category of groups.

Remember, a torus in $\mathscr{G}$ is an algebraic subgroup of $\mathscr{G}$ which becomes, after some field extension, isomorphic to a direct product of finitely many $\mathbf{G}_\mathrm{m}$. Let $\mathcal{T}/\mathbb{Q}$ be a maximal torus in $\mathscr{G}$, then the **rank** – or more precise the $\mathbb{Q}$-rank of $\mathscr{G}$ – is defined by

$$\operatorname{rank}_{\mathbb{Q}}(\mathscr{G}) := \dim_{\mathbb{Q}} \mathcal{T}(\mathbb{Q}) \tag{6.1}$$

An algebraic group is called **semi-simple** if the (solvable) radical of the identity component is trivial, or equivalently, if it has no connected, normal, abelian subgroups. All groups we will consider, as for example $\mathbf{Sp}_g/\mathbb{Q}$, are semi-simple algebraic groups.

Two subgroups $\Gamma_1, \Gamma_2 \subset \mathscr{G}(\mathbb{Q})$ are called **commensurable** if $\Gamma_1 \cap \Gamma_2$ has finite index in both groups Γ_1 and Γ_2. A subgroup $\Gamma \subset \mathscr{G}(\mathbb{Q})$ is called **arithmetic** if Γ is commensurable with $\mathscr{G}(\mathbb{Z})$. In particular, the group $\mathscr{G}(\mathbb{Z})$ is arithmetic.

A group is called **torsion free** if all elements different from the identity have infinite order. An element $\mathrm{g} \in \mathscr{G}(\mathbb{Q})$, where $\mathrm{g} \neq \mathrm{Id}$, is called **neat** if for a faithful complex values representation ρ the set of all eigenvalues of $\rho(\mathrm{g})$ generates a torsion free subgroup of $\mathbb{C}^\times$. A subgroup of $\mathscr{G}(\mathbb{Q})$ is called **neat** if all its elements different from Id are neat. In particular all neat arithmetic groups are torsion free. Any arithmetic group has a neat subgroup of finite index.

All algebraic groups $\mathscr{G}/\mathbb{Q}$ we consider in this thesis carry also the structure of Lie groups. To be more precise, the set of real valued points $\mathscr{G}(\mathbb{R})$ of $\mathscr{G}$ is a (real) Lie group. Remember, a group $\mathbf{G}$ is called (real) **Lie group** if $\mathbf{G}$ has the structure of a real differentiable manifold and the group action is compatible with this structure. The notions for algebraic groups and Lie groups are quite analogous. In the following, we use $\mathbf{G} := \mathscr{G}(\mathbb{R})$ for the group of real valued points.

6.1.2. Γ-Modules

Let $\mathscr{G}/\mathbb{Q}$ be an algebraic group over $\mathbb{Q}$. An **algebraic representation** $(\rho, \mathbb{M}_{\rho,\mathbb{Q}})$ over a field $k/\mathbb{Q}$ is a homomorphism of algebraic groups

$$\rho : \mathscr{G} \times_{\mathbb{Q}} k \to \mathbf{GL}(\mathbb{M}_{\rho,\mathbb{Q}} \otimes_{\mathbb{Q}} k) \tag{6.2}$$

where $\mathbb{M}_{\rho,\mathbb{Q}}$ is a finite dimensional $\mathbb{Q}$-vector space. If the group $\mathscr{G}$ is already defined over the integers, i.e. $\mathscr{G}/\operatorname{Spec}(\mathbb{Z})$ is a group scheme, we can replace this definition by: An **algebraic representation** $(\rho, \mathbb{M}_\rho)$ over $\mathbf{R}$ is a morphism

$$\rho : \mathscr{G} \times_{\operatorname{Spec}(\mathbb{Z})} \mathbf{R} \to \mathbf{GL}(\mathbb{M}_\rho \otimes_{\mathbb{Z}} \mathbf{R}) \tag{6.3}$$

where $\mathbf{R}$ is any ring extension of $\mathbb{Z}$ and $\mathbb{M}_\rho$ a free $\mathbb{Z}$-module of finite rank. If $\mathbf{R}$ is a field $k/\mathbb{Q}$ both definitions are equivalent. This construction is essentially the same as the definition of a Lie group representation. Remember that a **representation** $(\rho, \mathbb{M}_{\rho,\mathbb{R}})$ of the Lie group $\mathscr{G}(\mathbb{R})$ is a group homomorphism

$$\rho : \mathscr{G}(\mathbb{R}) \to \mathbf{GL}(\mathbf{M}_{\rho,\mathbb{R}}) \tag{6.4}$$

for a finite dimensional $\mathbb{R}$-vector space $\mathbf{M}_{\rho,\mathbb{R}}$. We make use of the fact that each vector space $\mathbb{M}_{\rho,\mathbb{R}}$ is given by an underlying free $\mathbb{Z}$-module $\mathbb{M}_\rho$, such that $\mathbb{M}_{\rho,\mathbb{R}} = \mathbb{M}_\rho \otimes_\mathbb{Z} \mathbb{R}$. Analogously, $\mathbb{M}_{\rho,\mathbb{Q}}$ is also given by a free $\mathbb{Z}$-module $\mathbb{M}_\rho$. In all these cases, we call $\mathbb{M}_\rho$ the **representation module** for ρ. Sometimes we identify the representation $(\rho, \mathbf{M}_{\rho,\mathbb{R}})$ with its underlying representation module $\mathbf{M}_\rho$. We can assume that all Lie group representations of a Lie group $\mathbf{G} = \mathscr{G}(\mathbb{R})$ come from an algebraic representation.

Furthermore, any algebraic representation $(\rho, \mathbb{M}_\rho)$ induces a natural action of the group $\mathscr{G}(\mathbf{R})$ on the module $\mathbb{M}_\rho \otimes_\mathbb{Z} \mathbf{R}$ by

$$\begin{aligned} \mathscr{G}(\mathbf{R}) \times \mathbb{M}_\rho \otimes_\mathbb{Z} \mathbf{R} &\to \mathbb{M}_\rho \otimes_\mathbb{Z} \mathbf{R} \\ (g, m) &\mapsto \rho(g) \cdot m. \end{aligned}$$

One of the most important types of objects in this thesis are so called Γ-modules for a (not necessarily arithmetic) subgroup $\Gamma \subset \mathscr{G}(\mathbf{R})$.

Definition 6.1.1 (Γ-module)**:** *Let* $\mathbf{R}$ *be some ring as above and* $\Gamma \subset \mathscr{G}(\mathbf{R})$ *be any subgroup. An* **$\mathbf{R}$-Γ-module** $\mathbb{M}$ *is an* $\mathbf{R}$*-module* $\mathbb{M}$ *together with a* Γ*-action defined by a representation* $(\rho_\mathbb{M}, \mathbb{M})$ *of* Γ*.* $\mathbf{R}$*-*Γ*-modules form a category. We denote the category of* $\mathbb{Z}$*-*Γ*-modules by* $\mathbf{Mod}_\Gamma$*. We call a* Γ*-module* **irreducible** *if there are no non-trivial* Γ*-submodules.*

Usually we assume in this thesis that $\mathbf{R}$ is the ring of integers $\mathbb{Z}$ or an extension of $\mathbb{Z}$ like the ring

$$\mathbb{Z}_\Gamma = \mathbb{Z}[\tfrac{1}{30}],$$

which is associated to the Siegel modular group Γ and which we consider later in Section 12.2 in context of the cohomology of arithmetic groups and which is the ring where we invert the orders of elements of finite order in Γ (see Definition 12.1.3). These modules form a subcategory of $\mathbf{Mod}_\Gamma$. If nothing else is mentioned we from now on always consider $\mathbb{Z}$-Γ-modules and simply call them Γ-modules.

In Chapter 11 we use Γ-modules which are given by highest weight modules (comp. Section 6.4.4) as coefficients of the cohomology groups. We define these modules in terms of Lie algebra representations. There is a strong connection between representations of Lie groups and representations of Lie algebras such that we can canonically define a group representation coming from a Lie algebra representation and the other way around (see Section 11.1).

6.2. Symmetric and Locally Symmetric Spaces

Let $\mathbf{G} = \mathscr{G}(\mathbb{R})$ be a real Lie group given by an algebraic group $\mathscr{G}/Q$ and let $\mathbf{K} \subset \mathbf{G}$ be a maximal compact subgroup of $\mathbf{G}$. Then the **symmetric space** related to $\mathscr{G}$ is defined by

$$\mathbb{X}_\mathscr{G} = \mathbb{X}_\mathbf{G} := \mathbf{K}\backslash\mathbf{G}. \tag{6.5}$$

If the underlying group is clear from its context we omit the group in the index. If $\Gamma \subset \mathscr{G}(\mathbb{Q})$ is a good[1] arithmetic group we have a natural action of Γ on $\mathbb{X}$ which is induced by the group multiplication. Therefore, we can form the quotient $\mathbb{X}/\Gamma$, which is called a **locally symmetric space** if Γ acts torsion free on $\mathbb{X}$ and therefore the quotient is a manifold. We denote the canonical projection onto this quotient usually by

$$\pi : \mathbb{X} \to \mathbb{X}/\Gamma. \tag{6.6}$$

The quotient $\mathbb{X}/\Gamma$ is in general not a manifold, but always a smooth orbifold. The quotient can have lots of singularities, which are difficult to control in general. Also if we restrict – as we can do – to consider only quotients by arithmetic groups the quotient can have a very bad structure. In particular, the volume of the quotient need not to be finite if the arithmetic group is too small. To avoid lots of trouble we can assume that Γ has **finite covolume**, i.e. $\mathbb{X}/\Gamma$ has finite volume. In case of the symplectic group we focus on this is not a restriction at all since $\mathbf{Sp}_g(\mathbb{Z})$ has finite covolume. It is known that $\mathbb{X}/\Gamma$ is a smooth manifold if the group Γ is torsion free (comp. Section 5.2). Let us now define by

$$\operatorname{vcd}(\Gamma) := \dim_{\mathbb{R}}(\mathbb{X}) - \operatorname{rank}_{\mathbb{Q}}(\Gamma) \tag{6.7}$$

the **virtual cohomological dimension** of Γ. A theorem of Serre shows that the definition is independent of the choice of Γ as arithmetic subgroup of $\mathscr{G}(\mathbb{Q})$ and thus this definition only depends on the algebraic group [Ser71]. Another way to define this number is to define it as the smallest integer q such that $H^q(\mathbb{X}/\Gamma, A)$ vanishes for all coefficients A for all degrees above q [BS73], which is a definition which is not very handy. By Theorem 12.3.4 of Borel and Serre, which we state later in the section on the cohomology of arithmetic groups, we will see that both definitions coincide.

For further details on symmetric and locally symmetric spaces we refer to the standard literature such as [Hel78]. A condensed introduction to this topic, which handles also the example we are most interested in (see Section 6.3.2), can also be found in [Bum04; Chap. 31].

6.3. Symplectic Groups and the Siegel Modular Variety

6.3.1. The Symplectic Group and the Siegel Modular Group

Now we consider the so called symplectic group which is the main object of interest in this thesis. The symplectic group was first studied by the Norwegian mathematician Niels Hendrik Abel (1802–1829). So it had been studied decades before the group[2] got its today's name which is due to Herman Weyl. Before Weyl lots of different notions were used to describe the same object. There was the notion of *abelian group* as well as *complex group* or *complex line group*. Each of them had the disadvantage that it was misleading because the used names were already used with a different meaning. That is in principle not a problem at all. In mathematics there are a lot of notions used in different contexts which have the

[1]Good will usually mean torsion free or even neat. Later we find that one can skip in many cases for our purposes assumptions like this.

[2]The formal definition of a group is due to Cayley in 1854, which means that it was not known as abstract concept at lifetime of Abel, who died in 1829.

same name, only to mention *simple*. But here it was used in more or less the same context. So Herman Weyl a new name for that object proposed in his first edition of [Wey46]. He wrote [Wey46; p. 165]:

> The name *complex group* formerly advocated by me in allusion to line complexes, as these are defined by the vanishing of antisymmetric bilinear forms, has become more and more embarrassing through collision with the word *complex* in the connotation of complex number. I Therefore, propose to replace it by the corresponding Greek adjective *symplectic*. Dickson calls the group the *abelian linear group* in homage to Abel who first studied it.
>
> Herman Weyl 1939

Within a few years this new name effectively replaced the previously used ones. The first mathematician who studied the newly named group extensively was Carl-Ludwig Siegel. His book [Sie43] is always a good reference. Other good references are for example the books of Eberhard Freitag [Fre83, Fre91] and the book of a student of Siegel named Helmut Klingen [Kli90]. But these books have their main focus on the theory of Siegel modular forms. Dedicated to questions more related to the topic of this thesis is the part by Gerard van der Geer [vdG08] in the book [Ran08]. It considers for example also Harder's conjecture (see [Har08b]), which was one motivation for this thesis (see Section 1.2). For some further information in direction of compactifications or more cohomological questions we refer to [Sch86, Nam80].

6.3.1.1. Definition of the Symplectic Group

We start with the most general definition.

Definition 6.3.1: *Let V be a free $\mathbb{Z}$-module equipped with a non-degenerate alternating bilinear form $\langle , \rangle : V \times V \to \mathbb{Z}$. Then we define the* **symplectic group** *on $(V, \langle , \rangle)$ to be*

$$\mathbf{Sp}(V, \langle , \rangle) = \mathbf{Aut}\,(V, \langle , \rangle) = \{g \in \mathbf{End}\,(V) : \langle g(v), g(w) \rangle = \langle v, w \rangle \ \forall v, w \in V\}\,.$$

The form $\langle , \rangle$ is called **symplectic form***.*

There are several choices to equip a module with a symplectic form. Furthermore, it is necessary that the rank of V is even, i.e. $\operatorname{rank}_{\mathbb{Z}}(V) = 2g$, because otherwise there is no symplectic form. Therefore, we can assume that $V = \mathbb{Z}^{2g}$. We identify the symplectic group with a matrix group, because $\mathbf{End}\,(V) = \mathbf{End}\left(\mathbb{Z}^{2g}\right) = \mathbf{GL}_{2g}(\mathbb{Z})$. To do this we define a symplectic form in a canonical way.

Let $\mathfrak{B} = (\mathbf{e}_1, \ldots, \mathbf{e}_g, \mathbf{f}_g, \ldots, \mathbf{f}_1) \subset \mathbb{Z}^{2g}$ be the set of standard basis vectors of $\mathbb{Z}^{2g}$. We define the **standard symplectic form** $\langle , \rangle : \mathbb{Z}^{2g} \times \mathbb{Z}^{2g} \to \mathbb{R}$ to be the linear extension of the following relations on the basis to a bilinear form. For all $1 \leq i, j \leq g$ we have

$$\left\langle \mathbf{e}_i, \mathbf{f}_j \right\rangle = -\left\langle \mathbf{f}_j, \mathbf{e}_i \right\rangle = \delta_{ij} \qquad \text{and} \qquad \left\langle \mathbf{e}_i, \mathbf{e}_j \right\rangle = \left\langle \mathbf{f}_i, \mathbf{f}_j \right\rangle = 0, \tag{6.8}$$

where δ_{ij} is the Kronecker δ. It is not difficult to check that this indeed defines a symplectic form. We know from linear algebra how to describe bilinear forms in a given basis by matrices. The standard symplectic form in the basis $\mathfrak{B}$ is given by

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^t \cdot J_g \cdot \mathbf{y} \qquad \text{with } J_g = \begin{pmatrix} & & & 1 \\ & & \iddots & \\ & -1 & 1 & \\ & & & \\ & \iddots & & \\ -1 & & & \end{pmatrix}. \tag{6.9}$$

For $g = 2$ we use $J_2 =: J$. Now a matrix $A \in \mathbf{GL}_{2g}(\mathbb{Z})$ is symplectic if and only if the following equation holds:

$$A^t \cdot J_g \cdot A = J_g. \tag{6.10}$$

We define for a ring extension $\mathbf{R}$ of $\mathbb{Z}$

$$\mathbf{Sp}_g(\mathbf{R}) = \mathbf{Sp}(\mathbb{Z}^{2g} \otimes_{\mathbb{Z}} \mathbf{R}, \langle,\rangle_{\mathbf{R}}) = \{A \in \mathbf{GL}_{2g}(\mathbf{R}) : A^t \cdot J_g \cdot A = J_g\} \tag{6.11}$$

where $\langle,\rangle_{\mathbf{R}}$ is the (unique) bilinear extension of $\langle,\rangle$ to $\mathbf{R}^d$, which is in terms of matrices given by the same defining antisymmetric matrix J_g as $\langle,\rangle$. Therefore, we omit the index in the following. Obviously we have $\mathbf{Aut}\left(\mathbb{R}^{2g}, J_g\right) = \mathbf{Sp}_g(\mathbb{R})$ and $\mathbf{Aut}\left(\mathbb{Z}^{2g}, J_g\right) = \mathbf{Sp}_g(\mathbb{Z})$. We call $\Gamma^g = \mathbf{Sp}_g(\mathbb{Z})$ the **Siegel modular group** (of degree g). For $g = 2$ we use only Γ instead of Γ^2. For $g = 1$ we have by definition $\mathbf{Sp}_1(\mathbf{R}) = \mathbf{SL}_2(\mathbf{R})$. It remains to introduce the **group of symplectic simmilitudes**

$$\mathbf{GSp}_g(\mathbf{R}) = \left\{A \in \mathbf{GL}_{2g}(\mathbf{R}) : A^t \cdot J_g \cdot A = \lambda \cdot J_g \text{ for a } \lambda \in \mathbf{R}^\times\right\}. \tag{6.12}$$

We introduce the usual notation and set $\Gamma_0 = \mathbf{GSp}_2(\mathbb{Z})$. Since $\mathbb{Z}^\times = \{\pm 1\}$ we easily find that

$$\Gamma_0 = \Gamma \rtimes \langle \operatorname{diag}(1, 1, -1, -1) \rangle. \tag{6.13}$$

6.3.1.2. Properties of Symplectic Groups

We want collect some properties of symplectic groups in general and the Siegel modular group (for $g = 2$) in particular. The matrix group $\mathbf{Sp}_g(\mathbb{R})$ is a real Lie group, i.e. it has the structure of a smooth real manifold and the group action is compatible with the differentiable structure. We can view the symplectic groups $\mathbf{Sp}_g$ as algebraic groups defined over the integers. The related functor is realized by the map $\mathbf{Sp}_g : B \mapsto \mathbf{Sp}_g(B)$, where B is some algebra, according to (6.11). The Siegel modular group Γ_g is a discrete algebraic subgroup of $\mathbf{Sp}_g$, which is also an arithmetic subgroup. The group $\mathbf{Sp}_g$ is as algebraic group and $\mathbf{Sp}_g(\mathbb{R})$ is as real Lie group simple and connected. But $\mathbf{Sp}_g(\mathbb{R})$ is not simply connected, so its fundamental group is isomorphic to $\mathbb{Z}$, whereas $\mathbf{Sp}_g(\mathbb{C})$ is simply connected.

By equation (6.10) one finds easily that any $A \in \mathbf{Sp}_g(\mathbb{R})$ has to have determinant ± 1. In fact, we have more:

Lemma 6.3.2: *For all $g \geq 1$ and any ring extensions $\mathbf{R}$ over $\mathbb{Z}$ we have a natural inclusion $\mathbf{Sp}_g(\mathbf{R}) \hookrightarrow \mathbf{SL}_{2g}(\mathbf{R})$. In particular, we have* $\det(A) = 1$ *for all $A \in \mathbf{Sp}_g(\mathbb{R})$.*

There are several ways to exclude -1 from being a valid value of the determinant. Later we need to compute the action of group elements on highest weight modules. To do this we have to decompose a given group element into special generators of the group which are unipotent, i.e. all eigenvalues are $+1$ (see Section 11.2.3). Therefore, we use the following theorem to prove the previous lemma.

Theorem 6.3.3: *The symplectic group* $\mathbf{Sp}_g(\mathbf{R})$ *is generated by its unipotent elements.*

The same statement is valid for all other classical groups. For the proof we refer to [GW09; Theo. 2.2.2]. The Lemma follows immediately since the special linear group is the group of all matrices of determinant $+1$. We postpone the detailed description of this generators to Section 11.2.3, where we also state an algorithm to obtain a decomposition into unipotent generators, because they are related to root systems (see Section 6.4.2).

6.3.1.3. Other Realizations

Note that in the equations (6.9) and (6.10) one can replace J_g by any other (integral) antisymmetric $2g \times 2g$-matrix of full rank. By the analogue of Sylvester's law of inertia in case of antisymmetric bilinear forms we know that in a suitable basis the defining matrix is an antisymmetric matrix with only zeroes and ± 1 as entries. The entries of the matrix are the values of the symplectic form evaluated on the basis vectors. Therefore, we may consider those forms which corresponds to the choice of a different ordering of the basis which also fulfil the relations from (6.8). This covers all standard cases of symplectic forms in literature.

We know by linear algebra that two symplectic groups given by different (integral) symplectic forms are conjugate to each other. The element with which the matrix is conjugated need not to be a symplectic matrix (with respect to one of both forms). If it is a symplectic matrix the symplectic groups defined for both forms coincide. Otherwise they are only isomorphic. In case of the group $\mathbf{Sp}_2(\mathbb{R})$ there are two classes of symplectic forms, which fall together if we are allowed to conjugate with non-symplectic elements in the general linear group.

We decide to choose the matrix J_g as matrix which defines the symplectic group, which also defines the used realization of the standard representation of the symplectic group. This choice is somewhat arbitrary, although there are some reasons to use this form and not an arbitrary one, which are related to some easier descriptions of deduced objects. It seems to be a kind of law in mathematics that whenever there is a choice between different possibilities which are in some way canonical one finds for each of them mathematicians who choose this in their work. We now have the problem that in the context of our work three different choices for a symplectic form occur. Each of them has some advantages for its context and disadvantages in other contexts. First, there is the classical representation which is used for example in the books [Sie43, Fre83, Fre91, Kli90] and very often in context of Siegel modular forms, because in that context the description is more easy. For example the action can be written in terms of block matrices *and* one can check (using only matrix multiplication, matrix addition and transposition) symplecticity in terms of the blocks. Furthermore, most things in this formulation can be written down completely analogous to the known formulas for $\mathbf{SL}_2(\mathbb{R})$. In this context the symplectic form is given by the block matrix

$$J_g^{\mathsf{var}} = \begin{pmatrix} 0 & \mathrm{Id} \\ -\mathrm{Id} & 0 \end{pmatrix}.$$

This corresponds to the same relations as in (6.8), but one uses the basis in a different order, namely $\mathfrak{B}^{\mathsf{var}} = (\mathbf{e}_1, \ldots, \mathbf{e}_g, \mathbf{f}_1, \ldots, \mathbf{f}_g)$. We do not use this form. Finally we have a third basis $\mathfrak{B}' = (\mathbf{f}_g, \mathbf{e}_1, \mathbf{f}_{g-1}, \mathbf{e}_2, \ldots, \mathbf{f}_2, \mathbf{e}_{g-1}, \mathbf{f}_1, \mathbf{e}_g)$ which occurs in the GAP library to model Lie algebras. This basis has the advantage that the matrices representing positive and negative root vectors are upper or lower triangular matrices, depending on whether they are associated to positive or negative roots. In this basis the symplectic form is given by the matrix

$$J'_g = \begin{pmatrix} & & & & & & & (-1)^g \\ & & & & & & \iddots & \\ & & & & & -1 & & \\ & & & & 1 & & & \\ & & & -1 & & & & \\ & & 1 & & & & & \\ & \iddots & & & & & & \\ -(-1)^g & & & & & & & \end{pmatrix}.$$

It is easy to transfer a symplectic group given by a form $J^{(1)}$ to $J^{(2)}$. This is essentially the well known base change for matrices and bilinear forms.

Lemma 6.3.4 (Base change)**:** *Let $J^{(1)}$ and $J^{(2)}$ be two integral matrices defining each a symplectic form. Assume that*

$$A^t \cdot J^{(1)} \cdot A = J^{(2)}.$$

Then we have

$$\mathbf{Sp}(\mathbb{R}^{2g}, J^{(1)}) = A \cdot \mathbf{Sp}(\mathbb{R}^{2g}, J^{(2)}) \cdot A^{-1}$$

We need this in some internal parts of our SAGE program later on, where we have to put the computation of stabilizers and cells. This part of the program uses a definition of the symplectic group by J_g, together with highest weight modules computed with GAP, which produces elements which are symplectic with respect to J'_g. As standard we use the symplectic elements defined by J_g and, if necessary, we convert the elements to the needed form.

6.3.2. Siegel Upper Half-Space and Siegel Modular Varieties

We now consider the symmetric space associated to the symplectic group $\mathbf{Sp}_g(\mathbb{R})$. We know that the unitary group $\mathbf{U}(g) \subset \mathbf{Sp}_g(\mathbb{R})$ (see e.g. [vdG08; p. 185]) is a maximal compact subgroup of the symplectic group. All other maximal compact subgroups are conjugated to $\mathbf{U}(g)$. We call the (hermitian) symmetric space

$$\mathfrak{S}_g = \mathbf{U}(g)\backslash\mathbf{Sp}_g(\mathbb{R}) \tag{6.14}$$

the **Siegel upper half-space**. The Siegel upper half-space is a simply connected complex manifold. This manifold is isomorphic to the space symmetric $g \times g$ matrices with positive definite imaginary part, i.e. we have

$$\mathfrak{S}_g \cong \left\{ S \in \mathrm{Mat}_g(\mathbb{C}) : S = S^t, \mathrm{Im}\,(S) > 0 \right\}. \tag{6.15}$$

There is another quite useful description of the Siegel upper half-space in terms of quadratic forms (see also Chapter 8).

Lemma 6.3.5: *If $\mathscr{Q}^{2g}_{>0}$ is the space of positive definite quadratic forms of dimension $2g$ (see Section 8.1 for its definition) then we have*

$$\mathfrak{S}_g = \left\{ Q \in \mathscr{Q}^{2g}_{>0} : Q^t \cdot J_g \cdot Q = J_g \right\}.$$

The proof is not difficult. One direction follows immediately from [MM93; (1.7)] and the other direction is a simple exercise in linear algebra (comp. [MM93; p. 589]).

There is a natural action of $\mathbf{Sp}_g(\mathbb{R})$, and therefore also of its discrete subgroup Γ^g, on $\mathfrak{S}_g$, which can be written – in a suitable basis[3] – as fractional linear operation on the set of symmetric matrices from (6.15). One finds, similar to the case of the group $\mathbf{SL}_2(\mathbb{R})$, that $\pm\mathrm{Id}$ act trivially on $\mathfrak{S}_g$. Therefore, the exact group of automorphisms of $\mathfrak{S}_g$ is not $\mathbf{Sp}_g(\mathbb{R})$, but the the group

$$\mathbf{PSp}_g(\mathbb{R}) := \mathbf{Sp}_g(\mathbb{R})/\{\pm\mathrm{Id}\}, \tag{6.16}$$

which is called the **projective symplectic group**. For the group action on $\mathfrak{S}_g$, there is no difference between the action of Γ^g and the action of $\mathbf{PSp}_g(\mathbb{Z})$. We now form the quotient $\mathfrak{S}_g/\Gamma^g$, which is called the **Siegel modular variety** of dimension g. Since Γ^g does not act torsion free on $\mathfrak{S}_g$, the quotient is a smooth complex orbifold of dimension $g(g+1)/2$ (comp. Section 5.2). If we replace Γ^g by a suitable torsion free subgroup $\Gamma' \subset \Gamma^g$, the quotient $\mathfrak{S}_g/\Gamma'$ becomes a smooth complex manifold. In other words, $\mathfrak{S}_g/\Gamma'$ is a locally symmetric space. We know by Siegel [Sie43, Sie55] that the volume of a fundamental domain of Γ^g, or equivalently of the quotient $\mathfrak{S}_g/\Gamma^g$, is finite and has the value

$$\mathrm{Vol}\left(\mathfrak{S}_g/\Gamma^g\right) = 2 \cdot \pi^{-\frac{1}{2}g(g+1)} \prod_{k=1}^{g} (k-1)! \cdot \zeta(2k) \tag{6.17}$$

where ζ is the Riemann ζ-function. For $g = 2$ we get as volume

$$\mathrm{Vol}\left(\mathfrak{S}_2/\Gamma\right) = 2 \cdot \pi^{-3} \cdot \zeta(2) \cdot \zeta(4) = \frac{\pi^3}{270}. \tag{6.18}$$

There are other ways to normalize the volume such that for torsion free groups the volume equals the Euler number of the manifold.

6.3.3. Elements of Finite Order in the Symplectic Group

In 1979 Balz Bürgisser (b. 1953) received his PhD at the ETH Zürich with a thesis [Bür79] wherein and in a sequel paper [Bür82] he constructed elements of finite order in some matrix groups of Lie type. He considered the groups $\mathbf{GL}_d(\mathbf{R})$, $\mathbf{SL}_d(\mathbf{R})$ and $\mathbf{Sp}_d(\mathbf{R})$ for an integrally closed domain $\mathbf{R}$ and showed that all possible finite orders of elements in $\mathbf{GL}_{2d}(\mathbf{R})$ also occur in $\mathbf{Sp}_d(\mathbf{R})$. In particular, all elements of finite order in $\mathbf{SL}_d(\mathbf{R})$ are also in $\mathbf{Sp}_d(\mathbf{R})$. Furthermore, he stated a criterion which allows to decide easily whether an order can occur in a group. We can use this criterion to check the plausibility of the results on stabilizers in the symplectic group later on. The result was partially known before in some special cases, but not in that generality which Bürgisser obtained. Similar ideas occur also in context of

[3]This is the basis in which the symplectic form is given by J^{var}_g.

the study of non-trivial automorphism groups of principally polarized abelian varieties (see Section 6.3.4).

The general idea to obtain the result is to look at the eigenvalues and the possible characteristic polynomials. One easily finds that the eigenvalues have to have absolute value 1 in all cases of matrices of finite order. Therefore, the characteristic polynomial χ has to be a product of cyclotomic polynomials $\Phi_{d_i} \in \mathbb{Z}[X]$, which have the primitive d_i-th roots of unity as roots. Since we do not need the results in complete generality, we only state the result for $\mathbf{Sp}_g(\mathbb{Z})$. In this case we have:

Theorem 6.3.6 (Bürgisser): *Let $m = \prod_{i=1}^k p_i^{\nu_i}$ for p_i prime and integers $\nu_i \geq 1$. We assume that $p_i < p_{i+1}$ for all $1 \leq i \leq k$ and set*

$$\widetilde{\varphi}(m) = \begin{cases} \sum_{i=2}^k \varphi(p_i^{\nu_i}) & \text{if } m \equiv 2 \bmod 4 \\ \sum_{i=1}^k \varphi(p_i^{\nu_i}) & \text{else} \end{cases}$$

where φ is the Euler φ-function. Then there is an element $A \in \mathbf{Sp}_g(\mathbb{Z})$ of order m if and only if $\widetilde{\varphi}(m) \leq 2g$.

The difference in the definition of $\widetilde{\varphi}$ between the case of the doubles of odd integers and the remaining cases has its origin in the action of -1 on roots of unity, i.e. if d is odd we have $\Phi_d(-X) = \Phi_{2d}(X)$. From this theorem we easily find that:

Corollary 6.3.7 (Finite Orders): *If $A \in \mathbf{Sp}_1(\mathbb{Z}) = \mathbf{SL}_2(\mathbb{Z})$ has finite order, then its order is $1, 2, 3, 4$ or 6. If $A \in \mathbf{Sp}_2(\mathbb{Z})$ has finite order, then its order is $1, 2, 3, 4, 5, 6, 8, 10$ or 12.*

The study of the properties of the function $\widetilde{\varphi}$ is a quite interesting business, but not topic of this thesis.

6.3.4. The Symplectic Group and the Moduli Space of Abelian Surfaces

In this section we collect some results on the connection of abelian surfaces and the symmetric space for the symplectic group. We use these results in Chapter 10 to check whether the results concerning the stabilizers can be correct.

For a general reference on abelian varieties we refer to the classic book of David Mumford [Mum70] on abelian varieties or the book of Christina Birkenhake and Herbert Lange [BL04] on complex abelian varieties, which is somehow closer to the topics we are interested in. There are several resources on special automorphism structures of abelian varieties which apply (partially) to our problem. The most useful of them are [HN65, BGL99a, BGL99b, BL94a, LR04a, LR04b, Gon09, GMG01, GMG04, GMG05, GR99, Fuj88].

Related to the more special questions, which we have to answer, there is the PhD-thesis [Sch97] of Dieter Schmidt, a student of Herbert Lange and Christina Birkenhake in Erlangen. In his thesis he discusses the automorphism structure of (complex) 2 and 3-dimensional abelian varieties. There seems to be an older, never published preprint of Shi-Shyr Roan [Roa90], which is cited at several places. Unfortunately it is not true that this was, as many papers refer, a preprint of the MPI for Mathematics in Bonn, although Roan was in Bonn at that time. However one is able to reconstruct the content from the citations in [BL04, Sch97]. Also useful are [Str10, BLS11] of Marco Streng, who studied so called (ℓ,ℓ)-endomorphisms which are exactly what we are interested in for $\ell = 1$, which is a case he originally did not consider.[4]

The basic idea is that one can identify the Siegel modular variety $\mathfrak{S}_g/\Gamma^g$ with the moduli space of principally polarized abelian varieties (comp. e.g. [BL04; pp. 204]), i.e. each point in $\mathfrak{S}_g/\Gamma^g$ corresponds to an isomorphism class of principally polarized abelian varieties. If we replace the group Γ^g by a (congruence) subgroup we get the moduli space with some additional level structure (see e.g. [BL04; pp. 216]). Here we only consider the case where $g = 2$, i.e. we only have principally polarized abelian surfaces. For simplicity we skip principally polarized in the following, since all abelian surfaces are principally polarized. Each point in $\mathfrak{S}_2/\Gamma$ corresponds to an isomorphism class of abelian surfaces. Each isomorphic abelian surface has isomorphic automorphism groups. It is not difficult to check that there is (up-to isomorphism) a one-to-one correspondence between stabilizers of a point $x \in \mathfrak{S}_2/\Gamma$ and the automorphism group of an abelian surface of the isomorphism class of abelian surfaces represented by x. For our purpose it is enough to get a complete list of all possible groups which may occur as stabilizer of a point in $\mathfrak{S}_2$. Most points have trivial stabilizers, i.e. they are equal to the group $\{\mathrm{Id}, -\mathrm{Id}\}$. The question is how to get a list of the remaining groups.

The general idea is not that difficult. One uses the fact that each abelian surface is a complex torus, i.e. it has the form $\mathbb{C}^2/\Lambda$, where $\Lambda \cong \mathbb{Z}^4$ is a suitable lattice compatible with the polarization. It is not difficult to see that the abelian surfaces which admit a non-trivial automorphism structure have to be abelian surfaces with complex multiplication (CM) (see [BL04; Cor.13.3.3]). The theory of complex multiplication was extensively studied by Gorō Shimura and his students (see e.g. [Shi63, Shi71, Shi98] for a definition). The CM-type Φ of an abelian surface A is a pair of pairwise non complex conjugated embeddings of a totally imaginary algebraic number field K/Q of degree four into the complex numbers $\mathbb{C}$, i.e. we can identify Φ with a pair of the form $\Phi = \{\alpha_1, \alpha_2\}$, where $\alpha_i \in \mathbb{C}$ and $\alpha_1 \neq \overline{\alpha_2}$. Let K/Q be a field as above and let Φ be a CM-type. Then it is not difficult to find out that the units $\mathscr{O}_K^\times$ are automorphisms of the lattice $\mathscr{O}_K \subset \mathbb{C}^2$ and thus of the abelian surface

$$A = A(K,\Phi) := K \otimes_{\mathbb{Q}} \mathbb{R}/\mathscr{O}_K,$$

where the tensor product is given by two embeddings of the CM-type. Further, we have that the diagonal matrix $\mathrm{diag}\,(\alpha_1, \alpha_2)$ is an automorphism of A. We should stress that not all automorphisms are units in $\mathscr{O}_K^\times$. One finds that all occurring fields K in this context have to be cyclotomic fields $K = \mathbb{Q}(\zeta_k)$, i.e. they are generated by a primitive root of unity $\zeta_k = \exp(2\pi \mathbb{i}/k)$ for some k. For the k only a few integers may occur, namely some of those which occur in the list stated in Corollary 6.3.7. A pair (A, f), where A is an abelian surface and $f \in \mathbf{Aut}\,(A) - \{\pm\mathrm{Id}\}$, is up to isomorphisms uniquely determined by the CM-type of the abelian variety A together with the eigenvalues of f, which can represent a CM-type as we saw above (see [BL04; Section 13.3]). In all case we identify f with its complex representation in $\mathbf{GL}_2(\mathbb{C})$. If one combines the above made observations one gets a classification of abelian surfaces with non-trivial automorphisms.

One finds that there are up to isogeny three different cases (comp. [BL04; Prop. 13.4.1]):

[4]We figured out together with Marco Streng, that all of his construction works fine for $\ell = 1$ and thus are applicable in our case.

No.	Variety A	$\mathbf{End}_{\mathbb{Q}}(A)$
1	A is simple	Skewfield of degree 1,2 or 4 over $\mathbb{Q}$
2	$E_1 \times E_2$ E_1 and E_2 are non-isogenous elliptic curves.	$K_1 \oplus K_2$ $K_i = \mathbf{End}_{\mathbb{Q}}(E_i) \subseteq \mathbb{Q}(\sqrt{-d_i})$
3	$E \times E$ E elliptic curve	$\mathrm{Mat}_2(K)$ $K = \mathbf{End}_{\mathbb{Q}}(E) \subseteq \mathbb{Q}(\sqrt{-d})$

To be isogenous to a product of elliptic curves does not mean to be isomorphic to a product of elliptic curves, because it might happen that there are some relations given by an isogeny. If there is such a relation the abelian surface does not split and is isomorphic to the product of the two curves modulo the isogeny. Let

$$E_\tau := \mathbb{C}/ < 1, \tau >_{\mathbb{Z}} . \tag{6.19}$$

Then up to isomorphism the only elliptic curves E_τ which have non-trivial automorphisms are those curves given by $\tau \in \{\mathbb{i}, \sqrt{-2}, \zeta_3\}$. One easily finds the following lemma (see [BL04; Cor. 13.3.5]):

Lemma 6.3.8: *Suppose f is an automorphism of an abelian surface A of order $d \geq 3$ where f acts by multiplication with a scalar, i.e. $f = a \cdot \mathrm{Id}$. Then $a = \zeta_d$ with $d = 3, 4$ or 6 and*

$$A = E \times E$$

where E is isomorphic to either $\mathbb{C}/\mathbb{Z}[\zeta_4]$ or $\mathbb{C}/\mathbb{Z}[\zeta_6]$, i.e. an elliptic curve admitting an automorphism of order d.

The automorphisms are then generated by ζ_3, ζ_4, or ζ_6 (see [BL04; Cor. 13.3.4]. The next case which might occur is covered by [BL04; Cor.13.3.6].

Theorem 6.3.9: *Let A be an abelian surface with an automorphism f of order $d \geq 3$. Let $\Phi_f = \{\alpha_1, \alpha_2\}$, where the $\alpha \in \mathbb{C}$ are eigenvalues of f. Suppose all eigenvalues of f in Φ_f are primitive and $\varphi(d) = 4$. Then A is isomorphic to $A(\mathbb{Q}(\zeta_d), \Phi_f)$ with d and CM-type Φ_f as in Table 6.1.*

One finds that (see [BL04; Cor.13.3.7, Rem. 13.3.8, Theo. 13.4.2]:

Theorem 6.3.10:

a) If an abelian surface A has an automorphism of order 5, then $A \cong A(\mathbb{Q}(\zeta_5), \{\zeta_5, \zeta_5^2\})$.

b) An abelian surface is simple if and only if it has an automorphism of order 5.

c) The abelian surface $A(\mathbb{Q}(\zeta_5), \{\zeta_5, \zeta_5^2\})$ is isomorphic to the Jacobian of the curve $y^2 = x(x^5 - 1)$.

d) We have

$$\mathbf{Aut}\left(A(\mathbb{Q}(\zeta_5), \{\zeta_5, \zeta_5^2\})\right) = \left\langle \begin{pmatrix} \zeta_{10} & \\ & \zeta_{10}^3 \end{pmatrix} \right\rangle = C_{10}.$$

Thus, it remains to consider the non-simple abelian surfaces which are isogenous to a product of elliptic curves. This fixes the abelian surface only up to isogeny, but in one isogeny class

Table 6.1.: *Abelian surfaces A with a non-trivial automorphism f of degree d with CM-type Φ_f from Theorem 6.3.9. In the last column we indicate whether the abelian surface is simple or not. This last column is not part of Theorem 6.3.9.*

d	A	f	Φ_f	simple
5	$A(\mathbb{Q}(\zeta_5), \{\zeta_5, \zeta_5^2\})$	$\operatorname{diag}\left(\zeta_5, \zeta_5^2\right)$	$\{\zeta_5, \zeta_5^2\}$	Yes
8	$E_{\sqrt{-2}} \times E_{\sqrt{-2}}$	$\begin{pmatrix} \sqrt{-2} & -1 \\ -1 & 0 \end{pmatrix}$	$\{\zeta_8, \zeta_8^3\}$	No
	$E_{\mathbb{i}} \times E_{\mathbb{i}}$	$\begin{pmatrix} 0 & \mathbb{i} \\ 1 & 0 \end{pmatrix}$	$\{\zeta_8, \zeta_8^5\}$	No
10	$A(\mathbb{Q}(\zeta_5), \{\zeta_5, \zeta_5^2\})$	$\operatorname{diag}\left(\zeta_{10}, \zeta_{10}^3\right)$	$\{\zeta_{10}, \zeta_{10}^3\}$	Yes
12	$E_{\mathbb{i}} \times E_{\mathbb{i}}$	$\mathbb{i} \cdot \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$	$\{\zeta_{12}, \zeta_{12}^5\}$	No
	$E_{\zeta_3} \times E_{\zeta_3}$	$\zeta_3^2 \cdot \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$	$\{\zeta_{12}, \zeta_{12}^7\}$	No

Table 6.2.: *Isogeny classes of abelian surface A with automorphism group G from Theorem 6.3.11. E and E' are generic elliptic curves with trivial automorphism groups isomorphic to C_2.*

G	Structure	A
$\left\langle \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \right\rangle$	$C_2 \times C_2$	$E \times E'$
$\left\langle \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & \mathbb{i} \end{pmatrix} \right\rangle$	$C_2 \times C_4$	$E \times E_{\mathbb{i}}$
$\left\langle \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & \zeta_6 \end{pmatrix} \right\rangle$	$C_2 \times C_6$	$E \times E_{\zeta_3}$
$\left\langle \begin{pmatrix} \mathbb{i} & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & \zeta_6 \end{pmatrix} \right\rangle$	$C_4 \times C_6$	$E_{\mathbb{i}} \times E_{\zeta_3}$

there is one maximal group, such that all occurring automorphism groups of surfaces in that class are isomorphic to a subgroup of this group. Let us start with the case where the two curves are non-isogenous. Then we have (see [BL04; 13.4.4])

Theorem 6.3.11: *Let $G = \mathbf{Aut}(A)$ be the automorphism group of an abelian surface A. Assume that G is finite and maximal in its isogeny class of abelian surfaces. If A is isogenous to a product of two non-isogenous elliptic curves then (A, G) is contained in Table 6.2.*

The last case is then given by (see [BL04; 13.4.5]):

Theorem 6.3.12: *Let $G = \mathbf{Aut}(A)$ be the automorphism group of an abelian surface A. Assume that G is finite and maximal in its isogeny class of abelian surfaces. If A is isogenous to a product $E \times E$ of two elliptic curves then (E, G) is contained in Table 6.3.*

This completes the list of possible groups. For more details of the proof and further constructions we refer to the literature cited above.

Table 6.3.: *Isogeny classes of elliptic curves E together with $G = \mathbf{Aut}\,(E \times E)$ from Theorem 6.3.12.*

No.	Structure	G	$\#G$	E
1	D_8	$\left\langle \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \right\rangle$	8	any
2	D_{12}	$\left\langle \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \right\rangle$	12	any
3	$C_8 \rtimes C_2$	$\left\langle \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \frac{1}{\sqrt{-2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \right\rangle$	16	$E_{\sqrt{-2}}$
4	$C_{12} \rtimes C_2$	$\left\langle \mathbb{i} \cdot \begin{pmatrix} 1 & 1 \\ -1 & 0 \end{pmatrix}, \begin{pmatrix} -1 & -1 \\ 0 & 1 \end{pmatrix} \right\rangle$	24	$E_{\mathbb{i}}$
5	$C_{12} \cdot C_2$	$\left\langle \mathbb{i} \cdot \begin{pmatrix} 1 & 1 \\ -1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & \mathbb{i} \\ \mathbb{i} - 1 & -1 \end{pmatrix} \right\rangle$	24	$E_{\mathbb{i}}$
6	$(C_4 \times C_4) \rtimes C_2$	$\left\langle \mathbb{i} \cdot \mathrm{Id}, \begin{pmatrix} \mathbb{i} & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 1 & 0 \end{pmatrix} \right\rangle$	32	$E_{\mathbb{i}}$
7	$(Q_8 \rtimes C_3) \rtimes C_2$	$\left\langle \begin{pmatrix} 1 & 1 \\ -2 & 1 \end{pmatrix}, \begin{pmatrix} -1 - \sqrt{-2} & -1 \\ 2 \cdot \sqrt{-2} & 1 + \sqrt{-2} \end{pmatrix}, \begin{pmatrix} -1 & \frac{\sqrt{-2}}{2} \\ \sqrt{-2} & 0 \end{pmatrix}, \begin{pmatrix} -1 & 0 \\ 2 + \sqrt{-2} & 1 \end{pmatrix} \right\rangle$	48	$E_{\sqrt{-2}}$
8	$(C_6 \times C_6) \rtimes C_2$	$\left\langle -\zeta_3 \cdot \mathrm{Id}, \begin{pmatrix} -\zeta_3 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \right\rangle$	72	E_{ζ_3}
9	$(Q_8 \rtimes C_3) \rtimes C_3$	$\left\langle \zeta_3 \cdot \mathrm{Id}_2, \begin{pmatrix} 1 + 2 \cdot \zeta_3 & -1 \\ -2 & -1 - 2 \cdot \zeta_3 \end{pmatrix}, \begin{pmatrix} -1 & \zeta_3^2 \\ -2 \cdot \zeta_3 & 1 \end{pmatrix}, \begin{pmatrix} -1 & -\zeta_3 \\ -2 \cdot \zeta_3 & 2 + \zeta_3 \end{pmatrix} \right\rangle$	72	E_{ζ_3}
10	$(Q_8 \rtimes C_3) \cdot C_3$	$\left\langle \begin{pmatrix} \mathbb{i} & 0 \\ 0 & -\mathbb{i} \end{pmatrix}, \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \frac{\mathbb{i}-1}{2} \begin{pmatrix} \mathbb{i} & -\mathbb{i} \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 0 & \mathbb{i} \\ 1 & 0 \end{pmatrix} \right\rangle$	96	$E_{\mathbb{i}}$

6.4. Lie Algebras and Highest Weight Modules

We assume that the reader is familiar with the basic notions from representation theory, root systems and Lie algebras, which can be found in the classical book of Hermann Weyl [Wey46] or the more recent books of William Fulton, Joe Harris, Roe Goodman, and Nolan Wallach [FH91, GW98, GW09] as well as the standard book of James E. Humphreys on Lie algebras [Hum78]. A reference that focuses more on Lie groups is the book of Daniel Bump [Bum04].

6.4.1. Lie-Algebras

Let $\mathbb{K}$ be either $\mathbb{R}$ or $\mathbb{C}$. Remember, a **Lie algebra** over the field $\mathbb{K}$ is a $\mathbb{K}$-vector space $\mathfrak{g}$ with an antisymmetric bilinear product

$$\begin{aligned}[,] : \mathfrak{g} \times \mathfrak{g} &\to \mathfrak{g} \\ (X, Y) &\mapsto [X, Y]\end{aligned} \tag{6.20}$$

which satisfies the **Jacobi identity**, i.e.

$$[X, [X, Z]] + [Y, [Z, X]] + [Z, [X, Y]] = 0 \tag{6.21}$$

holds for all $X, Y, Z \in \mathfrak{g}$. The bilinear map is called **Lie bracket**. A non-abelian[5] Lie algebra is called **simple** if it contains no non-trivial ideals. A Lie algebra is called **semi-simple** if it is the direct sum of simple Lie algebras. In the following, we only consider Lie subalgebras of the matrix algebra $\mathrm{Mat}_d(\mathbb{K})$, which becomes the Lie algebra $\mathfrak{gl}_d(\mathbb{K})$ together with the usual **commutator** $[X, Y] = X \cdot Y - Y \cdot X$.

Any (real) Lie group $\mathbf{G}$ defines an associated Lie algebra $\mathfrak{g} = \mathbf{Lie}(\mathbf{G})$. In our context this can be done quite explicitly. Assume we can realize the Lie group $\mathbf{G}$ as a subgroup of $\mathbf{GL}_d(\mathbb{R})$ for some integer d, then

$$\mathbf{Lie}(\mathbf{G}) = \{X \in \mathrm{Mat}_d(\mathbb{R}) : \exp(t \cdot X) \in \mathbf{G}\, \forall t \in \mathbb{R}\}. \tag{6.22}$$

We will write $\exp(\mathfrak{g}) := \mathbf{G}$ if $\mathfrak{g} = \mathbf{Lie}(\mathbf{G})$. But take care if $\mathbf{G}$ is not abelian or if the $X, Y \in \mathfrak{g}$ do not commute, i.e. $[X, Y] \neq 0$, $\exp$ is not a homomorphism, because

$$\exp(X + Y) \neq \exp(X) \cdot \exp(Y).$$

In this case one can apply the Baker-Campell-Hausdorff formula to evaluate the exponential series which involves lots of commutators (see e.g. [Hau06]). Unfortunately the result in these cases is usually an infinite series expansion. So we will try to avoid this. For the general formalism concerning the relation between Lie algebras and Lie groups we refer to the literature (see e.g. [Bum04, GW09]).

Recall that a **Cartan subalgebra** $\mathfrak{h}$ of a Lie algebra $\mathfrak{g}$ is a nilpotent[6] subalgebra of $\mathfrak{g}$ which is self-normalizing.[7] Each Cartan algebra of a semi-simple Lie algebra is an abelian Lie algebra.

[5] A Lie algebra $\mathfrak{g}$ is called **abelian** if for all possible $X, Y \in \mathfrak{g}$ we have $[X, Y] = 0$.
[6] A Lie algebra $\mathfrak{g}$ is called **nilpotent** if the series $\mathfrak{g} > [\mathfrak{g}, \mathfrak{g}] > [[\mathfrak{g}, \mathfrak{g}], \mathfrak{g}] > [[[\mathfrak{g}, \mathfrak{g}], \mathfrak{g}], \mathfrak{g}] > \ldots$ becomes zero.
[7] A Lie subalgebra $\mathfrak{h}$ of $\mathfrak{g}$ is called **self-normalizing** if it follows that $\mathbf{H}_2 \in \mathfrak{h}$ if $[\mathbf{H}_1, \mathbf{H}_2] \in \mathfrak{h}$ for all $\mathbf{H}_1 \in \mathfrak{h}$.

In case of subalgebras of a matrix algebra over an algebraically closed field a Cartan subalgebra consists only of diagonalizable matrices, i.e. it is conjugate to an algebra consisting only of diagonal matrices. We will only consider maximal Cartan subalgebras. One finds that if $\mathfrak{h} \subset \mathfrak{g}$ is a (maximal) Cartan algebra and $\mathfrak{g} = \mathbf{Lie}\,(\mathbf{G})$ for some Lie group $\mathbf{G}$ then there is a (maximal) torus $\mathbf{H} \subset \mathbf{G}$ such that $\mathfrak{h} = \mathbf{Lie}\,(\mathbf{H})$.

We can now state the symplectic Lie algebra $\mathfrak{sp}_g(\mathbb{K}) = \mathbf{Lie}\,\big(\mathbf{Sp}_g(\mathbb{K})\big)$ in a quite explicit way. Using the definition of the symplectic group given by the symplectic form defined by J_g (see equation (6.10)) we get

$$\mathfrak{sp}_g(\mathbb{K}) = \Big\{X \in \mathrm{Mat}_{2g}(\mathbb{K}) : J_g \cdot X = -X^t \cdot J_g\Big\}. \tag{6.23}$$

Therefore, we get as Cartan algebra

$$\mathfrak{h} = \big\{\mathrm{diag}\,\big(a_1, \ldots, a_g, -a_g, \ldots, -a_1\big) : a_i \in \mathbb{K}\big\} \tag{6.24}$$

which is associated to the maximal torus

$$\mathbf{H} = \Big\{\mathrm{diag}\,\Big(x_1, \ldots, x_g, x_g^{-1}, \ldots, x_1^{-1}\Big) : x_i \in \mathbb{K}^\times\Big\}. \tag{6.25}$$

Note: If the symplectic form which is used to define the symplectic group changes, this description can also change in some cases (comp. Section 6.3.1.3).

We remember that the symplectic Lie algebra is a simple Lie algebra and therefore in particular semi-simple.

Any Lie algebra $\mathfrak{g}$ acts on itself by multiplication from the left using the Lie bracket. This action is nothing else than the action of the so called **adjoint representation**, which is given by sending $X \in \mathfrak{g}$ to ad_X which is defined to be

$$\begin{aligned} \mathrm{ad}_X : \mathfrak{g} &\to \mathfrak{g} \\ Y &\mapsto \mathrm{ad}_X(Y) := [X, Y]. \end{aligned} \tag{6.26}$$

This is a **Lie algebra representation**, i.e. a Lie algebra homomorphism

$$\rho : \mathfrak{g} \to \mathfrak{gl}(\mathbb{M}_\rho) \tag{6.27}$$

for some $\mathbb{K}$-vector space $\mathbb{M}_\rho$, which means that

$$\rho([X, Y]) = [\rho(X), \rho(Y)] = \rho(X) \cdot \rho(Y) - \rho(Y) \cdot \rho(X). \tag{6.28}$$

This definition is completely analogous to that of representations of groups (comp. Section 6.1.2). Thus, we can replace again the the representation space over $\mathbb{K}$ by an underlying free $\mathbb{Z}$-module $\mathbb{M}_{\rho,\mathbb{Z}}$ for which $\mathbb{M}_\rho = \mathbb{M}_{\rho,\mathbb{Z}} \otimes_{\mathbb{Z}} \mathbb{K}$ holds.

Let from now on $\mathbb{K} = \mathbb{R}$ and $\mathfrak{g} = \mathbf{Lie}\,(\mathbf{G})$ where $\mathbf{G} = \mathscr{G}(\mathbb{R})$ for some algebraic group $\mathscr{G}$, i.e. $\mathbf{G}$ is a real Lie group.

Finally we have to recall the definition of the universal enveloping algebra $\mathfrak{U}(\mathfrak{g})$ of a Lie algebra $\mathfrak{g}$. We only give its direct definition and refer for topics like its universal property and further nice features to the standard books (see e.g. [Hum78; V.]). Let $T(\mathfrak{g}) = \bigoplus_{n \geq 0} \mathfrak{g}^{\otimes d}$ be the tensor algebra of $\mathfrak{g}$. By its definition $T(\mathfrak{g})$ has the structure of a $\mathfrak{g}$-module (see Section 6.4.4), which gets its $\mathfrak{g}$-structure by the natural extension of the adjoint representation to $T(\mathfrak{g})$. We have a natural embedding $i : \mathfrak{g} \hookrightarrow T(\mathfrak{g})$ and identify $\mathfrak{g}$ with its image $i(\mathfrak{g})$ in $T(\mathfrak{g})$. Let $\mathfrak{I}$ be the two-sided ideal in $T(\mathfrak{g})$ generated by elements of the form $a \otimes b - b \otimes a - [a, b]$ for all $a, b \in \mathfrak{g}$. Then the **universal enveloping algebra** $\mathfrak{U}(\mathfrak{g})$ of the Lie algebra $\mathfrak{g}$ is defined to be

$$\mathfrak{U}(\mathfrak{g}) := T(\mathfrak{g})/\mathfrak{I}. \tag{6.29}$$

The natural map $i : \mathfrak{g} \hookrightarrow T(\mathfrak{g})$ descends to a Lie algebra homomorphism from $\mathfrak{g}$ to $\mathfrak{U}(\mathfrak{g})$. If $\mathfrak{g}$ is the Lie algebra coming from a Lie Group $\mathbf{G}$ we can identify the Lie algebra with the space of left invariant vector fields on $\mathbf{G}$, which is isomorphic to the tangent space at the identity element in $\mathbf{G}$ (see e.g. [Bum04; Chap. 7]). In this differential geometric language, we can also identify the universal enveloping algebra with the algebra of left-invariant differential operators on $\mathbf{G}$ (see e.g. [Bum04; Chap. 10]).

6.4.2. Roots and Weights

Let $\mathfrak{g} = \mathbf{Lie}(\mathbf{G})$ for some (connected) semi-simple[8] Lie group $\mathbf{G}$ and let us fix a maximal torus $\mathbf{H}$ in $\mathbf{G}$. We denote the associated Cartan algebra by $\mathfrak{h} = \mathbf{Lie}(\mathbf{H})$. All sets defined in the following depend on the choice of the maximal torus. Usually one chooses the maximal torus to be the subgroup of diagonal matrices in $\mathbf{G}$ (comp. (6.25)), but it can be easily replaced by an arbitrary conjugate of this group in $\mathbf{G}$. We consider $\mathfrak{g}$ as Lie subalgebra of $\mathfrak{gl}_d(\mathbb{R}) = \mathrm{Mat}_d(\mathbb{R})$ for some integer d. Therefore, we can define a symmetric bilinear pairing on $\mathfrak{g}$ by

$$\langle X, Y\rangle_{\mathfrak{g}} = \mathrm{tr}\left(X^t \cdot Y\right). \tag{6.30}$$

Denote by $\mathfrak{h}^*$ the dual space of $\mathfrak{h}$ with respect to $\langle \_, \_\rangle_{\mathfrak{g}}$. There is a natural pairing $(\_, \_)$ between $\mathfrak{g}^*$ and $\mathfrak{g}$ given by

$$(\alpha, X) = \alpha(X) = \mathrm{tr}\,(\alpha \cdot X) \tag{6.31}$$

for $\alpha \in \mathfrak{g}^*$ and $X \in \mathfrak{g}$. We can associate a subspace of $\mathfrak{g}$ to each element $\alpha \in \mathfrak{h}^*$ by

$$\mathfrak{g}_\alpha := \{X \in \mathfrak{g} : [\mathbf{h}, X] = \mathrm{ad}_{\mathbf{h}}(X) = \alpha(\mathbf{h}) \cdot X \text{ for all } \mathbf{h} \in \mathfrak{h}\}. \tag{6.32}$$

We call a non-zero $\alpha \in \mathfrak{h}^*$ a **root** if $\mathfrak{g}_\alpha$ is non-zero. The space $\mathfrak{g}_\alpha$ is then called **root space**. One finds that all root spaces $\mathfrak{g}_\alpha$ are one-dimensional. Furthermore, one can show that the space $\mathfrak{g}_0$, which is not a root space, is equal to the Cartan algebra $\mathfrak{h}$. We denote the set of roots by $\Phi = \{\alpha_1, \ldots, \alpha_l\}$. The set Φ is called **root system** for $\mathfrak{g}$. Elements in root spaces are called **root vectors**. If $\mathfrak{h}^*$ is the dual Lie algebra of $\mathfrak{h}$ with respect to the pairing we introduced above, we fix a basis $\{\varepsilon_i\}$ of $\mathfrak{h}^*$ as described in [GW09; p. 91f.]. For the detailed description of the choice in case of classical (simple) Lie algebras we refer to that reference. For our purposes we only need the result for symplectic Lie algebras. For $\mathfrak{g} = \mathfrak{sp}_g$ we can choose the basis as follows: The element ε_i is determined by

$$\langle \varepsilon_i, A\rangle = a_i \quad \text{for } A = \mathrm{diag}\left(a_1, \ldots, a_g, -a_g, \ldots, -a_1\right) \in \mathfrak{h}^*.$$

The order of the a_i on the diagonal depends on the choice of the underlying symplectic form (comp. (6.25)). The shown form is that for J_g.

[8]A connected Lie group is semi-simple if and only if its Lie algebra is semi-simple.

A semi-simple Lie algebra $\mathfrak{g}$ is uniquely determined by its (irreducible) root system. There is a complete classification of irreducible root systems of simple Lie algebras. Over a field of characteristic zero there are four infinite families A_d, B_d, C_d and D_d and five exceptional cases of irreducible root systems, namely E_6, E_7, E_8, F_4, and G_2, which are called **Cartan types** of root systems. We are primarily interested in the Lie algebra of type C_2, because the Lie algebras $\mathfrak{sp}_g$ have Cartan type C_g. In all cases the index refers to the **rank** of the root system, i.e. to the dimension of the Cartan subalgebra.

A subset $\Delta = \{\alpha_1, \dots, \alpha_k\} \subset \Phi$ of roots is called a set of **simple roots** if every root can be uniquely written as integral linear combination of the α_i where all (integral) coefficients have the same sign. One can show that Δ generates $\mathfrak{h}^*$. Further, we define the set of **positive roots** $\Phi^+ = \{\alpha_1, \dots, \alpha_s\} \subset \Phi$ to be the subset of all non-negative linear combinations of elements in Δ. This induces a decomposition

$$\Phi = \Phi^+ \cup (-\Phi^+). \tag{6.33}$$

For the symplectic Lie algebra $\mathfrak{sp}_g$ of rank g, which is a Lie algebra of Cartan type C_g, we get an explicit description of its root system with the notions from above (comp. [GW09; 2.4.1, 2.4.3]). We take the set $\mathfrak{B} = \{\mathbf{e}_1, \dots, \mathbf{e}_g, \mathbf{f}_g, \dots, \mathbf{f}_1\}$ defined in Section 6.3.1.3 as basis. Let $\mathbf{E}_{i,j} \in \mathrm{Mat}_{2g}(\mathbb{Z})$ be the matrix with everywhere zeroes except a single entry in the j-th column and the i-th row, which is equal to 1. The $\mathbf{E}_{i,j}$ for $1 \le i,j \le 2g$ form a basis of $\mathrm{Mat}_{2g}(\mathbb{Z})$. Set for $1 \le i < j \le g$

$$\begin{aligned} \mathbf{x}_{\varepsilon_i - \varepsilon_j} &:= \mathbf{E}_{i,j} - \mathbf{E}_{2g+1-i,2g+1-j} \\ \mathbf{x}_{-(\varepsilon_i - \varepsilon_j)} &:= \mathbf{E}_{j,i} - \mathbf{E}_{2g+1-j,2g+1-i}, \end{aligned} \tag{6.34}$$

where $\{\varepsilon_i\}$ is the previously chosen basis of $\mathfrak{h}^*$. One easily checks that

$$[\mathbf{h}, \mathbf{x}_{\varepsilon_i - \varepsilon_j}] = \langle \varepsilon_i - \varepsilon_j, \mathbf{h} \rangle \cdot \mathbf{x}_{\varepsilon_i - \varepsilon_j} \tag{6.35}$$

for $\mathbf{h} \in \mathfrak{h}$. Thus, $\pm(\varepsilon_i - \varepsilon_j)$ for $1 \le i < j \le g$ is a root and its root space is

$$\mathfrak{g}_{\pm(\varepsilon_i - \varepsilon_j)} = \mathbf{x}_{\pm(\varepsilon_i - \varepsilon_j)} \mathbb{C}. \tag{6.36}$$

We set $\mathbf{y}_{\varepsilon_i - \varepsilon_j} = \mathbf{x}_{-(\varepsilon_i - \varepsilon_j)}$. Analogously, one shows that for $1 \le i \le j \le g$ the elements $\pm(\varepsilon_i + \varepsilon_j)$ are also roots with root vectors

$$\begin{aligned} \mathbf{x}_{\varepsilon_i + \varepsilon_j} &:= \mathbf{E}_{i,2g+1-j} + \mathbf{E}_{j,2g+1-i} \\ \mathbf{x}_{-(\varepsilon_i + \varepsilon_j)} &:= \mathbf{E}_{2g+1-j,i} + \mathbf{E}_{2g+1-i,j} =: \mathbf{y}_{\varepsilon_i + \varepsilon_j}. \end{aligned} \tag{6.37}$$

Their associated root spaces are $\mathfrak{g}_{\varepsilon_i + \varepsilon_j} = \mathbf{x}_{\varepsilon_i + \varepsilon_j} \mathbb{C}$ and $\mathfrak{g}_{-\varepsilon_i - \varepsilon_j} = \mathbf{y}_{\varepsilon_i + \varepsilon_j} \mathbb{C}$. We get

$$\Phi = \bigcup_{1 \le i < j \le g} \{\pm(\varepsilon_i - \varepsilon_j), \pm(\varepsilon_i + \varepsilon_j)\} \cup \bigcup_{1 \le k \le g} \{\pm 2\varepsilon_k\}. \tag{6.38}$$

It is not difficult to check with the definitions above that

$$\mathbf{y}_\alpha^t = \mathbf{x}_\alpha. \tag{6.39}$$

Let $\alpha_i = \varepsilon_i - \varepsilon_{i+1}$ for $1 \leq i < g$ and $\alpha_g = 2\varepsilon_g$. We can take as set of simple roots

$$\Delta = \left\{\alpha_1, \dots, \alpha_g\right\}. \tag{6.40}$$

The associated set of positive roots for $\mathfrak{sp}_g$ is then given by

$$\Phi^+ = \bigcup_{1\leq i<j\leq g} \{\varepsilon_i - \varepsilon_j, \varepsilon_i + \varepsilon_j\} \cup \bigcup_{1\leq k\leq g} \{2\varepsilon_k\}. \tag{6.41}$$

For the other Cartan types of root systems of simple Lie algebras as well as for the described case the construction of roots and associated root vectors can be found in [GW09; 2.4.1].

From now on we assume that we fixed a system of positive roots $\Phi^+ = \{\alpha_1, \dots, \alpha_s\}$. For each root $\alpha \in \Phi$ we define an associated coroot $\mathbf{h}_\alpha \in [\mathfrak{g}_\alpha, \mathfrak{g}_{-\alpha}]$ by the normalization

$$(\alpha, \mathbf{h}_\alpha) = 2. \tag{6.42}$$

We can define a lattice $\Lambda \subset \mathfrak{h}^*$ by

$$\Lambda = \left\{\mu \in \mathfrak{h}^* : (\mu, \mathbf{h}_\alpha) = \mu(\mathbf{h}_\alpha) \in \mathbb{Z}\ \forall \alpha \in \Phi\right\}. \tag{6.43}$$

This lattice is called **weight lattice** of $\mathfrak{g}$. An element in Λ is called an **integral weight**. With the choices for the ε_i from above Λ is generated by the ε_i, i.e. $\Lambda = \bigoplus_i \mathbb{Z} \cdot \varepsilon_i$. There is a subset

$$\Lambda^+ = \left\{\mu \in \mathfrak{h}^* : (\mu, \mathbf{h}_\alpha) = \mu(\mathbf{h}_\alpha) \in \mathbb{N}_0\ \forall \alpha \in \Phi\right\} \subset \Lambda. \tag{6.44}$$

The elements in Λ^+ are called **dominant weights**. The dominant weights are integral weights and generate the lattice Λ of integral weights. Note that they do not form a sublattice in Λ. The **fundamental weights** (with respect to some choice of simple roots Δ) are the set $\{\omega_1, \dots, \omega_g\}$ where $(\omega_i, \mathbf{h}_{\alpha_j}) = \delta_{i,j}$. We have $\Lambda^+ = \bigoplus_i \mathbb{N}_0 \cdot \omega_i$. For the weight modules we are interested in dominant weights are of particular interest (comp. Section 6.4.4). Hence, we usually write each (integral) weight $\lambda \in \Lambda$ as $\lambda = \sum_i \lambda_i \cdot \omega_i$ for some $\lambda_i \in \mathbb{Z}$ and identify λ with the collection (λ_i) of coefficients in this basis. If λ is dominant all λ_i are non-negative.

For the symplectic Lie algebra $\mathfrak{sp}_g$ we have $\omega_i = \sum_{k=1}^i \varepsilon_k$ for $1 \leq i \leq g$. In this case we can easily express a dominant weight $\lambda = \sum_{i=1}^g \lambda_i \cdot \omega_i$ in terms of the ε_i via

$$\lambda = \sum_{i=1}^g \lambda_i \cdot \omega_i = \sum_{i=1}^g \lambda_i \cdot \left(\sum_{j\leq i} \varepsilon_j\right) = \sum_{i=1}^g \underbrace{\left(\sum_{j=i}^g \lambda_j\right)}_{=:\widetilde{\lambda}_i} \cdot \varepsilon_i. \tag{6.45}$$

Since $\lambda_i \geq 0$ the coefficients $\widetilde{\lambda}_i$ are also non-negative integers. An easy computation shows that if $\lambda = \sum_{i=1}^g \widetilde{\lambda}_i \cdot \varepsilon_i \in \Lambda^+$, we have

$$\lambda = \sum_{i=1}^g \widetilde{\lambda}_i \cdot \varepsilon_i = \sum_{i=1}^g \underbrace{\left(\widetilde{\lambda}_i - \widetilde{\lambda}_{i+1}\right)}_{\geq 0} \cdot \omega_i, \tag{6.46}$$

where we set $\widetilde{\lambda}_{g+1} := 0$. Therefore, $\lambda = \sum_{i=1}^{g} \widetilde{\lambda}_i \cdot \varepsilon_i \in \Lambda^+$ if $\widetilde{\lambda}_1 \geq \widetilde{\lambda}_2 \geq \ldots \geq \widetilde{\lambda}_g \geq 0$. A weight is called **regular** if all $\geq$ are $>$.

All $\lambda = (\lambda_i) \in \Lambda$ define a rational character[9] of the maximal torus $\mathbf{H}$ via the map[10] (see e.g. [GW09; 2.4.1])

$$h = \operatorname{diag}(h_i) \mapsto \prod_i h_i^{\lambda_i} =: h^{\lambda}. \tag{6.47}$$

Since $\mathfrak{h}$ is abelian we know by a standard result in linear algebra that the family of endomorphisms given by $\operatorname{ad}_{\mathfrak{g}}(\mathfrak{h})$ are simultaneously diagonalizable (see [Hum78; 8.1])). In other words, $\mathfrak{g}$ decomposes into a direct sum of eigenspaces $\mathfrak{g}_\alpha$ under the action of $\mathfrak{h}$. Therefore, we get

$$\begin{aligned}\mathfrak{g} = \underbrace{\mathfrak{g}_0}_{=\mathfrak{h}} \oplus \bigoplus_{\alpha\in\Phi} \mathfrak{g}_\alpha &= \underbrace{\bigoplus_{\alpha\in\Phi^+} \mathfrak{g}_{-\alpha}}_{=\mathfrak{n}^-} \oplus \mathfrak{h} \oplus \underbrace{\bigoplus_{\alpha\in\Phi^+} \mathfrak{g}_\alpha}_{=\mathfrak{n}^+} \\ &= \mathfrak{n}^- \oplus \mathfrak{h} \oplus \mathfrak{n}^+.\end{aligned} \tag{6.48}$$

This decomposition is called **root space decomposition** (comp. [GW09; p. 93]). We will equip the Lie algebra $\mathfrak{g}$ with a standard basis whose structure constants are integral (comp. [Hum78; VII.25]). Let d be the rank of Φ, i.e. $d = \dim \mathfrak{h}$. We choose root vectors $\mathbf{x}_\alpha \in \mathfrak{g}_\alpha$ and $\mathbf{y}_\alpha \in \mathfrak{g}_{-\alpha}$ such that they generate the corresponding root spaces, i.e. they are unique up to a sign. Further, we set $\mathbf{h}_\alpha = [\mathbf{x}_\alpha, \mathbf{y}_\alpha]$. One easily finds that $\mathbf{h}_\alpha \in \mathfrak{h}$. For simplicity we set $\mathbf{y}_i := \mathbf{y}_{\alpha_i}$, $\mathbf{x}_i := \mathbf{x}_{\alpha_i}$ and $\mathbf{h}_i := \mathbf{h}_{\alpha_i}$. We normalize[11] the elements $\mathbf{y}_i$ and $\mathbf{x}_i$ by the following conditions:

$$\begin{aligned} [\mathbf{h}_i, \mathbf{h}_j] &= 0 & [\mathbf{x}_i, \mathbf{y}_j] &= \delta_{i,j} \cdot \mathbf{h}_i \\ [\mathbf{h}_j, \mathbf{x}_i] &= c_{i,j} \cdot \mathbf{x}_i & [\mathbf{h}_j, \mathbf{y}_i] &= -c_{i,j} \cdot \mathbf{y}_i \end{aligned} \tag{6.49}$$

where $c_{i,j} = (\alpha_i, \mathbf{h}_j) = \alpha_i(\mathbf{h}_j)$ for $\alpha_i, \alpha_j \in \Phi$. We have defined $3s$ elements in $\{\mathbf{y}_i, \mathbf{h}_j, \mathbf{x}_k\}$, but the dimension of $\mathfrak{g}$ is only $2s + d$ by the decomposition of (6.48). Therefore, the elements cannot be linearly independent. Furthermore, by a theorem of Claude Chevalley we find, that the $\mathbf{h}_i$ for $d < i \leq s$ can be written even as integral linear combinations of $\mathbf{h}_1, \ldots, \mathbf{h}_d$ and with the above made normalizations the set of $\{\mathbf{y}_i, \mathbf{h}_j, \mathbf{x}_k\}$ only depends on the choice of the root system together with its fixed system of simple roots (see e.g. [Hum78; VII.25]). In other words, the definition of the elements is in some way canonical. This leads to the following definition:

Definition 6.4.1 (Chevalley basis)**:** *The set $\{\mathbf{y}_1, \ldots, \mathbf{y}_s, \mathbf{h}_1, \ldots, \mathbf{h}_d, \mathbf{x}_1, \ldots, \mathbf{x}_s\}$, where the $\mathbf{y}_i, \mathbf{h}_i, \mathbf{x}_i$ are defined as above and fulfil the conditions from* (6.49)*, is called the* **Chevalley basis** *of $\mathfrak{g}$ (with respect to Δ).*

For the symplectic Lie algebra the elements in the Chevalley basis are exactly those $\mathbf{y}_\alpha$ and $\mathbf{x}_\alpha$ defined earlier in (6.34) and (6.37).

[9]The choice of a different basis of Λ does not affect the existence of such a character. It changes only its representation.

[10]If in the chosen basis of $\mathbf{H}$ the element $h \in \mathbf{H}$ does not have diagonal shape, one has to replace the h_i by the images of the i-th coordinate function (see e.g. proof of [GW09; Theo. 2.1.5]).

[11]The choice of a generator of a one-dimensional space involves a choice of a sign.

6.4.3. Kostant's $\mathbb{Z}$-Form and the Poincaré-Birkhoff-Witt Theorem

The construction we present in this section can be found in the original papers of Henri Poincaré [Poi00], George Birkhoff [Bir37], Ernst Witt [Wit37], and Bertram Kostant [Kos66] or little bit more convenient to the reader in [Hum78; VII.26].

Let $\mathfrak{g}$ be a semi-simple Lie algebra over a field of characteristic zero (this is not necessary for all computations but for some) and let Φ be its root system of $\mathfrak{g}$. Let, as before,

$$\{\mathbf{y}_1, \ldots, \mathbf{y}_s, \mathbf{x}_1, \ldots, \mathbf{x}_s, \mathbf{h}_1, \ldots, \mathbf{h}_d\}$$

be a fixed Chevalley basis[12] of $\mathfrak{g}$ with respect to the chosen Cartan subalgebra $\mathfrak{h}$ (see Definition 6.4.1). Remember that we have the root space decomposition (see equation (6.48))

$$\mathfrak{g} = \mathfrak{n}^- \oplus \mathfrak{h} \oplus \mathfrak{n}^+$$

where the $\mathfrak{g}_\alpha$ are one-dimensional and generated by a root vector $\mathbf{x}_i$ or $\mathbf{y}_i$, respectively. To shorten the notation we introduce for $i = 1, \ldots, s$ and $n, k \in \mathbb{N}$

$$\mathbf{x}_i^{(n)} = \frac{\mathbf{x}_i^n}{n!} \qquad \mathbf{y}_i^{(n)} = \frac{\mathbf{y}_i^n}{n!} \qquad \binom{\mathbf{h}_i}{k} = \prod_{j=1}^{k} \frac{(\mathbf{h}_i - j + 1)}{j}. \tag{6.50}$$

Further, we set for multiindices $N = (n_1, \ldots, n_s)$, $M = (m_1, \ldots, m_s)$, and $K = (k_1, \ldots, k_l)$ with n_i, m_i, $k_i \geq 0$

$$\mathbf{x}^N = \mathbf{x}_{\alpha_1}^{(n_1)} \cdots \mathbf{x}_{\alpha_s}^{(n_s)} \qquad \mathbf{y}^M = \mathbf{y}_{\alpha_1}^{(m_1)} \cdots \mathbf{y}_{\alpha_s}^{(m_s)} \qquad \mathbf{h}^K = \binom{\mathbf{h}_1}{k_1} \cdots \binom{\mathbf{h}_l}{k_l}. \tag{6.51}$$

This leads to the well known theorem which is named after Poincaré, Birkhoff and Witt.

Theorem 6.4.2 (Poincaré 1900, Birkhoff 1937, Witt 1937)**:** *Let $\mathfrak{g}$ be a semi-simple Lie algebra over a field of characteristic zero and the other notions defined as above, then the universal enveloping algebra $\mathfrak{U}(\mathfrak{g})$ is generated as free algebra over $\mathbb{C}$ by the monomials of the form*

$$\mathbf{y}^M \cdot \mathbf{h}^K \cdot \mathbf{x}^N$$

for multiindices M, K, N as above.

In the papers of Poincaré, Birkhoff and Witt they do not use the normalization introduced before, because they did not look at the underlying integral structure. But Bertram Kostant, who presented the following theorem in 1965 at a Symposium held at Boulder (Colorado), did exactly this and therefore we decided to restate the original Poincaré-Birkhoff-Witt Theorem in a modified version which is more related to Kostant's work and is in some sense stronger.

Theorem 6.4.3 (Kostant, 1966)**:** *Let $\mathfrak{g}$ be a semi-simple Lie algebra over a field of characteristic zero and the other notions defined as above, then the universal enveloping algebra $\mathfrak{U}(\mathfrak{g})$ is generated as a free $\mathbb{Z}$-module by the monomials of the form*

$$\mathbf{y}^M \cdot \mathbf{h}^K \cdot \mathbf{x}^N$$

for multiindices M, K, N as above. We call the lattice generated by these monomial an admissible lattice for $\mathfrak{U}(\mathfrak{g})$.

[12]Note: In GAP the order is exactly as indicated here, i.e. first the negative root vectors, then the positive root vectors, then the Cartan elements.

It is interesting to note that neither Birkhoff nor Witt mention the work of Poincaré in their publication. It is not known whether one of them knew of the preliminary version of their theorems stated by Poincaré 37 years earlier or not.

Beside the fact that the monomials $\mathbf{y}^M \cdot \mathbf{h}^K \cdot \mathbf{x}^N$ generate $\mathfrak{U}(\mathfrak{g})$ as free $\mathbb{Z}$-module we have some relations between the generators given by the so called **Serre relations**, which can be written down in terms of Lie brackets and elements in the Cartan matrix. These relations are used for example by GAP to simplify the expressions. Beside the relations stated in (6.49) during the definition of a Chevalley basis there are two additional relations, which involve the adjoint representation. Since we do not need these in the following, we do not state them explicitly.

6.4.4. Highest Weight Modules and Representations

Highest weight modules or highest weight representations are the building blocks to describe representations of Lie groups or Lie algebras and play an important role in this thesis as coefficient modules of the cohomology groups. Here we summarize here the essential definitions and facts on representations and their relation to highest weight modules.

Definition 6.4.4 ($\mathfrak{g}$-module)**:** *Let $\mathfrak{g}$ be a Lie algebra, $\mathbf{R}$ a ring extension of $\mathbb{Z}$ and V be a free $\mathbf{R}$-module with an action $\cdot : \mathfrak{g} \times V \to V$ of the Lie algebra. We call V a* **$\mathfrak{g}$-$\mathbf{R}$-module** *if the following conditions are fulfilled*

M1 $(aX + bY) \cdot v = a(X \cdot v) + b(Y \cdot v)$ *for $a, b \in \mathbf{R}$, $v \in V$ and $X, Y \in \mathfrak{g}$.*

M2 $X \cdot (av + bw) = a(X \cdot v) + b(X \cdot w)$ *for $a, b \in \mathbf{R}$, $v, w \in V$ and $X \in \mathfrak{g}$.*

M3 $[X, Y] \cdot v = X \cdot (Y \cdot v) - Y \cdot (X \cdot v)$ *for $v \in V$ and $X, Y \in \mathfrak{g}$.*

For $\mathbf{R} = \mathbb{Z}$ we call a $\mathfrak{g}$-$\mathbb{Z}$-module simply a **$\mathfrak{g}$-module**.

Note: By Theorem 6.4.3 we know that the universal enveloping algebra is given by its underlying integral structure. Therefore, we replaced – compared to some definitions in the literature – the field of real or complex numbers number by a ring $\mathbf{R}/\mathbb{Z}$ and vector spaces over these fields by free $\mathbf{R}$-modules (comp. the analogous remarks on group representations in Section 6.1.2).

Any Lie algebra representation $\phi : \mathfrak{g} \to \mathfrak{gl}(V)$ may be viewed as a $\mathfrak{g}$-module via the action

$$X \cdot v = \phi(X) \cdot v. \tag{6.52}$$

The other way around every $\mathfrak{g}$-module is also a Lie algebra representation of $\mathfrak{g}$. Thus, there is a one-to-one correspondence between representation modules for a Lie algebra representation of $\mathfrak{g}$ and $\mathfrak{g}$-modules.

A $\mathfrak{g}$-module is called **irreducible** if there are no non-trivial $\mathfrak{g}$-submodules. The construction is completely analogous to the construction of Γ-modules and representations of Lie groups (see Section 6.1.2). If V is an irreducible $\mathfrak{g}$-module of finite rank, we know that $\mathfrak{h}$ has to act diagonally on V. Therefore, we have a decomposition of V into eigenspaces under this action. Thus, we have

$$V = \bigoplus_{\lambda \in \mathfrak{h}^*} V(\lambda) \tag{6.53}$$

where $V(\lambda) = \{v \in V : h \cdot v = \lambda(h)v\}$. If $V(\lambda) \neq 0$ we call the $V(\lambda)$ a **weight space** or **weight module**.

Let $\alpha \in \Phi$, $X \in \mathfrak{g}_\alpha$, $h \in \mathfrak{h}$ and $v \in V(\lambda)$, then we have

$$h \cdot X \cdot v = (X \cdot h + [h, X]) \cdot v = (\lambda(h) + \alpha(h))\, X \cdot v. \tag{6.54}$$

Thus, elements in $\mathfrak{g}_\alpha$ map $V(\lambda)$ into $V(\lambda + \alpha)$. This observation leads to the definition of highest weight module.

Definition 6.4.5 (Highest Weight Module)**:** *A $\mathfrak{g}$-module $\mathbb{M}$ is a* **highest weight module** *or a* **highest weight representation**[13] *(with respect to a fixed set of positive roots Φ^+) if there is a weight $\lambda \in \mathfrak{h}^*$ and a non-zero vector $\mathbf{v}_0 \in \mathbb{M}$ such that*

a) $\mathfrak{n}^+ \cdot \mathbf{v}_0 = 0$

b) $h \cdot \mathbf{v}_0 = \lambda(h) \cdot \mathbf{v}_0$ *for* $h \in \mathfrak{h}$

c) $\mathbb{M} = \mathfrak{U}(\mathfrak{g}) \cdot \mathbf{v}_0$.

The vector $\mathbf{v}_0$ is called a **maximal vector** *of $\mathbb{M}$ and λ is called* **highest weight** *of $\mathbb{M}$.*

Some authors consider a highest weight module as $\mathfrak{U}(\mathfrak{g})$-module, but the objects are in both cases the same, because we can write all elements in $\mathfrak{U}(\mathfrak{g})$ by Theorem 6.4.3 in terms of a product of root vectors (applied to a highest weight vector). We know that for each $\lambda \in \mathfrak{h}^*$ there is irreducible highest weight representation for that weight. Furthermore, for a highest weight module the related highest weight is uniquely determined [GW09; 3.2.1]. We denote the highest weight module of weight λ by $\mathbb{M}_\lambda$. We need some properties of highest weight modules and collect them in the following theorem [Hum78; Theo. VI.20.2].

Theorem 6.4.6: *Let $\mathbb{M}$ be the highest weigh module of weight $\lambda \in \mathfrak{h}^*$ generated by a maximal vector $\mathbf{v}_0$. Let $(\alpha_1, \dots, \alpha_s) = \Phi^+$ be the chosen fixed system of positive roots. Then:*

a) $\mathbb{M}$ is spanned by the vectors $\mathbf{y}_1^{i_1} \cdots \mathbf{y}_s^{i_s} \cdot \mathbf{v}_0$ with $i_j \in \mathbb{N}_0$. These vectors have weights $\lambda - \sum_j i_j \cdot \alpha_j$. Thus, $\mathbb{M}$ is a semi-simple $\mathfrak{h}$-module. The set of generating vectors is called the Poincaré-Birkhoff-Witt basis (PBW-basis) of $\mathbb{M}$.

b) All weights μ of $\mathbb{M}$ satisfy $\mu \leq \lambda$, i.e.

$$\mu = \lambda - \sum_{\alpha \in \Phi^+} m_\alpha \cdot \alpha$$

with $m_\alpha \in \mathbb{N}_0$.

c) For all weights μ of $\mathbb{M}$ we have $\operatorname{rank} \mathbb{M}(\mu) < \infty$ for the weight space $\mathbb{M}(\mu)$, while $\operatorname{rank} \mathbb{M}(\lambda) = 1$.

[13]One identifies sometimes the representation with the module implicit together with the action of the group via the representation.

d) *Each submodule of $\mathbb{M}$ is a weight module. A submodule generated by a maximal vector of weight $\mu < \lambda$ is proper.*

Furthermore, we get the following decomposition from (6.53) and the previous theorem [GW09; Lemma 3.2.2]

$$\mathbb{M}_\lambda = \mathbf{v}_0 \cdot \mathbb{Z} \oplus \bigoplus_{\mu<\lambda} \mathbb{M}_\lambda(\mu) \tag{6.55}$$

where all submodules are of finite rank. One can use point a) of the theorem above and the previous equation to compute a basis of a highest weight module. The following lemma shows why the consideration of highest weight representations is of particular interest (see [GW09; Cor. 323]).

Lemma 6.4.7: *Let $\pi : \mathfrak{g} \to \mathfrak{gl}(V)$ be a non-zero irreducible Lie-algebra representation. Then there exists a unique dominant integral weight $\lambda \in \mathfrak{h}^*$ such that $\operatorname{rank} V(\lambda) = 1$, and for all other weights μ of vectors in V we have $\mu < \lambda$.*

From this we immediately get:

Corollary 6.4.8: *Every irreducible Lie algebra representation of a Lie algebra $\mathfrak{g}$ is contained in a uniquely determined highest weight module $\mathbb{M}_\lambda$ for a dominant integral weight $\lambda \in \mathfrak{h}^*$.*

We should emphasise that a finite rank highest weight representation for a highest weight λ need not be irreducible. We know that an irreducible highest weight module $\mathbb{M}_\lambda$ is of finite rank if and only if λ is integral and dominant (see [Hum78; Cor. 21.2]).

The next question we might ask is if a highest weight module can be written in a basis in the sense of point a) of the Theorem 6.4.6, how does the action of the Lie algebra $\mathfrak{g}$ in this basis then look like? This is essentially basic linear algebra, because if we have a basis element we can multiply the monomial by the Lie algebra element written, in terms of the Chevalley basis, as linear combination of the root vectors $\mathbf{x}_\alpha$ and $\mathbf{y}_\alpha$ with $\alpha \in \Phi^+$. By Definition 6.4.5 c) the result again has to be a vector in that highest weight module. Therefore, it again has to be written in the original basis of the highest weight module. This last part is in praxi the main difficulty, because we need to apply the relations to simplify the expressions and write them again in terms of only elements coming from negative roots, i.e. the $\mathbf{y}_\alpha$. For details how to do this in praxi we refer to Section 11.2.3.

The last important question we handle in this section is which rank/dimension a highest weight module for a given (integral) weight has. There are several ways to compute the dimension of a highest weight module. We use **Weyl's dimension formula** which is a consequence of Weyl's character formula [GW09; 7.1]

$$\operatorname{rank}_{\mathbb{Z}}(\mathbb{M}_\lambda) = \prod_{\alpha\in\Phi^+} \frac{(\lambda+\alpha,\alpha)}{(\rho,\alpha)}, \tag{6.56}$$

where

$$\rho = \tfrac{1}{2} \sum_{\alpha\in\Phi^+} \alpha. \tag{6.57}$$

All other formulas derive from this in some way.

Example 6.4.9. Let $\mathbb{M}_\lambda$ be the highest weight module for the Lie algebra $\mathfrak{sp}_g$ of integral weight $\lambda = \sum_i \lambda_i \varepsilon_i \in \mathfrak{h}^*$, i.e. $\lambda_i \in \mathbb{Z}$. Then we have

$$\operatorname{rank}_{\mathbb{Z}} \mathbb{M}_\lambda = \prod_{1 \le i < j \le g} \frac{(\lambda_i - g + i - 1)^2 - (\lambda_j - g + j - 1)^2}{(g - i + 1)^2 - (g - j + 1)^2} \prod_{1 \le i \le g} \frac{\lambda_i + g - i + 1}{g - i + 1} \tag{6.58}$$

6.4.5. The Standard Representation

Finally, we introduce the so called standard representation as highest weight representation (see [FH91] for further details). The **standard representation** $(\rho_{\mathfrak{g},0}, \mathbb{V})$ of $\mathfrak{g}$ is given by the natural inclusion map from $\mathfrak{g}$ into $\mathfrak{gl}(\mathbb{V})$. The standard representation is given by the representation of its root vectors as matrices, which we presented in (6.34) and (6.37) for the symplectic Lie algebra, and which can be obtained in the general case in a completely analogous way. The matrices in (6.34) and (6.37) are the images $\rho_{\mathfrak{g},0}(\mathbf{x}_i)$ and $\rho_{\mathfrak{g},0}(\mathbf{y}_i)$, where the $\mathbf{x}_i$ and $\mathbf{y}_i$ are elements of the Chevalley basis $\{\mathbf{y}_i, \mathbf{h}_j, \mathbf{x}_k\}$ of $\mathfrak{g}$ (comp. Definition 6.4.1). The standard representation is the representation of smallest rank which is faithful, i.e. the homomorphism $\rho_{\mathfrak{g},0} =: \rho_0$ is injective. Furthermore, this representation is given as a highest weight representation for the weight $\lambda_0 = \omega_1 = \varepsilon_1$, i.e. $\mathbb{V} := \mathbb{M}_{1,0,\dots,0}$. To shorten the notation we use $\mathbb{V}_{\mathbb{R}} := \mathbb{V} \otimes_{\mathbb{Z}} \mathbb{R}$.

Later in Chapter 11 we use the standard representation $\mathbb{V}$ as an essential building block for Γ-modules and thus for our coefficients of the cohomology. Therefore, we have a closer look how the matrix representation on the Chevalley basis of $\mathfrak{g}$ looks like.

Because we can view $\mathbb{V}$ as a highest weight module of weight λ_0, we have an additional way to write the standard representation, namely the PBW-basis of $\mathbb{V} = \mathbb{M}_{1,0,\dots,0}$ introduced in Theorems 6.4.3 and 6.4.2. According to Theorem 6.4.6 a) any basis element of $\mathbb{V}$ can be given – with the notation from (6.51) – as $\mathbf{y}^{\mu'} \cdot \mathbf{v}_0$, where $\mathbf{v}_0$ is a fixed highest weight vector and $\mu' = (\mu'_{\alpha_1}, \dots, \mu'_{\alpha_s})$ is a multiindex of length $s = |\Phi^+|$. The weight vectors $\mathbf{y}^{\mu'} \cdot \mathbf{v}_0$ have weight $\mu = (\mu_1, \dots, \mu_s) \le \lambda_0 = (1, 0, \dots, 0)$, i.e. $\mu = \lambda_0 - \mu' = \lambda_0 - \sum_{\alpha_i \in \Phi^+} \mu'_i \cdot \alpha$ for some $\mu'_i := \mu'_{\alpha_i} \in \mathbb{N}_0$.

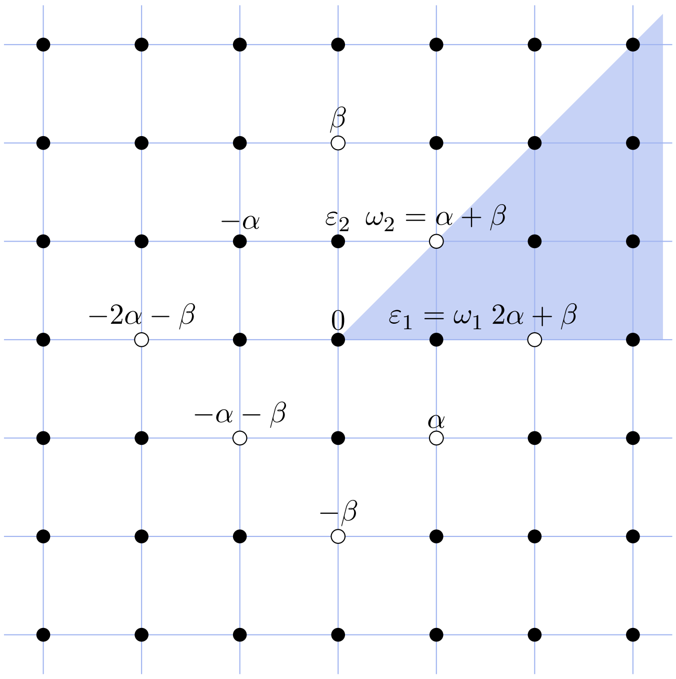
For simplicity we restrict outselves to $\mathfrak{g} = \mathfrak{sp}_2$, which is the case we need later on. In this case we have the following positive roots:

$$\begin{aligned} \alpha &= 2\omega_1 - \omega_2 = \varepsilon_1 - \varepsilon_2 \\ \beta &= -2\omega_1 + 2\omega_2 = 2\varepsilon_2 \\ \alpha + \beta &= \omega_2 = \varepsilon_1 + \varepsilon_2 \\ 2\alpha + \beta &= 2\omega_1 = 2\varepsilon_1. \end{aligned} \tag{6.59}$$

The simple roots are α and β. Let $t = \operatorname{diag}\left(t_1, t_2, t_2^{-1}, t_1\right)$, then the corresponding torus characters are given by

$$\begin{aligned} \alpha(t) &= t_1 \cdot t_2^{-1} \\ \beta(t) &= t_2^2 \\ (\alpha + \beta)(t) &= t_1 \cdot t_2 \\ (2\alpha + \beta)(t) &= t_1^2. \end{aligned} \tag{6.60}$$

Figure 6.1.: *Roots and weights for $\mathfrak{sp}_2$: the dots indicate the weight lattice Λ, where the empty dots are those weights which are roots. The weights in the shaded area are dominant weights.*



Now one has to find the weights μ of the elements of the PBW-basis, i.e. $\mathbf{y}^{\mu'} \cdot \mathbf{v}_0$, which have to fulfil

$$\mu = \lambda_0 - \mu' = \lambda_0 - \left(\mu'_\alpha \alpha + \mu'_\beta \beta + \mu'_{\alpha+\beta}(\alpha+\beta)\right) \tag{6.61}$$

with $\mu'_i \geq 0$. One can easily check that if we have chosen a highest weight vector $\mathbf{v}_0$, the root vector $\mathbf{y}_\beta$ annihilates $\mathbf{v}_0$, while the other products of negative root vectors with $\mathbf{v}_0$ are non-zero (see Table 6.4). Therefore, $\mathbf{y}_\beta \cdot \mathbf{v}_0$ cannot be a basis element and the PBW-basis of $\mathbb{V}$ is given by $\{\mathbf{v}_0, \mathbf{y}_\alpha \mathbf{v}_0, \mathbf{y}_{\alpha+\beta}\mathbf{v}_0, \mathbf{y}_{2\alpha+\beta}\mathbf{v}_0\}$. We stated the action of the roots on this basis together with the resulting weights μ in Table 6.4. From this one can easily read the matrix $\rho_0(X)$ for an element X of the Chevalley basis of $\mathfrak{g}/\mathfrak{h}$, which also gives the values on the Cartan elements by application of the commutator relation.

The resulting matrices written in the PBW-basis are exactly the same as those defined earlier in this section (see (6.34) and (6.37)).

6.5. Highest Weight Modules for $\mathrm{Sp}_2(\mathbb{Z})$

There is another (compared with that one presented in Section 6.4) way to describe highest weight modules, which is also used in context of cohomology of arithmetic groups, since it has the advantage that one does not use the theory of Lie algebras. In other words, in this section we consider highest weight modules as modules for the Lie group. For simplicity we restrict to the case of $\mathbf{Sp}_2/\mathbb{Q}$. The representations of Lie groups and Lie algebras can be identified using the exponential map whereas the underlying modules are the same. In

Table 6.4.: *Poincaré-Birkhoff-Witt basis with its weights* μ *and the action of roots on the basis.*

	$\mathbf{v}_0$	$\mathbf{y}_1\mathbf{v}_0$	$\mathbf{y}_3\mathbf{v}_0$	$\mathbf{y}_4\mathbf{v}_0$
$\mathbf{y}_\alpha \cdot \text{-}$	$\mathbf{y}_1\mathbf{v}_0$	0	$\mathbf{y}_4\mathbf{v}_0$	0
$\mathbf{y}_\beta \cdot \text{-}$	0	$-\mathbf{y}_3\mathbf{v}_0$	0	0
$\mathbf{y}_{\alpha+\beta} \cdot \text{-}$	$\mathbf{y}_3\mathbf{v}_0$	$-\mathbf{y}_4\mathbf{v}_0$	0	0
$\mathbf{y}_{2\alpha+\beta} \cdot \text{-}$	$\mathbf{y}_4\mathbf{v}_0$	0	0	0
μ	$\lambda_0 - 0$ $= \omega_1$	$\lambda_0 - \alpha$ $= -\omega_1 + \omega_2$	$\lambda_0 - (\alpha+\beta)$ $= \omega_1 + \omega_2$	$\lambda_0 - (2\alpha+\beta)$ $= -\omega_1$

Section 11.1 we have a closer look on this. There we focus on the computation of highest weight modules.

Let

$$\rho'_{1,0} : \mathbf{Sp}_2(\mathbb{Z}) =: \Gamma \to \mathbf{GL}(\mathbb{M}_{1,0}) \tag{6.62}$$

be the standard representation of $\mathbf{Sp}_2(\mathbb{Z})$ introduced above. Further, let

$$\rho'_{0,1} : \Gamma \to \mathbf{GL}(\mathbb{M}_{0,1}) \tag{6.63}$$

be the irreducible subrepresentation of the representation $\Lambda^2\mathbb{M}_{1,0}$ given by the decomposition

$$\Lambda^2\mathbb{M}_{1,0} = \mathbb{Z} \oplus \mathbb{M}_{0,1}. \tag{6.64}$$

Now we consider the module $\mathrm{Sym}^m(\mathbb{M}_{1,0}) \otimes \mathrm{Sym}^n(\mathbb{M}_{0,1})$ for a pair (m,n) of non-negative integers, which has a unique subrepresentation

$$\rho'_{m,n} : \Gamma \to \mathbf{GL}(\mathbb{M}_{m,n}) \tag{6.65}$$

which is defined by the requirement that is has the largest dominant weight among the highest weights of all subrepresentations. Then $\mathbb{M}_{m,n}$ is the unique irreducible submodule with highest weight $\lambda = m \cdot \omega_1 + n \cdot \omega_2$, i.e. the module defined in this way coincides with the highest weight module of weight λ introduced in Definition 6.4.5 (see e.g. [Har08b; p. 253ff.]).

Some authors like Gerard van der Geer and his collaborators prefer to define the representation $\rho'_{m,n}$ using the modules of the form $\mathrm{Sym}^i(\mathbb{M}_{1,0}) \otimes \mathrm{Sym}^j(\Lambda^2\mathbb{M}_{1,0})$, which leads to the very same modules which might be indexed in a different way.[14]

[14]This different ways to index correspond to different choices from bases of the roots. We use a basis containing the fundamental weights ω_i whereas others choose a basis formed by the ε_i.

Part II.

On Quadratic Forms and Cells

7. Background on Reduction

7.1. Classical Reduction Theory

In the theory of group actions it is an old problem that one is interested in finding minimal sets which consist only of elements which are not equivalent under the group action. Such a set is called fundamental domain. In general it is not so easy to give a good description of such a fundamental domain in an explicit way. One problem is that there is not a unique one, hence there are plenty of possible ways to construct such sets for one given group. To make it more complicated there are also minor differences between different definitions of such a domain (some of these domains are not even a domains since they are not open). We understand a fundamental domain as an open set such that there are no equivalent points under the group action in the set and the translates of its closure cover the whole space.

In some cases one can construct from a given set for a larger group a fundamental domain for subgroups by translating the given domain by any set of representatives of the quotient of the larger group by the smaller one. But it remains to determine at least one such set.

Now a **reduction theory** for a group is an algorithm which describes how to construct for a given group and a space on which the group acts a fundamental domain. There are a lot of things which one easily can do if one has a good, i.e. easy, description of the quotient of some set by a group acting on it.

The most classical example is the fundamental domain for the group $\mathbf{SL}_2(\mathbb{Z})$ acting on the upper half-plane, i.e. the complex numbers with positive imaginary part. Here one fundamental domain is given by

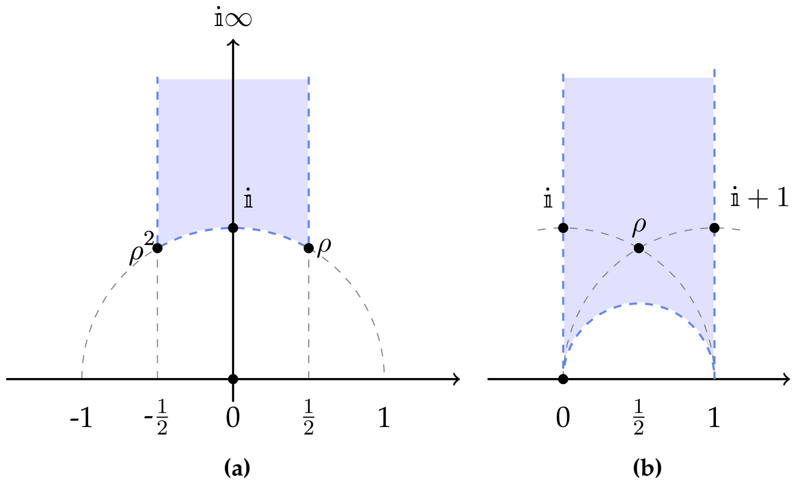
$$\{z \in \mathbb{C} : \mathrm{Im}\,(z) > 0\} \cap \{z \in \mathbb{C} : |z| > 1\} \cap \left\{z \in \mathbb{C} : |\mathrm{Re}\,(z)| < \tfrac{1}{2}\right\}. \tag{7.1}$$

This set has a nice description and can easily be plotted (see Figure 7.2a). To get the quotient one only has to identify the boundary in the right way, i.e. the two vertical lines are glued together as well as the right part of arc and the left part of the arc. In practice used there is a different construction for a fundamental domain for the group $\mathbf{SL}_2(\mathbb{Z})$, which is shown in Figure 7.2b. This is not a fundamental domain in the above mentioned way, since it is three times as large as the domain in Figure 7.2a) and thus there are point represented by more then one point in the set. But for some purposes this construction has some advantages.

As mentioned before it is not difficult to construct from this fundamental domain fundamental domains for finite index subgroups of $\mathbf{SL}_2(\mathbb{Z})$.

For the symplectic group $\mathbf{Sp}_g(\mathbb{Z})$ one can get an analogous description by a set of inequalities applying Hermann Minkowski's lattice reduction algorithm [Min05]. A priori, there are infinitely many of them, but one can show that finitely many are sufficient to describe the

Figure 7.1.: *The classical fundamental domain for the group* $\mathbf{SL}_2(\mathbb{Z})$ *(a) and an other commonly used fundamental domain (b).*



domain exactly. A special version of this reduction algorithm for the group $\mathbf{Sp}_g(\mathbb{Z})$ was given by Carl-Ludwig Siegel [Sie43]. But he was only able to simplify the system of inequalities for Minkowski's algorithm for arbitrary symplectic groups. The first list of a finite set of inequalities which describes all boundary components of the fundamental domain was given by a PhD-student of Carl-Ludwig Siegel in Göttingen named Erhard Gottschling in [Got59; Satz 1, Satz 2]. Gottschling gave a description in terms of 28 polynomial inequalities of degree 4 with 3 complex variables. The fundamental domain is then the intersection of the half-spaces defined by these inequalities. However neither Siegel nor Gottschling attempted to find a kind of cellular structure or a description of all components of the intersections. Thus, their description is not very helpful for our purpose because we cannot use their results (directly). One reason is that the structure is rather complicated and before one can work with that one has to get all components of the boundary in the Siegel upper half-space and and their behaviour under the group. This is not impossible, but also not very illuminating. In the next section we will give an other reason why for us an other variant of a reduction theory is more useful.

7.2. Topological Reduction Theory

It is the aim of a reduction theory to give a minimal description of the action of a group. In the previous section we considered fundamental domains which are minimal in a given space on which the group acts, i.e. for each orbit there is at most[1] one element. But one might ask whether this construction is also minimal in the context of topological questions like the

[1]The reason why we write 'at most' is the fact that, if the set is a domain, i.e. it is in particular open, the boundary does not belong to the domain and therefore there are some orbits – those which correspond to elements in the boundary of a fundamental domain – which are not represented. In the literature some authors call a set fundamental domain which has for each orbit exactly one representative in the set, others define a kind of closed fundamental domains which have at least one representative, i.e. they contain the whole boundary of the (open) fundamental domain and therefore some orbits have more than one representative.

computation of the cohomology of that group. Topological questions can be in some sense weaker since for example the cohomology of an object does not change if we replace the original space by a space of the same homotopy type. So we can extend the question and ask whether there is a homotopy equivalent space which carries enough information of the group action to compute the cohomology. The object we are looking for is some kind of a *minimal, good* topological model which is compatible with the group action. Minimal in this context should mean that it is of minimal dimension. Good means that it has a simple topological structure, e.g. is a locally finite CW-complex or an analogue in the context of orbifolds. It is an obvious advantage if the space of which we want to compute the cohomology has such a minimal form because the runtime of such a computation decreases if the dimension becomes smaller or the topological structure is more rigid.

Due to Theorem 12.3.4 we know that all cohomology groups of degree larger than the virtual cohomological dimension of the group vanish. Thus, there is a chance to find a homotopy equivalent space which has dimension equal to the virtual cohomological dimension of the group. In the case of the Siegel modular variety, which is of real dimension 6, the virtual cohomological dimension is only 4. The best model for our purpose should therefore have dimension 4.

In some sense this is nothing else than before. But we first replace the (locally) symmetric space by a subspace which is homotopy equivalent to the original one and has a simpler structure. To be precise, we replace the space by a suitable strong deformation retract[2] of the original space. All deformation retracts we will consider in this thesis are strong deformation retracts. Thus, we will skip this attribute in many cases. After this exchange we start with reduction theory for this deformation retract of the original space in the same fashion as described in the previous section. We will explain what this means more in detail later (see Section 9.1). Such a retract of a (locally) symmetric space is sometimes called **spine** of that space. Since in constructions of this type one uses sets of short vectors which form a system which is more round then others, one calls such a retract **well rounded**.

[2] A **(strong) deformation retract** of a topological space $\mathcal{X}$ onto a subspace $A \subset \mathcal{X}$ is a continuous family of continuous maps $f_t : \mathcal{X} \to \mathcal{X}$ with $t \in [0,1]$ such that $f_0 = \mathrm{Id}$, $f_1(\mathcal{X}) = A$, and $f_t \mid_A = \mathrm{Id}$ for all $t \in [0,1]$.

8. Quadratic Forms and Voronoï's Theory

The russian mathematician Georgy F. Voronoï (1868–1908) with ukrainian roots developed shortly before his early death in the year 1908 in a sequence of three papers [Vor08b, Vor08a, Vor09] a theory of reduction for quadratic forms or, to be more precise, he developed two independent theories. One of these is an important ingredient for the later work of Robert MacPherson together with his student Mark McConnell [McC91, MM93], which I used for my model of the Siegel modular threefold (see Chapter 9). The last part of that work was published in 1909 shortly after Voronoï's death. The method which is used in [McC91, MM93] is the method described in [Vor08b] and hence called Voronoï I. Today the method described in [Vor08a, Vor09] – called Voronoï II – is better known since it is used in algebraic geometry since years. So it was used for example in [AMRT75] for the toroidal compactification of arithmetic quotients or in the theory of lattices and packing problems (see e.g. [CS99]).

Since Voronoï I is important for my work I will sketch his main results of the reduction theory for $\mathbf{GL}_d(\mathbb{R})$. A very good reference for this area is the Habilitationsschrift of Achill Schürmann from 2007 [Sch07] which appeared also in the AMS University Lecture Series in 2009 [Sch09]. A much shorter introduction to Voronoï I can also be found either in the survey article [McC98] in the conference proceedings volume of a conference on Voronoï's impact in modern science held 1998 in Kyiv, or in the paper of Robert MacPherson and Mark McConnell [MM93].

8.1. Positive Definite Quadratic Forms

Quadratic forms are a quite old and fascinating subject in mathematics. Diophantus of Alexandria[1] studied questions related to integral solutions of equations given by quadratic forms. In modern times Pierre de Fermat (1607/8–1665) restarted this subject followed by Joseph-Louis de Lagrange (1736–1813), Adrien-Marie Legendre (1752–1833), and Carl Friedrich Gauß (1777–1855), who presented in his Disquisitiones Arithmeticae (1801) [Gau01] a description of the case of binary quadratic forms. But it was Charles Hermite (1822–1901) who first studied quadratic forms of arbitrary degree in a systematic way and not only special cases like binary, ternary, etc. forms.

A **quadratic form** is a map of the form

$$\mathbf{x} \mapsto Q[\mathbf{x}] := \mathbf{x}^t \cdot Q \cdot \mathbf{x} \tag{8.1}$$

where $Q := (q_{ij})_{i,j=1,\ldots,d} \in \mathrm{Mat}_d(\mathbb{R})$ is a $d \times d$-matrix and $\mathbf{x} = (x_1, \ldots, x_d)^t \in \mathbb{R}^d$ a column vector. We will only consider real or even integral quadratic forms, i.e. the matrix Q has real

[1]The exact time when Diophantus lived is not known. But because he cited Hypsikles of Alexandria we can assume that he lived after 150BC. For several reasons one assumes that he lived about 250AD [Mac].

or integral entries. Furthermore, we assume that all quadratic forms are symmetric, i.e. we have that $q_{ij} = q_{ji}$ or equivalently $Q = Q^t$. Therefore, we will omit the attribute symmetric for the rest of this thesis.

We identify the space of symmetric quadratic forms with the space of symmetric matrices.

$$\mathscr{Q}^d := \left\{ Q \in M_d(\mathbb{R}) : Q = Q^t \right\} = \operatorname{Sym}^2 \left(\mathbb{R}^{d^*} \right). \tag{8.2}$$

The subset of positive definite quadratic forms $\mathscr{Q}^d_{>0}$ is given by

$$\mathscr{Q}^d_{>0} := \left\{ Q \in \mathscr{Q}^d : Q[\mathbf{x}] > 0 \, \forall \mathbf{x} \in \mathbb{R}^d - \{0\} \right\}, \tag{8.3}$$

and the subset of positive semidefinite quadratic forms is given by

$$\mathscr{Q}^d_{\geq 0} := \left\{ Q \in \mathscr{Q}^d : Q[\mathbf{x}] \geq 0 \, \forall \mathbf{x} \in \mathbb{R}^d \right\}.$$

If the dimension is clear we omit the exponent d.The space $\overline{\mathscr{Q}^d_{>0}}$ is the closure of $\mathscr{Q}^d_{>0}$ in $\mathscr{Q}^d$. We can equip $\mathscr{Q}$ with an inner product

$$\langle Q, Q' \rangle_{\mathscr{Q}} = \operatorname{tr}\left(Q \cdot Q'\right) = \sum_{i,j=1}^{d} q_{ij} q'_{ij} \tag{8.4}$$

thus $\mathscr{Q}$ gets the structure of a euclidean vectorspace of real dimension $\binom{d+1}{2} = \frac{1}{2}d(d+1)$.

Furthermore, we find that the space of positive definite quadratic forms $\mathscr{Q}^d_{>0}$ is an open full dimensional convex polyhedral cone in $\mathscr{Q}^d$ with apex in 0. In other words, $\mathscr{Q}^d_{>0}$ is invariant under **homotheties**, i.e. multiplication with positive scalars $\lambda \in \mathbb{R}^+$. Under the map $Q \mapsto Q^{-1}$ (matrix inverse) the cone $\mathscr{Q}^d_{>0}$ is mapped to its dual cone with respect to the introduced inner product. Voronoï found that $\mathscr{Q}^d_{>0}$ is self dual. This fact was used by Avner Ash [Ash77] (for a summary see e.g. [McC98; (2.5)]) to extend the theory of self dual cones and to prove the existence of deformation retracts of lowest possible dimension for some groups using this extension. See Section 9.1 for more details on this.

8.2. Quadratic Forms and Symmetric Spaces

We have two natural actions of the group $\mathbf{GL}_d(\mathbb{R})$ on $\mathscr{Q}$, $\mathscr{Q}^d_{>0}$, and $\overline{\mathscr{Q}^d_{>0}}$. First, we have a right action of $\mathbf{GL}_d(\mathbb{R})$ given by

$$\begin{aligned} R_g : \mathscr{Q} &\longrightarrow \mathscr{Q} \\ Q &\longmapsto R_g(Q) := g^t \cdot Q \cdot g \end{aligned} \tag{8.5}$$

and secondly with have a left action, which is given by

$$\begin{aligned} L_g : \mathscr{Q} &\longrightarrow \mathscr{Q} \\ Q &\longmapsto L_g(Q) := g \cdot Q \cdot g^t. \end{aligned} \tag{8.6}$$

We have that $R_g = L_{g^t}$. Each of these two group actions has only one orbit, because we know from linear algebra that there exists a linear substitution $S \in \mathbf{GL}_d(\mathbb{R})$ such that for every $Q \in \mathscr{Q}$ in $S^t Q S = \mathrm{Id}_d$. In other words we have for every $Q \in \mathscr{Q}$ a decomposition

$$Q = S^t \cdot D \cdot S$$

with a diagonal matrix $D = \operatorname{diag}(D_1, \ldots, D_d)$ and a unipotent upper triangular matrix S. Since Q is positive definite all $D_i > 0$. Hence, we get $A = \sqrt{D} \cdot S \in \mathbf{GL}_d(\mathbb{R})$ and get the well known **Cholesky decomposition** $Q = A^t \cdot A$ which is unique up to orthogonal transformation, i.e. A and OA for any $O \in \mathbf{O}_d$ define the same quadratic form Q. For more information on the action of these groups on the cones see the book of Jean Pierre Serre [Ser73; V.1.1]. In the following, we will only consider the right action defined above, but all results stay (up to the changes related to this action) valid if we exchange the right action by the left action.

We call two quadratic forms $Q, Q' \in \mathscr{Q}$ **arithmetically equivalent** if there is an element $\gamma \in \mathbf{GL}_d(\mathbb{Z})$ such that $Q' = R_\gamma(Q) = \gamma^t \cdot Q \cdot \gamma$ holds. It is easy to check that this is an equivalence relation.

Proposition 8.2.1: *Let $Q, Q' \in \mathscr{Q}$ be two arithmetical equivalent positive quadratic forms then $Q[\mathbb{Z}^d] = Q'[\mathbb{Z}^d]$.*

Proof. If Q and Q' are arithmetically equivalent positive quadratic forms, there is a $\gamma \in \mathbf{GL}_d(\mathbb{Z})$ s.th. $Q' = \gamma^t \cdot Q \cdot \gamma$ holds. Then we have for any $\mathbf{x} \in \mathbb{Z}^d$ that $Q'[\mathbf{x}] = Q[\gamma \cdot \mathbf{x}]$ and $\gamma \cdot \mathbb{Z}^d = \mathbb{Z}^d$. □

This leads to the idea of the German mathematician Carl-Friedrich Gauß (1777– 1855), who started in 1840 to interpret the value of a quadratic form on lattice vectors. He used for a $Q \in \mathscr{Q}^d_{>0}$ the mentioned Cholesky decomposition $Q = A^t \cdot A$ by

$$Q[\mathbf{x}] = \mathbf{x}^t \cdot Q \cdot \mathbf{x} = \mathbf{x}^t \cdot A^t \cdot A \cdot \mathbf{x} = \|A \cdot \mathbf{x}\|^2. \tag{8.7}$$

This was the starting point of a theory due to Hermann Minkowski (1864–1909) which connects the theory of lattices with number theory known as geometry of numbers. If we have a basis $(\mathbf{a}_1, \ldots, \mathbf{a}_d)$ then we can identify the lattice

$$\Lambda = \mathbf{a}_1 \cdot \mathbb{Z} \oplus \ldots \oplus \mathbf{a}_d \cdot \mathbb{Z}$$

where the the $\mathbf{a}_i$ are column vectors of the matrix $A = (\mathbf{a}_1, \cdots, \mathbf{a}_d)$. If $(\mathbf{a}_1, \ldots, \mathbf{a}_d)$ is a basis of Λ, then this $d \times d$ Matrix A has rank d and thus $A \in \mathbf{GL}_d(\mathbb{R})$. Since $\mathbb{Z}^d$ and $\gamma \cdot \mathbb{Z}^d$ for some $\gamma \in \mathbf{GL}_d(\mathbb{Z})$ define the same lattice, they only differ by the choice of their bases, so we can write every lattice as $\Lambda = \gamma \cdot A \cdot \mathbb{Z}^d$ for a $A \in \mathbf{GL}_d(\mathbb{R})$ and $\gamma \in \mathbf{GL}_d(\mathbb{Z})$. Therefore, a lattice can be identified with an element out of the quotient $\mathbf{GL}_d(\mathbb{Z})\backslash\mathbf{GL}_d(\mathbb{R})$. As mentioned above the matrix $A \in \mathbf{GL}_d(\mathbb{R})$ is unique only up to action of the orthogonal group $\mathbf{O}_d$. This shows the following theorem:

Theorem 8.2.2: *There is a one-to-one correspondence between the space of quadratic forms of dimension d and the symmetric space for $\mathbf{GL}_d$:*

$$\underbrace{\mathscr{Q}^d_{>0}}_{\text{pos. def. quadratic forms}} \quad \overset{1:1}{\longleftrightarrow} \quad \underbrace{\mathbf{O}_d\backslash\mathbf{GL}_d(\mathbb{R}).}_{\text{bases up to orth. trans.}}$$

Furthermore, we have also a one-to-one correspondence between quadratic forms of dimension d up to arithmetic equivalence and a locally symmetric space

$$\underbrace{\mathscr{Q}^d_{>0}/\mathbf{GL}_d(\mathbb{Z})}_{\substack{\textit{pos. def. quadratic forms}\\ \textit{up to arithm. equiv.}}} \quad \overset{1:1}{\longleftrightarrow} \quad \underbrace{\overbrace{\mathbf{O}_d\backslash\mathbf{GL}_d(\mathbb{R})}^{\textit{space of lattices}}/\mathbf{GL}_d(\mathbb{Z})}_{\textit{lattices up to isometries}}.$$

Using e.g. the Iwasawa decomposition it is easy to see that an analogue result holds if we divide out homotheties. Denote by $\mathbb{P}\mathscr{Q}^d_{>0}$ the quotient of $\mathscr{Q}^d_{>0}$ by homotheties, which is a subset of the projectivization $\mathbb{P}\mathscr{Q}$ of the space of (symmetric) quadratic forms $\mathscr{Q}$. In other words we have an embedding

$$\mathbb{P}\mathscr{Q}^d_{>0} \hookrightarrow \mathbb{P}\mathscr{Q}. \tag{8.8}$$

We summarize:

Lemma 8.2.3: *There is a one-to-one correspondence between the space of quadratic forms of dimension d up to homothety and the symmetric space for $\mathbf{SL}_d$:*

$$\mathbb{P}\mathscr{Q}^d_{>0} \quad \overset{1:1}{\longleftrightarrow} \quad \mathbf{SO}_d\backslash\mathbf{SL}_d(\mathbb{R})$$

and analogously a one-to-one correspondence

$$\mathbf{SL}_d(\mathbb{Z})\backslash\mathbb{P}\mathscr{Q}^d_{>0} \quad \overset{1:1}{\longleftrightarrow} \quad \mathbf{SO}_d\backslash\mathbf{SL}_d(\mathbb{R})/\mathbf{SL}_d(\mathbb{Z}).$$

We will discuss some more details of this connection later in Section 8.4, where we discuss the result of Voronoï for the groups $\mathbf{GL}_d$ or $\mathbf{SL}_d$ resp., and some extension to the group $\mathbf{Sp}_g$ in Section 9.1.

8.3. Perfect Quadratic Forms

Let for the following all quadratic forms be integral. For a positive definite (integral) quadratic form $Q \in \mathscr{Q}^d_{>0}$ there is an $n \in \mathbb{N}$ such that the Diophantine equation $Q[\mathbf{x}] = n$ has an integer vector valued solution. We define the **arithmetic minimum** as

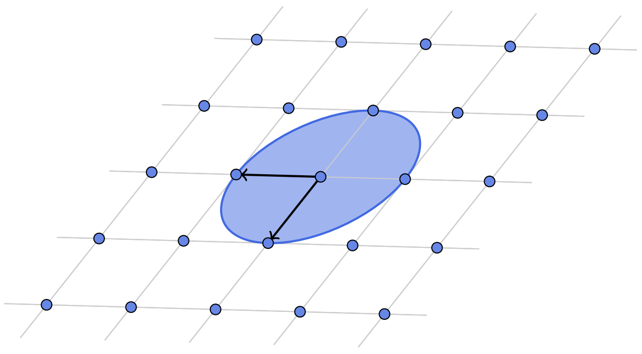
$$\mu(Q) := \min_{\mathbf{x}\in\mathbb{Z}^d-\{0\}} Q[\mathbf{x}]. \tag{8.9}$$

We have that $\mu(\lambda Q) = \lambda\cdot\mu(Q)$ for $\lambda \in \mathbb{R}^+$. For a positive definite quadratic form $Q \in \mathscr{Q}^d_{>0}$ the arithmetic minimum $\mu(Q)$ is always positive. Every positive definite quadratic form defines a real valued strictly convex function on $\mathbb{R}^d$. For every $c \in \mathbb{R}^+$ the set

$$E(Q,c) := \left\{\mathbf{x} \in \mathbb{R}^d : Q[\mathbf{x}] \leq c\right\} \tag{8.10}$$

forms an ellipsoid and the points of a constant value of a $Q \in \mathscr{Q}^d_{>0}$ form the surface of an ellipsoid. This give us a geometric interpretation of the arithmetic minimum $\mu(Q)$. The

Figure 8.1.: *Example for $d = 2$ for some $E(Q, \mu(Q))$.*



arithmetic minimum is the smallest natural number $n \in \mathbb{N}$ for which the intersection between $E(Q, n)$ and the lattice $\mathbb{Z}^d$ contains a non-zero element. Obviously the points which attain the arithmetic minimum lie on the boundary of the ellipsoid $E(Q, \mu(Q))$. The set of representatives of the arithmetic minimum

$$\mathrm{Min}\,(Q) = \{\mathbf{x} \in \mathbb{Z}^d : Q[\mathbf{x}] = \mu(Q)\} \tag{8.11}$$

is called set of **minimal vectors**. Because the boundary of the ellipsoid $E(Q, \mu(Q))$ is compact there are only finitely many points of $\mathbb{Z}^d$ where Q can take the minimum. Therefore, we have that $|\mathrm{Min}\,(Q)| < \infty$. A quadratic form $Q \in \mathscr{Q}^d_{>0}$ is called **perfect** if Q is uniquely determined by its arithmetic minimum $\mu(Q)$ and its set of minimal vectors $\mathrm{Min}\,(Q)$.

Lemma 8.3.1: *If $Q, Q' \in \mathscr{Q}^d_{>0}$ are arithmetically equivalent positive definite quadratic forms with $Q' = R_g(Q) = \gamma^t \cdot Q \cdot \gamma$ with $\gamma \in \mathbf{GL}_d(\mathbb{Z})$ then*

$$\mu(Q) = \mu(Q') \qquad \textit{and} \qquad \mathrm{Min}\,(Q') = \gamma^{-1} \cdot \mathrm{Min}\,(Q)\,.$$

Furthermore, Q is perfect if and only if Q' is perfect.

Proof. An easy calculation gives $Q'[\mathbf{x}] = Q[\gamma \cdot \mathbf{x}]$. The remaining part follows because every matrix $\gamma \in \mathbf{GL}_d(\mathbb{Z})$ maps $\mathbb{Z}^d$ isomorphically onto $\mathbb{Z}^d$. □

The matrix $A \in \mathbf{GL}_d(\mathbb{R})$ maps the standard lattice $\mathbb{Z}^d$ to the lattice $\Lambda = A \cdot \mathbb{Z}^d$ and the ellipsoid $E(Q, \mu(Q))$ is transformed to

$$A \cdot E(Q, \mu(Q)) = \sqrt{\mu(Q)} \cdot B_d, \tag{8.12}$$

where $B_d = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\| = \sqrt{\mathbf{x}^t \cdot \mathbf{x}} \leq 1\}$ is the euclidean unit ball. The arithmetic minimum $\mu(Q)$ of Q is the square of the length of the shortest non-zero lattice vector and the determinant $\det Q$ of the quadratic form Q may be interpreted geometrically as the square of the volume of a fundamental cell of the lattice. This is the point where we have the relation to different kinds of sphere packings (compare [CS99]). Let

$$\delta(Q) := \frac{\mu(Q)}{\sqrt[d]{\det Q}}. \tag{8.13}$$

This is a density function, which was introduced by Hermite, which measures the density of a sphere packing of the $\mathbb{R}^d$ by spheres centered at the lattice points. An **extremal** form is a local minimum of the density δ.

Furthermore, we have:

Theorem 8.3.2: *An integral quadratic form $Q \in \mathscr{Q}$ is perfect if and only if*

$$\mathrm{span}_{\mathrm{Cone}}\{\mathbf{x} \cdot \mathbf{x}^t : \mathbf{x} \in \mathrm{Min}\,(Q)\} = \mathscr{Q}.$$

Here $\mathrm{span}_{\mathrm{Cone}}$ means the conal span, i.e. the space which is generated by all non-negative linear combinations with real coefficients.

The form $\mathbf{x} \cdot \mathbf{x}^t$ has only rank 1. Since $\pm\mathbf{x}$ give the same rank-1 form $\mathbf{x}\mathbf{x}^t$ the number of elements in the set of minimal vectors $|\mathrm{Min}\,(Q)|$ of a perfect form $Q \in \mathscr{Q}^d_{>0}$ is at least $(d+1)d$. Since it is impossible that the $\mathbf{x} \cdot \mathbf{x}^t$ for $\mathbf{x} \in \mathrm{Min}\,(Q)$ span $\mathscr{Q}$ if they do not span $\mathbb{R}^d$, we have:

Proposition 8.3.3: *If $Q \in \mathscr{Q}^d_{>0}$ is (integral) perfect then $\mathrm{Min}\,(Q)$ spans $\mathbb{R}^d$ as real vectorspace.*

In fact, if $\mathrm{Min}\,(Q)$ does not span $\mathbb{R}^d$, the elements $\mathbf{x}\cdot\mathbf{x}^t$ can maximally span a $\binom{d}{2}$-dimensional subspace of $\mathscr{Q}$. Korkin and Zolotarev proved that extremal forms are perfect forms. Further, Voronoï proved that for $d \geq 6$ the converse is false (for a proof see [Bar57]).

We call a form Q **eutactic** if its dual form Q^{-1} is contained in the cone generated by the minimal vectors of Q. This cone is called Voronoi cone (see (8.15) for the definition). Voronoï introduced that notion only in the cases where Q is perfect and did not use the word eutactic, which was introduced later by Harold Scott MacDonald Coxeter (1907–2003) in 1951. But Voronoï proved in [Vor08b; 17]:

Theorem 8.3.4 (Voronoï, 1907)**:** *A quadratic form is extreme if and only if it is perfect and eutactic.*

Without explaining this in more detail this gives an easierer way to check perfect forms for extremality, which plays some role in Voronoï's algorithm for finding perfect forms (Algorithm 1)

8.4. Voronoï I

Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_s\}$ be any non empty finite set of column vectors $\mathbf{x}_i \in \mathbb{Z}^d$. We define a real cone generated by the set X by

$$\mathscr{R}(X) := \left\{ \sum_{i=1}^{s} \rho_i \mathbf{x}_i \cdot \mathbf{x}_i^t : \rho_i > 0 \right\} =: \mathrm{cone}\left(\mathbf{x} \cdot \mathbf{x}^t : \mathbf{x} \in X \right) \subset \mathscr{Q}. \tag{8.14}$$

For any finite set X of vectors in $\mathbb{Z}^d$ this set $\mathscr{R}(X)$ is an open rational polyhedral cone, i.e. the cone has only finitely many faces and these are given by rational equations. The closure of this cone is obviously given by

$$\overline{\mathscr{R}(X)} := \left\{ \sum_{i=1}^{s} \rho_i \mathbf{x}_i \cdot \mathbf{x}_i^t : \rho_i \geq 0 \right\}.$$

If we have such a set $X = \{\mathbf{x}_i\}$ we do not make a difference between sets which define the same cone $\mathscr{R}(X)$. Hence, we introduce an equivalence relation on the set of finite sets of column vectors in $\mathbb{Z}^d$ of the same length. Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_s\}$ and $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_s\}$, then we call X and Y equivalent if there is a permutation $\sigma \in \mathbb{S}_s$ such that $x_i = y_{\sigma(i)}$ or $x_i = -y_{\sigma(i)}$ for all $i = 1, \ldots, s$. To indicate the equivalence classes we will enclose the set in brackets $[X] := [\mathbf{x}_1, \ldots, \mathbf{x}_i]$. In the following, we will most time identify all sets of column vectors with their equivalence classes. This is useful to simplify notation in context of generating sets for cones and cells. In particular, if we write $X \subset X'$ for some sets of column vectors X and X' we mean that there is a representative X_0 of the class $[X]$ of X and a representative X_0' of the class $[X']$ of X' such that $X_0 \subset X_0'$. If $X \subset X'$ we have obviously by definition of the $\mathscr{R}$ that $\mathscr{R}(X) \subset \overline{\mathscr{R}(X')}$. Thus, we have a partial ordering on the set of cones of the form $\mathscr{R}$ induced by the partial ordering coming from the inclusion relations on the generating sets. In other words we have here the structure of a partially ordered set (poset) as well on the space of cones as on the space of generating sets which are compatible.

In the special case that $X = \mathrm{Min}\,(Q)$ for a $Q \in \mathscr{Q}^d_{>0}$ we define the **Voronoï cone** or **Voronoï domain** by

$$\mathscr{V}(Q) = \mathscr{R}(\mathrm{Min}\,(Q)) = \mathrm{cone}\left\{\mathbf{x} \cdot \mathbf{x}^t : \mathbf{x} \in \mathrm{Min}\,(Q)\right\}. \tag{8.15}$$

Remark: Also subcones of such a $\mathscr{V}(Q)$ are called Voronoï cones.

For every closed face $\overline{\mathscr{F}}$ of $\overline{\mathscr{V}(Q)}$ for $Q \in \mathscr{Q}^d_{>0}$ the relative interior $\mathrm{Int}\left(\overline{\mathscr{F}}\right) = \mathscr{F}$ of $\overline{\mathscr{F}}$, i.e. the interior of $\overline{\mathscr{F}}$ in its span, is a rational cone which either lies in $\mathscr{Q}^d_{>0}$ or is disjoint from $\mathscr{Q}^d_{>0}$.

Now we go the other way round. Let $\mathscr{X}$ be any closed non-zero face[2] of $\mathscr{V}(Q)$ for some $Q \in \mathscr{Q}^d_{>0}$, then $\mathscr{X}$ is generated by some subset of $X \subseteq \mathrm{Min}\,(Q)$. In other words, if we set

$$\mathfrak{X}(Q, \mathscr{X}) = \{\mathbf{x} \cdot \mathbf{x}^t \in \mathscr{X} : \mathbf{x} \in \mathrm{Min}\,(Q)\} \subset \mathscr{Q}^d \tag{8.16}$$

then we have $\mathscr{X} = \overline{\mathscr{R}\,(\mathfrak{X}(Q, \mathscr{X}))}$. One easily deduces

Corollary 8.4.1: *The Voronoï cone $\mathscr{V}(Q)$ of some $Q \in \mathscr{Q}^d_{>0}$ is full dimensional if and only if Q is perfect.*

We introduced in (8.4) the structur of an euclidean vectors space on $\mathscr{Q}$. We find that the Voronoï cone is the normal cone with respect to this inner product, i.e.

$$\left\{N \in \mathscr{Q}^d : \langle N, Q\rangle \leq \langle N, Q'\rangle \,\forall Q' \in \mathcal{P}_{\mu(Q)}\right\}$$

where $\mathcal{P}_\lambda$ is the **Ryshkov polyhedron** for a non-negative real parameter λ defined by

$$\mathcal{P}_\lambda = \left\{Q \in \mathscr{Q}^d_{>0} : \left\langle Q, \mathbf{x} \cdot \mathbf{x}^t\right\rangle \geq \lambda \forall \mathbf{x} \in \mathbb{Z}^d - \{0\}\right\}. \tag{8.17}$$

[2]Non zero mean different from the apex of the cone.

Algorithm 1: Voronoï's algorithm for finding perfect quadratic forms

```
Input   : Dimension $d$, $Q_0$ perfect quadratic form
Output : A complete list $\mathrm{List}_{\mathrm{perf}}$ of inequivalent perfect forms in $\mathscr{Q}^d_{>0}$
$\mathrm{List}_{\text{perf, new}} = \{Q_0\}$;
$\mathrm{List}_{\mathrm{perf}} = \{Q_0\}$;
repeat
    for $Q \in \mathrm{List}_{\textbf{perf, new}}$ do
        Compute $\mathrm{Min}\,(Q)$ and the describing inequalities of the polyhedral cone
            $$\mathcal{P}(Q) = \{Q' \in \mathscr{Q}^d : Q'[\mathbf{x}] \geq 0, \forall \mathbf{x} \in \mathrm{Min}\,(Q)\}$$
        Enumerate extreme rays $R_1, \dots, R_k$ of $\mathcal{P}(Q)$;
        Find contiguous perfect forms $Q_i = Q + \alpha R_i$ for $i = 1, \dots, k$ and
            $$\alpha = \min_{\rho \in \mathbb{N}} (\rho : \mu\,(Q + \rho R) = \mu(Q_0))$$
        if Test: $Q_i \not\sim Q$ for some $Q \in \mathrm{List}_{\textbf{perf}}$ then
            Add $Q_i$ to $\mathrm{List}_{\text{perf,new}}$, Add $Q_i$ to $\mathrm{List}_{\mathrm{perf}}$;
until $\mathrm{List}_{\textbf{perf, new}} = \emptyset$;
return $\mathrm{List}_{\mathrm{perf}}$
```

The Ryshkov polyhedra $\mathcal{P}_\lambda$ are for any λ an intersection of infinitely many halfspaces. But if $d \geq 1$ and $\lambda > 1$ the Ryshkov polyhedra $\mathcal{P}_\lambda$ are locally finite polyhedra. The theory of Ryshkov polyhedra can be used to give a geometric picture of Voronoï's construction and his algorithm to compute the reduction for spaces of quadratic forms (see e.g.[Gru07, Gru09]).

The right action of $\mathbf{GL}_d(\mathbb{Z})$ on $\mathscr{Q}$ introduced in (8.5) defines also an action on its subcones, in particular on $\mathcal{P}_\lambda$, where it permutes vertices, edges and facets of them. A lattice $A \cdot \mathbb{Z}^d$ is called **perfect** if $A^t \cdot A$ defines a perfect quadratic form. Bases of perfect lattices correspond to vertices of a locally finite Ryshkov polyhedron $\mathcal{P}_\lambda$ with $\lambda = 4 \cdot \mu(Q)^2$. Now Voronoï states:

Theorem 8.4.2 (G. Voronoï, 1907): *[Vor08b; p. 110] Up to arithmetic equivalence and homotheties there exist only finitely many perfect forms in a given dimension $d \geq 1$.*

One can now deduce from the statements from above that it is enough to know a complete (finite) list of all perfect forms to generate the set of all (positive definite, integral) quadratic forms via taking Voronoï cones and translating them under the action of of the group $\mathbf{GL}_d(\mathbb{Z})$. For more details on this we refer to the literature (see e.g. [CS99, Sch09]) and the description of Voronoï's reduction later in this chapter (see page 46 f.).

Therefore, it is an obvious question if there is an algorithm to give us such a list. Fortunately the answer to this question is *Yes*. Voronoï gave such an algorithm in [Vor08b] and presented a list of perfect forms up to arithmetic equivalence which he computed by hand for the cases up to $d = 5$. We state his algorithm without describing all the details in Algorithm 1. To start the algorithm one needs at least one perfect form, which is always possible (see e.g. [CS99; Section 6.1]).

Table 8.1.: *Known number of perfect and extremal quadratic forms up to arithmetic equivalence with their finders [Vor08b, Bar57, Sch07].*

d	number of perfect forms	number of extremal forms	author(s)	year
2	1	1	Lagrange	1773
3	1	1	Gauß	1840
4	2	2	Korkin & Zolotarev	1877
5	3	3	Korkin & Zolotarev	1877
6	7	6	Barnes	1957
7	33	30	Jaquet-Chiffelle	1993
8	10916	2408	Dutour, Sikirić, Schürmann, Vallentin, Riener	2005
9	$> 5\times 10^5$		Dutour, Sikirić, Schürmann, Vallentin, Riener	2005

The first implementation on a computer was done by Larmouth in 1971, who was able to verify the results of Barnes [Bar57] up to $d = 6$. Today there are many implementations of Voronoï's algorithm available, for example in computer algebra systems GAP or Magma. With some changes Voronoï's algorithm is still used to find all perfect forms. But todays algorithms are much faster than Voronoï's original version of that algorithm. For example, one uses some variation of the LLL-algorithm [LLL82] to find the $\mathrm{Min}\,(Q)$ and the α in Algorithm 1 which is much faster than the original version of Voronoï. Further, there are today better methods than in 1908 to determine whether forms are equivalent. For more details on the algorithm and its implementations we refer e.g to [Sch07; 3.1.4-3.1.5] and [Sch09]. Table 8.1 shows the known numbers of perfect quadratic forms. Until $d = 6$ the computation can be done easily with usual computers, but the number of perfect forms and therefore the runtime grows quite fast with ascending d. In other words, it is computationally hard to compute the case $d = 9$ or even higher d also with modern computer systems and algorithms. Fortunately we need only the results for $d = 4$ which are known since 1887. One can find a complete list of inequivalent perfect quadratic forms and their sets of minimal vectors for $d \le 7$ in [CS88]. For $d = 4$ we have only two classes of perfect quadratic forms Q_1 and Q_2 (see Table 8.2), which are related to lattices of type D_4 and A_4, respectively. For them there are 12 or 10 pairs of minimal vectors. A list of these minimal vectors can be found in Table 8.2.

Voronoï's first reduction **(Voronoï I)** is based on perfect forms and Theorem 8.4.2. We describe briefly the idea how it works. For any (integral) $Q \in \mathscr{Q}^d_{>0}$ we have that $\mathscr{V}(Q)$ is a fulldimensional cone, i.e. $\dim \mathscr{V}(Q) = \dim \mathscr{Q}^d = \binom{d+1}{2}$, if and only if Q is perfect. Furthermore, the set of all Voronoï cones forms a partially ordered set (poset) with respect to inclusion and defines a polyhedral subdivision of

$$\mathcal{P}^d := \mathrm{cone}\{\mathbf{x}\mathbf{x}^t : \mathbf{x} \in \mathbb{Z}^d\},$$

which is the rational closure of $\mathscr{Q}^d_{>0}$ for which

$$\mathscr{Q}^d_{>0} \subset \mathcal{P}^d \subset \mathscr{Q}^d_{\ge 0} \tag{8.18}$$

holds (see e.g. [Sch07; p. 54]). Since further only Voronoï cones of perfect forms are full

Table 8.2.: *List of perfect quadratic forms (up to arithmetic equivalence) and their sets of pairs of minimal vectors in dimension 4. For each minimal vector also -1 times that vector is minimal. [CS88]*

Quadratic Form	**Corres-ponding Lattice**	**Minimal Vectors**
$Q_1 = \begin{pmatrix} 2 & 1 & 1 & 2 \\ 1 & 2 & 1 & 2 \\ 1 & 1 & 2 & 2 \\ 2 & 2 & 2 & 4 \end{pmatrix}$	D_4	$\begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \begin{pmatrix}1\\-1\\0\\0\end{pmatrix}, \begin{pmatrix}1\\0\\-1\\0\end{pmatrix}, \begin{pmatrix}1\\0\\0\\-1\end{pmatrix}, \begin{pmatrix}0\\1\\-1\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\\-1\end{pmatrix}, \begin{pmatrix}0\\0\\1\\-1\end{pmatrix}, \begin{pmatrix}0\\1\\1\\-1\end{pmatrix}, \begin{pmatrix}1\\0\\1\\-1\end{pmatrix}, \begin{pmatrix}1\\1\\0\\-1\end{pmatrix}$
$Q_2 = \begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{pmatrix}$	A_4	$\begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\\1\end{pmatrix}, \begin{pmatrix}1\\0\\0\\-1\end{pmatrix}, \begin{pmatrix}1\\0\\-1\\0\end{pmatrix}, \begin{pmatrix}1\\-1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\\-1\end{pmatrix}, \begin{pmatrix}0\\1\\0\\-1\end{pmatrix}, \begin{pmatrix}0\\1\\-1\\0\end{pmatrix}$

dimensional (compare Prop. 8.3.3) we have

$$\mathcal{P}^d = \bigcup_{Q \text{ perfect}} \mathscr{V}(Q)\,. \tag{8.19}$$

One gets by (8.18) and (8.19) a covering by perfect Voronoï cones, on which the group $\mathbf{GL}_d(\mathbb{Z})$ acts by permutations. This action is induced by the multiplication of the elements in the defining sets of the cones with a matrix in $\mathbf{GL}_d(\mathbb{Z})$. We will take a closer look at this action later on page 55 in case of the action of the symplectic group, which is essentially the same. We postpone the discussion how the action looks like in detail to that place. In fact, Voronoï proved that:

Definition 8.4.3 (Voronoï)**:** *Let $\mathcal{P}(\mathbb{Z}^d)$ be the set of all sets of column vectors in $\mathbb{Z}^d$. The* **Voronoï I decomposition** *is defined by*

$$\mathscr{Q}^d_{>0} = \bigcup_{X \in \mathfrak{R}} \mathscr{R}(X),$$

where

$$\mathfrak{R} = \left\{ X \in \mathcal{P}(\mathbb{Z}^d) : \exists \textit{ perfect } Q \in \mathscr{Q}^d_{>0} : X \subset \operatorname{Min}(Q)\,, \mathscr{R}(X) \subset \mathscr{V}(Q) \textit{ face} \right\}_{/\sim}.$$

We say $X \sim X'$ if the cones spanned by X and X' coincide, i.e. $\mathscr{R}(X) = \mathscr{R}(X')$.

We set

$$\mathscr{R}(\mathfrak{R}) := \{\mathscr{R}(X) : X \in \mathfrak{R}\}\,. \tag{8.20}$$

Each $\mathscr{F} \in \mathscr{R}(\mathfrak{R})$ is also called a **Voronoï cone**.

Thus, we have a one-to-one correspondence between full dimensional Voronoï cones and some subsets of $\mathcal{P}(\mathbb{Z}^d)$. Let $R \subset \mathscr{Q}^d_{>0}$ be a full dimensional Voronoï cone, then there is at least one finite set $X \in \mathcal{P}(\mathbb{Z}^d)$ such that $R = \mathscr{R}(X)$ and X spans $\mathbb{R}^d$. Let

$$\mathscr{S}(X) = \left\{Q \in \mathscr{Q}^d_{>0} : [\mathrm{Min}\,(Q)] = [X]\right\}, \tag{8.21}$$

then we have, if $\mathscr{S}(X) \neq \emptyset$, that $\mathscr{R}(Y) = \mathscr{R}(X) = R$ for $Y \in \mathscr{S}(X)$. One can easily show that $\mathscr{R}(X) = \mathscr{S}(X)$ if X spans $\mathbb{R}^d$ (use [McC91; (2.5)]). Therefore, a full dimensional Voronoï cone R defines up to the previously introduced equivalence relation (see p. 44) a unique class $[X]$ and we write $X = X(R)$ if $R = \mathscr{R}(X(R))$. This relation decends to non maximal Voronoï cones because they are subcones of maximal cones and their defining sets correspond to sets which span lower dimensional spaces for which we can do the same.

The action defined in (8.5) on quadratic forms induces an associated action on maximal Voronoï cones in the following way: Let $R = \mathscr{R}(X(R))$ be a maximal Voronoï cone, then there is a quadratic form $Q \in \mathscr{Q}^d_{>0}$ such that $[X(R)] = [\mathrm{Min}\,(Q)]$. Since $X(R)$ has to span $\mathbb{R}^d$ the form Q has to be perfect by Corollary 8.4.1. We know by Lemma 8.3.1 that for any $\gamma \in \mathbf{GL}_d(\mathbb{R})$ we have that $\mathrm{Min}\left(R_\gamma(Q)\right) = \gamma^{-1}\cdot\mathrm{Min}\,(Q)$. This defines an action on the maximal cones, which descends also to their subcones. Let $X \in \mathfrak{R}$ and $\gamma \in \mathbf{GL}_d(\mathbb{R})$ then we have

$$R_\gamma(\mathscr{R}(X)) = \gamma^t \cdot \mathscr{R}(X) \cdot \gamma = \mathscr{R}(\gamma^{-1} \cdot X). \tag{8.22}$$

Further, Voronoï gave in [Vor08b] the following theorem, which shows that the previous definition makes sense.

Theorem 8.4.4 (Main Theorem of Voronoï's Reduction Theory, 1907): *a) For each Voronoï cone $\mathscr{F} \in \mathfrak{R}$, the set $\mathfrak{X}(Q, \mathscr{F})$ is independent of Q. More precisely, if $Q, Q' \in \mathscr{Q}^d_{>0}$ are perfect forms and if $\mathscr{L}$ and $\mathscr{L}'$ are non-zero closed faces of $\mathscr{V}(Q)\,, \mathscr{V}(Q')$ respectively such that $\mathscr{F} = \mathrm{Int}\,(\mathscr{L}) = \mathrm{Int}\left(\mathscr{L}'\right)$ then*

$$\mathfrak{X}(Q, \mathscr{F}) = \mathfrak{X}(Q', \mathscr{F}).$$

Hence, we will write $\mathfrak{X}(Q, \mathscr{F}) = \mathfrak{X}(\mathscr{F})$.

b) $\mathscr{R}$ defines a locally finite decomposition into open cones (cells), i.e.

$$\mathscr{Q}^d_{>0} = \bigsqcup_{X \in \mathfrak{R}} \mathscr{R}(X)$$

and a given $\mathscr{F} \in \mathfrak{R}$ is only in the closure of finitely many elements of $\mathfrak{R}$.

c) The action of $\mathbf{GL}_d(\mathbb{Z})$ on $\mathfrak{R}$ has only finitely many orbits.

Proof. Part b) is proved in [Vor08b; 21,24]. Part a), which implies that in b) the union is disjoint, is proved in [Vor08b; 20] and Part c) follows from Theorem 8.4.2. □

Since there are only finitely many perfect forms modulo $\mathbf{GL}_d(\mathbb{Z})$ and homothethies and hence also modulo $\mathbf{SL}_d(\mathbb{Z})$, we have only finitely many orbits in the whole decomposition in b) of Theorem 8.4.4. In Chapter 9 we will explain how to get from this result a cellular decomposition in our case.

9. Reduction Theory for the Symplectic Group

One should always generalize.

(Carl Jacobi)

Before we extend the method of Robert MacPherson and Mark McConnell to the symplectic group $\Gamma = \mathbf{Sp}_2(\mathbb{Z})$ we sketch their strategy (see Section 9.1.1) and give a short review on [MM89, MM93] without proofs (see Sections 9.1.2 and 9.1.3). For proofs we refer to their original articles. Our extension of their results is based on the fact that – as MacPherson and McConnell mention in [MM93; p. 620] – their construction for the retract of the Siegel upper half-space $\mathfrak{S}_2$ does not depend on the choice of a neat subgroup of Γ. In a subsequent publication [McC98] Mark McConnell has removed the condition that the group has to be neat for the case of subgroups of the group $\mathbf{SL}_n(\mathbb{Z})$ and replaced it by subgroup of finite index. It seems that at least the authors and some specialists in this area like Paul Gunnells know or perhaps only expect that a weaker version of the main theorem [MM93; Theorem 8.8] is true for symplectic groups which are not torsion free. We will see later (in Section 9.2) that indeed an analogue theorem to [MM93; Theorem 8.8] holds for any subgroup of Γ of finite index. In 2009 Paul Gunnells mentioned to me that he thinks that the result of [MM93] should be true also for the symplectic group. The analogue we will proof provides an orbicell decomposition which is a classical CW-complex if the subgroup of Γ is torsion free. This construction is the natural generalization of a cell complex to this context and is enough to compute later the cohomology of the Siegel modular variety. To prove this result one only has to go through the original proof and adapt, if necessary, some parts to the case that there is torsion. A similar approach was worked out by Paul Gunnells et. al. for the case of the group $\mathbf{SL}_4(\mathbb{Z})$, which according to him is much simpler (see [AGM02, AGM08, AGM10, Gun09]). But up to now no result for the symplectic group in this direction was published or used for other computations. So we decided to work out this more in detail to get a suitable topological description of the Siegel modular variety.

9.1. Reduction Theory for Neat Symplectic Groups by MacPherson and McConnell

I describe the method developed by Mark McConnell and Robert MacPherson in [MM93] to construct an explicit reduction theory for some subgroups of $\Gamma := \mathbf{Sp}_2(\mathbb{Z})$. Their method is based on Voronoï's reduction theory described in Chapter 8. The rough idea is to replace the Siegel upper half-space and its quotients by equivalent spaces which are as easy as possible, but have the same cohomology (compare Section 7.2). In particular, one tries to reduce the dimension of the considered space. Due to Theorem 12.3.4 of Armand Borel and Jean-Pierre Serre we can hope to find a 4-dimensional deformation retract $\mathbb{W}$ of the 6-dimensional space $\mathfrak{S}_2$ such that the quotient $\mathbb{W}/\Gamma$ is homotopy equivalent to $\mathfrak{S}_2/\Gamma$. As reference for this one can use the original paper [MM93] and its three satellite papers [MM89, McC91, McC98].

Since we are not interested in any kind of compactification we skip the part of [MM93] dedicated to this problem. In presence of the torsion elements in particular the behaviour of the boundary is significantly more complicated than in cases where the group is torsion free. Basically the restriction to neat subgroups of Γ in [MM93] is caused by the fact that the authors consider also compactifications of Siegel modular varieties such that they had to ensure that the action of the group extends in a nice way to the boundary of the variety, which is the case if the considered group is neat. For our case – we only consider the cohomology of the open space $\mathfrak{S}_2/\Gamma$ and not of a compactification of $\mathfrak{S}_2/\Gamma$ – we will see that we can remove those restriction, but we have to pay the price that we have to replace in general the cellular structure by a suitable analogue (see Section 9.2).

9.1.1. Overview of the Strategy of MacPherson and McConnell

The construction of MacPherson and McConnell roughly consists of four steps.

Step 1 Construct a decomposition of $\mathscr{Q}^4_{>0}$ into polyhedral cones.

Step 2 Reduce $\mathscr{Q}^4_{>0}$ modulo homotheties and get a decomposition of $\mathbb{X}_{\mathbf{SL}_4} = \mathbb{P}\mathscr{Q}^4_{>0}$ into polyhedral cells.

Step 3 Embed $\mathfrak{S}_2 \hookrightarrow \mathbb{P}\mathscr{Q}^4_{>0}$ and get a decomposition for $\mathfrak{S}_2$ induced by the decomposition of $\mathbb{P}\mathscr{Q}^4_{>0}$. One has to prove that the decomposition gives cells and is equivariant and to find a set of generators. This is the most difficult part of [MM93].

Step 4 Take the polyhedral decomposition for $\mathfrak{S}_2$ and take the first barycentric subdivision and remove all cells of that which are not dual cells of the original decomposition. This gives a deformation retract.

We will not present all details and most time refer for proofs to the original paper since especially the part that the resulting objects are really cells is rather technical and hence not helpful for the understanding of the method.

9.1.2. Using an Embedding

The idea of MacPherson and McConnell is extremely simple, but has some technical difficulties in detail as we will see later. We have seen in Lemma 8.2.3 that

$$\mathbb{X}_{\mathbf{SL}_d} = \mathbb{P}\mathscr{Q}^d_{>0} := \mathscr{Q}^d_{>0/_\sim}.$$

Further, we get from (8.5) for $Q = \mathrm{Id}$ a map $\mathbf{GL}_{2g}(\mathbb{R}) \supset \mathbf{Sp}_g(\mathbb{R}) \to \mathscr{Q}^{2g}$ by $x \mapsto x^t \cdot x$, which descends to a map on $\mathfrak{S}_g = \mathbb{X}_{\mathbf{Sp}_g}$

$$\mathfrak{S}_g \longhookrightarrow \mathscr{Q}^{2g}_{>0}. \tag{9.1}$$

By Lemma 6.3.5 and the fact that $\mathscr{Q}^{2g}_{>0}$ is a cone we find that the embedded image of $\mathfrak{S}_g$ under this map is preserved by the group of symplectic simmilitudes $\mathbf{GSp}_g(\mathbb{Z})$ and hence also by the group $\mathbf{Sp}_g(\mathbb{Z})$. The group $\mathbf{GSp}_g(\mathbb{Z})$ is the largest group with that property. By composition of this embedding with the projection map

$$p : \mathscr{Q}^{2g} - \{0\} \twoheadrightarrow \mathbb{P}\mathscr{Q}^{2g}_{>0}$$

we get an embedding

$$\begin{tikzcd} \mathfrak{S}_g \arrow[r, hook] \arrow[rrr, bend right, "j"'] & \mathscr{Q}^{2g}_{>0} \arrow[r, two heads, "p"] & \mathbb{P}\mathscr{Q}^{2g}_{>0} \arrow[r, hook] & \mathbb{P}\mathscr{Q}^{2g}. \end{tikzcd} \tag{9.2}$$

In the following, we will identify $\mathfrak{S}_g$ with its image $j(\mathfrak{S}_g)$ [MM93; (1.8)]. From the definition of the cone $\mathscr{R}(X)$ in (8.14) for any finite set of column vectors $X = [\mathbf{x}_1, \ldots, \mathbf{x}_s]$ we get that the image under the projection p of $\mathscr{R}(X)$ is a rational polyhedron, which we denote by $\mathbb{P}\mathscr{R}(X)$. We call $\mathbb{P}\mathscr{R}(X)$ a **Voronoï cell** with respect to X.

We introduced in Section 8.4 an easy order relation on cones coming from inclusion relations of their generating sets. We can use this poset structure to determine closures of cones. We want to emphasize that here it is important that this partial ordering is defined in terms of the equivalence classes of the column vectors defined earlier (see page 44) and not necessarily on the sets itself although we identify the classes with a suitable representative. This leads to:

Lemma 9.1.1: *A cone $\mathscr{R}(X')$ lies in the closure $\overline{\mathscr{R}(X)} \subset \mathbb{X}_{\mathbf{SL}_d}$ of $\mathscr{R}(X)$ if and only if $X \subset X'$.*

Let $\mathfrak{R}$ be the set of sets of column vectors from Definition 8.4.3. Then we have

$$\mathfrak{S}_g \subset \mathbb{P}\mathscr{Q}^{2g}_{>0} = \bigsqcup_{X \in \mathfrak{R}} \mathbb{P}\mathscr{R}(X) \tag{9.3}$$

and hence

$$\mathfrak{S}_g = \bigsqcup_{X \in \mathfrak{R}} \underbrace{\left(\mathbb{P}\mathscr{R}(X) \cap \mathfrak{S}_g\right)}_{=:\mathscr{B}(X)} = \bigsqcup_{X \in \mathfrak{R}} \mathscr{B}(X). \tag{9.4}$$

We call $\mathscr{B}(X)$ a **cell** if it is non empty and homeomorphic to some $\mathbb{R}^n$. Now there are two problems left:

Problem A Is $\mathscr{B}(X) \neq \emptyset$ and how can we decide whether $\mathscr{B}(X)$ is different from the empty set or not?

Problem B Is $\mathscr{B}(X)$ homeomorphic to $\mathbb{R}^n$ form some n (or to a single point for $n = 0$), i.e. is $\mathscr{B}(X)$ a cell?

The order relation for the cones descends to an order relation for the cells.

Corollary 9.1.2: *A cell $\mathscr{B}(X')$ lies in the closure $\overline{\mathscr{B}(X)} \subset \mathfrak{S}_2$ of $\mathscr{B}(X)$ if and only if $X \subset X'$. This defines a partial ordering $\lesssim$ on the set of X and makes them a partially ordered set (poset).*

First, there are obviously $X \in \mathfrak{R}$ such that $\mathscr{B}(X) \neq \mathbb{P}\mathscr{R}(X)$, because

$$\dim \mathfrak{S}_g \;\leq\; \dim \mathbb{X}_{\mathbf{SL}_{2g}}$$

with equality if and only if $g = 1$. Further, there is no general argument known to prove that the $\mathscr{B}(X)$ are cells [MM93]. A further question is whether a resulting cell complex will be regular, i.e. the closure of each cell is homeomorphic to some closed ball. There are strange things which could happen since there is no hidden linear structure (see page 56 for more explanation). For example, the boundaries of closures of cells could be non-simply-connected homology spheres instead of true spheres. Due to all these difficulties MacPherson and McConnell restricted themselves to the case $g = 2$ and did explicit the computations only in that case. We will do the very same and restrict our considerations from now on to the case $g = 2$.

According to Table 8.2 we have only two sets of 10 and 12 pairs of minimal vectors for the two different classes of perfect quadratic forms for $\mathbf{SL}_4(\mathbb{Z})$ or $\mathbf{GL}_4(\mathbb{Z})$ respectively, which are related to extremal root lattices of type A_4 and D_4. All subsets of these $\mathbf{x}_i$ are the candidates for sets X which we have to check first for equivalence and then for intersection.

But we and also MacPherson and McConnell are in the lucky situation that Mikhail Štogrin [Š74] and four years later independently Ronnie Lee and Robert Henry Szczarba [LS78] found a list of 21 classes of cones/cells represented by subsets of the minimal vectors from Table 8.2 for the larger group $\mathbf{GL}_4(\mathbb{Z})$. From the 21 classes only 18 are are not of boundary type (see below), thus we only have to check these 18 classes, which is not difficult, to answer Problem A. We get [MM93; pp. 586-588]

Theorem 9.1.3 (MacPherson, McConnell 1993)**:** *Let Σ be the set of sets of column vectors of the matrices in Table 9.1 which are not of boundary type then $\mathscr{B}(X) \neq \emptyset$ if and only if $[X] = [\gamma \cdot S]$ for some $S \in \Sigma$ and $\gamma \in \Gamma_0 := \mathbf{GSp}_2(\mathbb{Z})$. We call Σ the set of* **standard elements**. *All elements of Σ are inequivalent under the action of Γ_0. The dimension is given by $\dim_{\mathbb{R}} \mathscr{B}(X) = \mathrm{rank}\ (X)$ where $\mathrm{rank}\ (X)$ is the rank stated in Table 9.1. If x is a translate of an element in Σ we define the rank to be the rank of its preimage in Σ.*

We call $\mathscr{B}(X)$ of transverse, non-transverse, or boundary type if X is a translate of a set X, which has that type according to Table 9.1. These types are related to the intersection behaviour of cells for $\mathbf{SL}_4$ under the inclusion map (for an explanation see [MM89]). We call X a Name-set if X is a translate of name Name. We will say that all cell in one orbit have the type Name of the related standard element in Σ. If we have to distinguish more than one cell of the same type we use indices which usually start with 0 like Pyramid$_0$ or Pyramid$_{12}$. An explanation of the names of the cell types is given in [MM89; § 5.3]. For a proof of Theorem 9.1.3 we refer to [MM93]. The action of the group Γ_0 on the sets X is defined by multiplication from the left on the set of column vectors. this compatible with the action on cones defined earlier in (8.22).

Note: Not all of the 18 classes of non boundary type of the 21 $\mathbf{GL}_4(\mathbb{Z})$-classes occur in the list stated in Table 9.1, since some have empty intersection with the embedding. So there are indeed empty intersections. Further, there are two Γ_0-classes (Hexagon and Vertebra) which belong to the same class for $\mathbf{GL}_4(\mathbb{Z})$. So the $\mathbf{GL}_4(\mathbb{Z})$ class decomposes into two connected components which are not equivalent under the action of Γ_0. This is caused by the fact we will discuss later on that the Siegel upper half-space is not a linear symmetric space and therefore we have some strange behaviour under intersections (comp. page 56).

Table 9.1.: *List of standard elements from [MM93; pp.587-588]. Abbreviation for types: T=Transverse, N=Non-Transverse, B=Boundary.*

Rank	Name	Type	Set
6	Desargues	T	$\begin{bmatrix}1&0&0&0&1&1&1&0&1&1\\0&1&0&0&1&0&0&1&1&0\\0&0&1&0&0&1&0&0&1&1\\0&0&0&1&0&0&-1&1&0&-1\end{bmatrix}$
6	Reye	T	$\begin{bmatrix}1&0&0&0&1&1&0&0&1&1&1&0\\0&1&0&0&1&0&1&0&1&1&0&1\\0&0&1&0&0&-1&0&1&0&-1&-1&1\\0&0&0&1&0&0&-1&-1&-1&0&1&-1\end{bmatrix}$
5	DR	T	$\begin{bmatrix}1&0&0&0&1&1&1&0&1\\0&1&0&0&1&0&0&1&0\\0&0&1&0&0&1&0&0&1\\0&0&0&1&0&0&-1&1&-1\end{bmatrix}$
5	RR	T	$\begin{bmatrix}1&0&0&0&1&0&0&1&1\\0&1&0&0&1&1&0&1&0\\0&0&1&0&0&0&1&-1&-1\\0&0&0&1&0&-1&-1&0&1\end{bmatrix}$
4	Hexagon	N	$\begin{bmatrix}1&0&0&0&1&0\\0&1&0&0&0&1\\0&0&1&0&0&1\\0&0&0&1&1&0\end{bmatrix}$
4	Square	T	$\begin{bmatrix}1&0&0&0&1&1&0&1\\0&1&0&0&1&0&1&0\\0&0&1&0&0&0&0&1\\0&0&0&1&0&-1&1&-1\end{bmatrix}$
4	Triangle	T	$\begin{bmatrix}1&0&0&0&1&1&0&1\\0&1&0&0&1&0&1&0\\0&0&1&0&0&1&0&1\\0&0&0&1&0&0&1&-1\end{bmatrix}$
3	Vertebra	N	$\begin{bmatrix}1&0&0&0&1\\0&1&0&0&0\\0&0&1&0&0\\0&0&0&1&1\end{bmatrix}$
3	Crystal	N	$\begin{bmatrix}1&0&0&0&1&0\\0&1&0&0&0&1\\0&0&1&0&1&0\\0&0&0&1&0&1\end{bmatrix}$
3	Pyramid	T	$\begin{bmatrix}1&0&0&0&1&0&1\\0&1&0&0&1&1&0\\0&0&1&0&0&0&1\\0&0&0&1&0&1&-1\end{bmatrix}$
2	RedSquare	N	$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$
2	LagrangianTriple	B	$\begin{bmatrix}1&0&1\\0&1&1\\0&0&0\\0&0&0\end{bmatrix}$
1	LagrangianPair	B	$\begin{bmatrix}1&0\\0&1\\0&0\\0&0\end{bmatrix}$
0	Point	B	$\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$

By simple computations one can show that:

Proposition 9.1.4: *The Γ and Γ_0 orbits of Σ coincide.*

We set $\mathfrak{W} := \Gamma\Sigma$, then we get from (9.4) a locally finite decomposition (Theorem 8.4.4)

$$\mathfrak{S}_2 = \bigsqcup_{X\in\mathfrak{W}} \mathscr{B}(X). \qquad \textbf{(Voronoï decomposition)} \tag{9.5}$$

MacPherson and McConnell showed that:

- The $\mathscr{B}(X)$ are open cells for $X \in \mathfrak{W}$ with $\dim_{\mathbb{R}} \mathscr{B}(X) = \mathrm{rank}\,(X)$ [MM93; pp. 589-594, pp. 600-602].
- The (relative) closure $\overline{\mathscr{B}(X)} \subset \mathfrak{S}_2$ is contained in a union of Voronoï cells $\mathscr{B}(X')$ for some $X' \in \mathfrak{W}$. [MM93; pp. 594-599, pp. 602-603]

Since $\mathbb{P}\mathscr{R}(X)$ are rational polyhedra the $\mathscr{B}(X)$ are also polyhedra. For finite sets X the $\mathscr{B}(X)$ have only finitely many faces. Therefore, $\mathfrak{S}_2$ gets by this construction the structure of a CW-complex. Let $\Gamma_0 = \mathbf{GSp}_2(\mathbb{Z})$. It is not difficult to check that there is a group action of Γ_0 on the $\mathscr{B}(X)$, which is the right action on the cones $\mathscr{R}(X)$ given by (8.22) via the action of Γ_0 on the column vectors in X. Hence, the action is given by

$$R_\gamma\left(\mathscr{B}(X)\right) = \mathscr{B}(\gamma^{-1}\cdot X) \tag{9.6}$$

for a $\gamma \in \Gamma_0$. We find also by application of the definitions that this action is cellular on the decomposition (9.5), i.e. it acts by permutation of the cells. There is the same action on other kinds of cells, e.g. $\mathscr{C}(X)$ defined later. This action is not fixed point free, since Γ_0 has torsion and hence there are fixed point components inside the cells. We want to emphasize that this does not mean that if $\gamma\cdot\mathscr{B}(X) = \mathscr{B}(X)$ for some $X \in \mathfrak{W}$ that $\mathscr{B}(X)$ is fixed pointwise under $\gamma \in \Gamma_0$. But we know by Avner Ash [Ash80, Ash84] (see also [McC91; (2.5)]) that the stabilizer of each cell is finite. We denote $\mathscr{B}(X)$ modulo the action of Γ_0 by $\mathscr{B}_{\Gamma_0}(X)$. Later we use the analogous notation also for other kinds of cells, like $\mathscr{C}(X)$, and subgroups of Γ_0, in particular for the Siegel modular group $\Gamma \subset \Gamma_0$. It turns out in our special situation that the quotients of $\mathscr{B}(X)$ by Γ_0 and its subgroup Γ coincide, because on one side $-\mathrm{Id}$ acts trivially and on the other side we have the equivalence relations on the set of column vectors (comp. Prop. 9.1.4).

Let $\Gamma' \subset \Gamma$ be a neat arithmetic subgroup, then one gets by results of the Japanese mathematician Ichirō Satake [Sat60b, Sat60a] that the quotient

$$\mathfrak{S}_2/\Gamma' = \bigsqcup_{X\in\mathfrak{W}/\Gamma'} \mathscr{B}_{\Gamma'}(X) \tag{9.7}$$

has the structure of a regular CW-complex. Satake's work, which is used in [MM93, McC98], applies only for neat subgroups. Later for our purpose we have to take care of this. We summarize the results of MacPerson and McConnell on this decomposition:

Theorem 9.1.5 (MacPherson, McConnell 1993)**:** *Let Σ be the set of standard elements and $\mathfrak{W} = \Gamma\cdot\Sigma$ as above then*

$$\mathfrak{S}_2 = \bigsqcup_{X\in\mathfrak{W}} \mathscr{B}(X)$$

is a decomposition of $\mathfrak{S}_2$ into polyhedral cells on which the group Γ_0 acts cellularly. If $\Gamma' \subset \Gamma$ is a torsion free arithmetic group then there is a decomposition

$$\mathfrak{S}_2/\Gamma' = \bigsqcup_{X \in \mathfrak{W}/\Gamma'} \mathscr{B}_{\Gamma'}(X),$$

which is a finite regular cell complex.

We will prove in Section 9.2 that a very similar result is valid for the group Γ itself. For completeness we want to mention that the construction of MacPherson and McConnell extends easily to the so called partial **Satake compactification** $\mathfrak{S}_2^*$ of $\mathfrak{S}_2$ with respect to the standard representation of the symplectic group. For this one replaces $\mathfrak{W}$ by $\mathfrak{W}^* := \Gamma_0\Sigma^*$, where Σ^* is the set of all elements in Table 9.1 including those of boundary type. Then the partial Satake compactification $\mathfrak{S}_2^*$ with respect to the standard representation of $\mathbf{Sp}_2(\mathbb{R})$ can be obtained as the union of $\mathfrak{S}_2$ with its rational boundary components, i.e.

$$\mathfrak{S}_2^* = \mathfrak{S}_2 \cup R_1 \cdot \mathbf{Sp}_2(\mathbb{Q}) \cup R_2 \cdot \mathbf{Sp}_2(\mathbb{Q}),$$

where

$$R_1 = \left\{ \begin{pmatrix} A & 0 \\ 0 & 0 \end{pmatrix} \in \mathrm{Mat}_4(\mathbb{R}) : A \in R_2(\mathbb{R}), A = A^t, A > 0 \right\}$$
$$R_2 = \left\{ \begin{pmatrix} 1 & \\ & 0 \end{pmatrix} \in \mathrm{Mat}_4(\mathbb{R}) \right\}.$$

The R_i are called **standard rational boundary components** of $\mathfrak{S}_2$ and their translates $R_\gamma(R_i)$ under some $\gamma \in \mathbf{Sp}_2(\mathbb{Q})$ are called **rational boundary components** of $\mathfrak{S}_2$.

Note: The decomposition of $\mathfrak{S}_2$ in Theorem 9.1.5 is *not* locally finite near its rational boundary components [MM93; (6.1)]. But for retraction we will have a locally finite decomposition (see Theorem 9.2.11).

9.1.3. Construction of the Retract

Theorem 9.1.5 gives us a decomposition of $\mathfrak{S}_2$. This is not the decomposition we are looking for, since the dimension is still 6 and not 4 as we hope to find due to Theorem 12.3.4. In this section we now describe how to construct from this decomposition a lower dimensional one which has the right properties. MacPherson's and McConnell's approach uses a bunch of different tools to prove that their construction has the right properties. They use for example the so called Whitney stratification, so called warped/non-warped pairs and properties of tubular neighbourhoods which we will not explain since they are rather technical and not needed for the understanding of the results. For more details we refer to the original paper and the references there in [MM93; (6)-(7)].

The main problem of the authors is the fact that for $g \geq 2$ the space $\mathfrak{S}_g$ is not a linear symmetric space. But for example the space $\mathbb{X}_{\mathbf{SL}_d}$ is a linear symmetric space. We will explain what this means in a minute. As a consequence one has the problem that some known concepts which apply (only) for linear symmetric spaces are not applicable in our case.

Definition 9.1.6: *A symmetric space $\mathbb{X}_{\mathscr{G}}$ is called a* **linear symmetric space** *if there are an embedding $i : \mathbb{X}_{\mathscr{G}} \hookrightarrow \mathbb{R}^d$ of the symmetric space $\mathbb{X}_{\mathscr{G}}$ and a faithful representation $\rho : \mathscr{G}(\mathbb{R}) \to \mathbf{GL}_d(\mathbb{R})$ such that the map i becomes equivariant.*

This means in particular that the induced action on the embedded image has to map rays to rays and also to preserve the closed cone $\overline{\mathscr{Q}^d_{>0}}$. This kind of structure is strongly related to the theory of so called self adjoint homogenous cones [AMRT75, Ash77]. It might be important to mention that in a linear symmetric space one is able to construct for any set of points a convex hull which is equivariant. It is clear that this is extremely useful if we want to construct an equivariant cell decomposition. If the space is not linear the boundaries of the closures could be non-simply-connected homology spheres. There are known retractions for all (irreducible) linear symmetric spaces [AMRT75, Ash77, Sou78, Ash84]. A bit less than half of all negatively curved symmetric spaces are linear (see [MM93; p. 579]).

Unfortunately $\mathfrak{S}_2$ is not a linear symmetric space which is [MM93; (0.3)]. As a consequence, the embedding $\mathfrak{S}_2 \hookrightarrow \mathscr{Q}$ is preserved by the action of Γ (Lemma 6.3.5), but the intersection of the cone $\mathscr{R}(X)$ with the embedded image is not in all cases a full cone. The intersection can have three different types (transversal, non-transversal and boundary), which behave completely differently. If $\mathbb{P}\mathscr{R}(X)$ for some set X intersects (the embedded image of) $\mathfrak{S}_2$ transversely, then it is clear that $\mathscr{B}(X) = \mathbb{P}\mathscr{R}(X) \cap \mathfrak{S}_2$ is a cell. In the other cases one has to work more [MM93]. This small difference causes also much more technical problems if we want to construct a Γ-equivariant retraction of $\mathfrak{S}_2$, which we describe now.

Avner Ash described in [Ash77] a method to obtain an equivariant retraction with respect to some arithmetic group if one has a decomposition into polyhedra. This construction was then used by MacPherson and McConnell in [MM93; (8)] to derive a Γ'-equivariant retraction for a torsion free or neat arithmetic subgroup $\Gamma' \subset \Gamma$ (Theorem 9.1.10). To get a retraction by directly applying [Ash77] it is enough to demand from the group to be torsion free. Neatness is only important to extend the results also to the boundaries. We sketch here only the method of Ash and present later some more details of the construction in our case, i.e. if torsion is allowed (see Section 9.2).

Let $\Gamma' \subset \Gamma$ be a neat arithmetic subgroup. To get the demanded retraction we start with the extended version of the decomposition from Theorem 9.1.5 to the compactified quotient, i.e. we replace Σ by Σ^* and $\mathfrak{S}_2$ by $\mathfrak{S}_2^*$ in Theorem 9.1.5. It will turn out that for us we can even start directly with the decomposition from Theorem 9.1.5 since the additional parts do not occur in the new decomposition which we want to construct. This decomposition is a Γ'-admissible decomposition (see [Ash77; p. 72]). We can now follow step by step the construction in [Ash77; §2].

Thus, we take first the first barycentric subdivision $B(\mathfrak{S}_2^*/\Gamma')$ of the decomposition of $\mathfrak{S}_2^*/\Gamma'$. We know by [MM93; (8.4)] that $B(\mathfrak{S}_2^*/\Gamma')$ is the order complex of the poset $(\mathfrak{W}/\Gamma', \lesssim)$ where $\lesssim$ is the partial ordering relation defined in Corollary 9.1.2. The vertices, i.e. the zero dimensional cells, in $B(\mathfrak{S}_2^*/\Gamma')$ are in one-to-one correspondence to elements in $\mathfrak{W}/\Gamma'$ (see [Ash77] or [MM93; (8.5)]), therefore we can associate to each vertex in $B(\mathfrak{S}_2^*/\Gamma')$ an element in $\mathfrak{W}/\Gamma'$. We introduced in (8.4) an inner product on the cone of positive definite quadratic forms.

Table 9.2.: *Dimension of the cells $\mathscr{C}(X)$ for $X \in \Sigma$.*

Cell name	Desargues	Reye	DR	RR	Triangle	Square	Hexagon	Vertebra	Crystal	Pyramid	RedSquare
$\dim(\mathscr{C}(X))$	0	0	1	1	2	2	2	3	3	3	4

Using this inner product we can introduce a dual cell structure on $\mathfrak{S}_2^*$. Let

$$\mathscr{B}^\vee(X) := \left\{Q \mapsto \langle Q', Q\rangle_{\mathscr{Q}} : Q' \in \mathscr{B}(X)\right\} \subset \left(\mathscr{Q}^d_{>0}\right)^\vee \tag{9.8}$$

be the dual cell to $\mathscr{B}(X)$ for some $X \in \mathfrak{W}$. According to Voronoï we can identify $(\mathscr{Q}^d_{>0})^\vee$ again with $\mathscr{Q}^d_{>0}$. Therefore, we can view the $\mathscr{B}^\vee(X)$ as subset in $\mathscr{Q}^d_{>0}$. Denote by

$$\mathscr{C}(X) := \mathscr{B}^\vee(X) \tag{9.9}$$

the dual cell to $\mathscr{B}(X)$. We define analogously $\mathscr{C}_{\Gamma'}(X) := \mathscr{B}_{\Gamma'} \vee (X)$ for some $X \in \Gamma' \backslash \mathfrak{W}$. By taking the dual of a cell the dimension is reversed, i.e. the dimension of the dual cell is the difference between the dimension of the whole space and the dimension of the cell. Therefore, we have by Theorem 9.1.3

$$\begin{aligned}\dim_{\mathbb{R}} \mathscr{C}(X) = \dim_{\mathbb{R}} \mathscr{C}_{\Gamma'}(X) &= \dim_{\mathbb{R}} \mathfrak{S}_2 - \dim_{\mathbb{R}} \mathscr{B}(X) \\ &= 6 - \operatorname{rank}(X).\end{aligned} \tag{9.10}$$

Definition 9.1.7: *We denote by $\mathbb{W}_{\Gamma'}$ the full subcomplex of the cell complex $B(\mathfrak{S}_2^*/\Gamma')$ spanned by those vertices $\sigma \in B(\mathfrak{S}_2^*/\Gamma')$ for which $\sigma = \mathscr{C}_{\Gamma'}(X)$ for some $X \in \mathfrak{W}/\Gamma'$.*

Note: $\mathbb{W}_{\Gamma'}$ is the order complex of $(\mathfrak{W}/\Gamma', \lesssim)$. Thus, it makes here no difference whether we consider the barycentric subdivision for $B(\mathfrak{S}_2^*/\Gamma')$ or for $B(\mathfrak{S}_2/\Gamma')$ since we take only the elements corresponding to the elements in $\mathfrak{W}/\Gamma'$.

By definition $\mathbb{W}_{\Gamma'} \subset \mathfrak{S}_2/\Gamma'$. Therefore, we can define

Definition 9.1.8 (Retract)**:** *Let $\mathbb{W} = \pi^{-1}(\mathbb{W}_{\Gamma'})$ for $\pi : \mathfrak{S}_2 \to \mathfrak{S}_2/\Gamma'$. We denote the restriction of the projection map π to $\mathbb{W}$ by $\widetilde{\pi}$.*

Note: This definition is – as mentioned in [MM93; p. 620] – independent of the chosen group Γ'. For its construction we will later give a proof which is not given in [MM93] (see Lemma 9.2.1).

By [MM93; (8.7)] we have:

Lemma 9.1.9 (Retract)**:** *The cell complex $\mathfrak{S}_2/\Gamma'$ is a strong deformation retract of $\mathfrak{S}_2/\Gamma'$.*

At the end we want to mention that this result extends easily to $\mathfrak{S}_2^*/\Gamma'$ in the following way. Let $\mathbb{W}^*_{\Gamma'}$ be the full subcomplex spanned by the vertices in $B(\mathfrak{S}_2^*/\Gamma')$ corresponding to dual cells defined by elements in $\mathfrak{W}^*/\Gamma'$. Then we can define $\mathbb{W}^*$ to be its inverse image under the projection π. This $\mathbb{W}^*$ defines then a deformation retract of the partial Satake compactification with respect to the standard representation. This leads to the main theorem of MacPherson and McConnell [MM93; Theorem 8.8]:

Theorem 9.1.10 (MacPherson, McConnell 1993)**:** *Let $\Gamma' \subset \Gamma_0$ be a neat (or torsion free) subgroup. Then there is a Γ'-equivariant strong deformation retraction $\mathbb{W}$ of $\mathfrak{S}_2$ which has the structure of a locally finite Γ'-equivariant regular cell complex with only finitely many Γ'-orbits of cells. The face poset of $\mathbb{W}$ is $\mathscr{C}(\mathfrak{W})$, partially ordered by inverse inclusion, and Γ' acts compatibly on $\mathbb{W}$ and $\mathfrak{W}$.*

Note: It is essential that the retract now has the structure of a locally *finite* cell complex. This is not true for the cell complex before the retraction.

The last part of [MM93] is dedicated to a much more explicit construction for the case that the group which is considered is a principal congruence subgroup $\Gamma(p)$ of odd prime level $p \equiv 3 \bmod 4$. This construction uses McConnell's work on configuration in projective geometry in connection to cone decomposition [McC91] and is also applicable for some groups different from the symplectic group. In particular, it works for $\mathbf{GL}_d$ and $\mathbf{SL}_d$ [MM89]. A more detailed analysis for the spaces related to these groups and their Satake compactifications is contained in [AM97, McC99].

Remark (Why not use projective configurations with a covering?)**:**
It is known that we could replace the study of a quotient by a group with torsion by the study of a branched covering, which is a quotient by a torsion free group. So one might ask the question why not use these projective configurations, then take a covering and simply compute the cohomology from that. This is of course possible, but the problem in practice is that the covering group, which in our case for $\Gamma(p)$ is $\mathbf{Sp}_2(\mathbb{F}_p)$, is quite large. For example, the smallest case where $\Gamma(p)$ is torsion free (and also neat) with $p \equiv 3 \bmod 4$ is $\Gamma(3)$. Here we have a branched 51840-fold covering of our space. This leads in the further computations to linear systems, which have to be solved, which are 51840 times as large as the original ones. In the best case the runtime to solve linear systems grows quadratically in the dimension n. For a generic dense case one has, using Strassen's trick[1], a complexity of $\mathcal{O}(n^{\log_2(7)})$, which is asymptotically faster than the classical gaussian elimination, which needs $\mathcal{O}(n^3)$, because $\log_2(7) = 2.80735... < 3$ (see [vG03; p. 327ff.]). Therefore, we have to assume in the case above that the computation for the larger system needs roughly between 2.69×10^9 (sparse case) and 1.72×10^{13} (dense case) times as long as for the original system. After this estimate we thought that this approach is probably not the best solution for our problem.

The analogue of Ash's construction [Ash77; §3] shows that $\mathbb{W}$ can be interpreted as the Poincaré dual of the decomposition of $\mathfrak{S}_2$. The sets $\mathscr{C}(X)$ form an analytic Whitney stratification of $\mathfrak{S}_2$ [MM93; (0.2)] and we can view $\mathbb{W}$ as a subcomplex of the original decomposition of $\mathfrak{S}_2$.

[1]Volker Strassen proposed in 1969 [Str69] a faster alternative to the classical gaussian elimination which uses an iterated splitting of the matrix into blocks of half size. If the dimension is $n = 2^k$ one can show that one needs at most $6 \times n^{\log_2(7)}$ and if n is arbitrary one needs at most $42 \times n^{\log_2(7)}$. The fastest known methods today are variations of Strassen's approach. The best known method for dense systems has a complexity of $\mathcal{O}(n^{2.376})$ [vG03; p. 329].

9.2. Extension to the Torsion Case

Now we extend the results of MacPherson and McConnell from Theorem 9.1.10, which are only valid for neat subgroups of Γ_0, to arbitrary arithmetic subgroups of Γ of finite index. This result is stated in Theorem 9.2.11. There are some differences to the result in Theorem 9.1.10. First, the resulting decomposition in the quotient is not a regular CW-complex, because not all objects in the decomposition are cells (they are only orbicells). Furthermore, the result seems not to extend (in general) to a suitable description of compactifications of the quotient. This is not really surprising because even in the absence of torsion we have to handle fancy identifications in the boundaries.

The first step in this extension was the above mentioned remark that $\mathbb{W}$ does not depend on the choice of Γ'. This is easy to see. According to Theorem 9.1.5 we have a set Σ such that $\mathfrak{W} = \Gamma \cdot \Sigma$. This is obviously independent of Γ'. Hence, we get:

Lemma 9.2.1 (Independence): *The construction of the space $\mathbb{W}$ is independent of the chosen Γ'. More precisely, if Σ is the set of standard elements from Theorem 9.1.3 and $\mathscr{C}(X)$ the cell dual to $\mathscr{B}(X)$ for $X \in \mathfrak{W} = \Gamma \cdot \Sigma$, then*

$$\mathbb{W} = \bigsqcup_{X \in \mathfrak{W}} \mathscr{C}(X).$$

A remark on the notation: we will very often identify $\mathbb{W}$ with the set of cells which are defining $\mathbb{W}$, hence we use terms like $\sigma \in \mathbb{W}$.

In some way this lemma suggests that the complete construction from [MM93] should be valid more or less for arbitrary subgroups of Γ and therefore also the group Γ itself. This leads to the question whether Theorem 9.1.10 stays true if there is torsion and how the constructions differ between the case where the group has torsion elements and the case where it has not. While reading [MM93] I got the strong feeling that torsion in some way only matters if we consider also the behaviour on the boundaries or compactifications. Since we are only interested in the cohomology of the space itself and not its compactifications, this should not be a serious problem. My idea was that the only thing one should change in the presence of torsion elements is that then the image of cells in the quotient are no longer copies of cells in $\mathfrak{S}_2$, but they are cells on $\mathfrak{S}_2$ modulo the action of the stabilizer, i.e. orbicells (see Section 5.3). However, the space $\mathbb{W}$ seems to be independent of the group used for its construction in Definition 9.1.8.

Now we have to verify step by step some properties of $\mathbb{W}$ with respect to the group Γ which were proved in [MM93] in case of neat subgroups of Γ_0. Usually one can replace Γ by a subgroup Γ' in the following statements and they stay valid.

Proposition 9.2.2 (Cellularity): *The group Γ acts cellularly on the space $\mathbb{W}$ by the action induced by the action of Γ on $\mathfrak{W}$.*

Proof. Let $\sigma \in \mathbb{W}$ be a cell, then there is an element $X \in \mathfrak{W}$ such that $\sigma = \mathscr{C}(X)$. By Theorem 9.1.5 we have $\mathfrak{W} = \Gamma\Sigma$. Therefore, there is an element $\gamma_{X,S} \in \Gamma$ such that $\gamma_{X,S}^{-1} \cdot X = S$ is a standard element.[2] Both stabilizers are conjugated.). By (9.6) the group Γ acts on the cells

[2]The element $\gamma_{X,S}$ is not unique. One can replace $\gamma_{X,S}$ by a translate under an element of the stabilizers of the sets X and S.

by $R_\gamma\left(\mathscr{C}(X)\right) = \mathscr{C}(\gamma^{-1}\cdot X)$. Hence, we get that

$$\begin{aligned} R_\gamma(\sigma) &= R_\gamma\left(\mathscr{C}(X)\right) = \mathscr{C}(\gamma^{-1}\cdot X) \\ &= \mathscr{C}(\underbrace{\gamma^{-1}\cdot\gamma_{X,S}}_{=:\gamma'}\cdot S) = \mathscr{C}(\underbrace{\gamma'\cdot S}_{=:X'}) \\ &= \mathscr{C}(X'), \end{aligned}$$

where $X' \in \mathfrak{W}$ by construction. So the group Γ acts by permuting the cells. □

The next proposition is a complete triviality.

Proposition 9.2.3 (Orbits)**:** *The action of the group Γ on $\mathfrak{W}$, and therefore on $\mathbb{W}$, has only finitely many orbits.*

Proof. This follows immediately from the fact that $\mathfrak{W} = \Gamma\cdot\Sigma$ (see Theorem 9.1.3) and Σ consists of exactly 11 elements (see Table 9.1). In other words, $\mathbb{W}$ decomposes into 11 orbits under Γ generated by the 11 sets in Σ. □

If $\Gamma' \subset \Gamma$ is a subgroup of finite index Proposition 9.2.3 is still valid, since if Γ decomposes $\mathbb{W}$ into m orbits and $[\Gamma : \Gamma'] = d$ then there are at most $d\cdot m$ different orbits. There might be some orbits which are stable under the subgroup while others may decompose. Hence

Corollary 9.2.4: *Let $\Gamma' \subset \Gamma$ be a subgroup of finite index. Then Γ' has only finitely many orbits on $\mathfrak{W}$ and therefore on $\mathbb{W}$.*

By the original proof in [MM93] it is clear that for any $X \in \mathfrak{W}$ the set $\mathscr{C}(X)$ is a cell. Since there the groups are neat the cells $\mathscr{C}(X)$ have a trivial stabilizer, i.e. plus and minus identity are the only possible elements in the isotropy group of the cell. Therefore, also the associated set $\mathscr{C}_\Gamma(X)$ in the quotient is a cell. If there are torsion elements this is no longer true, because the non-trivial stabilizer causes additional problems.

Lemma 9.2.5: *Let $\sigma \in \mathbb{W}$ be a cell, then σ/Γ is an orbicell in $\mathbb{W}/\Gamma$ and $\sigma/\Gamma = \sigma/\Gamma_\sigma =: \sigma_\Gamma$, where*

$$\Gamma_\sigma = \left\{\gamma \in \Gamma : R_\gamma(\sigma) = \sigma\right\}$$

is the stabilizer of the cell σ. We use, as before, $\mathscr{C}_\Gamma(X) = \mathscr{C}(X)/\Gamma_X$ where $\Gamma_X = \Gamma_{\mathscr{C}(X)}$ with the previous notation.

Note that the stabilizer does not necessarily stabilize the cell pointwise.

Proof. By Proposition 9.2.2 the action of Γ is cellular. Hence, $\gamma \in \Gamma$ either maps a cell in the decomposition to another cell or to itself. Therefore, if $\gamma \notin \Gamma_\sigma$ it does not act on σ and we have $\sigma/\Gamma = \sigma/\Gamma_\sigma$. We know that the stabilizer Γ_σ is discrete since Γ is discrete. Γ_σ is compact because Γ_σ is the intersection of a group which is conjugate to the maximal compact group $\mathbf{U}(2)$ with Γ. Therefore, Γ_σ is finite. Since σ is homeomorphic to an open ball and Γ_σ is a finite group of isometries, the quotient is an orbicell. □

We know that a cell $\mathscr{B}(X')$ lies in the closure $\overline{\mathscr{B}(X)} \subset \mathfrak{S}_2$ of $\mathscr{B}(X)$ if and only if $X \subset X'$ (comp. Corollary 9.1.2). Going from cells to dual cells reverses the inclusion compared to the order relation for the $\mathscr{B}(X)$ (or $\mathscr{R}(X)$).

Corollary 9.2.6: *A cell $\mathscr{C}(X')$ lies in the closure $\overline{\mathscr{C}(X)} \subset \mathbb{W}$ of $\mathscr{C}(X)$ if and only if $X \supset X'$. This defines a partial ordering relation on $\mathbb{W}$.*

This is the most important tool for our algorithms to determine closures and boundaries of cells in Chapter 10. Before we proceed with the proof that our decomposition of the quotient has the structure of a finite orbicell complex, we need to have a better description of the closure of cells in $\mathbb{W}$ and its quotients.

Proposition 9.2.7 (Closure I)**:** *The closure of a cell $\sigma \in \mathbb{W}$ in $\mathbb{W}$ is a finite union of cells $\sigma_i \in \mathbb{W}$. The closure of each cell is homeomorphic to a closed ball., i.e. $\mathbb{W}$ has the structure of a locally finite regular cell complex. If $\sigma = \mathscr{C}(X)$, we have*

$$\overline{\sigma} = \bigsqcup_{\substack{X' \supset X \\ \exists \gamma \in \Gamma : \gamma X' \in \Sigma}} \mathscr{C}(X')$$

Proof. By construction $\mathbb{W}$ is a cell complex. By [MM93; Proposition 8.3] we have that the closure of a cell is a closed cell, i.e. homeomorphic to a closed ball. Therefore, we have a regular cell complex (comp. e.g. Theorem 9.1.10). This cell complex is locally finite because by Theorem 8.4.4 of Voronoï the polyhedral cone we are starting with is locally finite. □

We now want to descend to the quotient.

Corollary 9.2.8 (Closure II)**:** *Let $\Gamma' \subset \Gamma$ be a subgroup of finite index. The closure of an orbicell $\sigma_{\Gamma'} \in \mathbb{W}/\Gamma'$ in $\mathbb{W}/\Gamma'$ is a finite union of orbicells $\sigma_{i,\Gamma'} \in \mathbb{W}/\Gamma'$ which are images of cells $\sigma_i \in \mathbb{W}$ under the projection map $\widetilde{\pi}$.*

Proof. Let $\widetilde{\sigma_{\Gamma'}}$ be a lift of the orbicell $\sigma_{\Gamma'}$ to $\mathbb{W}$. By Proposition 9.2.7 the closure $\overline{\widetilde{\sigma_{\Gamma'}}}$ of $\widetilde{\sigma_{\Gamma'}}$ in $\mathbb{W}$ is a union of cells σ_i in $\mathbb{W}$. By Proposition 9.2.3 and Corollary 9.2.4 there are only finitely many Γ'-orbits, hence the cells $\sigma_i \in \mathbb{W}$ descend to only finitely many orbicells $\sigma_{i,\Gamma'}$ in $\mathbb{W}/\Gamma'$ (comp. Lemma 9.2.5). □

Remark: Unlike the cells in $\mathbb{W}$ (Proposition 9.2.7) or the cells in the quotient in case of neat subgroups in the original work of MacPherson and McConnell [MM93] we cannot ensure that in the general case the closure is homeomorphic to a closed ball. If there is torsion then the boundary will be identified in a non-trivial way in the quotient. Hence, we have in general not a regular cell structure in the quotient.

Lemma 9.2.9 (Finite cell complex)**:** *Let $\Gamma' \subset \Gamma$ be a subgroup of finite index. Then we have that*

$$\mathbb{W}/\Gamma' = \bigsqcup_{X \in \mathfrak{W}/\Gamma'} \mathscr{C}_{\Gamma'}(X)$$

has the structure of a finite orbicell complex.

Proof. This follows immediately from the fact that there are only finitely many Γ'-orbits (Corollary 9.2.4) and Proposition 9.2.7. □

The only thing which now is still missing is that this is indeed a deformation retract for our quotient.

Proposition 9.2.10 (Retract): *$\mathbb{W}$ is a Γ-equivariant strong deformation retract of $\mathfrak{S}_2$. If $\Gamma' \subset \Gamma$ is a subgroup of finite index then $\mathbb{W}/\Gamma'$ is a strong deformation retract of $\mathfrak{S}_2/\Gamma'$.*

Proof. By [MM93; Prop. 8.7] we have that $\mathbb{W}$ is a strong deformation retract of $\mathfrak{S}_2$. According to Theorem 9.1.10 $\mathbb{W}$ is Γ'-equivariant for any neat $\Gamma' \subset \Gamma$. Furthermore, we know by Proposition 9.2.7 that $\mathbb{W}$ is a locally finite cell complex on which by Proposition 9.2.2 the group Γ acts cellularly. Hence, we can make $\mathbb{W}$ into a Γ-equivariant deformation retract, because by Lemma 9.2.1 $\mathbb{W}$ does not depend on the chosen neat $\Gamma' \subset \Gamma$ which was used during its construction. The proof of the analogous statement in the neat case can be found in [MM93; §8], where it is split into several partial results. This proves the first part.

Further, each cell has only finitely many faces (see Proposition 9.2.7), therefore $\mathbb{W}$ is a Γ-admissible decomposition in the sense of [Ash77; p. 72]. Now we can consider $\mathbb{W}/\Gamma$ as a finite subcomplex of the finite cell complex $\mathfrak{S}_2/\Gamma$ by the embedding of cells in in the construction of $\mathbb{W}$ (comp. Section 9.1.3). We are now able to define exactly the retraction from $\mathfrak{S}_2/\Gamma$ onto $\mathbb{W}/\Gamma$ in the same way as described in [Ash77; p. 75f.] (which is the same as in [MM93; Lemma 8.7]), but we have to replace the (relatively open) polyhedral cones there by the cones modulo their stabilizers, which are again polyhedral cones, but not necessarily relatively open. With the previous results it is easy to check that this defines now indeed also a strong deformation retraction to $\mathbb{W}/\Gamma$. We refer to [Ash77; p. 74f.] for all details, which are easy to adapt. □

An other method than to use [Ash77; §3] to prove the Proposition 9.2.10 is to use the result of MacPherson and McConnell that there is a retraction for some neat subgroup $\Gamma' \subset \Gamma$ from [MM93] and to consider the (branched) covering of $\mathbb{W}/\Gamma$ by $\mathbb{W}/\Gamma'$. But this is essentially the same since the lift of the retraction map involves to divide out the non-trivial stabilizers at some point.

We can now collect the results from above.

Theorem 9.2.11 (Main Theorem): *Let $\Gamma' \subset \Gamma$ be an arithmetic subgroup of finite index. Then there is a Γ'-equivariant strong deformation retraction $\mathbb{W}$ of $\mathfrak{S}_2$ which has the structure of a regular locally finite Γ'-equivariant regular cell complex. The cell complex $\mathbb{W}$ is independent of Γ' and decomposes into finitely many Γ'-orbits of cells. Let $\mathfrak{W} = \Gamma \cdot \Sigma$, where Σ is the set of standard elements from Theorem 9.1.3, then the face poset of $\mathbb{W}$ is $\mathscr{C}(\mathfrak{W})$, partially ordered by inverse inclusion, and Γ' acts compatibly on $\mathbb{W}$ and $\mathfrak{W}$. We have that*

$$\mathbb{W} = \bigsqcup_{X \in \mathfrak{W}} \mathscr{C}(X).$$

Further, there is a strong deformation retract of $\mathfrak{S}_2/\Gamma'$ which is equal to $\mathbb{W}/\Gamma'$ and hence we have an orbicell decomposition

$$\mathbb{W}/\Gamma' = \bigsqcup_{X \in \mathfrak{W}/\Gamma'} \mathscr{C}_{\Gamma'}(X).$$

So we got an extension of the original result from [MM93; Theorem 8.8] (see Theorem 9.1.10) which is valid for arbitrary arithmetic subgroups and therefore in particular for the Siegel modular group Γ itself. The only malus is that it is not possible to get a cell decomposition in the classical sense, but only an orbicell decompostion.

This result is in some way consistent to some phenomena in this area. For example, the result on the virtual cohomological dimension of [BS73] (see Theorem 12.3.4) was first only considered for torsion free groups and then extended to the general case by showing that it does not depend on the choice of the subgroup.

We use the results presented in this section in Chapter 10 to compute complete lists of (orbi-)cells in $\mathbb{W}$ (or $\mathbb{W}/\Gamma$) which are consistent to some known local consideration on cells in $\mathbb{W}$ done with this method by the authors in [MM89] and some notes from Paul Gunnells on the work of Robert MacPherson on arithmetic groups [Gun06], where he states the local results on the closure of one 4-cell in $\mathbb{W}$ from [MM89]. These known results are not enough to compute later the cohomology (see Section 13.3).

10. Computing a Cellular Decomposition

Variatio delectat.

(Marcus Tullius Cicero)

10.1. Introduction

In this chapter we describe the methods we used to compute lists of cells in $\mathbb{W}$, their stabilizers (and thus also orbicells in $\mathbb{W}/\Gamma$), boundaries, or neighbours. All these computations are more or less purely combinatorial and use – in the sense of the famous sentence of the Roman philosopher and statesman Marcus Tullius Cicero[1] – most of the time different variants of the same basic idea (see Section 10.4.1). One can summarize the main tools for this task as follows

a) Theorem 9.2.11 together with the definition of the set Σ of standard elements.

b) The order relation on the sets $\mathfrak{W}$ and $\mathbb{W}$ from Corollary 9.2.6.

c) Some basic linear algebra.

The method proposed by us might be not the fastest possible method, but it has the big advantage to be quite simple and that it is obvious that it returns the right results. A sequence of tests showed that our method is fast enough to give us the results we are looking for in a few minutes. So it was not necessary to speed-up things at this point.

The computations for this part are more or less completely done in SAGE. Only the part which allows us to identify the abstract group structure of stabilizers uses explicitly the GAP `SmallGroups` Library [BEO02].[2] In particular, in this part we used the possibility of SAGE to create objects which carry not only the information of the defining set $X \in \mathfrak{W}$, from which all other necessary information could be deduced, but also some additional information, like the rank, which make the handling easier and faster since they do not have to be computed several times. We created a class to describe cells in $\mathbb{W}$ together with their properties. A list of available commands in Chapter A in the Appendix. We cannot state the full source code of our program. Therefore, we put the parts of the SAGE source code, which are useful to understand the program in Appendix B. The full source code is available either on the CD-ROM, which is included in the printed copy of this thesis, or direct from the author. In this chapter we only present the ideas of the used algorithms and explain at which points there were some technical problems and how we handled them. Furthermore, we will give some example outputs of our program. Additional results can be found on the CD-ROM, which is include in the printed copy of this thesis.

[1]"Variatio delectat" from Marci Tullii Ciceronis "De natura deorum" (Liber I, 9, 22). Also known in a variant (sic!) from the poet Phaedrus ("Fabulae" Liber II, prologus 10)

[2]SAGE uses some parts of GAP inside its own algorithms as subroutines for some tasks for matrix groups.

It is necessary to discuss some technical questions like how to represent a cell (see Section 10.3) and the choice of lifts (see Section 10.2) before we are going in medias res because it is essential for all algorithms to represent the objects in a unique way such that one can compare them fast.

10.2. The Question of Lifts

Let $\widetilde{\pi} : \mathbb{W} \to \mathbb{W}/\Gamma$ be the projection map to the quotient of the retract (comp. Sections 6.2 and 12.1). By the choice of the set of standard elements Σ in Theorem 9.1.3 and Table 9.1 we implicitly made a choice for a lift under $\widetilde{\pi}$ of each orbicell in the decomposition of $\mathbb{W}/\Gamma$. Since each cell in $\mathbb{W}$ lies in the orbit of exactly one standard element we find

$$\mathbb{W}/\Gamma = \bigsqcup_{\sigma \in \Sigma} \widetilde{\pi}(\sigma) \tag{10.1}$$

If $\sigma \in \mathbb{W}/\Gamma$ is an orbicell then

$$\widetilde{\sigma} := \widetilde{\pi}^{-1}(\sigma) \cap \{\mathscr{C}(X) : X \in \Sigma\} \tag{10.2}$$

is this chosen lift for the orbicell σ. We know by Lemma 9.2.5 that then $\sigma = \widetilde{\sigma}/\Gamma_{\widetilde{\sigma}}$ where $\Gamma_{\widetilde{\sigma}}$ is the stabilizer of the lifted cell $\widetilde{\sigma}$ in $\mathbb{W}$. Remember that the stabilizer need not to stabilize the cell pointwise. Thus, we have a one-to-one correspondence between elements in Σ and orbicells in $\mathbb{W}/\Gamma$. Assume now that we replace a chosen lift by a translate, say $\widetilde{\sigma}' = \gamma \cdot \widetilde{\sigma}$ for some $\gamma \in \Gamma$, then we have $\gamma^{-1} \cdot \Gamma_{\widetilde{\sigma}'} \cdot \gamma = \Gamma_{\widetilde{\sigma}}$ and thus

$$\widetilde{\pi}(\widetilde{\sigma}') = \widetilde{\pi}(\widetilde{\sigma}) = \widetilde{\sigma}/\Gamma_{\widetilde{\sigma}} = \widetilde{\sigma}'/\Gamma_{\widetilde{\sigma}'} = \sigma. \tag{10.3}$$

From now on we fix for each orbicell in $\mathbb{W}/\Gamma$ a lift by (10.2). It is a more complicated question how to fix a preimage for a single point $x \in \mathbb{W}/\Gamma$ or some arbitrary subset of $\mathbb{W}/\Gamma$. This is not important for the computation of cells or their stabilizers, etc., but we discuss this shortly to make clear what kind of problems can occur in this context in general. The basic problem is that even if we fix a lift for each cell, we might, depending on the stabilizer of x, have more than one lift of x in the chosen lift of the cell in which x lies. At most there are finitely many such lifts. Let us now have a closer look at this. We know that for each point $x \in \mathbb{W}/\Gamma$ there is a unique orbicell σ_x such that $x \in \sigma_x$ and $\sigma_x = \mathscr{C}_\Gamma(X)$ for one $X \in \Sigma$. Now choose an arbitrary point $\widetilde{x_0} \in \widetilde{\sigma_x} \cap \widetilde{\pi}^{-1}(x)$, i.e. $\widetilde{x_0}$ is one of the lifts of x in the cell $\widetilde{\sigma_x}$. Then we find

$$\left|\widetilde{\pi}^{-1}(x) \cap \widetilde{\sigma_x}\right| = [\Gamma_{\widetilde{\sigma_x}} : \Gamma_{\widetilde{x_0}}] \tag{10.4}$$

different preimages of x which lie inside the uniquely determined cell $\widetilde{\sigma_x}$. It follows by cellurarity of the action that the stabilizer of a point in a cell has to be contained in the stabilizer of the whole cell. The number of preimages in the cell does not depend on the choice of $\widetilde{x_0}$ because all possible finitely many different $\Gamma_{\widetilde{x_0}}$ are conjugated subgroups of $\Gamma_{\widetilde{\sigma_x}}$ and therefore have the same index. If the stabilizer of a point inside the cell equals the stabilizer of the cell there is exactly one preimage. In all other cases we have more than one possible points in the unique cell $\widetilde{\sigma_x}$.

There is no canonical choice to distinguish a single out of these points. Thus, one has to choose one out of these arbitrarily if there are no further restrictions to the choice. Further restrictions we can consider are for example if we have more than one point, say a small connected open neighbourhood U of x, then it is useful to choose the preimages of x simultaneously together with all points in U such that also the preimage is connected. But take care, even if $U \subset \mathbb{W}/\Gamma$ is simply connected there is no reason that any connected preimage of U in $\mathbb{W}$ has to be simply connected. In particular, it can (but need not to) have several holes, if U contains more than one orbicell. Therefore, we will make the choice of preimages of sets which are neither orbicells nor unions of orbicells if necessary depending on the context. In all other cases we take the choice from above.

These chosen preimages have an interesting property. Using Corollary 9.2.6 together with Table 9.1 shows that all cells $\mathscr{C}(X)$ for $X \in \Sigma$ lie in the boundary of $\mathscr{C}(\text{RedSquare})$, because each defining set $X \in \Sigma$ contains the standard basis $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4\}$. This is due to the fact that a collection can only define a cell in the inner if the collection generates $\mathbb{Z}^4$. This follows easily from Theorem 8.3.2 and Proposition 8.3.3 together with the definition of the Voronoï I decomposition (Definition 8.4.3) and is a quite useful property for the following computations.

In Section 13.3 we will find that it is not enough to choose only one preimage per type of cells. According to Section 13.3, we need to choose a representative per orbit of 4-cells which lie around each chosen lift of a k-cell for $k \leq 3$ under the stabilizer of that particular k-cell.

In later computations (see Section 10.7.3.1) we will see that in case of the three different types of 3-cells there is for each cell of type Crystal and Vertebra only one orbit of 4-cells around them under the stabilizer of the central 3-cell, whereas the neighbours of dimension 4 of cells of type Pyramid decompose into three orbits under the corresponding stabilizer, two of length one and one of length two. Therefore, we need two additional representatives of cells of type RedSquare beside the standard 4-cell RedSquare_0 represented by the identity matrix. In further steps this has to be iterated with 3-cells around 2-cells and so on. It is obvious that the set of standard cells is not large enough to do this, since therein is only one representative per type. We find analogous situation for cells of other dimensions. For an efficient computation is is useful to reduce the number of chosen representatives as much as possible to reduce the complexity of computations.

10.3. How to Represent Cells

The most important question for doing computations in this part is: What are cells for the computer and how to generate and represent them. We have to include in this decision which kind of operations we have to perform with cells. For us this means that we have to be able to check the inclusion relations for cells and to compute the action of a given group element. Therefore, we will also give a short description how to do this (see Section 10.3.2.2 and 10.3.2.3).

Group elements are in our program given as `MatrixGroupElement`, which is a predefined data type in SAGE, which uses parts of the `MatrixGroup` construction from GAP. Each element in a SAGE `MatrixGroup` can be given as a matrix.

It is a little bit more complicated to get a suitable description for (orbi-)cells. We decided to represent an orbicell in $\mathbb{W}/\Gamma$ as well as a cell in $\mathbb{W}$ by a representative in $\mathbb{W}$ or, to be more precise, by an element in $\mathfrak{W}$. This has the advantage that the computations are much simpler in $\mathbb{W}$ or $\mathfrak{W}$ and the disadvantage that to describe an orbicell in $\mathbb{W}/\Gamma$ we have to deal with lifts and maybe several equivalent definitions for the same orbicell. The first problem was discussed in the previous section for cells and is therefore not a problem at all. The problem of having several descriptions which define the same object occurs not only for orbicells in $\mathbb{W}/\Gamma$, but also for cells in $\mathbb{W}$ itself. In $\mathbb{W}$ two sets which are equivalent under the equivalence relation defined on page 44 define the same cell. In $\mathbb{W}/\Gamma$ all cells of the same type in $\mathbb{W}$ define the same orbicell in the quotient. Up to further notice we are only dealing with cells in $\mathbb{W}$. We describe first how to represent cells in $\mathbb{W}$.

10.3.1. Cells in $\mathbb{W}$ - Dealing with Equivalent Defining Sets

According to the results in Chapter 9 we know that every cell σ – independent of whether a cell in $\mathbb{W}$ or in $\mathbb{W}/\Gamma$ – is defined by an element $X \in \mathfrak{W}$, i.e. a collection of integral column vectors. Therefore, a minimal definition has to contain only this set X. We defined on page 44 an equivalence relation on the set of collections of integral column vectors in the context of cones. We reformulate this together with the results from Corollary 9.2.6:

Proposition 10.3.1: *Let $X = (\mathbf{x}_1, \dots, \mathbf{x}_k)$ and $X' = (\mathbf{x}'_1, \dots, \mathbf{x}'_l)$ with $\mathbf{x}_i, \mathbf{x}'_i \in \mathbb{Z}^4$. We have that $\mathscr{C}(X) = \mathscr{C}(X')$ if $k = l$ and there is a permutation $\sigma \in \mathbb{S}_k$ and $\epsilon_1, \dots, \epsilon_k \in \{\pm 1\}$ such that*

$$(\mathbf{x}_{\sigma(1)}, \dots, \mathbf{x}_{\sigma(k)}) = (\epsilon_1 \cdot \mathbf{x}'_1, \dots, \epsilon_k \cdot \mathbf{x}'_k)$$

and therefore $[X] = [X']$.

Recall that if we enclose a collection of vectors in $[\dots]$ we consider the collection as collection up to equivalence as described in Proposition 10.3.1. We choose a representative in each class of collections of column vectors in a way such that we are able to compute this representative effectively. For that purpose we introduce a standardization of the signs of the columns and their order in following way:

- If the first non-zero entry of a column is negative, replace the column by its negative.
- Sort the columns in the following way:
  - Count the number of zeroes in each column. Take first the columns with the most zero entries, then the others with ascending number of non-zero entries.
  - Inside a block with the same number of zeroes the columns are ordered in such a way that in each row the size of the entries is descending.

An element which has the above stated form we call an element of **standard form**.

We decided to store sets of columns in a matrix because then the group action on a cell is a matrix multiplication from the left with the inverse of the matrix representing the group element (comp. equation (9.6)). See Sage Source Code B.1 for the implementation of the function `getstandardform` in SAGE, which converts a given matrix, i.e. collection of column vectors, into the standard form.

Here an example in SAGE with a random matrix with six columns:

```
sage: m1 = random_matrix(ZZ,4,6, density=0.5); m1
    [  0   0   0   4   1  -1]
    [  0 -14  -3   0   0  -2]
    [ 17   0   0   0   0  -2]
    [  0   1   0  -1   0   1]
sage: getstandardform (m1)
    [ 1  0  0  4  0  1]
    [ 0  3  0  0 14  2]
    [ 0  0 17  0  0  2]
    [ 0  0  0 -1 -1 -1]
```

In our program we will store every defining set for cells in this form. If we perform any kind of operation on a cell we apply the induced action an the set of standard form and store, after the operation is done, the obtained result again in standard form. This allows us to compare cells (for equality or inclusion relations) simply by comparing the stored generating sets in standard form.

This is a huge advantage compared with the case where we compare matrices in non standardized forms because then the only thing one can do is to compare all possible combinations of signs and permutations, which costs much more time. For example, for the smallest set of four columns one has to compare in the worst case up to

$$2^4 \times 4! = 384$$

matrices. In case of the largest occurring collection in Σ with 12 columns there are up to

$$2^{12} \times 12! = 1,961,990,553,600 \approx 1.96 \times 10^{12}$$

possible matrix representations to check. The matrices themselves are not that huge that one matrix comparison operation (about 215ns) needs a mentionable amount of time, but the potentially huge number of necessary comparisons which has to be done changes the things. It is quite obvious that it is not very intelligent to compare up to 10^{12} matrices, which needs roughly 116 hours, in each check of equality of cells. In SAGE we can use the very efficient implemented sorting algorithms coming from Python in the command `sorted` such that the standardization after each operation is much less time consuming then one naively performed comparison operation for a large example (see e.g. Table 10.1). Almost all parts of our algorithms to compute closures, stabilizers, etc. require a great many comparisons, therefore it is important to have a storing format which allows fast comparisons.

10.3.2. Cells as Objects - Unifying Descriptions

From some point of view it is enough to represent a cell by the matrix in standard form which encodes the defining set in $X \in \mathfrak{W}$. But then we have – as mentioned above – to deduce the other informations by often time-consuming computations. So it is better to remember some additional data like the rank or the standard set in which orbit X is contained. These information are important if we work also with cells in the quotient or if we construct a

Table 10.1.: *Comparison of runtimes for the comparison operation. The examples are random matrices of full rank with four rows and the number of columns stated in the first column. We measured the time for a single comparison operation which checks if two matrices are equal. Further, we computed the maximal possibly necessary number of matrix comparisons in case of the comparison of cells in non standard form. The product of these numbers gives the maximal total time for one cell comparison, which is also stated in the table. In addition we measured the time to transform and store a matrix in standard form using the* `getstandardform` *command. The maximal total time in this case is then the sum of the time per comparison and the time to convert the matrix to standard form, which is roughly the same as the time which is needed to convert the matrix, therefore we do not list it in the table. All timings were done on the ray server* `harmonia` *at MPIM in Bonn and are the arithmetic mean of 625 runs.*

		Non Std. Form		**Std. Form**
columns	**time per comparison**	**maximal number of comparisons**	**maximal total time**	**time to get standard form**
4	219ns	384	84.1μs	391μs
6	214ns	4.6×10^4	9.86ms	630μs
8	216ns	1.03×10^7	2.23s	685μs
10	216ns	3.71×10^9	13.38min	751μs
12	213ns	1.96×10^{12}	116.08h	789μs

resolution to compute the cohomology. We know by Theorem 9.2.11 that every set $X \in \mathfrak{W}$ is an image of a set $X_0 \in \Sigma$ under the group Γ. So a the cell σ can be given by an element in Σ together with (at least) one element in Γ. In other words, we have to store besides the defining set also the standard set from which the the defining set can obtained and a group element which maps one onto the other. It is not useful to recompute this information every time it is used, because one has a similar problem than in the previous section. this information is also used to identify cells with its corresponding orbicells in the quotient. It might also be useful to have more information, as the dimension or the rank of the cell or later if we have computed them, the cells in the boundary or the stabilizer of the cell.

10.3.2.1. Cells as Objects

A good concept to handle this is to use so called objects. This concept is known in many programming languages such as C++ or Python (and so in SAGE). Since we do not assume that all readers are familiar with this concept we describe what this means in context of our example, the description of cells.

An object is some kind of data type which has some underlying attributes, e.g. an integer which is the dimension or a matrix which is a defining set of the cell. Furthermore, an object can carry information which operations are allowed with or on it and how to perform these operations. In other words an object is a kind of container for some data, called attributes, and functions, called methods, related to this data. This concept is very familiar to mathematicians since usually we have complicated structures which we have to encode in some way. A good example might be an integer. An integer has its value, but we also have to know how to multiply this number with other objects of the same type, i.e. an integer. Objects allow a great degree of abstractness in the code. We defined in our package `Cells` cells as objects carrying lots of different attributes and methods. The more structure we define for an object, the slower might be the access (in particular the creation of new objects depends on this). Parts of the data are necessary, e.g. the defining set and the rank. Other parts, like the closure of a cell and its stabilizer, could be computed, but are stored after the fist computation as an attribute of the cell to avoid to compute it again and again. This is extended in the way that we cached this data in addition on the disc to avoid the recomputation for the next time.

For example, the method `stabilizer` returns the attribute `_stabilizer` if the attribute is set, otherwise it checks whether a cached version can be found on the disc and if this also fails, it starts to compute the stabilizer and stores the result. We use this also for other time consuming parts like the closure of a cell. In this way we can avoid the time consuming recomputation of results, because we can load them from disc after they were computed once. To define an object in SAGE we use so called classes. The class which defines cells is called `Cell`. One can create a new object of a class by calling the class. This means in case of the class `Cell` evaluating the command `Cell (matrix, key1=val1,...)`, where the parameter `matrix` is required and further optional arguments (keys) can be set, otherwise a default value is used. For the class `Cell` there are the following arguments. We list them together with their default parameter if there is one.

matrix The matrix containing the defining set as columns. Any matrix is converted to standard form during initialization this ensures that in any case the matrix has standard form.

name='Cell' The name which is returned when calling the cell. For cells coming from a standard set we give by default a string concatenating the cell type with an integer.

latex_name='Cell' SAGE objects allow to return a LaTeX string. This string can contain LaTeX formating commands.

cell_type=None A string equal to name of the standard set, i.e. one of the following strings: `['RedSquare', 'Vertebra', 'Crystal', 'Pyramid', 'Triangle', 'Square', 'Hexagon', 'RR', 'DR', 'Desargues', 'Reye']` or `None` if the type is not known. This attribute is used to identify orbicells in the quotient since they are uniquely determined by its type.

cell_rank=None An integer equal to the rank of the defining set in sense of Theorem 9.1.3.

bd_type=False A boolean, i.e. `True` or `False`, which indicates whether the cell is of boundary type. This information is not used so far.

transv=None A boolean or `None`. It is set to `None` if the is of boundary type or it is not known whether the cell is transverse or not. This information is not used so far.

map2standard=None A symplectic matrix γ such that γ applied to the cell is one of the standard cells defined by an elements in Σ.

closure=None List of objects from the class `Cells` which lie in the closure of the cell.

stabilizer=None An object of the SAGE class `MatrixGroup` equals to the stabilizer of the cell.

These arguments are stored as attributes of the class `Cell` and can be used to compute further data for the cell. By default some parameters, like stabilizer or closure, are set to be `None`, although we could compute them during the initialization of the new object. We postpone their computation to the first use, such that the initialization of a new object is faster and we avoid the expensive (in time) computation for those objects where this information is maybe never used. After the first use the related attribute is set from `None` to its real value and in some cases also stored in addition on disc. We will explain this later. There is one important exception to this: if we generate a cell as image under some group operation, we compute the new values, for example for the attributes `_stabilizer` and `_closure`, from the original object, because they can be obtained simply by translation, conjugation, or simple copying of the original value.

Beside the class `Cell`, which describes arbitrary cells, we defined a further class called `StandardCell` which inherits from`Cell` the definition and its properties with only a simple modification which allows to define the matrices of the standard set as lists of integers which refer to the positions of the occurring columns. Here we use the fact that there is only a rather small number of different columns which are combined in different ways to get the standard sets. This makes it easier to define the standard elements. In other words, the only difference in handling the objects of the class `StandardCell` is during the initialization.[3] All objects defined by elements coming from Σ are collected in a predefined list of objects called `StandardElements`, such that they are easily accessible for further use.

In the following two sections we have a short look at the two most important basic operations for cells: The action of group elements, i.e. matrices, and the comparison of two cells. These are two of several class methods which are included in definition of the class `Cell`.

10.3.2.2. Action on Cells

We defined the operation of group elements as class method in the class `Cell`. Here we use the possibility to overload operators. This means that we replace the `*` operator for multiplication by our own definition on the objects of type `Cell`. Beside this short form we have also the class method `apply` which does exactly the same. We decided only to implement the right action on the cells, because that is is the only action we used up to now. However we want for simple handling that in the implementation multiplication with a group element g from the right *and* the left side returns the result of the right action R_g of the group on cells.

We illustrate how the group action can be computed with an example how we can use the group action in our program.

[3]and they return `True` in the method `is_StandardCell`.

```
sage: C= RedSquare; C.matrix()
    [ 1  0  0  0 ]
    [ 0  1  0  0 ]
    [ 0  0  1  0 ]
    [ 0  0  0  1 ]
sage: g=matrix([[-1,0,1,0], [0,1,0,-1], [0,0,1,0],
    [0,0,0,-1]]);
sage: g
    [ -1  0  1   0]
    [  0  1  0  -1]
    [  0  0  1   0]
    [  0  0  0  -1]
sage: g*C
    [1 0 1 0]
    [0 1 0 1]
    [0 0 1 0]
    [0 0 0 1]
sage: C.apply(g)
    [1 0 1 0]
    [0 1 0 1]
    [0 0 1 0]
    [0 0 0 1]
sage: C*g==g*C
    True
```

The standard multiplication in Python of object with an element, which is defined in the class method `__mul__`, is left multiplication, i.e. one computes `C*g`. Left here refers to the position where the object of the class in which this multiplication is defined is placed. To get a code which looks more like the usual notation we overload also the right multiplication, which is defined in the method `__rmul__`, such that we have a multiplication also from the other side. For the class `Cell` both multiplication methods are, as shown in the example, set to give the same result. This allows us to read the code in the usual way, i.e. the group acts from the left on the cell.

Remember, a group element, which is encoded as an object of the class `Matrix`[4] or an object of the class `MatrixGroupElement`, operates on the cell by multiplication with the inverse of the matrix on the set of columns from the left (see (9.6)). The Sage Source Code B.2 in Appendix B shows also – we mentioned that before – that if further attributes of the cell are known the corresponding attributes are computed from that of the original cell. To be mentioned are in particular the closure and the stabilizer, which can be obtained by translation or conjugation with the element which is applied to the cell. Other attributes as the rank and the type stay the same under the action and are simply copied.

After the multiplication is done a new object of the class `Cell` is created by the data computed from the original object. In this creation operation the application of the function `getstandardform` is hidden. As last part we return the newly created cell.

[4]Note: `Matrix` is the class in SAGE. In addition there is a function which is a matrix constructor called `matrix` which parses the arguments to the constructor of the class. This function `matrix` is used in the example above.

10.3.2.3. Comparison of Cells

Most parts of our algorithm consist of comparisons of cells, so we are now going to have a closer look on the comparison of cells. This method is simple, but also enormously important. Assume we have two cells σ_1 and σ_2 and we want to know how they are related. We say $\sigma_1 = \sigma_2$ if they are equal (it is up to now only a tautology). We say $\sigma_1 < \sigma_2$ if σ_1 lies in the boundary of σ_2 and $\sigma_2 < \sigma_1$ if σ_2 lies in the boundary of σ_1. According to Corollary 9.2.6 we can test the property for σ_1 to be in the boundary of σ_2 by testing whether the columns of σ_2 are contained in the set of columns of σ_1. To check equality we simply compare the defining matrices in standard form of both cells (comp. Section 10.3.1). We implemented this in the function `is_in_closure_of` (see Sage Source Code B.3) which then is used in the method `__cmp__` which in Python is used to check the relations `<`,`>`,`>=`,`<=`,`!=` and `==`. To be precise we overload the standard comparison operator coming from the method `__cmp__` with our own comparison function described above. The method `__cmp__` has to return real numbers and is interpreted by Python in the following way:

$$\texttt{\_\_cmp\_\_}(\sigma_1, \sigma_2) \begin{cases} < 0 & \text{if } \sigma_1 < \sigma_2 \\ = 0 & \text{if } \sigma_1 = \sigma_2 \\ > 0 & \text{if } \sigma_1 > \sigma_2 \end{cases} \tag{10.5}$$

We decided to set the value of `__cmp__` in the top case equal to -1 and in the bottom case equal to 1. Furthermore, if neither the cells are equal, nor one lied in the boundary of the other we return the value None (see Sage Source Code B.4). To return the value None allows to separate those cases and to avoid a crash during execution of the program. We want to emphasize that it is is enough to us the class method `is_in_closure_of`, since we check first whether both are equal. The class method `is_in_boundary_of`, which also exists, uses also `is_in_closure_of` and checks in addition for inequality.

Note: This check only works in case of cells in $\mathbb{W}$. There is no such simple check for orbicells in $\mathbb{W}/\Gamma$. For them one has to compute for each type the closure and can then deduce from that which cell types occur in the boundary of a given type and which do not.
Now we give a short example how `__cmp__` is used:

```
sage: C1= Vertebra; C1.matrix()
     [1 0 0 0 1]
     [0 1 0 0 0]
     [0 0 1 0 0]
     [0 0 0 1 1]
sage: C2=RR; RR.matrix()
     [ 1  0  0  0  1  0  0  1  1]
     [ 0  1  0  0  1  1  0  1  0]
     [ 0  0  1  0  0  0  1 -1 -1]
     [ 0  0  0  1  0 -1 -1  0  1]
sage: C= RedSquare; C.matrix()
     [ 1  0  0  0 ]
     [ 0  1  0  0 ]
     [ 0  0  1  0 ]
     [ 0  0  0  1 ]
sage: C2==C1, C2<C1, C<C2, C1>C, C>=C, C2!=C
    False, False, True, True, True, True
```

10.4. Computing Closures of Cells

Computing the closure of a cell in $\mathbb{W}$ is, beside the computation of the stabilizer, which we handle later on, the most important operation in our program. The implementation has two steps. First, one has to ensure that one is in the boundary of the cell $\mathscr{C}(\textsc{RedSquare})$, then one can use a very simple method to get a list consisting of all cells which lie in the boundary of a given cell, which we describe in Section 10.4.1. Here one uses the fact that there is only one orbit of 4-cells and all representatives of lower dimensional cells are chosen such that they lie also in the boundary of the chosen 4-cell $\mathscr{C}(\textsc{RedSquare})$. We know by Theorem 9.1.10 that $\mathbb{W}$ is a regular cell complex. Therefore, the boundary of each cell is a union of finitely many lower dimensional cells.

For the later use we need a good description how the standard 4-cell and its boundary components as well as and its surrounding cells together with their boundary look in detail (see Section 10.7). Therefore, we need to be able to compute the closure for any cell, but we we really compute the closure for $\mathscr{C}(\textsc{RedSquare})$ and some 4-cells lying around.

10.4.1. The Basic Idea

Without loss of generality we assume that the cell for which we want to compute the closure is the standard cell $\mathscr{C}(\textsc{RedSquare})$, which is given by the four by four unit matrix. Since all cells are translates of the standard cells and therefore the closure of $\mathscr{C}(\textsc{RedSquare})$ contains at least one representative of each orbit under Γ we can obtain the closure of all other 4-cells also by translation. Furthermore, it is easy to see from the fact that $\mathbb{W}$ is a regular cell complex (see Theorem 9.1.10) that if a lower dimensional cell lies in the boundary of $\mathscr{C}(\textsc{RedSquare})$ also its closure has to lie in the boundary of $\mathscr{C}(\textsc{RedSquare})$.

According to Chapter 9 we have 11 orbits of cells under the action of Γ. Each orbit is defined by a standard element $X \in \Sigma$. The cell $\mathscr{C}(\textsc{RedSquare})$ is the only 4-cell among these standard cells, thus its closure is the union of $\mathscr{C}(\textsc{RedSquare})$ itself with cells of lower dimension, which are given by some translates of all of the 10 classes coming from elements in $\Sigma - \{\textsc{RedSquare}\}$. It is necessary for a cell $\mathscr{C}(X)$ for some $X \in \mathfrak{W}$ to be in the boundary of $\mathscr{C}(\textsc{RedSquare})$ that $[\textsc{RedSquare}] \subset [X]$. If we assume that X has standard form we only have to check $\textsc{RedSquare} \subset X$ or equivalently to use the methods `__cmp__` or `is_in_boundary_of` discussed above.

Furthermore, we know that the set X has to be a translate of an element in $\Sigma - \{\textsc{RedSquare}\}$. What does this mean? Let $X = [\mathbf{x}_1, \ldots, \mathbf{x}_k] \in \mathfrak{W} = \Gamma \cdot \Sigma$ for some integer $k \geq 4$ and $\gamma \in \Gamma$, then we have by (9.6) and the definition of the cells $\mathscr{C}(X)$ that

$$R_\gamma(\mathscr{C}(X)) = \mathscr{C}(\gamma^{-1} \cdot X) \tag{10.6}$$

and

$$\gamma^{-1} \cdot X = \gamma^{-1} \cdot [\mathbf{x}_1, \ldots, \mathbf{x}_k] = [\gamma^{-1} \cdot \mathbf{x}_1, \ldots, \gamma^{-1} \cdot \mathbf{x}_k]. \tag{10.7}$$

By assumption $\mathscr{C}(X)$ is a cell in the boundary of $\mathscr{C}(\textsc{RedSquare})$, therefore we know that $[\mathbf{e}_1, \ldots, \mathbf{e}_4] \subset X$. If the defining set X has standard form we have $\mathbf{x}_1 = \mathbf{e}_1, \ldots, \mathbf{x}_4 = \mathbf{e}_4$.

Since X is in the orbit of some $X_0 \in \Sigma$ we find an element $\gamma \in \Gamma$ and an element $X^0 = [\mathbf{x}_1^0, \dots, \mathbf{x}_k^0] \in \Sigma$ such that $X^0 = \gamma^{-1} \cdot X$. Now we have that

$$\gamma^{-1} \cdot \mathbf{e}_i =: \mathbf{g}_i \in X^0 \tag{10.8}$$

where the $\mathbf{g}_i$ are on the one hand the i-th column vectors of the matrix representing γ^{-1}, on the other hand they are up to a factor $\epsilon_i \in \{\pm 1\}$ some of the column vectors in the collection X^0. Thus, we have that there are $\epsilon_1 \dots, \epsilon_4 \in \{\pm 1\}$ and a permutation $\sigma \in \mathbb{S}_4$ such that the matrix of γ^{-1} is given by the following column representation

$$\gamma^{-1} = (\epsilon_1 \cdot \mathbf{g}_{\sigma(1)}, \epsilon_2 \cdot \mathbf{g}_{\sigma(2)}, \epsilon_2 \cdot \mathbf{g}_{\sigma(3)}, \epsilon_2 \cdot \mathbf{g}_{\sigma(4)}). \tag{10.9}$$

Since either $\mathbf{g}_i \in X^0$ or $-\mathbf{g}_i \in X^0$ we find a permutation $\tau \in \mathbb{S}_k$ such that $\mathbf{g}_{\sigma(i)} = \mathbf{x}_{\tau(i)}$. We summarize:

Lemma 10.4.1 (Prima Variatio): *Let $\mathscr{C}(X)$ for some $X = [\mathbf{x}_1, \dots, \mathbf{x}_k] \in \mathfrak{W}$. The cell $\mathscr{C}(X)$ is in the closure of $\mathscr{C}(\text{RedSquare})$ if and only if there are an element $X^0 \in \Sigma$, $\epsilon_1, \dots, \epsilon_4 \in \{\pm 1\}$, a permutation $\tau \in \mathbb{S}_k$, and a matrix $\gamma \in \Gamma$ such that $[\gamma^{-1}X] = [X^0]$ where γ^{-1} has the column representation $\gamma^{-1} = \left(\epsilon_1 \cdot \mathbf{x}^0_{\tau(1)}, \epsilon_2 \cdot \mathbf{x}^0_{\tau(2)}, \epsilon_3 \cdot \mathbf{x}^0_{\tau(3)}, \epsilon_4 \cdot \mathbf{x}^0_{\tau(4)}\right)$.*

We define

$$\Gamma(X) = \bigcup_{\sigma \in \mathbb{S}_k/\mathbb{S}_4} \bigcup_{\substack{(\epsilon_1,\dots,\epsilon_4) \\ \in \{\pm 1\}^4}} \left\{ \gamma \in \Gamma : \gamma^{-1} = \left(\epsilon_1 \cdot \mathbf{x}^0_{\tau(1)}, \epsilon_2 \cdot \mathbf{x}^0_{\tau(2)}, \epsilon_3 \cdot \mathbf{x}^0_{\tau(3)}, \epsilon_4 \cdot \mathbf{x}^0_{\tau(4)}\right) \right\} \tag{10.10}$$

to be the set of all elements in Γ which can be constructed from columns in X with some signs. We can now use Lemma 10.4.1 to compute the $\gamma \in \Gamma$ which define the cells in the closure of $\mathscr{C}(\text{RedSquare})$.

Corollary 10.4.2 (Closure): *We have*

$$\overline{\mathscr{C}(\text{RedSquare})} = \bigcup_{X \in \Sigma} \bigcup_{\gamma \in \Gamma(X)} \mathscr{C}(\gamma^{-1} \cdot X)$$

where the set $\Gamma(X)$ is the set defined in (10.10).

It is not so difficult to get the list $\Gamma(X)$ for $X \in \Sigma$ by simply computing all permutations with all possible sets of signs, and then to check whether the resulting element is a symplectic matrix. The only problem might be that the number can be quite huge. There are for a standard set with k columns $4! \times 2^4 \times \binom{k}{4}$ different matrices which have to be checked whether they are symplectic or not. Let us call these matrices **candidate matrices**. This means we have to check between 384 matrices for the set RedSquare and 190080 matrices for cells of type Reye for each set in Σ. All in all there are 448512 different candidate matrices. Neither are all this matrices symplectic nor define they different cells because of the possible action of stabilizers. So we first have to check whether they are in Γ and then whether they define a new element in the closure. We will propose a method to reduce the number of candidates.

Remark: Someone might insist that one can also get the elements by solving linear equations, but this means to solve not only one but lots of linear equations – namely equations coming from changing the representative of the class – which is much more expensive (comp. Table 10.1) than the combinatorial construction (with some improvements we still have to describe), because we have no advantage from using our standard form for the sets.

To reduce the number of candidate matrices we have to check we decompose them. We can write these matrices in the form

$$\begin{aligned}\gamma^{-1} &= \left(\epsilon_1 \cdot \mathbf{x}^0_{\tau(1)}, \epsilon_2 \cdot \mathbf{x}^0_{\tau(2)}, \epsilon_3 \cdot \mathbf{x}^0_{\tau(3)}, \epsilon_4 \cdot \mathbf{x}^0_{\tau(4)}\right) \\ &= \begin{pmatrix} \epsilon_1 & & & \\ & \epsilon_2 & & \\ & & \epsilon_3 & \\ & & & \epsilon_4 \end{pmatrix} \left(\mathbf{x}^0_{\tau(1)}, \mathbf{x}^0_{\tau(2)}, \mathbf{x}^0_{\tau(3)}, \mathbf{x}^0_{\tau(4)}\right).\end{aligned} \tag{10.11}$$

We know that $\gamma^{-1} \cdot X^0$ and $-\gamma^{-1} \cdot X^0$ define the same cell since $-\mathrm{Id}$ acts trivially on $\mathfrak{W}$ and $\mathbb{W}$. This means that we can generate all necessary elements from only half of the candidates. We denote by $\mathbf{D}$ the set of diagonal matrices with ± 1 entries on the diagonal which contains for each such matrix m only m or $-m$. We call $\mathbf{D}$ the **set of sign matrices**. We decompose this set into the set of elements which are symplectic, which is denoted by $\mathbf{D}_s$, and the complement $\mathbf{D}_n$, which containes the non symplectic sign matrices. So we get

$$\mathbf{D} = \mathbf{D}_s \sqcup \mathbf{D}_n. \tag{10.12}$$

In our example we have $2^4 = 16$ possible distributions of ± 1 on the diagonal, but only $2^3 = 8$ are in $\mathbf{D}$ because the others are redundant. Out of these 8 matrices 6 are non symplectic and 2 are symplectic.

Let $\mathbf{M}(X)$ be the set of four by four matrices which can be constructed from columns from the set $X \in \mathfrak{W}$. We call $\mathbf{M}(X)$ the **set of column matrices for** X. The basic idea is that one decomposes each candidate matrix into a (unique) product of a sign matrix in $\mathbf{D}$ with a column matrix in $\mathbf{M}(X)$ for some $X \in \mathfrak{W}$. Since matrices form a group under multiplication we can reduce the number of times where we have to check for symplecticity, because if both matrices are symplectic the product has to be symplectic, if one is symplectic but the other is not the product cannot be symplectic and we have not to consider these products only if both matrices are not symplectic one has to check the product for symplecticity.

To reduce the number of checks needed to decompose $\mathbf{M}(X)$ into symplectic and non symplectic elements we use an additional test which we apply to the elements before we start to check whether a matrix is symplectic or not. We simply compute the absolute value of the determinant of the element and check whether the result is one. This needs roughly 4.2μs compared to 1.72ms to check the symplecticity of a matrix.[5] So we got a decomposition $\mathbf{M}(X)$ into

$$\mathbf{M}(X) = \mathbf{M}_s(X) \sqcup \mathbf{M}_n(X) \sqcup \mathbf{M}_o(X), \tag{10.13}$$

where $\mathbf{M}_o$ are the elements with a determinant different from ± 1 (most of them are even singular), $\mathbf{M}_n(X)$ are those of the remaining elements which are not symplectic, and $\mathbf{M}_s(X)$ are the symplectic elements in $\mathbf{M}(X)$.

Due to the fact that all elements in $\mathbf{D}$ have to have either determinant 1 or -1 we can remove the elements in $\mathbf{M}_o(X)$ from our further considerations. All in all we reduced the task of checking all candidate matrices for symplecticity using the following cases. Assume for this that a candidate matrix m is written in the form $m = d \cdot m_c$ where $d \in \mathbf{D}$ and $m_c \in \mathbf{M}(X)$ for some $X \in \mathfrak{W}$.

[5]Both computations done on the ray server `harmonia` at the MPIM in Bonn. We took for each the arithmetic mean over 625 runs and repeated that several times to ensure that the result is stable.

Table 10.2.: *Candidate matrices by cell type: We counted the number of candidate matrices depending on their kind. This data is stated in the shaded columns. There are three kinds of matrices: those which fail the first test in the algorithm which checks whether the absolute value of their determinant is one, those which fail the test for symplecticity, and those which pass both test. The last two columns show the number of checks for symplecticity and the number of computations of determinants of 4×4-matrices which are done during the reduction of the number of candidate matrices before checking whether the cells defined by the matrices are already known.*

Cell		Number of matrices in			Tests for	
Type	**columns**	$\mathbf{M}_s(X)$	$\mathbf{M}_n(X)$	$\mathbf{M}_o(X)$	**sympl.**	**det**
Desargues	10	2040	2970	30	20820	5040
Reye	12	4392	7404	84	51912	11880
DR	9	1224	1778	22	12468	3024
RR	9	1080	1914	30	13428	3024
Hexagon	6	144	198	18	1404	360
Square	8	665	999	16	7009	1680
Triangle	8	600	1066	14	7476	1680
Vertebra	5	48	66	6	468	120
Crystal	6	144	210	6	1476	360
Pyramid	7	264	568	8	3984	840
RedSquare	4	0	22	2	156	24
	Sum:	10601	17195	236	120601	28032

Case 1 If $d \in \mathbf{D}_s$ and $m_c \in \mathbf{M}_s(X)$ keep m.

Case 2a If $d \in \mathbf{D}_s$ and $m_c \notin \mathbf{M}_s(X)$ remove m from the list of candidate matrices.

Case 2b If $d \notin \mathbf{D}_s$ and $m_c \in \mathbf{M}_s(X)$ remove m from the list of candidate matrices.

Case3 If $d \in \mathbf{D}_n$ and $m_c \in \mathbf{M}_n(X)$ check whether $m \in \mathbf{Sp}_2(\mathbb{Z})$. If so keep m, otherwise remove m from the list of candidate matrices.

In this way we can reduce the number of checks for being symplectic. But we add some time to construct the lists of symplectic and non-symplectic elements. Before this reduction we had to check for all $2^4 \cdot \sum_{X\in\Sigma} \#\mathbf{M}(X) = 448512$ candidate matrices whether they are symplectic or not. Each summand in the formula above has the form $4! \times \binom{k}{4}$ if the set X has k columns. If we assume that each check needs 1.72ms (see above) we need 16.3s only for these checks. If we use the decomposition described above we first have to check for all column matrices, i.e $\sum_{X\in\Sigma} \#\mathbf{M}(X) = 28032$ many times, the determinant which needs 4.2μs per time. Therefore, we need about 118.ms to do this.

It remains to check for $\sum_{X\in\Sigma}(\#\mathbf{M}(X) - \#\mathbf{M}_o(X))$ matrices the symplecticity. Altogether we have to compute

$$2^3 + \sum_{X\in\Sigma} (\#\mathbf{M}(X) - \#\mathbf{M}_o(X) + \#\mathbf{D}_n \cdot \#\mathbf{M}_n(X)) = 120609$$

times the condition to be symplectic and not 448512. But we pay for this by the additional computation of some determinants, but this is not that expensive. This optimized version needs for the part to perform all checks for symplecticity 207.4s compared with 771.4s in the non optimized case[6]. This means a speed-up for this part of about 73.11%.

The other advantage is that we have already removed the doublets coming from the action of $-\mathrm{Id}$, so that the output of symplectic elements is only half as large as in the non optimized case. This reduces the time for the remaining part, where we have to check whether we got new cells, to only a half.This is the most time consuming part of the algorithm. Furthermore, we can reduce the number of elements which have to be checked by the simple fact that the closure is the union of the cell with cells of lower dimension. Therefore, we can skip all cells which have dimension larger than or equal to the dimension of the cell of which we want to compute the closure. The only cell of the same dimension which we have to include is the inner cell itself. For the details of the algorithm we refer to the implementation of the function `getclosure`, which is given in an abbreviated form in Sage Source Code B.5.

The runtime corresponds directly to the number of candidate matrices one has to check. Therefore, the most time consuming parts during the computation are obviously those coming from defining sets which have the most columns, i.e. in particular the cells of the types Desargues, Reye, DR, and RR. The only exception is the computation of the closure of cells of type Desargues and Reye itself, because there is no need to compute the closure of points since they are closed. If we want to compute the closure of a cell with dimension larger than zero we have to check the column matrices $\mathbf{M}(X)$ for all X which define cells of lower dimension. As a direct consequence of this we would expect that the computation of the closure of cells of the same dimension should need roughly the same time.

Furthermore – we mentioned this already before – if a cell lies in the boundary of a larger cell so does its closure, too. This can be used to reduce the time for the computation of the closure of all cells which lie in the boundary of a cell whose closure is known, because we can take the already computed closure of the larger cell and check the elements in this list whether they are in the closure of the smaller cell or not. In our implementation we apply this to all cells in the boundary of $\mathscr{C}$(RedSquare). Thus, we first compute the closure of $\mathscr{C}$(RedSquare) and then the closure of other cells. In Table 10.3 we compare the time to compute the closure, on the one hand using the precomputed closure for $\mathscr{C}$(RedSquare) and on the other hand computing each closure on scratch. The time to get the closure of $\mathscr{C}$(RedSquare) which is about 8.5 minutes on `theia`, is therefore the biggest part while computing the structure of the cell complex.

10.4.2. Computing Closures

We can now assume that we can compute the closure for arbitrary cells in the closure of $\mathscr{C}$(RedSquare). If we have an arbitrary cell $\mathscr{C}(X)$ for some $X \in \mathfrak{W}$ we can get the closure of $\mathscr{C}(X)$ by translating it to a suitable cell in the closure of $\mathscr{C}$(RedSquare). We can then there compute the closure and translate the closure back to the cell we started with. The main problem is to find an element $\gamma \in \Gamma$ which maps $\mathscr{C}(X)$ to a cell $\mathscr{C}(X^0)$ in the closure of $\mathscr{C}$(RedSquare) for which we can compute the closure. The set X^0 can be an element from Σ,

[6]In all timings in this section the time for generating the lists as object (not to compute their content) is omitted. The generation of lists has to be done only a couple of times in the whole program and the time for each generation – between 9μs and 8ms depending on the length – is negligible compared with the operations we have to perform several times.

Table 10.3.: *Time to compute the closure of standard cells with and without using the precomputed closure of $\mathscr{C}$(RedSquare), using the command `getclosure`. The computations are done on the ray servers `harmonia` and `theia` at the MPIM in Bonn. To force not to use the precomputed results we removed temporarily the related part from the program. This leads to a very tiny speed-up which is – as we see in the timings – over compensated by the other slower operations. All times are the arithmetic mean of 5 runs of the program where we do not use the precomputed data and the arithmetic mean of 625 runs in the other cases.*

	Time on					
	`harmonia`		`theia`		`NOETHER`	
Cell-Type	**with**	**without**	**with**	**without**	**with**	**without**
Desargues	1.17μs	59.5ms	3.32μs	76.5ms	406ms	0.03s
Reye	1.21μs	63.2ms	3.41μs	67.0ms	532ms	0.03s
DR	1.19μs	369.08s	3.30μs	314.65s	355ms	323.08s
RR	1.22μs	369.64s	3.37μs	315.21s	371ms	322.13s
Hexagon	1.19μs	494.73s	1.81μs	422.85s	230ms	433.15s
Square	1.22μs	497.67s	1.00μs	422.77s	317ms	435.01s
Triangle	1.16μs	496.66s	3.77μs	423.73s	311ms	443.47s
Vertebra	1.18μs	571.68s	3.58μs	487.10s	215ms	505.19s
Crystal	1.15μs	567.17s	3.30μs	486.09s	250ms	499.87s
Pyramid	1.16μs	570.36s	3.36μs	487.89s	273ms	503.81s
RedSquare	-	606.23s	-	512.00s	-	538.58s

but it does not need to be. There are essentially two ways to get a suitable $\gamma \in \Gamma$.

a) The easiest method is to use the attribute `map2standard` if this is different from `None`. In this case it gives exactly one of the γ we are looking for. In practice this will happen only if $\mathscr{C}(X)$ is still in the closure of the standard 4-cell $\mathscr{C}$(RedSquare) or we got it by applying a translation to such an element.

b) Use the function `getmap2standard`, which computes such a map and sets the related attribute of the cell. This function uses similar techniques to those used before to get the closure for cells. Then we can proceed with the previous case.

As illustrated in case of the closure of $\mathscr{C}$(RedSquare), solving linear equations is not the method of choice to get a map to a standard cell, because for each element also here we have to solve lots of linear systems.

The starting point to find a map to a standard element is that we have only one type of 4-cell among the standard cells and all other 4-cells are translates of the cell $\mathscr{C}$(RedSquare). Let $\mathscr{C}(X^4)$ be an arbitrary 4-cell, i.e. there is a $\gamma \in \Gamma$ such that

$$R_\gamma(\mathscr{C}(X^4)) = \mathscr{C}(\text{RedSquare}). \tag{10.14}$$

Now it is important to remember that

$$\text{RedSquare} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}.$$

Hence, we have

$$[\gamma^{-1} \cdot X^4] = [\textsc{RedSquare}] \tag{10.15}$$

or equivalently

$$[X^4] = [\gamma \cdot \textsc{RedSquare}] = [\gamma]. \tag{10.16}$$

We know by Theorem 9.1.10 and Lemma 9.2.1, together with the fact that all standard cells lie in the closure of $\mathscr{C}(\textsc{RedSquare})$, that any cell lies in the closure of some 4-cell. Furthermore, we have that if we take a map which maps an arbitrary 4-cell $\mathscr{C}(X^4)$ onto $\mathscr{C}(\textsc{RedSquare})$, also the closure of $\mathscr{C}(X^4)$ is mapped onto the closure of $\mathscr{C}(\textsc{RedSquare})$. Therefore, it is sufficient for the computation of the closure of an arbitrary cell $\mathscr{C}(X)$ to find one 4-cell $\mathscr{C}(X^4)$ for which $\mathscr{C}(X)$ lies in its boundary. But it is not necessary to get all 4-cells for which $\mathscr{C}(X)$ lies in its boundary. To get all such cells – also for other cells than 4-cells – is the thing we have to do if we compute the neighbours of a cell (see Section 10.4.3).

For runtime reasons Python uses, if possible, so called iterators for the iteration in `for`-loops. In particular, if we iterate over sets which are constructed with `range`, `tuples`, or `arrangements`, the iteration over one of these sets is not implemented by computing the full list of elements and running through this list. Instead of that the iterator over this set is used. This is an object which has an actual value and some functions which give the next or the previous element in the list. After each cycle of the loop the next value is computed from the actual value which then replaces the actual value. This has the advantage that if – as in this example – there is an early termination of the loop, we avoid the generation of the remaining elements of the list. This is much faster and uses less memory. So this part of the program will be much faster then the one which computes the neighbours (see Section 10.4.3).

It remains to find a way to get as fast as possible a suitable 4-cell $\mathscr{C}(X^4) = \mathscr{C}([\gamma])$ for some $\gamma \in \Gamma$. We use for this a variation of the method we used before to get the closure for a cell in the closure of $\mathscr{C}(\textsc{RedSquare})$. If $\mathscr{C}([\gamma])$ is a 4-cell such that $\mathscr{C}(X)$ lie in its boundary, we have to have by Corollary 9.2.6 that $[\gamma] \subset [X]$, i.e. the columns of γ occur (up to their sign) in the list of columns of X. Therefore, we get a similar result to Lemma 10.4.1.

Lemma 10.4.3 (Secunda Variatio): *Let $\mathscr{C}(X)$ for $X = [\mathbf{x}_1, \ldots, \mathbf{x}_k] \in \mathfrak{W}$ be any cell and $\mathscr{C}(X^4)$ a cell which lies in the orbit of the standard 4-cell $\mathscr{C}(\textsc{RedSquare})$, i.e. there is a $\gamma \in \Gamma$ such that $R_\gamma(\mathscr{C}(X^4)) = \mathscr{C}(\textsc{RedSquare})$. The cell $\mathscr{C}(X)$ lies in the closure of $\mathscr{C}(X^4)$ if and only if $[X^4] = [\gamma]$ for a $\gamma \in \Gamma$ which then has to have the column representation $\gamma = (\epsilon_i \cdot \mathbf{x}_{\tau(i)})_{i=1,\ldots,4}$, where $\epsilon_i \in \{\pm 1\}$, $\tau \in \mathbb{S}_k$.*

The set of all such γ is in bijection with the set $\Gamma(X)$ introduced earlier with the difference that here we want that γ and not γ^{-1} has the desired form. It might also be a method for a later further speed-up of our algorithm to use precomputed lists of matrices of this form.

The implementation of the function `getmap2standard` can use then the same techniques as described in Section 10.4.1. But since we need only a single element we can return the first symplectic matrix which fulfils the conditions from Lemma 10.4.3. Thus, we do not need to use the decomposition into a sign matrix and a column matrix, because if there is a symplectic column matrix $m \in \mathbf{M}(X) \cap \mathbf{Sp}_2(\mathbb{Z})$ (and there is always one) we can return the first m we have found and are finished.

10.4.3. Computing Boundaries and Skeletons

In some sense boundaries and skeletons are derived constructions from the closure of a cell or a union of cells. For simplicity we restrict ourselves to the case of single cells. We obtain the boundary of a cell by taking its closure and removing the cell from that list. To get the k-skeleton – here we understand as the k-skeleton of a cell the k-skeleton of its closure – of a cell we take from the list containing the closure only those cells of dimension lower or equal to k. By default $k = 1$ in our program. We also implemented a realization of the 1-skeleton as graph in SAGE. Here one can use the plotting libraries to get a two or even a three dimensional picture of the cells.

10.5. Computing Stabilizers of Cells

The technique to compute the stabilizer of a given cell $\mathscr{C}(X)$ for some $X = [\mathbf{x}_1, \ldots, \mathbf{x}_k] \in \mathfrak{W}$ is a further variant of the idea which was used to compute the closure. The stabilizer Γ_X of a cell is the group of all elements $\gamma \in \Gamma$ such that $[\gamma \cdot X] = [X]$. We assume that $\mathscr{C}(X)$ is a cell in the closure of $\mathscr{C}(\textsc{RedSquare})$. As before, we have that the standard basis vectors $\mathbf{e}_1, \ldots, \mathbf{e}_4$ are in the collection X. If X is given in standard form we have $\mathbf{x}_1 = \mathbf{e}_1, \ldots, \mathbf{x}_4 = \mathbf{e}_4$. Therefore, we have that

$$\gamma(\mathbf{x}_1, \ldots, \mathbf{x}_k) = (\gamma \cdot \mathbf{x}_1, \ldots, \gamma \cdot \mathbf{x}_k) = (\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_2, \mathbf{g}_3, \gamma \cdot \mathbf{x}_4, \ldots \gamma \cdot \mathbf{x}_k) \tag{10.17}$$

where $\mathbf{g}_i$ is the i-th column of γ. Therefore, an element $\gamma \in \Gamma_X$ has to have the following column representation

$$\gamma = (\epsilon_1 \cdot \mathbf{x}_{\tau(1)}, \epsilon_2 \cdot \mathbf{x}_{\tau(2)}, \epsilon_3 \cdot \mathbf{x}_{\tau(3)}, \epsilon_4 \cdot \mathbf{x}_{\tau(4)}) \tag{10.18}$$

where $\tau \in \mathbb{S}_k$ is a permutation and $\epsilon_i \in \{\pm 1\}$. This is only a necessary condition, which needs not to be sufficient, because these elements stabilize at least 4 of the k columns, but not necessarily all k. Therefore, one has to check in addition whether the elements of this form really stabilize the cell. Thus

Lemma 10.5.1 (Tertia Variatio)**:** *The stabilizer Γ_X of X is given by*

$$\Gamma_X = \bigcup_{\tau \in \mathbb{S}_k/\mathbb{S}_4} \bigcup_{\substack{(\epsilon_1, \ldots, \epsilon_4) \\ \in \{\pm 1\}^4}} \left\{\gamma \in \Gamma : \gamma = (\epsilon_1 \cdot \mathbf{x}_{\tau(1)}, \epsilon_2 \cdot \mathbf{x}_{\tau(2)}, \epsilon_3 \cdot \mathbf{x}_{\tau(3)}, \epsilon_4 \cdot \mathbf{x}_{\tau(4)})\right\} \cap \Gamma_X.$$

To compute Γ_X we generate a list of elements of the demanded form with the very same algorithm as described in Section 10.4.1 for computing the elements mapping a cell to a standard cell (see in particular Sage Source Code B.5). In this way we get a list of matrices. From this list we have to remove those elements which do not stabilize the other columns of the defining set of the cell $\mathscr{C}(X)$. As in the previous variations on this method, this method is much faster than to compute the stabilizer by solving lots of linear equations.

Assume now $\mathscr{C}(X)$ for some $X \in \mathfrak{W}$ is an arbitrary cell which not necessarily lies in the closure of $\mathscr{C}(\textsc{RedSquare})$. Then we find, using Lemma 10.4.3, an element $\gamma \in \Gamma$ such that $R_\gamma(\mathscr{C}(X)) = \mathscr{C}(\gamma^{-1} \cdot X)$ is a cell in the closure of $\mathscr{C}(\textsc{RedSquare})$. Then we have

$$\Gamma_X = \gamma^{-1} \cdot \Gamma_{\gamma^{-1} \cdot X} \cdot \gamma. \tag{10.19}$$

Table 10.4.: *Runtime to compute stabilizers for cells in the closure of* RedSquare$_0$ *by cell type. All timings were done on the ray server* `theia` *at MPIM in Bonn and are the arithmetic mean of 25 runs if possible each over a different representatives of the same class.*

Cell Type	Time [s]	Std. Dev.
Desargues	129.44	0.69
Reye	305.57	0.94
DR	78.20	0.23
RR	78.05	0.09
Triangle	43.34	0.05
Square	43.34	0.09
Hexagon	9.85	0.04
Vertebra	3.44	0.04
Crystal	9.62	0.07
Pyramid	21.85	0.08
RedSquare	1.36	-

It is simple to get from this equality together with Lemma 10.5.1 the stabilizer Γ_X of $\mathscr{C}(X)$. Our SAGE program provides the function `getstabilizer` which is also used to compute, if necessary, the value of the class attribute `_stabilizer`, and the class method `stabilizer` of an object of the class `Cell`, which either returns the value of the attribute `_stabilizer` if it is different from `None` or computes the stabilizer using the function `getstabilizer`. As in case of the function `getclosure`, the attribute `_closure`, and the class method `closure`, we use also a combination of cached and newly computed results.

For some purposes we are interested to know in an abstract sense how the groups look like. In other words we identify up to isomorphism the groups, which are realized in a first instance in terms of matrix groups, with finite groups, which we can write in terms of e.g. cyclic groups C_n or dihedral groups D_n. In our SAGE program we realize stabilizers as objects in the class `MatrixGroup`, which is realized in SAGE via a wrapper from GAP. The matrix group is initialized by a set of generators, which can be either matrices defined over some ring or matrix group elements, which again are given by some matrix. By default we consider the groups over the rational numbers $\mathbb{Q}$. But all groups are as finite groups already realized over $\mathbb{Z}$. Therefore, we can reduce the matrices modulo some prime p. If p is large enough, i.e. it is coprime to the order the group, the finite groups over $\mathbb{F}_p$ we obtain in this way is isomorphic to the group over the integers. For subgroups of Γ is is sufficient that $p \geq 7$ (comp. Section 6.3.3). In our SAGE program we wrote a function `makefinitegrp` which reduces a matrix group modulo p and returns this group as a `MatrixGroup` over $\mathbb{F}_p$. By default we reduce module `7`, but the function can take an optional parameter `p` to choose any prime we want. In GAP one can use for matrix groups over a finite field (or permutation groups) the `SmallGroups` Library [BEO02] to get the description we are looking for. For this we use our function `getgroupstructure`, which has as an optional parameter the prime p, with which the group is reduced.

One has to mention that here we cannot reduce the time for the computation of the stabilizer by deducing them via a filtering procedure as we can for the closure, because there is – except for the group Γ itself – no larger group containing other stabilizers. We study the intersection behaviour of stabilizers for different cells later in Section 10.7.5.

10.6. Computing Neighbours

For the computation of the cohomology – as we will explain in Section 13.3 – it is essential to know for cells – besides their closure, which we already discussed in Section 10.4.2 – their surrounding cells of higher dimension, which we call neighbours, of the considered cell. To be precise, the set of neighbours of a cell $\mathscr{C}(X)$ is the set of those cells whose closure has non-zero intersection with the cell $\mathscr{C}(X)$. We can apply Lemma 10.4.3 to get a list of all 4-cells whose closure contains the cell $\mathscr{C}(X)$. Due to the fact that the (only) 4-cell is represented by the set of the four standard unit vectors in $\mathbb{Z}^4$, we know that all other 4-cells are represented by a collection of four vectors which are, up to their order and sign the column vectors of a symplectic matrix. In other words, the elements we need are represented by the matrices which map $\mathscr{C}(X)$ onto a cell which lies in the boundary of the standard 4-cell RedSquare. Since we can compute the closure for each cell $\mathscr{C}(Y) \in \mathbb{W}$ (see Sections 10.4.1 and 10.4.2), it is not difficult to identify all cells whose closure contains $\mathscr{C}(X)$ by applying a suitable filter to the list of all cells which are in the closure of at least one of the 4-cells which lie around $\mathscr{C}(X)$. If we know the closure of the standard 4-cell RedSquare it is quite easy to compute all these closures fast, as we explained earlier. Unlike in the method to compute the map in `getmap2standard`, here we have to find all 4-cells with this property and not only one, so that we cannot leave the loop earlier. Therefore, the runtime depends essentially only on the numbers of columns and symplectic matrices which we can obtain from them. Thus, the runtime does not differ much for cells of the same type. Some timings with the standard deviation of the considered sample can be found in Table 10.5.

The SAGE source code of the function `getneighbours` can be found in the Appendix in section B.3. In Section 10.7.3.1 we will analyse the structure of the set of neighbours associated to a given cell under the stabilizer of that cell in detail.

Table 10.5.: *Time to compute all cells in the neighbourhood of a cell of given type in the boundary of* RedSquare$_0$. *For the computation of neighbours there are in some cases large differences between different representatives of the same cell type, in particular for those with a larger number of columns. Therefore, we include also the standard deviation for the samples of the test. All timings were done on the ray server* `theia` *at MPIM in Bonn and are the arithmetic mean of 25 runs, if possible each for a different representative of the same class.*

Cell Type	**Time** [s]	**Std. Dev.**
Desargues	353.67	40.69
Reye	949.75	102.25
DR	151.75	11.94
RR	197.16	28.09
Triangle	33.82	1.61
Square	34.75	1.34
Hexagon	26.99	3.76
Vertebra	1.81	0.12
Crystal	4.53	0.49
Pyramid	11.68	1.27

10.7. Results

In the following sections we give an overview on results we can obtain using our program in SAGE. Some additional results can be found in the different lists on the CD-ROM, which is included in the printed copy of this thesis. In this section we restrict ourselves to the interpretation and visualization of these results.

10.7.1. The Structure of the Stabilizers of Cells

We start with the description of the stabilizers of cells since we need them to understand the orbicells in the quotient which we handle later in context of the closures of cells. There are two things we can do to verify our results by known results. On one hand we know by the result of Balz Bürgisser [Bür82] which finite orders can occur for elements in Γ and thus in the stabilizers (see Section 6.3.3 and therein in particular Theorem 6.3.6 and Corollary 6.3.7). However the knowledge of orders cannot tell us which groups occur. On the other hand we know that $\mathfrak{S}_2/\Gamma$ is isomorphic to $\mathcal{A}_2$, i.e. the moduli space of principally polarized abelian surfaces (see Section 6.3.4). Since $\mathbb{W}/\Gamma$ is a deformation retract of that space, the points and therefore also the cells correspond to some isomorphism classes of abelian surfaces. The stabilizers of lifts of points in $\mathbb{W}/\Gamma$ correspond to the automorphism groups of these varieties. As mentioned in Section 6.3.4 there are several results on abelian surfaces with a non-trivial automorphism group, i.e. the only automorphisms are $\pm\mathrm{Id}$, and which groups can occur as automorphism groups of abelian surfaces. Unfortunately the known results are not stated in a sufficiently explicit way to directly check the correctness of our results, for several reasons. Some of the authors handle only special cases like abelian surfaces which are isomorphic to a product of elliptic curves [GR99] or abelian varieties with non-trivial automorphism groups which are isolated in $\mathcal{A}_2$ [GMG05]. Other authors determine only the orders of the groups occurring as automophism groups or focus not on the automorphisms, but rather on the structure of the abelian varieties themselves. These kinds of results are not very helpful at all to check the results in detail. There are few results which give a more complete picture of the situation in the case of abelian surfaces, which is interesting for us. In these works usually the groups are realised as abstract groups or subgroups of the endomorphism ring of the varieties in terms of groups of units of number fields with complex multiplication. We give a much more detailed picture of the situation because we compute for all cases the stabilizers of cells explicitly as matrix groups, which we then identify with abstract groups. Among these few results one can find the most complete results in the thesis of Dieter Schmidt from Erlangen [Sch97] and in the work of Shi-Shyr Roan on automorphisms of complex 2- and 3-tori [Roa90]. The work of Roan is unfortunately unpublished, but it is presented e.g. in [BL04], which we used as reference. In addition there is a recent preprint [BLS11] of Reinier Broker, Kristin Lauter, and Marco Streng, which gives also a full classification of all abelian surfaces with some predescribed endomorphism structure. Their result is somehow more general, but can be easily specialized to our case. For more information on this topic we refer to Section 6.3.4 and the references therein. We use the known results briefly sketched in that section to compare, up to some level, the results of our computation of stabilizers to them at the end of this section. There are some problems during this comparison. One reason is that the results are formulated in at least three different languages. There are some given as abstract groups, others in terms of CM-fields, and our own results as matrix groups. Furthermore, it is not so easy to make the connection between cells and a particular abelian surface, because we have no explicit map from our purely combinatoric model to the moduli space. We can only deduce the connection from some obstructions. For example, we can compare stabilizers of cells to automorphism groups of abelian surfaces to make a connection. However this does not work to differentiate between cells with isomorphic stabilizers. Finally we do not have a one-to-one correspondence between the list of possible automorphism groups and the list of stabilizers of cells, because the automorphism groups correspond to stabilizers of points in $\mathbb{W}/\Gamma$. But since the elements of the stabilizer need not to stabilize the cell point-wise, single points within cells might have different (larger) groups as stabilizer, which we do not identify at this point.

To compute the stabilizer of a cell in $\mathbb{W}$ we use the algorithm described in Section 10.5. We computed the stabilizers for all standard cells. The stabilizers of the other cells of the same type are are by equation (10.19) isomorphic to that of the standard cell of that type, since we have only to conjugate with integral matrices. To get a better picture how this groups look like we did several things. First, we computed for the matrix groups we found the order of each element and – this requires the knowledge of the cells in the boundary – determined whether the elements act on $\mathbb{W}$ orientation preserving or orientation reversing (see Table 10.6). Further, we use the GAP `SmallGroups` Library [BEO02] to identify the abstract structure of the groups we got in terms of simple components such as cyclic groups or dihedral groups, the number of generators, and whether the group is abelian or not (see Table 10.7).

Due to the results of Bürgisser [Bür82] we expected to find elements of the orders 1, 2, 3, 4, 5, 6, 8, 10 and 12 in the group Γ (see Section 6.3.3). A comparison with the orders of elements stated in Table 10.6 shows that elements of all expected orders really occur in some of the stabilizers. Furthermore, we find that all elements in the groups have characteristic polynomials which are products of cyclotomic polynomials. This property is stable under conjugation because conjugated matrices have the same characteristic polynomial (see Table 10.8). Therefore, stabilizers of cells of the same type have the same characteristic polynomial. The possible orders of the elements, which are annihilated by these polynomials, are also determined by the orders. However the knowledge of all characteristic polynomials of elements in a stabilizer of a cell give some deeper insight than only to know the order of the elements. Nevertheless also this additional information does not give the concrete subgroups of Γ.

Let us now have a closer look at the structure of the stabilizers. As mentioned above we can use the connection to abelian surfaces to check whether our results are plausible (see Section 6.3.4). Some possible automorphism groups do not occur – as mentioned above – in our list of stabilizers. The reason is that some groups occur only as stabilizer of points and not as stabilizer of cells. One example of an abelian surface whose automorphism group does not occur is $A = E \times E_{\mathbb{i}}$ where E is a generic elliptic curve with trivial automorphism group and $E_{\mathbb{i}} = \mathbb{C}/\mathbb{Z}[\mathbb{i}]$. Furthermore, we have the problem that one can equip abelian surfaces which are isomorphic to the product of two isogenous elliptic curves with quite different automorphism structures, corresponding to different CM-structures. We restrict ourselves to presenting here only three examples of the correspondence, which belong to three different types of abelian surfaces which have a non-trivial automorphism group. The other cases can be checked analogously.

Table 10.6.: *Stabilizers of standard cells. All stabilizers of cells of the same type are isomorphic. We computed the orders of the groups and all their elements, and determined whether their operation is orientation preserving (O) or non orientation preserving (N), and collected their numbers in this table.*

Cell type	**Order**	**Order of Elements**									**Orientation preserving**	
		1	2	3	4	5	6	8	10	12	O	N
Desargues	10	1	1			4			4			
Reye	48	1	13	8	6		8	12				
DR	2	1	1								2	
RR	12	1	7	2			2				12	
Hexagon	72	1	9	8	6		36			12	72	
Square	4	1	3								2	2
Triangle	2	1	1								2	
Vertebra	24	1	3	2	4		6			8	12	12
Crystal	24	1	9	2	6		6				12	12
Pyramid	4	1	3								2	2
RedSquare	32	1	7		16			8			16	16

Table 10.7.: *Structure of Stabilizers. We use* GAP *with its* `SmallGroups` *Library [BEO02] to get the structure of the stabilizer in terms of elementary groups. Then we state the* GAP *group ID which is a pair consisting of the order of the group and a consecutive number, which identifies the group in the database. I the last two columns we give the number of generators and an information whether the group is abelian or not.*

Cell Type	**Structure**	GAP **Group Id**		**# Generators**	**Abelian**
Desargues	C_{10}	10	2	2	Yes
Reye	$\mathbf{GL}_2(\mathbb{F}_3) = Q_8 \rtimes D_3$	48	29	3	No
DR	C_2	2	1	1	Yes
RR	$D_6 = \mathbb{S}_3 \times C_2$	12	4	3	No
Hexagon	$C_3 \times ((C_6 \times C_2) \rtimes C_2)$	72	30	5	No
Square	$C_2 \times C_2$	4	2	2	Yes
Triangle	C_2	2	1	1	Yes
Vertebra	$C_{12} \times C_2$	24	9	3	Yes
Crystal	$(C_6 \times C_2) \rtimes C_2$	24	8	4	No
Pyramid	$C_2 \times C_2$	4	2	2	Yes
RedSquare	$(C_4 \times C_4) \rtimes C_2$	32	11	3	No

Table 10.8.: *Characteristic polynomials of elements in stabilizers of cells. If there is an element in a group of given type with a characteristic polynomial stated in the first row, the corresponding field is marked with a* x.

Type	Φ_1^4	Φ_2^4	$\Phi_1^2\Phi_2^2$	Φ_3^2	Φ_6^2	Φ_4^2	$\Phi_3\Phi_4$	$\Phi_6\Phi_4$	$\Phi_3\Phi_6$	$\Phi_1^2\Phi_3$	$\Phi_1^2\Phi_6$	$\Phi_1^2\Phi_4$	$\Phi_2^2\Phi_3$	$\Phi_2^2\Phi_6$	$\Phi_2^2\Phi_4$	Φ_5	Φ_{10}	Φ_8	Φ_{12}
Desargues	x	x														x	x		
Reye	x	x	x															x	
DR	x	x																	
RR	x	x	x	x	x														
Hexagon	x		x	x	x				x	x	x								x
Square	x	x	x																
Triangle	x	x																	
Vertebra	x	x					x	x		x	x	x	x	x					
Crystal	x	x	x	x	x	x			x										
Pyramid	x	x	x																
RedSquare	x	x	x			x						x			x			x	

Example 1: Desargues According to Theorem 6.3.9 there is a simple abelian surface which is isomorphic to the complex 2-torus given by

$$\mathscr{O}_{\mathbb{Q}(\zeta_5)} \otimes_{\mathbb{Z}} \mathbb{R}/\mathscr{O}_{\mathbb{Q}(\zeta_5)}$$

where ζ_5 is a primitive fifth root of unity and $\mathscr{O}_{\mathbb{Q}(\zeta_5)}$ the ring of integers in the CM-field $\mathbb{Q}(\zeta_5)$. This abelian surface is also isomorphic to the Jacobian of the curve $y^2 = x(x^5 - 1)$ (see [BL04; p. 421]). Their automorphism group is generated by a primitive tenth root of unity ζ_{10} and therefore isomorphic to C_{10}, which is the stabilizer of a cell of type Desargues (see Table 10.7). We have to mention that $-\mathrm{Id}$ acts trivially as well on the complex torus as on the retract $\mathbb{W}$, Therefore, the action is equal to the action of the group which is generated by ζ_5, which is a root of Φ_5 and Φ_{10}. We know by a result on abelian surfaces that these points have to be isolated fixed points [GMG05].

Example 2: RedSquare Consider the CM abelian surface $A = E_{\mathbb{i}} \times E_{\mathbb{i}}$ whose CM-structure is given by $\{\zeta_8, \zeta_8^5\}$, then we know that

$$\mathbf{Aut}\,(A) = (C_4 \times C_4) \rtimes C_2 = \left\langle \mathbb{i} \cdot \mathrm{Id}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} \mathbb{i} & 0 \\ 0 & 1 \end{pmatrix} \right\rangle$$

(see e.g. [BL04; Cor. 13.3.6, Thm.13.4.5]), which is exactly the stabilizer of a cell of type RedSquare. The structure of the automorphism group is somehow reflected in the occurring characteristic polynomials. There are, beside the elements with characteristic polynomials with roots equal only to ± 1 and elements with characteristic polynomial equal to the cyclotomic polynomial Φ_8, only polynomials which have at least one irreducible factor equal to Φ_4, which is the only factor with non-trivial roots, i.e. such roots which are neither 1 nor -1.

Example 3: Vertebra Consider the abelian surface $A = E_{\mathbb{i}} \times E_{\zeta_6}$ then we know that

$$\mathbf{Aut}\,(A) = C_4 \times C_6$$

(see e.g. [BL04; Thm.13.4.4]). This is the stabilizer of a cell of type VERTEBRA. Here we have a much simpler correspondence between the subgroups of the abelian group $C_4 \times C_6$ and the irreducible factors of the characteristic polynomials of elements in the stabilizer of the cells of type VERTEBRA.

Also the other results we obtained are consistent with the possible groups (see e.g. [BL04, BLS11] and Section 6.3.4). For the computation of the cohomology we have also to compute the intersections of stabilizers. This can be done by comparison of the list of matrices. Results on this can be found in `intersections.pdf` on the CD-ROM, which is included in the printed copy of this thesis.

10.7.2. The Structure of the Cells in $\mathbb{W}$

In this section we give a short overview which information we can get by the method stated earlier in this chapter. Here we focus on the geometric realization of the cells in $\mathbb{W}$. From this we will deduce in Section 10.7.2 also – upto some level – also a picture of the orbicells in $\mathbb{W}/\Gamma$. It is important to emphasize that all pictures come from the combinatorial/topological model. In particular, there is no need that the lengths or angles in the pictures we will show are related to any isometric embedding into a suitable space. However we believe that it is useful to have some kind of simplified geometric picture in mind while dealing with such abstract objects.

The question we want to answer is how do the cells in $\mathbb{W}$ look like. It is easily possible to give a good picture how k-cells in $\mathbb{W}$ look like for $k = 0, \ldots, 3$. Unfortunately our world – at least the part of the world we can see with our eyes – is three dimensional, so we have a little problem to get a suitable picture of the shape of a 4-cell. We will try to do this similarly as in the lower dimension by giving a description how the boundary, which is three dimensional, looks like. The reader with a good spatial awareness might get some idea how this four dimensional object is constructed from knowing the structure of the boundary. For cells of dimension less or equal to three and orbicells of dimension less or equal to two we can draw some nice picture (see Figure 10.1).

There are eleven types of standard cells coming from the eleven sets in Σ. There are one type of a four dimensional cell, three of dimension three, three of dimension two, and two of dimension one and zero each. We computed, using the algorithms described above, lists of boundaries of cells of each type and of neighbours of them. We state the number of them by type together – one above the diagonal one below – in Table 10.9. Using these numbers we get a good description of the cells and how they are related to cells of other types. We find for example that in the boundary of $\mathscr{C}$(REDSQUARE) there are 512 different cells or that around the cell $\mathscr{C}$(REDSQUARE) there are 117 other 4-cells, separated from $\mathscr{C}$(REDSQUARE) by its 512 boundary components. Sometimes it is not that easy to imagine how 4-cells meet in some 3-cells, but we have to think for that in four dimensions (or even higher to get a better embedding[7]).

[7]I really like the embedding theorems from complex analysis for complex manifolds. However in this case they

Table 10.9.: *Cellular decomposition - neighbours and boundaries. How to read this table: In the area above the diagonal (darkly shaded) we find the number of cells of the type stated in the first row which border on a single cell of the type in the first column. This is the number of neighbours of a cell. In the area below the diagonal (lightly shaded) we list the number of cells of the type stated in the first row which lie in the boundary of cell of type in the first column. For example, in the boundary of one cell of* RedSquare*-type we find 48 cells of* Desargues*-type, whereas 15 cells of* RedSquare*-type meet in one cell of* Desargues*-type. These results are consistent with those stated in [MM89; p. 279].*

Cells in the Boundary / Cells in Neighbourhood	Desargues	Reye	DR	RR	Triangle	Square	Hexagon	Vertebra	Crystal	Pyramid	RedSquare
Desargues			5	0	5	5	0	5	5	5	15
Reye			24	8	48	24	4	24	16	36	42
DR	1	1			2	2	0	3	3	3	11
RR	0	1			6	0	1	6	3	6	15
Triangle	1	2	2	1				1	1	2	7
Square	2	2	4	0				2	2	1	8
Hexagon	0	6	0	6				6	0	0	9
Vertebra	12	12	36	12	12	12	2				3
Crystal	12	8	36	6	12	12	0				3
Pyramid	2	3	6	2	4	1	0				4
RedSquare	48	28	176	40	112	64	4	4	4	32	

For an easier use we enumerated the different defining sets of cells of the same type according to their order of generation in the algorithm and gave them the name Type_i, where i is the counter. The standard cells have the number zero. Thus, $\text{RedSquare} = \text{RedSquare}_0$, $\text{Crystal} = \text{Crystal}_0$ and so on. A list of all 513 cells which lie in the closure of $\mathscr{C}(\text{RedSquare})$, with their defining sets can be found in `cells_in _closure_of_RedSq.pdf` on the CD-ROM included in the printed copy of this thesis.

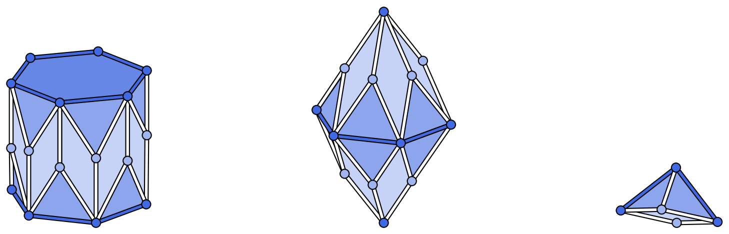
We can get a kind of geometric visualization by the combinatorics of their inclusion relations. For example, if we have something which has a boundary consisting of three vertices and three edges, we can conclude that the shape of this kind of cells has to be something like a triangle. We can do the same in all other cases. We start a survey on the structure of cells starting with the lowest dimensional cells.

10.7.2.1. Dimension 0 and 1 – Vertices and Edges.

Already by their dimension we know that cells of the types Desargues and Reye are vertices, i.e. points, in the decomposition. Each point of Desargues-type in $\mathbb{W}$ is an isolated fixed point of Γ with a stabilizer of order 10 in Γ. They are the only fixed points whose order is divisible by 5. The vertices of type Reye have quite a large (compared with most other groups occurring in this context) stabilizer. The order is 48 and they are the only points, except an isolated fixed point in each 4-cell, whose stabilizers contain elements of order 8

dos not help to understand the structure at all.

Figure 10.1.: *3-cells with their boundaries consisting of the 2-cells, the 1-cells, and the 0-cells. The different cells are coloured in different kinds of blue which are the same as before in the pictures of the 2-cells.*



(a) *Vertebra* **(b)** *Crystal* **(c)** *Pyramid*

(see remarks in Sections 10.7.1 and Table 10.6 on stabilizers and their order).

We know also by their dimension that cells of the types DR and RR have to be line segments. Our computations show now that a cell of type DR connects one cell of type Desargues with an other of type Reye. Furthermore, we get that the closure of a cell of type RR contains, beside the cell of type RR, two different cells of type Reye. These connecting property is the origin of the names of the defining sets DR and RR.

10.7.2.2. Dimension 2

The two dimensional cells in $\mathbb{W}$ have shapes as indicated by their names (that is the reason to name the defining sets in the way they are named). So we have triangles which have as vertices two of type Reye and one of type Desargues, as well as the three edges of types DR and RR (twice). The quadrangle or square has only edges at its boundary of type DR and therefore two vertices of each type. The hexagon shaped cell has only vertices of type Reye and thus also only edges of type RR.

10.7.2.3. Dimension 3 and 4

Next we study the structure of the three different types of 3-cells. We can do this essentially in the same way as we did before in case dimension two. Here we can include the knowledge of the structure of the lower dimensional (orbi-)cells.

The most simple cell is the cell of type Pyramid, which has the shape of a four-sided pyramid with one cell of type Square in the bottom and four triangular cells forming the pyramid above (see Figure 10.1 (c)). The other two types of 3-cells have a more fancy structure.

The boundary of a cell of type Crystal contains two stars, one placed at the top, one at the bottom, each build from six cells of type Square, which are arranged around a common point of type Reye. Between these two sets there is a chain of six vertices of type Reye and six edges of type RR. In each vertex of this chain one cell of type Square from the top meets one cell of type Square from the bottom. The remaining gaps are filled with 12 triangles, six above and six below the central chain (see Figure 10.1 (b)).

The cells of Vertebra-type are the only 3-cells with some hexagonal faces. The two hexagons are placed at two opposite ends of the boundary. They are rotated with an angle of $\pi/6$, i.e. a twelfth rotation, against each other. The boundary contains in addition – as all other kinds of 3-cells – triangles and quadrangles. Twelve triangles – six in in the upper and six in the lower part – meet with their edge of type RR one of the edges of the hexagons of type RReach. The remaining point of each triangle is a vertex of type Desargues, which is connected via an edge of type DR with a vertex of type Reye, which lies in the boundary of the opposite hexagon. The twelve gaps between the triangles and the edges of type DR are filled with cells of type Square (see Figure 10.1 (a)).

In the boundary of a 4-cell we find four cells of type Vertebra, they form – glued along their hexagonal faces – a solid torus, which is in some sense the vertebral column of the boundary. This chain contains 48 vertices of type Desargues and 24 of type Reye. These are all but four vertices in the full closure of the 4-cell. The four remaining vertices are of type Reye and can be placed on a circle, which lies in the hole of the solid torus. We find that any pair of two adjacent vertices on this circle forms, together with some of the vertices from the chain of closures of cells of type Vertebra, a cell of type Crystal. So there are four of these. We have now all vertices and edges of the boundary of the cell of type RedSquare. However there are some holes in this solid construction. These 32 gaps have all the shape of a pyramid and are filled with cells of type Pyramid. All in all there are 512 cells of the ten different classes in the boundary of each cell of type RedSquare (see Table 10.9).

10.7.3. The Structure of Orbits

There are at least two reasons for us to try to understand the orbits of cells under some groups. These groups are on one hand the full Siegel modular group Γ and on the other hand stabilizers of cells. In the first case we get a description of the orbicells in the quotient, which we discuss in Section 10.7.4. In this section we focus on the action of finite groups, i.e. stabilizers of cells. One reason to study the orbits under these groups is that we want to understand how the orbicells in the quotient look like, i.e. we have to compute the quotient of the cells described in Section 10.7.2 under the related stabilizers. The other reason is that for the computation of the cohomology later on we will need to know what the structure of the set of neighbours of a given cell under its associated stabilizer looks like (see Section 13.3).

Therefore, it is of fundamental interest to know not only the cells in the boundary (see Section 10.4) and the neighbours (see Section 10.6) of a given cell, but also their behaviour under the stabilizer of the central cell. One might expect that the cells are distributed equally onto the orbits. But contrary to this, we will find that the structure is a little bit more complicated

10.7.3.1. The Structure of Neighbours

The neighbours of a cell $\mathscr{C}(X)$ are, as explained in Section 10.6, those cells $\mathscr{C}(Y)$ whose closure has non-empty intersection with the closure of $\mathscr{C}(X)$. Therefore, the dimension of cells in the neighbourhood of $\mathscr{C}(X)$ is always larger than the dimension of the central cell $\mathscr{C}(X)$. We computed for each cell $\mathscr{C}(X)$ in the boundary of RedSquare first its set of neighbours before we determined for all of them in which orbit they lie under the stabilizer Γ_X of $\mathscr{C}(X)$. We have at least one orbit per type of cells, since these are in the same orbit under the full Siegel modular group Γ. The results of these computations can be found in Table 10.10.

One observes several things: The action of the stabilizers is almost never transitive on the set of neighboured cells of the same type, i.e. we find more than one orbit per cell type. This makes some things in Section 13.3 a little bit more complicated that it would be in the transitive case because then there are several orbits under the stabilizer and not only one. Further, we find that the length of orbits of the same type is not always constant. This phenomenon occurs only in few cases, namely for cells of type RedSquare around the cells Pyramid$_i$ for $i = 0, \ldots, 31$ under $\Gamma_{\text{Pyramid}_i}$, the cells of type RedSquare around Square$_i$ for $i = 0, \ldots, 63$ under Γ_{Square_i}, and finally for cells of type RedSquare as well as for those of type Crystal around Reye$_i$ for $i = 0, \ldots, 27$ under Γ_{Reye_i}. In all of these four cases the set of orbits splits into two different kinds of orbits. This is somehow astonishing, because one might expect that the situation is completely symmetric under the group action. Although there is this broken symmetry, the structure is the same if we consider another cell of the same type as central cell.

10.7.3.2. The Structure of the Boundary

In some way the consideration of orbits of cells in the boundary of a given cell is 'dual' to the situation presented in the previous section. As before, we compute for all cells in the boundary of the standard 4-cell RedSquare their orbits under the group $\Gamma_{\text{RedSquare}}$ (see Table 10.11 (a)). Furthermore, we did the same for all other types of cells (see Table 10.11 (b)-(h)). We find that for all cells of the same type the structure is the same. As also in case of orbits of neighbours under stabilizers we find here an analogous phenomenon, which affects the same types of cells as in the previously considered orbits of neighboured cells. To be accurate we have the following: There are orbits of two different lengths for the cells of type Reye and Square around a cell of type RedSquare under the corresponding stabilizer, and for cells of type Reye under the stabilizer of a cell of type Crystal around that cell. The details can be obtained from Table 10.11.

10.7.4. The Structure of the Orbicells in $\mathbb{W}/\Gamma$

The description of orbicells in the quotient needs some tinkering with scissors and glue. In particular, in case of higher dimensional orbicells we have to do some fancy identifications of the cells in $\mathbb{W}$. We have to distinguish between the description of an orbicell and the description of the behaviour of the lifts of its boundary components under the map $\widetilde{\pi}$ to the quotient. We have to emphasize that there are two kinds of identifications which take place under $\widetilde{\pi}$, those which come from elements in the stabilizer of the considered cell in $\mathbb{W}$ or the lift of an orbicell in $\mathbb{W}/\Gamma$ and those which come from group elements which map one cell onto a different cell in $\mathbb{W}$. The elements of the stabilizer identify also parts of the cell itself – remember an orbicell $\mathscr{C}_\Gamma(X)$ in the quotient is the image of the cell $\mathscr{C}(X) \in \mathbb{W}$ under its stabilizer Γ_X – and maybe also parts of the boundary to see how they fit in the whole

Table 10.10.: *Orbits of cells in the neighbourhood of a given cell under its stabilizer for non-trivial cases in the boundary of the standard 4-cell* RedSquare$_0$. *(a)* Desargues$_i$ *for* $i = 0, \dots, 47$. *(b)* Reye$_i$ *for* $i = 0, \dots, 27$. *(c)* DR$_i$ *for* $i = 0, \dots, 175$.*(d)* RR$_i$ *for* $i = 0, \dots, 39$. *(e)* Triangle$_i$ *for* $i = 0, \dots, 111$. *(f)* Square$_i$ *for* $i = 0, \dots, 63$. *(g)* Hexagon$_i$ *for* $i = 0, \dots, 3$. *(h)* Vertebra$_i$ *for* $i = 0, \dots, 3$. *(i)* Crystal$_i$ *for* $i = 0, \dots, 3$. *(j)* Pyramid$_i$ *for* $i = 0, \dots, 31$.

Cell Type	**Orbits**	**Structure** (# of orb.×length)
DR	1	1×5
Triangle	1	1×5
Square	1	1×5
Vertebra	1	1×5
Crystal	1	1×5
Pyramid	1	1×5
RedSquare	3	3×5

(a)

Cell Type	**Orbits**	**Structure** (# of orb.×length)	
DR	1	1×24	
RR	2	2×4	
Triangle	2	2×24	
Square	2	2×12	
Hexagon	1	1×4	
Vertebra	1	1×24	
Crystal	3	2×6	1×4
Pyramid	3	3×12	
RedSquare	4	3×12	1×6

(b)

Cell Type	**Orbits**	**Structure** (# of orb.×length)
Triangle	2	2×1
Square	2	2×1
Vertebra	3	3×1
Crystal	3	3×1
Pyramid	3	3×1
RedSquare	11	11×1

(c)

Cell Type	**Orbits**	**Structure** (# of orb.×length)
Triangle	2	2×1
Square	2	2×1
Vertebra	3	3×1
Crystal	3	3×1
Pyramid	3	3×1
RedSquare	11	11×1

(d)

Cell Type	**Orbits**	**Structure** (# of orb.×length)
Vertebra	1	1×1
Crystal	1	2×1
Pyramid	1	1×1
RedSquare	5	7×1

(e)

Cell Type	**Orbits**	**Structure** (# of orb.×length)	
Vertebra	1	1×2	
Crystal	2	2×1	
Pyramid	1	1×1	
RedSquare	5	3×2	2×1

(f)

Cell Type	**Orbits**	**Structure** (# of orb.×length)
Vertebra	1	1×6
RedSquare	1	1×9

(g)

Cell Type	**Orbits**	**Structure** (# of orb.×length)
RedSquare	1	1×3

(h)

Cell Type	**Orbits**	**Structure** (# of orb.×length)
RedSquare	1	1×3

(i)

Cell Type	**Orbits**	**Structure** (# of orb.×length)	
RedSquare	3	2×1	1×2

(j)

Table 10.11.: *Orbits of cells in the boundary of a given cell under its stabilizer for non-trivial cases in the boundary of the standard* 4*-cell* RedSquare$_0$. *(a)* RedSquare$_0$. *(b)* Crystal$_i$ *for* $i = 0, \ldots, 3$. *(c)* Vertebra$_i$ *for* $i = 0, \ldots, 3$. *(d)* Pyramid$_i$ *for* $i = 0, \ldots, 31$. *(e)* Triangle$_i$ *for* $i = 0, \ldots, 111$. *(f)* Square$_i$ *for* $i = 0, \ldots, 63$. *(g)* Hexagon$_i$ *for* $i = 0, \ldots, 3$. *(h)* RR$_i$ *for* $i = 0, \ldots, 39$.

Cell Type	**Orbits**	**Structure** (# of orb.×length)	
Desargues	3	3×16	
Reye	4	1×4	3×8
DR	11	11×16	
RR	5	5×8	
Triangle	7	7×16	
Square	5	3×16	2×8
Hexagon	1	1×4	
Vertebra	1	1×4	
Crystal	1	1×4	
Pyramid	2	2×16	

(a)

Cell Type	**Orbits**	**Structure** (# of orb.×length)	
Desargues	1	1×12	
Reye	3	1×2	2×3
DR	3	3×12	
RR	1	1×6	
Triangle	1	1×12	
Square	2	2×6	

(b)

Cell Type	**Orbits**	**Structure** (# of orb.×length)
Desargues	1	1×12
Reye	1	1×12
DR	3	3×12
RR	1	1×12
Triangle	1	1×12
Square	1	1×12
Hexagon	1	1×2

(c)

Cell Type	**Orbits**	**Structure** (# of orb.×length)
Desargues	1	1×2
Reye	3	3×1
DR	3	3×2
RR	2	2×1
Triangle	2	2×2
Square	1	1×1

(d)

Cell Type	**Orbits**	**Structure** (# of orb.×length)
Desargues	1	1×1
Reye	2	2×1
DR	2	1×1
RR	1	1×1

(e)

Cell Type	**Orbits**	**Structure** (# of orb.×length)
Desargues	1	1×2
Reye	2	2×1
DR	2	2×2

(f)

Cell Type	**Orbits**	**Structure** (# of orb.×length)
Reye	1	1×6
RR	1	1×6

(g)

Cell Type	**Orbits**	**Structure** (# of orb.×length)
Reye	1	1×6
RR	1	1×6

(h)

ensemble of orbicells. Our consideration starts usually first with the identification under the stabilizer elements, followed by the identification of the remaining boundary, which are not yet identified under the action of the stabilizer, components according to their types. The last part does not change the shape of the orbicells. We will give a detailed construction of this in case of orbicells whose dimension is two and less. For 3- and 4-orbicells it is more difficult to get a picture out of this construction. The more advanced reader with much better spatial awareness then myself may get a glimpse how they look like also in the case of 3- or even 4-orbicells in the quotient.

If we want to get a picture of the orbicells in the quotient, we need to include the action of the elements of the stabilizer of the cell, which fold and/or cut and glue the cells in $\mathbb{W}$. To visualize this we will identify parts of their boundary according to the observations made in Section 10.7.3.2. This describes how the orbicells $\mathscr{C}_\Gamma(X) = \mathscr{C}(X)/\Gamma_X$ look like in $\mathbb{W}/\Gamma$. We complete the picture how the different orbicells together with their boundary fit in the quotient $\mathbb{W}/\Gamma$ by identifying in addition also the remaining orbicells in the boundary according to their type. Analogously to the cells in $\mathbb{W}$ we will give a short survey starting with the orbicells of lowest dimension.

10.7.4.1. Dimension 0 and 1 – Vertices and Edges.

As already discussed in Section 10.7.2.1 the dimension forces the cells of type Desargues and Reye to be vertices. In this case orbicells are, independent of the stabilizer, even cells in the classical sense.

Also by dimension we know that the cells of type RR and DR have to be some kind of line segments in $\mathbb{W}$. Since all cells of type Reye become equivalent in the quotient and there is no element in the stabilizers of cells of type RR which interchanges the two end points of type Reye, a cell of type RR in the quotient together with its identified boundary forms a loop, whereas cells of type DR stay line segments with two different endpoints also in the quotient. Furthermore, we can conclude that the stabilizers of cells of types DR and RR stabilize the cell even pointwise. (For 0-cells this is obviously also true.) Therefore, also in dimension one all orbicells are even cells in the classical sense. Finally we want to mention that there are no lines connecting two points of type Desargues.

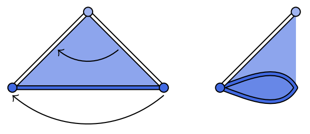
10.7.4.2. Dimension 2

To get a picture of the image of the 2-cells in the quotient, i.e. their images under $\widetilde{\pi}$, we have to cut, fold and glue them and their boundary components according to their stabilizers. According to Table 10.11 all vertices and all edges of the same type lie also in the same orbit under the stabilizers, i.e. in these cases the stabilizer acts transitively on the sets of vertices and edges, i.e. all cells in the boundary. Therefore, we have here

$$\overline{\mathscr{C}_\Gamma(X)} = \overline{\mathscr{C}(X)}/\Gamma, \tag{10.20}$$

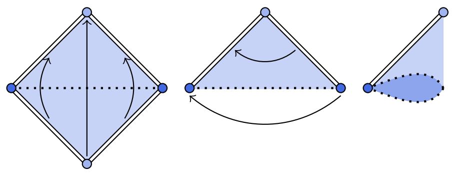
i.e. the orbits under Γ and Γ_X coincide. We tried to illustrate the construction we describe now in some pictures (see Figures 10.2, 10.3 and 10.4).

Figure 10.2.: *2-cell of type* Triangle *and cellular tinkering to the corresponding orbicell in the quotient. The dark vertices are those of type* Reye, *the light vertex is the vertex of type* Desargues. *The dark edge is of type* RR, *the other two light edges are of type* DR. *The arrows indicate how to identify the parts.*



(a) *Original* **(b)** *Result*

Figure 10.3.: *2-cell of type* Square *and cellular tinkering to the corresponding orbicell in the quotient. The dark vertices are those of type* Reye, *the light vertices are of type* Desargues. *All edges are of type* DR. *The arrows indicate how to identify the parts. The dotted line shows the branching points which have different stabilizers then the other points.*



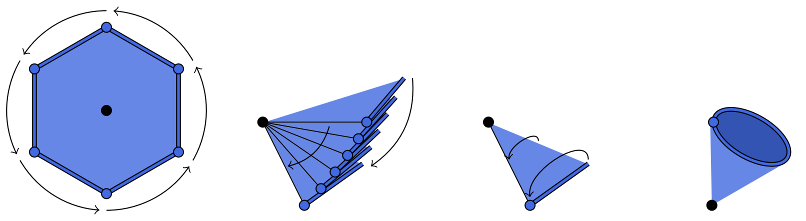
(a) *Original* **(b)** *Intermediate* **(c)** *Result*

Let us start our considerations with the triangular cell. By the computation of the stabilizers of cells of type Triangle we found that the stabilizer consists only of $\pm\mathrm{Id}$, therefore also in the quotient the orbicell has the shape of an open triangle and is therefore a cell. But under the quotient map $\widetilde{\pi}$ both vertices of type Reye and both edges of type DR are identified. Therefore, the picture of the closure of the cell of type Triangle in the quotient looks like the surface of a cone (see. Figure 10.2).

The cells of type Square in $\mathbb{W}$ have a stabilizer of order 4 in Γ which descends for the action on $\mathbb{W}$ to a subgroup of order two in $\mathbf{PSp}_2(\mathbb{Z})$. Therefore, we have to fold and glue the square along one of the two diagonals. Since there is no element mapping the two vertices of type Reye onto each other, but there is an element which interchanges the two cells of type Desargues in the boundary, there has to be a subset of the cell which is stable under this map and which connects the two points of type Reye. This subset, which is the diagonal connecting the two points of type Reye, consists of branching points for the covering of the orbicell in the quotient by its preimage in $\mathbb{W}$. We indicated this subset by a dotted line in Figure 10.3. Note that the element of order 2 in $\mathbf{PSp}_2(\mathbb{Z})$ reverses the orientation of the orbicell. We see that the shape of the image of a cell of type Square in the quotient is a kind of triangle which forms, as in the previous case, a kind of cone if we include the identification by elements which are not in the stabilizer. The bottom side of the cone-like shape shown in Figure 10.3 (b) are only branching points which are *inside* the orbicell and *not* a boundary component. This is the first example of an orbicell which is not a cell in the usual sense.

In case of the Hexagon-type cell we have a very similar situation to that of the cells of type Square if we only look to them as objects in $\mathbb{W}$. We easily find that the shape of cells of Hexagon-type is a hexagon whose boundary consists of six vertices of type Reye joint by six edges of type RR. The stabilizer is much larger (it has order 72) as in case of cells of type SQUARE, but all its elements act orientation preserving on the hexagon. A closer look to its structure and its action on the boundary shows that although the group is much more complicated the action on the hexagon is essentially the action of C_6, the cyclic group of order order 6. Therefore, the only point which is stable under each element of Γ_{HEXAGON} is the *central point* of the hexagon, indicated in Figure 10.4. This central point has the maximal stabilizer equal to Γ_{HEXAGON}. All other points are rotated with multiples of $\pi/3$ around this central point. Therefore, all other points have a much smaller stabilizer which has only order 12 and is the subgroup of Γ_{HEXAGON} which stabilizes the complete cell pointwise. Thus, the orbicell $\mathscr{C}_\Gamma(\text{HEXAGON})$ is a sixfold covering with an single branching point which has the shape of a cone (see Figure 10.4). In this case one easily find that this cone in the quotient is homeomorphic to an open disc and therefore different from the orbicell of type SQUARE, also a classical cell.

Figure 10.4.: *2-cell of type* HEXAGON *and cellular tinkering to the corresponding orbicell in the quotient. The central branching point is indicated by by a black dot. In the intermediate step we cut the hexagon into six triangular leaves around the branching point. Here only the thin solid black lines in the front of the picture belong to the triangle. In the result these six leaves are glued together to a sixfold covered cone (black arrows in (b) and (c)) with a loop in the boundary coming from the boundary of the hexagon.*



(a) *Original* **(b)** *Intermediate 1* **(c)** *Intermediate 2* **(d)** *Result*

10.7.4.3. Dimension 3 and 4

Now we ask what we can say about the image of the 3- and 4-cells under the projection map $\widetilde{\pi}$ in the quotient. We start with the cell of type PYRAMID. The stabilizer of the pyramidal cell in Γ is isomorphic to $C_2 \times C_2$ which descends to a stabilizer of order two in $\mathbf{PSp}_2(\mathbb{Z})$. This is exactly the same stabilizer as the stabilizer of the cell of type SQUARE in the bottom. Therefore, in the quotient the orbicell is a twofold branched cover. The preimages branching points in a fixed lift of the orbicell form in $\mathbb{W}$ a triangle shaped set. The vertices of this triangular shaped set are the cells of type DESARGUES in top of the pyramid and the two vertices of type REYE in the boundary of the rectangular cell in the bottom. The stabilizer of the cell identifies also parts of the boundary components: The rectangular cell is – as described above – folded and glued to a triangle, two pairs of triangles and the corresponding edges and vertices are identified. It becomes rather complicated if we include the behaviour of the remaining identifications under $\widetilde{\pi}$. This means that we identify the remaining 2-, 1- and 0-cells of the same type. The result of this is not so easy to draw. We leave it as an exercise to the interested reader. Here we have the same problem than in case of the orbicell of type SQUARE that the object in the quotient is an orbicell but not a cell. The same is true for the remaining objects in the quotient.

More complicated to describe are the remaining two types of 3-orbicells. We start with the cells of type Vertebra in $\mathbb{W}$. These cells have a stabilizer which acts as a C_{12}. This action consists of of two types of operations. First rotations around the axis connecting the isolated fixed points of the two hexagons. This axis stays stable also under the second component of the action of the stabilizer, which interchanges the two hexagons (and forced by this also the other cells), and therefore under all elements in Γ_{Vertebra}. This flip does not preserve the orientation of the cell. However there is a subset of the cell which stays stable under this flip which defines some additional branching points in the quotient. In the intersection of this subset and the axis is a fixed point with maximal stabilizer equal to Γ_{Vertebra}. Note that also the cells of type Square stay stable under the flip of the two hexagons (but not the cells in their boundary, they are interchanged). Furthermore, under this flip pairs of triangles are interchanged. This can be combined with a rotation around the central axis. In this way we get a description of the covering and branching behaviour of an orbicell of type Vertebra in the quotient. To complete the picture how this orbicell fits in the quotient we have now in addition to identify the images of all remaining cells of the same type in the boundary.

The last type of 3-cells we have to consider are those of type Crystal. The stabilizer of cells of this type is isomorphic to $(C_6 \times C_2) \rtimes C_2$ (see Table 10.7). The action of the stabilizer consists, similar to the previous case, of a rotation around a central axis connecting the top and the bottom point of type Reye of the cell shown in Figure 10.1 (b), together with an interchange of the part above and below the hexagon consisting of the remaining six vertices of type Reye. Also in this case this flip reverses the orientation of the cell. Therefore, we have as sets of branching points the central axis and the plane defined by the six vertices of type Reye. The intersection point of these two sets is the fixed point with maximal stabilizer. Further, there are elements of the stabilizer which interchange pairs of triangles. This gives us a picture of the covering and branching of the orbicell in the quotient. As last part of this construction we have to identify the cells in the boundary according to their types.

Now we have to finish our considerations of the structure of the orbicells in the quotient by handling the only 4-orbicell. This is from our point of view the most complicated object due to the lack of intuition in handling four dimensional object. Nevertheless we try to give a sketch in the same way we did before in case of lower dimensional parts of its boundary. There is exactly one representative of each type of orbicells in the closure of the 4-orbicell in $\mathbb{W}/\Gamma$. Their description was already given above. Remember, the closure of $\mathscr{C}_\Gamma(\text{RedSquare})$ is equal to $\mathbb{W}/\Gamma$. Thus, we have to complete our picture of $\mathbb{W}/\Gamma$ by looking to the behaviour of $\mathscr{C}(\text{RedSquare})$ under its stabilizer. The stabilizer acts transitively on the sets of cells of types Crystal and Vertebra. Each of these sets consists of four elements. For each cell $\mathscr{C}(X)$ for some $X \in \mathfrak{W}$ which is of type either Crystal or Vertebra in the boundary of $\mathscr{C}(\text{RedSquare})$ there is a subgroup of $\Gamma_{\text{RedSquare}}$ which is isomorphic to $D_4 = C_4 \rtimes C_2$ which also stabilizes $\mathscr{C}(X)$ (comp. Table 10.12). The stabilizer $\Gamma_{\text{RedSquare}}$ is isomorphic to $(C_4 \times C_4) \rtimes C_2 = D_4 \times D_4$ which has order 32. We know by our computations of $\Gamma_{\text{RedSquare}}$ that the highest order of an element in this group is eight. Furthermore, the order of the stabilizer in $\mathbf{PSp}_2(\mathbb{Z})$ is only 16. Therefore, it is not surprising that the set of 32 cells of type Pyramid splits into two disjoint orbits of 16 cells each.[8] On these two orbits the stabilizer of the 4-cell act transitively. There are several cyclic subgroups therein which have orbits of length two or four. But no element from the stabilizer of the 4-cell – except $\pm\text{Id}$ – stabilizes

[8]If the 32 cells of type Pyramid are enumerated as mentioned before in the order of their generation, i.e. we have $\text{Pyramid}_0, \ldots, \text{Pyramid}_{31}$. The first orbit consists of the cells with the indices $\{0, 1, 2, 3, 12, 13, 14, 15, 20, 21, 22, 23, 24, 25, 30, 31\}$. The other orbit consist of the remaining 16 cells.

a pyramid. Thus, we have, after we applied the action of the stabilizer to the boundary, two pyramids, and one of the other 3-cells each. The lower dimensional components are not changed up to now since their stabilizers intersected with the stabilizer of the 4-cell consist only of $\pm\mathrm{Id}$. First, we have now to identify the remaining two classes of pyramids, then we can replace all 3-cells together with their boundaries by the description stated above. Finally we have to identify also the remaining classes of images of lower dimensional cells according to the results of Section 10.7.3.2 for the picture of lower dimensional branching point inside RedSquare. In the center of the 4-cell there is a distinguished point which has the full stabilizer of the cell as stabilizer. This point lies in the intersection of all fixed point components of the subgroups of $\Gamma_{\text{RedSquare}}$ which are isomorphic to D_4. A deeper analysis of these groups and the permutations as discussed in Section 10.7.3.2 defined by them can give a more detailed picture of their fixed point components, which we skip at this place. Parts of that will be covered of the discussion of intersections of stabilizers in the following Section 10.7.5.

10.7.5. Intersection of Stabilizers

The groups which are results of the computation of stabilizers in our program are according to Section 10.7.1 objects in the class `MatrixGroup`. These objects are internally represented and initialized by a finite set of generators, which are matrices or equivalently matrix group elements. In our program the set of generator which is given to the constructor of the class consists of all elements of the stabilizer. There is a method of that class called `list()` which returns a complete list of elements, if the group is finite, and raises an exception otherwise. Therefore, we can compute quite simple the intersection of two such groups. In our program we implemented for this the function `intersectionofgroups(G1,G2)`, which takes the lists of elements, computes their intersection and returns the `MatrixGroup` generated by these elements, which is nothing else than the group consisting of exactly these elements in the intersection.

As we will see e.g. in Proposition 13.3.10 we have to find for certain pairs of cells the intersection of their stabilizers, to be able to compute cohomology groups. Therefore, we computed for all cells in the boundary of RedSquare the intersection of their stabilizer with $\Gamma_{\text{RedSquare}}$ (see Table 10.12).

We find an interesting coincidence between those types of cells in the boundary of RedSquare which have two kinds of orbits of different length and those types which have two different types of intersections (comp. Table 10.11). In some way this is not really surprising, since if under $\Gamma_{\text{RedSquare}}$ we have a shorter orbit one might guess that in these cases we could have a larger intersection, which is true. To compute the cohomology in degree one we need the intersections of the stabilizer of one chosen lift of a 4-cell, i.e. RedSquare, with stabilizers of cells in its boundary. Furthermore, we need the intersections of stabilizers of some representatives of 3-cells with stabilizers of at least one representative of each orbit under the stabilizer of that 3-cell to compute the cohomology in degree two and so forth. Therefore, we also computed the intersections of stabilizers of all other cells with stabilizers of cells in their boundary. Moreover, we also computed, because it was easier to do it with all already computed cells, the intersection of stabilizers of all cells of one type with the stabilizers of all cells of those types, which occur in its boundary. Now we find that in

Table 10.12.: *Intersections of the stabilizers for all cells in the closure of* RedSquare$_0$ *with the stabilizer* $\Gamma_{\text{RedSquare}_0}$ *of the standard 4-cell.*

Cell σ	**Group Structure**
Desargues$_i$ for $i = 0, \dots, 47$	C_2
Reye$_i$ for $i = 0, 1, 18, 19$	D_8
Reye$_i$ for $i \neq 0, 1, 18, 19$	$C_2 \times C_2$
DR$_i$ for $i = 0, \dots, 175$	C_2
RR$_i$ for $i = 0, \dots, 39$	$C_2 \times C_2$
Triangle$_i$ for $i = 0, \dots, 111$	C_2
Square$_i$ for $i = 8, 9, 14, 15, 16, 17, 18, 19, 30, 31, 32, 33, 52, 53, 54, 55$	$C_2 \times C_2$
Square$_i$ for the other cases	C_2
Hexagon$_i$ for $i = 0, \dots, 3$	D_8
Vertebra$_i$ for $i = 0, \dots, 3$	$C_4 \times C_2$
Crystal$_i$ for $i = 0, \dots, 3$	D_8
Pyramid$_i$ for $i = 0, \dots, 3, 12, \dots, 15, 20, \dots, 25, 30, 31$	C_2
Pyramid$_i$ for the other cases	$C_2 \times C_2$

most cases these intersections of stabilizers of cells which do not lie in the boundary of a cell have a trivial intersection, i.e. the group is isomorphic to C_2, but in some case we also find larger groups as e.g. some which are isomorphic to $C_2 \times C_2$. This is not necessary to know for the computation of cohomology, but give some deeper insight into the structure of cells in $\mathbb{W}$. A list with the some more intersections of stabilizers can be found in the file `intersections.pdf` which is include in the CD-ROM, which is included in the printed copy of this thesis.

Part III.

Cohomology

11. Γ-Modules and their Computation

The coefficient systems of the cohomology groups we want to compute are given by Γ-modules and systems of sheaves associated to them (see Section 12.1). In this chapter we describe the construction and computation of the class of Γ-modules which we use later as coefficients. The techniques used in this context are more or less standard techniques in representation theory. The problems raise up if we are going into details of the computation of Γ-modules. It is quite easy to compute highest weight modules for a simple or semi-simple Lie algebra $\mathfrak{g}$ and thus, by the results stated in Section 6.4.4, irreducible $\mathfrak{g}$-modules. Unfortunately, the transfer from Lie algebras to Lie groups is given by the exponential map, which is only locally a homomorphism. Therefore, we have to decompose our problem into parts which have a good local description (see Section 11.1).

11.1. Construction of Γ-modules

We introduced in Section 6.1.2 Γ-modules for some groups $\Gamma \subset \mathbf{G}$ and in Section 6.4.4 $\mathfrak{g}$-modules for a Lie algebra $\mathfrak{g} = \mathbf{Lie}(\mathbf{G})$. Using the Poincaré-Birkhoff-Witt theorem (see Theorem 6.4.2) one can quite easily state bases for $\mathfrak{g}$-modules associated to some highest weight λ. These $\mathfrak{g}$-modules are highest weight modules for some representations of Lie algebras. Using PBW-bases we can define the action of Lie algebra elements on these $\mathfrak{g}$-modules in terms of matrices, which realizes the Lie algebra representation (comp. Sections 6.4.3 and 6.4.5). We need a description of Γ-modules and not of $\mathfrak{g}$-modules. Therefore, we have to make out of such a $\mathfrak{g}$-module the associated Γ-module. To describe a Γ-module we need to describe representations of the group $\mathbf{G}$ or a discrete subgroup $\Gamma \subset \mathbf{G}$. We will describe in the following the connection between Lie algebra representations and representations for the associated Lie group before we are going more into the details of how to compute them.

Let $\mathfrak{g} = \mathbf{Lie}(\mathbf{G})$ the (simple) Lie algebra for $\mathbf{G}$, where $\mathbf{G}$ is a (real) Lie group. We assume that $\mathfrak{g}$ is realized as Lie subalgbra of the matrix algebra $\mathfrak{gl}_d(\mathbb{R})$ via the standard representation

$$\rho_{\mathfrak{g},0} : \mathfrak{g} \hookrightarrow \mathfrak{gl}(\mathbb{V}_{\mathbb{R}}) = \mathfrak{gl}_d(\mathbb{R}) \supset \mathfrak{g} = \mathbf{im}\,(\rho_{\mathfrak{g},0}) \tag{11.1}$$

of $\mathfrak{g}$. For us $\mathbf{G}$ will be $\mathbf{Sp}_2(\mathbb{R})$ and $\mathfrak{g} = \mathfrak{sp}_2$. Let further Γ be an arithmetic subgroup of $\mathbf{G}$. For us Γ will be the Siegel modular group or some subgroup of it. The construction in this section will work for an arbitrary simple Lie algebra and is not restricted to the symplectic Lie algebra. Therefore, we will formulate the results also in this generality.

As already mentioned above there is a strong relation between a Lie group $\mathbf{G}$ and its associated Lie algebra $\mathfrak{g} = \mathbf{Lie}(\mathbf{G})$ (see (6.22)).

$$\mathfrak{g} = \mathbf{Lie}(\mathbf{G}) = \{X \in \mathrm{Mat}_d(\mathbb{R}) = \mathfrak{gl}_d(\mathbb{R}) : \exp(t \cdot X) \in \mathbf{G}\,\forall t \in \mathbb{R}\}.$$

This connection can be used to make out of a Lie algebra representation a Lie group representation. This works as follows: Let $\rho_{\mathfrak{g}} : \mathfrak{g} \to \mathfrak{gl}(\mathbb{M} \otimes_{\mathbb{Z}} \mathbb{R}) \cong \mathfrak{gl}_n(\mathbb{R})$ be a Lie algebra representation with some $\mathfrak{g}$-module $\mathbb{M}$ of rank n, e.g. $\mathbb{M}$ is a highest weight module for some weight λ defined over the integers.[1] We define a representation of $\mathbf{G}$

$$\rho_{\mathbf{G}} : \mathbf{G} \to \mathbf{GL}(\mathbb{M} \otimes_{\mathbb{Z}} \mathbb{R}) \tag{11.2}$$

such that the action of Lie algebra elements is compatible with the action of Lie group elements via the exponential map. This means that the diagram

$$\begin{tikzcd}
\mathfrak{g} \arrow[r, "\exp"] \arrow[d, "\rho_{\mathfrak{g}}"'] & \mathbf{G} \arrow[d, "\rho_{\mathbf{G}}"] \\
\mathfrak{gl}(\mathbb{M} \otimes_{\mathbb{Z}} \mathbb{R}) \arrow[r, "\exp"'] & \mathbf{GL}(\mathbb{M} \otimes_{\mathbb{Z}} \mathbb{R})
\end{tikzcd} \tag{11.3}$$

should be commutative. This suggests – if we could invert the exponential map – to set

$$\rho_{\mathbf{G}}(\gamma) := \exp(\rho_{\mathfrak{g}}(\exp^{-1}(\gamma))) \tag{11.4}$$

for all $\gamma \in \mathbf{G}$. There is one problem: The exponential map is only locally an isomorphism onto its image and need not to be – and is not – globally even a homomorphism from $\mathfrak{g}$ to $\mathbf{G}$ (see Section 6.4.1). This means that we cannot compute the exponential map by evaluation of the exponential series globally for arbitrary elements $X \in \mathfrak{g}$. However we can do this locally. Thus, we need to decompose any element $\gamma \in \mathbf{G}$ into a product of elements for which we know that they are images of Lie algebra elements and what their preimages are.

According to (6.48) we have the root space decomposition

$$\mathfrak{g} = \mathfrak{n}^- \oplus \mathfrak{h} \oplus \mathfrak{n}^+,$$

where $\mathfrak{n}^\pm = \bigoplus_{\alpha\in\Phi^\pm} \mathfrak{g}_\alpha$ and all $\mathfrak{g}_\alpha$ for $\alpha \in \Phi$ are one-dimensional. Furthermore, we know from Sections 6.4.2 and 6.4.3 that the root elements $\mathbf{x}_\alpha$ and $\mathbf{y}_\alpha$ for $\alpha \in \Phi^+$ generate $\mathfrak{g}$ as Lie algebra and are as matrices nilpotent and hence we know that the matrices $\exp(t \cdot \mathbf{n}_\alpha)$ are unipotent elements in $\mathbf{G}$ for all $t \in \mathbb{R}$, which also generate $\mathbf{G}$. To simplify the notation we introduced here for $\alpha \in \Phi$

$$\mathbf{n}_\alpha := \begin{cases} \mathbf{x}_\alpha & \text{if } \alpha \in \Phi^+ \\ \mathbf{y}_{-\alpha} & \text{if } \alpha \in \Phi^-. \end{cases} \tag{11.5}$$

The exponential map $\exp(t \cdot \mathbf{n}_\alpha)$ is well defined as homomorphism for all $t \in \mathbb{R}$ and $\alpha \in \Phi$ and can be computed by evaluating the exponential series, which is here just a polynomial because the $\mathbf{n}_\alpha$ are nilpotent. This was shown for the case $\mathfrak{g} = \mathfrak{sp}_g$ in Sections 6.4.3 and 6.4.5, where we explicitly constructed the corresponding matrix representations. For other simple Lie algebras $\mathfrak{g}$ we refer to the literature (see e.g. [GW09; 2.4.1]). Hence, we find for each $\gamma \in \mathbf{G}$ a decomposition

$$\gamma = \prod_i \exp(t_i \cdot \mathbf{n}_{\beta_i}) \tag{11.6}$$

[1] By the results of Kostant (see e.g. Theorem 6.4.3) all structure constants are integral and thus we can assume that there is such a module defined over the integers.

where $t_i \in \mathbb{R}$ and $\beta_i \in \Phi$. This is exactly the decomposition into unipotent elements stated in Theorem 6.3.3. We present in Section 11.2.3 an algorithm to compute this decomposition for a given simple Lie algebra $\mathfrak{g}$ explicitly, which works in particular for integral elements in $\mathfrak{g}$. In that section we present the details in case of $\mathfrak{g} = \mathfrak{sp}_g$, which is the only one we need. However the implementation works also fine for other families of simple Lie algebras in characteristic zero, which are not relevant for the considerations in this thesis.

We can now apply successively – since the operations defined in (11.3) are associative – the inverse map $\exp^{-1}$ to the factors $\exp(t_i \cdot \mathbf{n}_{\beta_i})$, because on each one-parameter subgroup $\{\exp(t \cdot \mathbf{n}_\beta) : t \in \mathbb{R}\}$ the exponential map is a local isomorphism. To each element

$$\exp^{-1}(\exp(t_i \cdot \mathbf{n}_{\beta_i})) = t_i \cdot \mathbf{n}_{\beta_i}$$

we can now apply the Lie algebra representation $\rho_\mathfrak{g}$. The resulting elements

$$\rho_\mathfrak{g}(t_i \cdot \mathbf{n}_{\beta_i}) = t_i \cdot \rho_\mathfrak{g}(\mathbf{n}_{\beta_i})$$

are again nilpotent since the representation $\rho_\mathfrak{g}$ is a homomorphism. Thus, we can apply the exponential map and we get for each factor a unipotent element in $\mathbf{GL}(\mathbb{M} \otimes_\mathbb{Z} \mathbb{R})$. Using the associativity of the action we can multiply all transferred factors together and get the representation we are looking for. We summarize:

Theorem 11.1.1: *Let $\mathfrak{g} = \mathbf{Lie}\,(\mathbf{G})$ be a simple Lie algebra associated to a Lie group $\mathbf{G}$ with a fixed system of positive roots Φ^+ as above. Let further $\Gamma \subset \mathbf{G}$ be an arithmetic subgroup and $(\rho_\mathfrak{g}, \mathbb{M})$ a Lie algebra representation of $\mathfrak{g}$ for some $\mathfrak{g}$-module $\mathbb{M}$ defined over the integers. Then there is an associated group representation*

$$\rho_\mathbf{G} : \mathbf{G} \to \mathbf{GL}(\mathbb{M} \otimes_\mathbb{Z} \mathbf{R})$$

such that Diagram 11.3 commutes. Let $\gamma \in \mathbf{G}$ be a group element, then the representation $\rho_\mathbf{G}$ is given by

$$\rho_\mathbf{G}(\gamma) = \prod_i \exp(t_i \cdot \rho_\mathfrak{g}(\mathbf{n}_{\beta_i})),$$

if

$$\gamma = \prod_i \exp(t_i \cdot \mathbf{n}_{\beta_i})$$

is a decomposition into finitely many factors with $t_i \in \mathbb{R}$ and $\beta_i \in \Phi^+ \sqcup -\Phi^+$. This defines a $\mathbf{G}$-structure on $\mathbb{M} \otimes_\mathbb{Z} \mathbb{R}$ and therefore also a Γ-$\mathbb{R}$-module structure on $\mathbb{M} \otimes_\mathbb{Z} \mathbb{R}$.

Obviously we can obtain a representation by restriction of $\rho_\mathbf{G}$ to a subgroup $\Gamma \subset \mathbf{G}$. The interesting point is that according to Theorem 6.4.3 our chosen bases have an integral structure which is compatible with the exponential map (comp. [Kos66]). The normalization made in (6.50) is made in a way that integral elements in $\mathfrak{g}$, i.e. elements in

$$\mathfrak{g}_\mathbb{Z} = \mathfrak{g} \cap \mathfrak{gl}(\mathbb{V}) \tag{11.7}$$

are mapped onto integral elements in the group, i.e. $\mathscr{G}(\mathbb{Z}) := \mathbf{G} \cap \mathbf{GL}(\mathbb{V})$ if $\mathscr{G}/\mathbb{Q}$ is the underlying algebraic group to $\mathbf{G} = \mathscr{G}(\mathbb{R})$. Hence, we find that the representation $\rho_\mathbf{G}$ is an algebraic representation which is already defined over the integers. Therefore, the construction in Theorem 11.1.1 descends completely to the level of algebraic groups which are defined over the integers.

Therefore, we can in particular associate to each highest weight representation $(\rho_{\lambda,\mathfrak{g}}, \mathbb{M}_\lambda)$ of $\mathfrak{g} = \mathfrak{sp}_g$ of weight λ a unique group representation $(\rho_{\lambda,\Gamma^g}, \mathbb{M}_\lambda)$ of $\Gamma^g \subset \mathbf{Sp}_g(\mathbb{Z})$ and make $\mathbb{M}_\lambda$ into a Γ-module. We will describe in the following sections how to get a basis of a highest weight module of given weight and how to compute the group action on this module.

11.2. Computation of Γ-modules

The algorithm to compute the basis of highest weight modules and the action of Lie algebra elements uses Kostant's construction [Kos66] of lattice bases for Lie algebras an the PBW-basis [Poi00, Bir37, Wit37] for the universal enveloping algebra of $\mathfrak{sp}_2$ (see Sections 6.4.3 and 6.4.4). For the GAP-routines on Lie algebras we use one finds some short description in the GAP Reference Manual [The08; Chap. 60 and 61].

We describe in this section how to adapt and extend the existing routines for Lie algebras and their actions to get a SAGE-package which allows us to compute bases for highest weight modules and other Γ-modules as well as the actions of Lie group elements on them. For our purpose we use the existing routines in GAP and the additional GAP-package `QuaGroup`. This GAP-package was developed by Willem A. de Graaf in 2002 to extend the existing GAP-routines for Lie algebras to quantum Lie algebras and reimplements also some existing modules much more efficient than in the original version. The ability to deal with quantum Lie algebras is less important for us. However it offers also the ability to compute efficiently the action of (quantum) Lie algebra elements on certain modules which is not possible in GAP without `QuaGroup`-package. We use this to compute the actions of the related Lie Group on the weight modules and later to compute their invariants under these groups.

We describe as much of the internal data structure which is used in GAP to represent weight modules over semi simple Lie algebras, elements in them, and elements in the universal enveloping algebra as is necessary to understand our algorithms (see [The08; §61.13]). These structures are essentially the same than in the `QuaGroup` package. In quantum Lie algebras there is an additional parameter `_q` the so called quantum parameter. One gets classical Lie algebras by setting quantum-parameter `_q:=1`. In other words, the classical Lie algebras are only a special case this much more general framework.

All algorithms are defined over a field of characteristic zero. But we know by the result of Kostant [Kos66] that the used objects are indeed already in the integral lattice of the Lie algebra $\mathfrak{sp}_2$ or of the universal enveloping algebra $\mathfrak{U}(\mathfrak{sp}_2)$ (comp. Theorem 6.4.3).

For us a Γ-module $\mathbb{M}$ will always be an object given by the following data:

- A Cartan type of a simple Lie algebra defined over a field of characteristic zero, which is chosen to be $\mathbb{Q}$. Usually the Lie algebra will be of type C_2. The algorithm also works for other Cartan types like D_{12} or B_7.
- A weight $\lambda = \lambda_1 \cdot \omega_1 + \ldots + \lambda_g \cdot \omega_g$, which will be given by a list of non-negative integers

[lambda_1,...lambda_g], i.e. the highest weights we consider are always dominant weights.

We will describe how to obtain in our SAGE-package, which uses in this part mostly functions from GAP, out of these data a PBW-basis of a highest weight module $\mathbb{M}_\lambda$ and the corresponding matrix representation for the symplectic group (or one of the other associated types of of Lie groups). Here we restrict ourselves to the functions which are actively used and refer for most functions in the background to the manual of our program in the Appendix in Chapter A. Some more information can also obtained from the parts of source code your will find in the Appendix in Chapter B.

11.2.1. Basic GAP-Objects used in Our Program

The basic objects in the program are Lie algebras and some derived objects. Simple Lie algebras are basic objects in GAP, which we wrap into SAGE. A simple Lie algebra is given in our package by the command `SimpleLieAlgebra(type,rank, base_ring=QQ)` where,

- `type` is a string equal to *"A"*, *"B"*, ..., *"G"*, i.e. one Cartan family.
- `rank` is an integer (≥ 1 for type *"A"*, ≥ 2 for type *"B"* or *"C"*, ≥ 4 for type *"D"*, equal to $6, 7$ or 8 for type *"E"*, equal to 4 for type *"F"*, or equal to 2 in case of type *"G"*) equal to the rank of the simple Lie algebra.
- `base_ring` is any SAGE ring, but some further parts work only if it is a field of characteristic zero. Thus, we set by default `base_ring=QQ`.[2]

The objects of type `SimpleLieAlgebra` carry lots of additional attributes, such as the associated root system, which can be obtained by the method `RootSystem()` and carries further attributes like its positive, negative, or simple roots. We do not go too much into detail at this place. Here is a short example output:

```
sage: L = SimpleLieAlgebra("C", 2); L
    Algebra( Rationals, [ v.1, v.2, v.3, v.4, v.5, v.6, v.7,
        v.8, v.9, v.10 ] )
sage: L.Dimension()
    10
sage: R = L.RootSystem()
    <root system of rank 2>
sage: R.PositiveRoots()
    [ [ 2, -1 ], [ -2, 2 ], [ 0, 1 ], [ 2, 0 ] ]
sage: L.DimensionOfHighestWeightModule([10,3])
    2090
```

The dimension of a highest weight module is computed using the Weyl character formula (6.56), which is efficiently implemented in GAP. The most important objects are integral lattice elements in the universal enveloping algebra $\mathfrak{U}(\mathfrak{g})$ for a simple Lie algebra $\mathfrak{g}$ which form

[2]In SAGE we have that `QQ` are the rational numbers, `RR` are the real numbers and `CC` are the complex numbers.

a basis according to the Theorems 6.4.2 and 6.4.3. There are two existing implementations which we can use during this computation, one in the standard library of GAP, the other in the quagroup-package of GAP. The last one is somewhat faster for most cases, but according to the authors there is no good heuristic in which cases. Nevertheless we decided to use the quagroup-package, also because in the standard GAP libraries there is no implementation of the algebra action we need later on. The usage of both implementations is identical. With our wrapper to SAGE we can get a list of these lattice elements by the function `latticegeneratorsinUEA(L)`, where `L` is a simple Lie algebra. Note: The function `latticegeneratorsinUEA(L)` returns an object of the SAGE-type `list`, which starts indexing with 0, whereas the class method `L.latticegeneratorsinUEA()` returns a GAP list, which starts indexing with 1, as can be seen in the following example:

```
sage: L=p.SimpleLieAlgebra("C",2)
sage: Ugen = latticegeneratorsinUEA(L); Ugen
    [y1, y2, y3, y4, x1, x2, x3, x4, ( h9/1 ), ( h10/1 )]
sage: Ugen[1]
    y2
sage: L.latticegeneratorsinUEA()[1]
    y1
```

The occurring elements `yi, xj` and `( hk/1 )` are exactly the elements in a Chevalley basis of `L` with the notation introduced in (6.50). We should note that the first g of the `yi` as well as of the `xj` (corresponding to the negative and positive roots, respectively) correspond to the chosen simple roots. The last g elements are the Cartan elements, which are given by the Lie bracket applied to the elements corresponding to simple roots. With the elements in `Ugen` we can perform additions and scalar multiplications, and we can multiply them with an other element out of `Ugen`. The latter multiplication is the multiplication in the universal enveloping algebra of `L`. For example, we can calculate:

```
sage: Ugen[1]*Ugen[1]
   2*y2^(2)
sage: 6*Ugen[4]*Ugen[2]
   12*y2+6*y3*x1
```

The results are reduced and represented in the PBW-basis of the module with the normalization from (6.50). In the used GAP-implementation the reduction uses Gröbner bases for the monomials in the PBW-basis of the universal enveloping algebra. Since Gröbner bases are efficiently implemented in GAP (see [The08; §64.17]), this allows a fast reduction. Unfortunately the part on computation of PBW-bases is badly documented in the GAP-references as well for Lie algebras as for the quagroup-package. Therefore, one has to go into the source code of the used functions to get a glimpse of how it works in detail.

GAP contains highest weight modules as native objects, which we wrapped into our SAGE-program as the command `HighestWeightModule(L, weight)`, where `L` is a simple Lie algebra and weight is – as mentioned above – a list of non-negative integers. Highest weight modules are given by a PBW-basis, which can be obtained using Theorem 6.4.6. To return this basis (as GAP list) there is the class function `Basis()`.

For some reason elements in highest weight modules are in GAP represented as objects in the class `LeftAlgebraModuleElement` and in particular not as elements in the also existing class `WeightRepElement`. However, if `v` is an element of a highest weight module, then `ExtRepOfObj(v)` is a `WeightRepElement`. The constructor of elements in the class `HighestWeightModule` inherits from the element constructor in the class `VectorSpace` in GAP. In the package `quagroup` there is the possibility to apply an element `u` in the object `latticegeneratorsinUEA(L)` to a `WeightRepElement` `b`, which uses overloading of the GAP-operator `POW(u,b)/u^b`. Therefore, we have to convert an element in a highest weight module with `ExtRepOfObj()` to a weight representation element before we can apply a universal enveloping algebra element given in terms of lattice elements in the universal enveloping algebra.

A `WeightRepElement` is represented in GAP by a list of the form `[v1,c1,v2,c2, ...]`, where the `vi` are basis vectors, and the `ci` coefficients. Furthermore, a basis vector `v` in this list is a weight vector, which is represented by a list of the form `[k,mon,wt]`, where `k` is an integer, which indicates the index of the basis element (Note: Here GAP indexing is used, which starts with 1), `mon` is an `UEALatticeElement`, which is chosen such that the result of applying `mon` to a fixed highest weight vector `v0` is `v`, and `wt` is the weight of `v`. A `WeightRepElement` `v` is printed as `mon*v0`, while the command `ExtRepOfObj(v)` returns the corresponding list as described above. Here is an example for the usage of the mentioned objects:

```
sage: L = SimpleLieAlgebra("C", 2)
sage: M = HighestWeightModule(L,[2,1]);  M
    <35-dimensional left-module over Algebra( Rationals,
    [ v.1, v.2, v.3, v.4, v.5, v.6, v.7, v.8, v.9, v.10 ] )>
sage: B = M.Basis(); B
    Basis( <35-dimensional left-module over Algebra(
    Rationals, [ v.1, v.2, v.3, v.4, v.5, v.6, v.7, v.8, v.9,
    v.10 ] )>,
      [ 1*v0, y1*v0, y2*v0, y1*y2*v0, y3*v0, y1^(2)*v0,
      y1*y3*v0, y1^(2)*y2*v0, y4*v0, y2*y3*v0, y1*y4*v0,
      y1^(2)*y3*v0, y1*y2*y3*v0, y2*y4*v0, y3^(2)*v0,
      y1*y2*y4*v0, y1*y3^(2)*v0, y3*y4*v0, y1^(2)*y4*v0,
      y2*y3^(2)*v0, y1*y3*y4*v0, y1^(2)*y2*y4*v0, y4^(2)*v0,
      y2*y3*y4*v0, y3^(3)*v0, y1*y4^(2)*v0, y1*y2*y3*y4*v0,
      y2*y4^(2)*v0, y3^(2)*y4*v0, y1*y2*y4^(2)*v0,
      y3*y4^(2)*v0, y2*y3^(2)*y4*v0, y4^(3)*v0,
      y2*y3*y4^(2)*v0, y2*y4^(3)*v0 ] )
sage: Ugen = latticegeneratorsinUEA (L);
sage: b=gap.ExtRepOfObj(2*B[6]+B[9]); b
    2*y1^(2)*v0+y4*v0
sage: gap.ExtRepOfObj(b)
    [ [ 6, y1^(2), [ -2, 3 ] ], 2, [ 9, y4, [ 0, 1 ] ], 1 ]
sage: Ugen[6]^b               #Ugen[6]=x3
    -1*y1*v0
sage: (Ugen[1]+4*Ugen[2])^b)  #Ugen[1]+4*Ugen[2]=y2+4*y3
    -2*y1*y3*v0+2*y1^(2)*y2*v0+2*y4*v0+-16*y1*y4*v0
    +8*y1^(2)*y3*v0+y2*y4*v0+4*y3*y4*v0
```

This completes the list of objects we use in our own program part which we present in the next section.

11.2.2. Computing Highest Weight Modules

Let in this section Γ be an arithmetic subgroup of $\mathbf{G} = \mathscr{G}(\mathbb{R})$, whose associated simple Lie algebra is $\mathfrak{g} = \mathbf{Lie}\,(\mathbf{G})$. The examples for Γ-modules we consider are irreducible Γ-modules which are given by a highest weight module $\mathbb{M}_\lambda$ for some dominant weight λ together with an action of the group Γ. As module $\mathbb{M}_\lambda$ is the highest weight module given by the simple Lie algebra $\mathfrak{g}$ together with some $\lambda \in \Lambda^+$. A basis of this module can be obtained using the existing class method `Basis()` for a highest weight module as described in the previous section. Thus, it remains to implement an action of the group Γ on this module. This will be done in three steps:

Step 1: Compute the matrix representation $\rho_\mathfrak{g}(u)$ for a lattice generator u in the universal enveloping algebra $\mathfrak{U}(\mathfrak{g})$ on a highest weight module $\mathbb{M}_\lambda$ equipped with the PBW-basis given by the class method `Basis()`.

Step 2: Decompose a given element $\gamma \in \Gamma$ according to Theorem 11.1.1 into unipotent elements in $\mathbf{G}$, which are given by images $\exp(\rho_\mathfrak{g}(u))$. We postpone the detailed description of this step to Section 11.2.3.

Step 3: Combine the results of the two previous steps to a matrix representation of γ on the module $\mathbb{M}_\lambda$.

The first step is elementary linear algebra of the first year undergraduate lectures. The columns of the matrices are given as the images of the elements of the PBW-basis elements. Therefore, it is easy to get the matrix representation for Lie algebra elements or, to be precise, of their images under the natural inclusion into the universal enveloping algebra. This can be done for any $\mathfrak{g}$-module which is equipped with a PBW-basis, and not only for (irreducible) highest weight modules. We computed for all in this thesis occurring their PBW-bases. They can be found in the file `modules.pdf` on the CD-ROM, which is include in the printed copy of this thesis. In SAGE Source Code B.9 we state the function which computes the image of an arbitrary universal enveloping algebra element under a given highest weight representation.

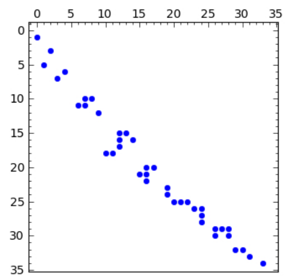
By default the returned matrix is a sparse matrix in SAGE. Here there is one reason to use GAP via a wrapper in SAGE, because GAP was not made for efficient linear algebra and has in particular no standard support for sparse matrices.[3] The resulting matrices for all root vectors are (very) sparse (see Figures 11.1 and 11.2).

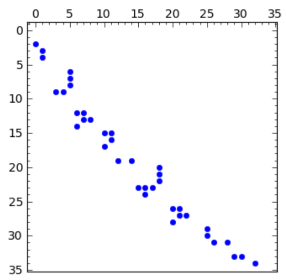
Although it is not very time consuming – compared with the total runtime of our program – to compute the matrices for a given weight[4], we will pass a precomputed list containing them to most of the following functions.

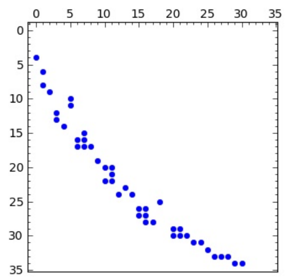
[3]There is an additional package `gauss` which offers some basic support of sparse matrices, but we do recommend to use the more advanced Python libraries used in SAGE [Tea11b; §41.11, §41.13, and §41.15].

[4]For example, to compute one of these matrices for $\mathbb{M}_{1,2}$ one needs 328ms, whereas in case of $\mathbb{M}_{8,6}$ one needs 54.5s.

Figure 11.1.: *Shape of matrices given by* `actionmatrixofUEAelement` *for negative root vectors on* $\mathbb{M}_{1,2}$*, which has dimension* 35*. Non-zero entries are indicated by dots.*


(a) $\mathbf{y}_\alpha$


(b) $\mathbf{y}_\beta$


(c) $\mathbf{y}_{\alpha+\beta}$

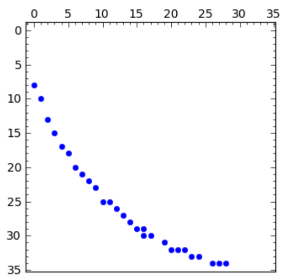
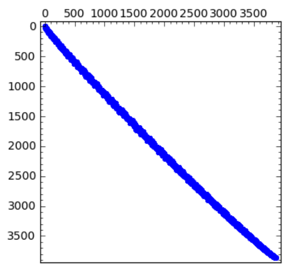
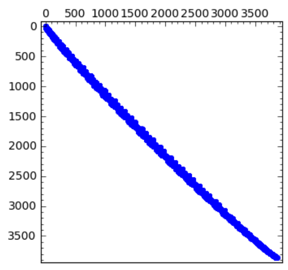
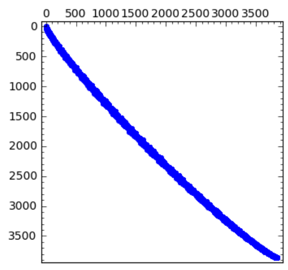
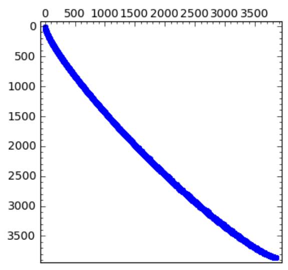

(d) $\mathbf{y}_{2\alpha+\beta}$

Figure 11.2.: *Shape of matrices given by* `actionmatrixofUEAelement` *for negative root vectors on* $\mathbb{M}_{8,6}$*, which has dimension* 3864*. Non-zero entries are indicated by dots.*


(a) $\mathbf{y}_\alpha$


(b) $\mathbf{y}_\beta$


(c) $\mathbf{y}_{\alpha+\beta}$


(d) $\mathbf{y}_{2\alpha+\beta}$

To get the image in $\mathbf{G}$ one has to apply the exponential map to these matrices. It turned out that the evaluation using the existing function in SAGE is much slower than a naively implemented exponential map, which simply evaluates the exponential series until the monomial in the (nilpotent) matrix yields zero.

The solution for the second step needs some more space. Therefore, we decided to postpone the details of that to Section 11.2.3, where we derive the algorithm to obtain such a decomposition. For the moment we assume that for each $\gamma \in \Gamma$ there is a decomposition into unipotent matrices which are images of nilpotent elements $\mathbf{n}_{\beta_i}$ in $\mathfrak{g}$ of the form

$$\gamma = \prod_i \exp(t_i \cdot \mathbf{n}_{\beta_i}) \tag{11.8}$$

for some numbers t_i and roots $\beta_i \in \Phi^+$ according to Theorem 11.1.1. The list of pairs (t_i, β_i) can be obtained by a structured Gaussian elimination process (see Theorem 11.2.5). According to the remark made above it is an easy task to compute $\exp(t_i \cdot \mathbf{n}_{\beta_i})$, which we simply have to multiply to get the matrix $\rho_{\mathbf{G}}(\gamma)$. This completes the computation of the underlying group representation on a given highest weight module.

For non highest weight modules it is easy to get the corresponding matrices by decomposing the space into a direct sum, i.e. essentially a list, of irreducible highest weight representations. For this one can also use some build in GAP-function like `DirectSumOfAlgebraModules` [The08; §61.14]. How to handle this one has to decide case by case since the different methods of implementations differ depending on the particular circumstances very much in their runtime. The action of the group element on the whole module can obtained on each direct summand by the method described above. The resulting matrix is simply a block diagonal matrix with blocks equal to a matrix on a highest weight module. If no decomposition into irreducible highest weight modules is known, but a PBW-basis, then it is even simpler, because the variable `HWM` can be any left module for a simple Lie algebra. Thus, we can use the ability of GAP/SAGE to define a (weight-)module by fixing a basis (see e.g. [The08; §60]).

We implemented the function `getreprofgroupelement` in a way which allows to use some precomputed list of generators and their images under the exponential map[5], if they are available, to avoid to compute the same objects several times in our SAGE program. This does not make a big difference for the computation of $\rho_{\mathbf{G}}(\gamma)$ for a single element $\gamma \in \Gamma$, but matters much more if we have to use it for lots of group elements. The resulting function is quite simple as explained above (see SAGE Source Code B.10). Most of the difficulties are hidden in the function `getdecomposition`, which we will explain in the next section. After that we will give in Section 11.2.4 some detailed examples.

11.2.3. Computing a Group Element Decomposition

As explained in Section 11.1 we need to decompose a given Element $\gamma \in \Gamma$ into a product of unipotent elements. We stated in Theorem 6.3.3 that Γ is generated by unipotent elements.

[5]To be precise we do use not the generators itself. Instead we use lists of functions which map the parameter `T` to $\exp(\mathrm{T} \cdot \rho(\mathbf{n}_\alpha))$, where ρ is either the standard representation ρ_0 or some highest weight representation for the considered Lie algebra. This avoids the evaluation of the exponential map. But one need still to evaluate the functions, which is impossible to avoid, which is if the highest weight representation has larger ranks the most time consuming part of these algorithm (see Section 11.2.4).

It turns out that is possible to use some generic (unipotent) generators coming from the structure of the root system (comp. Section 6.4.4). We describe this decomposition this in a slightly more general framework than we need for our actual computations. This can be regarded as a kind of proof of Theorem 6.3.3. We concentrate on the case where $\gamma \in \mathbf{Sp}_g(\mathbf{R})$ where $\mathbf{R}$ is any euclidean ring, i.e. we are able to perform the euclidean algorithm in $\mathbf{R}$. For our application $\mathbf{R}$ will be a subring of $\mathbb{R}$, namely the integers $\mathbb{Z}$, but all algorithms work fine also in the more general context.

Some set of generators of the symplectic group $\mathbf{Sp}_g(\mathbf{R}^{2g}, J_g^{\mathsf{var}})$ and an algorithm to compute them is well known (see e.g. the Appendix of [Fre83]). But there are two problems: First, the algorithms decompose in upper triangular matrices and the matrix J_g^{var}, which defines the symplectic form, and second all described algorithms are given for a realization of the symplectic group which is different from the one we are using (comp. Section 6.3.1.3). The first point is not a real objection since one easily finds a decomposition of J_g^{var} into upper and lower triangular unipotent matrices. The second problem requires some more work. One possible solution is to use the decomposition algorithm from [Fre83] and to transfer each element via Lemma 6.3.4 to our standard description. But this solution is much more expensive than a direct computation of the decomposition into the right set of generators. In this section we present an algorithm we implemented in SAGE which solves this problem. The algorithm incorporates the GAP implementation for the computation of simple Lie algebras and gets the underlying Lie algebra as additional input to determine generic generators for the group, therefore our implementation works also fine for a decomposition of a group whose (simple) Lie algebra $\mathfrak{g}$ is known or computable. The algorithm we developed consists of two steps:

a) Compute a set unipotent generators (”generic generators”) or the group coming from the root space decomposition of of the Lie algebra $\mathfrak{g}$.

b) Compute the decomposition of a given matrix into the generic generators.

The algorithm we will give works also fine for groups arising from Lie algebras of Cartan type different from C_g. So for example it has been tested to decompose also group elements in $\mathbf{SL}_d(\mathbb{Z})$ for $d \leq 10$ into unipotent generators. The construction is – with minor changes – also valid for all other families of simple Lie algebras, but for simplicity we give the proof only in case of the symplectic groups $\mathbf{G} := \mathbf{Sp}_g(\mathbf{R})$, which corresponds to the Lie algebra $\mathfrak{g} = \mathfrak{sp}_g$ of rank g which is of type C_g, and not for the other types.

11.2.3.1. Computing the Generic Generators

Assume we fixed a Chevalley basis of $\mathfrak{g}$ with respect to a Cartan subalgebra $\mathfrak{h}$ coming from a suitable maximal torus and the (previously chosen) fixed system of simple roots (comp. Sections 6.4 and 11.2.2). Denote by Φ^+ the corresponding system of positive roots. Let $\mathbb{V}$ be the standard representation of $\mathfrak{g}$, which we consider as free $\mathfrak{g}$-$\mathbb{Z}$-module (see Section 6.4.4). According to Section 6.4.4 $\mathbb{V}$ is the highest weight module for the weight $\lambda = (1, 0, \ldots, 0)$. We know by Theorem 6.4.6 that a basis of a highest weight module and thus $\mathbb{V}$ is given by a highest weight vector together with a collection of monomials in the $\mathbf{y}_i$. In the case $g = 2$ the standard representation $\mathbb{V}$ is generated by $(\mathbf{v}_0, \mathbf{y}_1 \cdot \mathbf{v}_0, \mathbf{y}_3 \cdot \mathbf{v}_0, \mathbf{y}_4 \cdot \mathbf{v}_0)$ (comp. Section 6.4.5). Assume we fixed such a PBW-basis of $\mathbb{V}$ (with respect to $\mathfrak{g}$, a maximal Cartan algebra $\mathfrak{h}$, and a Chevalley basis of $\mathfrak{g}$). Now we will identify the Lie algebra $\mathfrak{g}$ with its image under the standard representation. In other words, we identify $\mathfrak{g}$ with the matrix group which acts on the module $\mathbb{V}$ equipped with the PBW-basis by matrix multiplication from the left. We call the matrices $\rho_0(X)$ for $X \in \{\mathbf{y}_\alpha, \mathbf{x}_\alpha\}$ for $\alpha \in \Phi^+$ the **generic Lie algebra generators** associated to a system Φ^+ of positive roots. As before, we identify the root vectors with their images under the standard representation ρ_0. Thus, the generic Lie algebra generators are the matrices $\mathbf{y}_i = \mathbf{y}_{\alpha_i}$ and $\mathbf{x}_i = \mathbf{x}_{\alpha_i}$. We already explained how to obtain these matrices for $\mathfrak{g} = \mathfrak{sp}_g$ in Section 6.4.5 (see in particular (6.34) and (6.37)). But also for general simple Lie algebras $\mathfrak{g}$ it is not difficult to obtain the matrix representations of the $\mathbf{y}_i$ and $\mathbf{x}_j$ (see e.g. [GW09; 2.4.1]) and from them together with Theorem 6.4.6 one easily gets a PBW-basis of the standard representation $\mathbb{V}$ of the simple Lie algebra $\mathfrak{g}$.

We can now apply the Lie algebra element written in terms the root vectors to one of the weight vectors simply by formal multiplication of expressions. After one has reduced the result by the relations from (6.49) we can write the result again in terms of the fixed basis, which by a standard argument from linear algebra gives the matrix we are looking for. The reduction is done quite efficiently using Gröbner bases as mentioned earlier in Section 11.2.1. This is exactly the same construction we did for general highest weight modules in Section 11.2.2. The matrices are, as predicted by the theory, nilpotent upper or lower triangular matrices, depending on whether the element corresponds to a positive or a negative root[6]. Therefore, the exponential map applied to them is a polynomial with non vanishing zeroe order term in the nilpotent matrices which is unipotent. Via the exponential map we can associate to each positive root $\alpha \in \Phi^+$ two one-parameter subgroups in $\mathbf{G}$ by

$$\begin{aligned} \alpha &\mapsto \mathbf{Y}_\alpha = \{\exp(T \cdot \mathbf{y}_\alpha) \text{ for } T \in \mathbf{R}\} \\ \alpha &\mapsto \mathbf{X}_\alpha = \{\exp(T \cdot \mathbf{x}_\alpha) \text{ for } T \in \mathbf{R}\} \end{aligned} \tag{11.9}$$

We identify the positive and negative root vectors with the nilpotent matrices representing them in the chosen basis and analogously the Lie group elements with the corresponding matrices. To simplify the notation we set for $T \in \mathbf{R}$

$$\mathbf{Y}_\alpha(T) = \exp(T \cdot \mathbf{y}_\alpha) \qquad \mathbf{X}_\alpha(T) = \exp(T \cdot \mathbf{x}_\alpha). \tag{11.10}$$

Remark (Important): For technical reasons we use in this section until further notice mostly the symplectic group with respect to the symplectic form defined by J'_g (comp. Section 6.3.1.3) because GAP and therefore also our program part which uses GAP in some parts uses for Lie algebras this modified symplectic form. If we use the standard symplectic form in this section we will will explicitly mention that. As stated in Lemma 6.3.4 we can easily transform the elements to our standard symplectic form.

In case of $g = 2$ and the basis $\mathfrak{B}'$ we have the following collection of elements in $\mathfrak{sp}_2$ and subgroups in $\mathbf{Sp}_2(\mathbf{R})$ (comp. (6.39) and Table 6.4 or (6.34) together with (6.37)):

[6]This depends on the chosen basis. In the basis used by GAP, i.e. the symplectic form is chosen to be given by the basis $\mathfrak{B}'$, negative roots corresponds to lower triangular matrices whereas upper triangular matrices correspond to positive roots. Unfortunately this does not stay true if we conjugate the matrices such that the result is symplectic with respect to the standard basis $\mathfrak{B}$. Also in this case we have only upper and lower triangular matrices but this property does not coincide with with the property to come from a positive or negative root.

$$\begin{aligned}
\mathbf{x}_1 &= \begin{pmatrix} 0&1&0&0\\ 0&0&0&0\\ 0&0&0&1\\ 0&0&0&0 \end{pmatrix} & \mathbf{X}_1 &= \left\{ \begin{pmatrix} 1&T&0&0\\ 0&1&0&0\\ 0&0&1&T\\ 0&0&0&1 \end{pmatrix} : T \in \mathbf{R} \right\} \\
\mathbf{x}_2 &= \begin{pmatrix} 0&0&0&0\\ 0&0&-1&0\\ 0&0&0&0\\ 0&0&0&0 \end{pmatrix} & \mathbf{X}_2 &= \left\{ \begin{pmatrix} 1&0&0&0\\ 0&1&-T&0\\ 0&0&1&0\\ 0&0&0&1 \end{pmatrix} : T \in \mathbf{R} \right\} \\
\mathbf{x}_3 &= \begin{pmatrix} 0&1&0&0\\ 0&0&0&0\\ 0&0&0&1\\ 0&0&0&0 \end{pmatrix} & \mathbf{X}_3 &= \left\{ \begin{pmatrix} 1&T&0&0\\ 0&1&0&0\\ 0&0&1&T\\ 0&0&0&1 \end{pmatrix} : T \in \mathbf{R} \right\} \\
\mathbf{x}_4 &= \begin{pmatrix} 0&0&0&1\\ 0&0&0&0\\ 0&0&0&0\\ 0&0&0&0 \end{pmatrix} & \mathbf{X}_4 &= \left\{ \begin{pmatrix} 1&0&0&T\\ 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1 \end{pmatrix} : T \in \mathbf{R} \right\} \\
\mathbf{y}_i &= \mathbf{x}_i^t & \mathbf{Y}_i &= \left\{ \mathbf{X}_i(T)^t : T \in \mathbf{R} \right\}
\end{aligned} \tag{11.11}$$

To get the corresponding elements with respect to our standard choice for the symplectic form given by J_g we have to conjugate all elements in the groups above with the matrix

$$m = \begin{pmatrix} 1&&&\\ &&1&\\ &1&&\\ &&&1 \end{pmatrix} \in \mathbf{GL}_4(\mathbb{Z}). \tag{11.12}$$

We get the following collection of one-parameter subgroups of $\mathbf{Sp}_2(\mathbf{R})$:

$$\begin{aligned}
\mathbf{X}_1 &= \left\{ \begin{pmatrix} 1&0&T&0\\ 0&1&0&T\\ 0&0&1&0\\ 0&0&0&1 \end{pmatrix} : T \in \mathbf{R} \right\} \\
\mathbf{X}_2 &= \left\{ \begin{pmatrix} 1&0&0&0\\ 0&1&0&0\\ 0&-T&1&0\\ 0&0&0&1 \end{pmatrix} : T \in \mathbf{R} \right\} \\
\mathbf{X}_3 &= \left\{ \begin{pmatrix} 1&T&0&0\\ 0&1&0&0\\ 0&0&1&-T\\ 0&0&0&1 \end{pmatrix} : T \in \mathbf{R} \right\} \\
\mathbf{X}_4 &= \left\{ \begin{pmatrix} 1&0&0&T\\ 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1 \end{pmatrix} : T \in \mathbf{R} \right\} \\
\mathbf{Y}_i &= \left\{ \mathbf{X}_i(T)^t : T \in \mathbf{R} \right\}.
\end{aligned} \tag{11.13}$$

In case of $g = 3$ we can give a similar list with 9 pairs of nilpotent elements in $\mathfrak{sp}_3$ which lead to the same number of unipotent one-parameter subgroups in $\mathbf{Sp}_3(\mathbf{R})$.

We prove in the rest of this section the following theorem and give an algorithm to get a decomposition for a given element $\gamma \in G$. Although we consider only the case of the symplectic groups we remember that the statement in the theorem as well as the following constructive proof is, with minor changes, which do not change the algorithms we use, in the proof also valid for the groups which are associated to the other types of simple Lie algebras.

Theorem 11.2.1: *The group $G = \mathbf{Sp}_g(\mathbf{R})$ is generated by the matrices $\mathbf{Y}_\alpha(T)$ and $\mathbf{X}_\alpha(T)$ with $T \in \mathbf{R}$ and $\alpha \in \Phi^+$.*

This theorem is a more precise version of Theorem 6.3.3 (comp. [GW09; Theor. 2.2.2]). It follows by induction from the fact that that $\mathbf{Y}_\alpha(T)$ and $\mathbf{X}_\alpha(T)$ generate for a single $\alpha \in \Phi^+$

a subgroup of G which is isomorphic to $\mathbf{SL}_2(\mathbf{R})$ and the fact that the symplectic form splits compatible with this embedding (comp. [GW09; Sections 2.2.1 and 2.2.2]).

We will give here a construction to decompose an arbitrary element $\gamma \in G$ into generators of the form $\mathbf{Y}_\alpha(T)$ and $\mathbf{X}_\alpha(T)$, where $\alpha \in \Phi^+$ and $T \in \mathbf{R}$. This construction uses a kind of structured Gaussian elimination together with a variation of the extended euclidean algorithm. An different version of this method can be found e.g. in [Fre83; Anhang V.], where it is used to prove an analogous statement. For the our construction we use induction over rank g of the Lie algebra $\mathfrak{g}$. We start with $g = 1$.

For $g = 1$ we know that $G = \mathbf{Sp}_1 = \mathbf{SL}_2$. It is commonly known that the matrix group $\mathbf{SL}_2(\mathbf{R})$ for a euclidean ring $\mathbf{R}$ is generated by the elements

$$S = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \qquad\qquad T(x) = \begin{pmatrix} 1 & x \\ 0 & 1 \end{pmatrix} \tag{11.14}$$

with $x \in \mathbf{R}$. A proof can be found in the standard literature, see e.g. [KK07; Kap. II. §2]. There is only one positive root α because the root system has rank one and we immediately find that $T(x) = \mathbf{X}_\alpha(x)$. Furthermore, we have

$$S = \mathbf{X}_\alpha(1) \cdot \mathbf{Y}_\alpha(-1) \cdot \mathbf{X}_\alpha(1). \tag{11.15}$$

Thus, we can write the generators in the asserted form. For the decomposition of a given two by two matrix into generators we refer to the description how to do this in larger dimension (see page 118 ff.), which applies exactly in the same way.

Let us now consider $g \geq 2$. We need some simple transformations. Consider

$$\begin{aligned} \mathbf{Z}_\alpha :=& \;\;\mathbf{Y}_\alpha(1) \cdot \mathbf{X}_\alpha(-1) \cdot \mathbf{Y}_\alpha(1) \\ =& -\mathbf{Y}_\alpha(-1) \cdot \mathbf{X}_\alpha(1) \cdot \mathbf{Y}_\alpha(-1) \\ =& \;\;\mathbf{X}_\alpha(-1) \cdot \mathbf{Y}_\alpha(1) \cdot \mathbf{X}_\alpha(-1) \\ =& -\mathbf{Z}_\alpha^t \\ =& \;\;\mathbf{Z}_\alpha^{-1}. \end{aligned} \tag{11.16}$$

We can form a group from these elements, $\Pi_\mathfrak{g} := \langle \mathbf{Z}_\alpha : \alpha \in \Phi \rangle \subset \mathbf{Sp}_g(\mathbb{Z})$. We will have a closer look at these transformations, which we call **symplectic permutations**. According to (6.34) and (6.37) we have three different shapes of root vectors $\mathbf{x}_\alpha$, depending on which form the root α has. Let $1 \leq i < j < g$. Then we have the following three cases:

Case 1 ($\alpha = \varepsilon_i - \varepsilon_j$): Here we have $\mathbf{x}_{\varepsilon_i - \varepsilon_j} = \mathbf{E}_{i,j} - \mathbf{E}_{2g+1-i,2g+1-j}$. One easily checks that $\mathbf{E}_{i,j} \cdot \mathbf{E}_{k,l} = \delta_{j,k}\mathbf{E}_{i,l}$. Thus, we get

$$\begin{aligned} \mathbf{X}_{\varepsilon_i-\varepsilon_j}(T) &= \mathrm{Id}_{2g} + T \cdot \mathbf{E}_{i,j} - T \cdot \mathbf{E}_{2g+1-i,2g+1-j} \\ \mathbf{Y}_{\varepsilon_i-\varepsilon_j}(T) &= \mathrm{Id}_{2g} + T \cdot \mathbf{E}_{j,i} - T \cdot \mathbf{E}_{2g+1-j,2g+1-i} \end{aligned} \tag{11.17}$$

for $T \in \mathbf{R}$. It is an easy exercise to show that

$$\begin{aligned} \mathbf{Z}_{\varepsilon_i-\varepsilon_j} &= \mathbf{X}_{\varepsilon_i-\varepsilon_j}(-1) \cdot \mathbf{Y}_{\varepsilon_i-\varepsilon_j}(1) \cdot \mathbf{X}_{\varepsilon_i-\varepsilon_j}(-1) \\ &= \mathrm{Id}_{2g} - \mathbf{E}_{j,j} - \mathbf{E}_{i,i} - \mathbf{E}_{2g+1-i,2g+1-i} - \mathbf{E}_{2g+1-j,2g+1-j} \\ &\quad \underbrace{-\mathbf{E}_{i,j} + \mathbf{E}_{j,i}}_{i \leftrightarrow j} + \underbrace{\mathbf{E}_{2g+1-i,2g+1-j} - \mathbf{E}_{2g+1-j,2g+1-i}}_{2g+1-i \leftrightarrow 2g+1-j} \end{aligned} \tag{11.18}$$

During the computation it turns out that all $\delta_{r,s}$ which are not one in a trivial way vanish, because $1 \le i < j \le g$. This computation shows that $\mathbf{Z}_{\varepsilon_i - \varepsilon_j}$ permutes (up to sign) the indices i and j as well as the indices $2g+1-i$ and $2g+1-j$ and thus $\mathbf{Z}_{\pm(\varepsilon_i-\varepsilon_j)} \in \mathbb{S}_{2g} \times \{\pm 1\}^{2g}$.

Case 2 ($\alpha = \varepsilon_i + \varepsilon_j$): This case is completely analogous to the previous one. One only has to use $\mathbf{x}_{\varepsilon_i+\varepsilon_j} = \mathbf{E}_{i,2g+1-j} + \mathbf{E}_{j,2g+1-i}$ for $1 \le i < j \le g$ which leads to

$$\begin{aligned} \mathbf{X}_{\varepsilon_i+\varepsilon_j}(T) &= \mathrm{Id}_{2g} + T \cdot \mathbf{E}_{i,2g+1-j} + T \cdot \mathbf{E}_{j,2g+1-i} \\ \mathbf{Y}_{\varepsilon_i+\varepsilon_j}(T) &= \mathrm{Id}_{2g} + T \cdot \mathbf{E}_{2g+1-j,i} + T \cdot \mathbf{E}_{2g+1-i,j} \end{aligned} \tag{11.19}$$

for $T \in \mathbf{R}$ and hence we get

$$\begin{aligned} \mathbf{Z}_{\varepsilon_i+\varepsilon_j} = \mathrm{Id}_{2g} &- \mathbf{E}_{i,i} - \mathbf{E}_{j,j} - \mathbf{E}_{2g+1-i,2g+1-i} - \mathbf{E}_{2g+1-j,2g+1-j} \\ &+ \underbrace{\mathbf{E}_{2g+1-j,i} - \mathbf{E}_{i,2g+1-j}}_{i \leftrightarrow 2g+1-j} + \underbrace{\mathbf{E}_{2g+1-i,j} - \mathbf{E}_{j,2g+1-i}}_{j \leftrightarrow 2g+1-i}, \end{aligned} \tag{11.20}$$

which is a permutation (up to sign) of the indices i and $2g+1-j$ and of the indices j and $2g+1-i$. Thus, we have again $\mathbf{Z}_{\pm(\varepsilon_i+\varepsilon_j)} \in \mathbb{S}_{2g} \times \{\pm 1\}^{2g}$.

Case 3 ($\alpha = 2\varepsilon_i$): This is in some sense – there is a factor 2 – a special case of the previous one. Here we have $\mathbf{x}_{2\varepsilon_i} = \mathbf{E}_{2g+1-i,i}$ and hence

$$\mathbf{X}_{2\varepsilon_i}(T) = \mathrm{Id}_{2g} + T \cdot \mathbf{E}_{2g+1-i,i} \tag{11.21}$$

for $T \in \mathbf{R}$. Thus, we get

$$\mathbf{Z}_{2\varepsilon_i} = \mathrm{Id}_{2g} + \underbrace{\mathbf{E}_{2g+1-i,i} - \mathbf{E}_{i,2g+1-i}}_{i \leftrightarrow 2g+1-i} \tag{11.22}$$

and therefore $\mathbf{Z}_{\pm 2\varepsilon_i} \in \mathbb{S}_{2g} \times \{\pm 1\}^{2g}$.

We can conclude:

Proposition 11.2.2: *Let $1 \le i < j \le 2g$. Then we set*

$$d^{\pm}_{i,j} := \mathrm{diag}\Big(1, \ldots, 1, \overset{i\text{-th}}{\overset{\downarrow}{\pm 1}}, 1, \ldots, 1, \overset{j\text{-th}}{\overset{\downarrow}{\mp 1}}, 1 \ldots, 1\Big) \in \mathrm{Mat}_{2g}(\mathbb{Z}).$$

We have that $\Pi_{\mathfrak{g}} \subset \mathbb{S}_{2g} \times \{\pm 1\}^{2g}$ and furthermore that

$$\begin{aligned} \Pi_{\mathfrak{g}} \cong \Big\langle (i,j)(2g+1-i, 2g+1-j) \times d^{\pm}_{i,j} \cdot d^{\pm}_{2g+1-i,2g+1-j}, \\ (i, 2g+1-i)(j, 2g+1-j) \times d^{\pm}_{i,2g+1-j} \cdot d^{\pm}_{j,2g+1-i}, \\ (i, 2g+1-i) \times d^{\pm}_{i,2g+1-i} \quad : \quad 1 \le i < j \le g \Big\rangle \\ = \Big\langle (i,j)(2g+1-i, 2g+1-j) \times d^{\pm}_{i,j} \cdot d^{\pm}_{2g+1-i,2g+1-j}, \\ (i, 2g+1-i) \times d^{\pm}_{i,2g+1-i} \quad : \quad 1 \le i < j \le g \Big\rangle. \end{aligned}$$

Note that the matrices $d^{\pm}_{i,j}$ are not symplectic matrices since they have determinant -1. Although the generate of course the symplectic diagonal matrices with only ± 1 on the diagonal. It is not difficult to see that $\Pi_{\mathfrak{g}}/_{\sim}$, where $\sim$ is the equivalence relation given by forgetting the sign change of the operation, is a proper subgroup of the symmetric group $\mathbb{S}_{2g}$, i.e.

$$\Pi_{\mathfrak{g}}/_{\sim} \subsetneqq \mathbb{S}_{2g}. \tag{11.23}$$

For our computations we need a simple corollary out of the previous considerations.

Corollary 11.2.3: *For each pair (i,j) with $1 \le i,j \le 2g$ there is a permutation $p \in \Pi_{\mathfrak{g}}$ such that for a column vector $\mathbf{a} := (a_1, \dots, a_{2g})^t \in \mathbf{R}^{2g}$*

$$p(\mathbf{a}) = \big(a_1, \dots, a_{i-1}, \overset{\overset{i\text{-th}}{\downarrow}}{\pm a_j}, p(a_{i+1}), \dots, p(a_{j-1}), \overset{\overset{j\text{-th}}{\downarrow}}{\mp a_i}, p(a_{j+1}), \dots, p(a_{2g})\big)^t$$

holds.

In other words, we can (up to a sign) permute any element below the i-th row into the i-th row without changing the elements above. This corollary follows immediately from the knowledge of the generators. One only has to distinguish some combinations of the indices to be in the first and/or the second block of g elements. The proof is left to the reader as an exercise.

11.2.3.2. Computing the Decomposition

We are now able to compute the decomposition of a given element $\gamma \in \mathbf{Sp}_g(\mathbf{R})$ into the unipotent elements given by the generic group generators defined in the previous section.

Proposition 11.2.4 (Symplectic Gaussian Elimination I)**:**
Let $\mathbf{R}$ be a euclidean ring and $\gamma = (a_{i,j})_{1\le i,j\le 2g} \in \mathbf{Sp}_g(\mathbf{R})$ then there is an element

$$M_0 = \prod_i \mathbf{X}_{\alpha_{m_i}}(T_i)^{\nu_i}\mathbf{Y}_{\alpha_{n_i}}(S_i)^{\mu_i}$$

with integers $\mu_i, \nu_i \in \mathbb{N}_0$ and $m_i, n_i \in \mathbb{N}$ such that $\alpha_{m_i}, \alpha_{n_i} \in \Phi^+$, i.e. M_0 can be written in terms of the generators in Theorem 11.2.1, such that

$$M_0 \cdot \gamma = \begin{pmatrix} a'_{1,1} & * & \cdots & * \\ 0 & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & * & \cdots & * \end{pmatrix}.$$

If the $a_{1,i}$ are coprime we have that $a'_{1,1} \in \mathbf{R}^{\times}$. In this case we can force $a'_{1,i} = 1$.

Proof. We give an explicit construction in two steps, which might be iterated:

Step 1: According to Corollary 11.2.3 we find an $M_1 \in \Pi_{\mathfrak{g}}$ such that we have

$$M_1 \cdot \gamma = \begin{pmatrix} \tilde{a}_{1,1} & * & \cdots & * \\ \tilde{a}_{2,1} & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{a}_{2g,1} & * & \cdots & * \end{pmatrix}$$

where $\tilde{a}_{i,1} = \pm a_{\sigma(i)}$ for all i for some permutation $\sigma \in \mathbb{S}_{2g}$ and

$$0 < |\tilde{a}_{1,1}| \leq |\tilde{a}_{i,1}|$$

for all $2 \leq i \leq 2g$ where $\tilde{a}_{i,1} \neq 0$. It is here enough to apply the permutation which interchanges only $a_{1,1}$ with a non vanishing $a_{i,1}$ of smallest absolute value. In praxi it is sufficient to apply a permutation such that the first entry is non-zero and smaller than or equal to the next non-zero entry in the first column (compare Step 2). Nevertheless it will be faster if we have the absolute minimum.

Step 2: By (11.11) we know how the unipotent matrices $\mathbf{X}_i(T)$ and $\mathbf{Y}(T)$ for $T \in \mathbf{R}$ look like. They add the T-fold of row i_1 to row i_2 *and* the $-T$-fold of row j_1 to row j_2 for some integers $1 \leq i_1, i_2, j_1, j_2 \leq 2g$ (see (11.17) to (11.21) for the exact possible combinations indices). This make the situation a little bit more fancy than in case of $\mathbf{SL}_n(\mathbf{R})$ or $\mathbf{GL}_n(\mathbf{R})$ (see [Fre83; p. 324]) since in many cases the elements $\mathbf{X}_\alpha(T)$ act on a pair of rows and not on each row independently. Let us assume

$$M_1 \cdot \gamma = \begin{pmatrix} \tilde{a}_{1,1} & * & \cdots & * \\ 0 & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ 0 & \vdots & & \vdots \\ \tilde{a}_{j,1} & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \tilde{a}_{2g,1} & * & \cdots & * \end{pmatrix}$$

where $\tilde{\alpha}_{j,1} \neq 0$. Then we find a unipotent element $\mathbf{X}_\alpha(T)$ for some positive root $\alpha \in \Phi^+$ such that

$$\mathbf{X}_\alpha(T) \cdot M_1 \cdot \gamma = \begin{pmatrix} \tilde{a}_{1,1} & \tilde{a}_{1,2} & \cdots & \tilde{a}_{1,2g} \\ 0 & & * & \\ \tilde{a}_{j,1} \pm T \cdot \tilde{a}_{1,1} & & \vdots & \\ * & & * & \end{pmatrix}.$$

It is important to remember that the chosen $\mathbf{X}_\alpha(T)$ cannot change the entries in the first rows above the j-th row, it only changes rows below (comp. Corollary 11.2.3). We now apply the euclidean algorithm to find a $T_1 \in \mathbf{R}$ such that the absolute value

$$|\tilde{a}_{j,1} \pm T_1 \cdot \tilde{a}_{1,1}| \leq |\tilde{a}_{j,1} \pm T \cdot \tilde{a}_{1,1}|$$

for all $T \in \mathbf{R}$, i.e. it is minimal. Because $\tilde{a}_{1,1}, \tilde{a}_{j,1} \neq 0$ and $\tilde{a}_{j,1} \geq \tilde{a}_{1,1}$ we will get

$$0 \leq |\tilde{a}_{j,1} \pm T_1 \cdot \tilde{a}_{1,1}| < |\tilde{a}_{1,1}|$$

Now there are two cases: either $\tilde{a}_{j,1} \pm T_1 \cdot \tilde{a}_{1,1} = 0$, then repeat Step 2 for the next non-zero entry in the first column, or

$$0 < \tilde{a}_{j,1} \pm T_1 \cdot \tilde{a}_{1,1} < \tilde{a}_{1,1},$$

then restart with Step 1 and interchange the first row with an other row for which the first entry is the smallest. The first candidate is the j-th row but it might happen that simultaneously the other row which was changed by $\mathbf{X}_\alpha(T_1)$ has an even smaller non vanishing absolute value. In this case we exchange those rows.

Since we are working in a euclidean ring we will get in every run of these two steps either at least one additional zero in the first row or we can replace the element in the upper left corner by a smaller one. This will stop after finitely many steps. It follows from the well known properties of the iterated application of the euclidean algorithm that the found $a'_{1,1} \in \mathbf{R}^\times$ if there is no common (non-trivial) factor for all entries in the first column. $\square$

We can now assume that we can transform a given element in $\mathbf{Sp}_g(\mathbf{R})$ by unipotent transformations into the form stated in Proposition 11.2.4. We can now proceed in the same way and apply the method from Proposition 11.2.4 to the second column and the diagonal entry therein, which we maximise by a suitable symplectic permutation. By Corollary 11.2.3 this does not change the row above . This eliminates the entries in the second column below the diagonal. We iterate this procedure for the next columns/rows. After at most g steps we transformed the given element γ into an upper triangular matrix which is still symplectic.

The implementation of this procedure is done in two functions of our SAGE-package. The source code of them together with some auxiliary functions are collected in the Appendix B. The permutation needed in Step 1 of the proof of Proposition 11.2.4 can be computed with our function `rowpermutation` stated in SAGE Source Code B.11. This function can do more than the things described above. First, it performs the permutation, but in addition it ensures that the element with lowest absolute value is positive[7]. Furthermore, it is possible to apply the function in the following steps, which are controlled via the variable `step`, to the other columns, permute in the actual step the smallest element to the right position and switch, if necessary, the sign of the new entry to be positive. Furthermore, we collect the necessary elementary operations (comp. (11.10)) in a list `collection`. Such an elementary operation $\texttt{mat} = \exp(\texttt{T} \cdot \mathbf{n}_{\texttt{idx}})$ is stored in the form `[[idx,T], mat]`, where $\texttt{T} \in \mathbf{R}$ and `idx` is the index of the root in the list of roots Φ. The row operations from Step 2 of the proof, which reduce the size of the entries below the diagonal in the actual column indicated by the variable `step`, will be performed by the function `rowtransformation` (see SAGE Source Code B.13). This function uses in the reduction step the function `div_min`, stated in SAGE Source Code B.12, which is a variation of the extended euclidean algorithm and computes for given integers `a` and `b` with $|\texttt{a}| \leq |\texttt{b}|$ a pair `[s,t]`, such that $\texttt{t} = \min(|\ \texttt{s*a+b}\ |)$. Here it was quite useful to compute first a lookup table, in which we stored which generic group generator affects which rows. As keys we use the indices of the entries, therefore some elements will occur twice, because they exchange a pair of rows. The storage works for `rowtransformation` as before.

The result of this part is a (symplectic) upper triangular matrix, which need not to be unipotent, because on the diagonal there might be some of the elements different from 1. In fact, the diagonal entries have to be elements in $\mathbf{R}^\times$ (comp. Prop. 11.2.4). In the cases we are interested in it is $\mathbf{R} = \mathbb{Z}$ and thus we have only ± 1 on the diagonal. But during the normalization in each step we know that all diagonal entries are positive, i.e. in this case equal to $+1$. For

[7]There are matrices of the form $\mathrm{diag}(\underbrace{1,\ldots,1}_{\text{step-1 times}}, -1,\ldots,-1, \underbrace{1,\ldots,1}_{\text{step-1 times}}) \in \mathbf{Sp}_g(\mathbb{Z})$, which can be computed by the function `negative_identity(L, step=cur_step)` (see SAGE Source Code B.8), which uses products of symplectic permutations. This element is equal to id restricted to some $2g-2(\texttt{step}-1)$-dimensional subspace, on which the group act as a symplectic group of lower rank. These elements can of course be written in terms of unipotent transformations. We can use thes matrices to change the sign in the actual `step` without changing the signs of the normalizations in the previous steps.

the general case one easily finds a diagonal matrix $\operatorname{diag}(r_1, \ldots, r_g, r_g^{-1}, \ldots, r_1^{-1}) \in \mathbf{Sp}_g(\mathbf{R})$ with $r_i > 0, r_i \in \mathbf{R}^\times$ which can be written analogously as product of elements of the form $\mathbf{Y}_\alpha(r) \cdot \mathbf{X}_\alpha(-r) \cdot \mathbf{Y}_\alpha(r)$ for $\alpha \in \Phi^+$ and $r \in \mathbf{R}^\times$ (comp. (11.16)).

In this way we get a unipotent upper triangular (symplectic) matrix. It remains to transform this matrix into Id_{2g} with transformations which can be written in terms of the upper triangular matrices $\mathbf{X}_\alpha(T)$. This is much easier than the previous steps of the transformation, since all operations commute and we Therefore, do not have to take care of the order of the operations we have to perform. Furthermore, we can immediately read from the matrix what coefficients T we have to choose. Here we use again the lookup table from above. We iterate over the keys (i,j) of this table, check whether the element $a_{i,j}$, which is changed by the transformation for this key, is non-zero, if so we take apply $\mathbf{X}_\alpha(-a_{i,j})$ corresponding to the value of the key, which annihilates the non-zero element $a_{i,j}$. Then we take the next key from the lookup table and proceed in the same way with the new matrix. It will happen that a transformation occurs as value of two different keys in the lookup table, since some elements changes pairs of rows, this does not make a problem, because if the corresponding elements are already zero, nothing will be done. In all tested cases it was much more expensive to implement additional checks which avoid to have these elements twice. As before, we collect the necessary operations. We did not include this part of the transformation in a separated function, but in the main function `getdecomposition` (see SAGE Source Code B.14).

In this way we can transform any $\gamma \in \mathbf{Sp}_g(\mathbf{R})$ by consecutive application of a collection of unipotent elements in $\mathbf{Sp}_g(\mathbf{R})$ to the identity matrix. To get the decomposition one has to write the elements we collected in the opposite order and to switch the signs of the T in each factor. Here we have to mention that it is in practice not necessary to reverse the order of the elements in the program, because we can construct the list by the Python command `append`, which appends the elements to the list from the left to the right and therefore the elements are already arranged in the right order. This leads to the following theorem:

Theorem 11.2.5 (Symplectic Gaussian Elimination II):
Let $\mathbf{R}$ be a euclidean ring and $\gamma = (a_{i,j})_{1 \le i,j \le 2g} \in \mathbf{Sp}_g(\mathbf{R})$, then there are integers $\mu_i, \nu_i \in \mathbb{N}_0$ and $n_i \in \mathbb{N}$ and ring elements $S_i, T_i \in \mathbf{R}$, such that $\alpha_{n_i} \in \Phi^+$ and

$$\gamma = \prod_i \mathbf{X}_{\alpha_{n_i}}(T_i)^{\nu_i} \mathbf{Y}_{\alpha_{n_i}}(S_i)^{\mu_i}.$$

The parameters $n_i, \nu_i, \mu_i, T_i, S_i$ can be computed explicitly (see SAGE *Source Code B.14).*

This is a kind of structured version of the Gaussian elimination algorithm for symplectic matrices. Structured Gaussian elimination, which preserves some given structures like sparsity of the matrix, is widely known in different areas of mathematics (see e.g. [Vas07; §11.3]). In SAGE Source Code B.14 of the function `getdecomposition` you find the implementation of the algorithm described above. We use usually precomputed lists of generators[8], as well of sign and permutation matrices, because they do not change from one step to the other. These precomputed lists do also not change if we change the weight of the considered representation module. If they are not passed to the function, the necessary lists are computed. In

[8]In fact, we do not store the matrices of the generators. It is better to store directly a list of polynomial functions of the form $T \mapsto \exp(T \cdot \rho_0(\mathbf{n}_i))$ for the standard representation ρ_0, which avoids the more expensive evaluation of the exponential map.

addition there is a function called getdecompositionoflist which works in the same way as getdecomposition but take instead a single matrix a list of them as input and computes if necessary the list of generators only ones.

We also included some checks. First of all, we check whether the element is the identity element, then no computation is needed. Second we check whether the transformation leads to the identity matrix, if not the function returns None together with an error message. The latter case can only happen if the matrix we are going to decompose is not in the group related to the Lie algebra L which was passed to the function.

As already mentioned the function getdecomposition works also for other groups than $\mathbf{Sp}_g(\mathbf{R})$. For example, if we have the Lie algebra $\mathfrak{sl}_n$ with root system A_{n-1} the decomposition algorithm works completely in the same way (see Section 11.2.4.2). The automatically generated lists of generators are the known generic generators, which have ones on the diagonal and at all other positions zeros except one non-zero entry.

11.2.4. Some Examples

11.2.4.1. The Symplectic Group $\mathrm{Sp}_2(\mathbb{Z})$

In this section we give some practical examples of the usage of the previously defined action of group elements on a highest weight module. We have already seen some examples of highest weight modules and their bases in Section 11.2.1. For random testing we introduced a function randomgroupelemenetbyLiealgebra(L,number_of_generators=20), which generates a random group element, i.e. a matrix in the Lie group whose associated Lie algebra is L. By default the number of generators is 20. For more details we refer to the manual of our program package in Appendix in Chapter A. This is very comfortable since we need not to find symplectic elements by ourself.[9] Furthermore, the elements generated in this way are all symplectic with respect to the basis $\mathfrak{B}^{\mathsf{var}}$, such that we can use them without further transformation (comp. Section 6.3.1.3). We start with some examples on the usage of the random element generator and the getdecomposition function.

```
sage: L = SimpleLieAlgebra("C", 2)
sage: m1 = randomgroupelemenetbyLiealgebra(L); m1
    [ 3  7 -2  0]
    [ 1 10 -2  2]
    [-2 -2  1  1]
    [ 0  6 -1  2]
sage: m2 = randomgroupelemenetbyLiealgebra(L); m2
    [-1  0 -2  2]
    [ 0  0 -1  0]
    [ 1  1  0 -4]
    [-1  0 -3  1]
sage: m2dec = getdecomposition(m2,L); len(m2dec)
```

[9]That would not be a problem at all, since we generated lots of symplectic elements as elements in the stabilizer, but that elements are somehow special. For example, they have all finite order and no entries with larger absolute values.

```
        20
    sage: m2dec[3]
        [[0, 1], [1 0 0 0]
        [1 1 0 0]
        [0 0 1 0]
        [0 0 1 1]]
    sage: m1dec = getdecomposition(m2,L); len(m1dec)
        22
```

We see that the number of generators we obtain in the decomposition need not to be minimal, because the element `m1` was generated as product of 20 matrices, but our decomposition returns 22 elements.

We continue our example with the function `getreprofgroupelement`, which compute the matrix of the action of group elements on a given module. We choose the module $\mathbb{M}_{2,1}$, which has dimension 35. Here is not the place to state this matrices with all its entries. But later we will also give a picture how they look like (see Figure 11.3).

```
    sage: M = HighestWeightModule(L, [2,1])
    sage: M1 = getreprofgroupelement(m1,M); M1
         35 x 35 sparse matrix over Rational Field
    sage: M1.density()
        1152/1225
    sage: M2 = getreprofgroupelement(m2,M); M2
         35 x 35 sparse matrix over Rational Field
    sage: n(M2.density())
         0.440816326530612
    sage: m3= p.random_matrix_by_Lie_algebra(L,
    number_of_generators=3); m3
    [ 1 -1 -1  0]
    [ 0  1  1  0]
    [ 0  0  1 -1]
    [ 0  0  0  1]
    sage: M3 = getreprofgroupelement(m3,M)
    sage: n(M3.density())
        0.251428571428571
    sage: m3.density()
         1/2
```

Thus, we see that the density of the matrices on the module seems to be not directly correlated to the density of the matrix in the Lie group. We computed for the matrices `m1`, `m2` and `m3` from the examples above their densities (in the standard represenation) together with their densities in several matrix representations which are shown in Table 11.1 and in the pictures of the shapes of some of them in Figure 11.3. There is no obvious rule how dense the resulting matrices are compared with the original ones, except that the structure in some sense seems to be constant. This means if we have a look at Figure 11.3 we immediately find that for example the third column has always the shape of an upper triangular matrix,

Table 11.1.: *Densities for the highest weight representation for the elements `m1`, `m2`, `m3` for different weights. The standard representation is stated in the first row (λ=(1,0))*

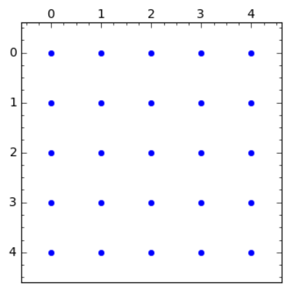
		Density		
Weight	**Dimension**	`m1`	`m2`	`m3`
(1,0)	4	0.8750	0.6250	0.5000
(0,1)	5	1.0000	0.6800	0.4800
(2,1)	35	0.9404	0.4408	0.2514
(5,0)	56	0.8957	0.4550	0.2181
(4,1)	105	0.9457	0.4051	0.2039
(4,2)	220	0.9732	0.3838	0.2014
(5,4)	880	0.9898	0.3646	0.1988
(4,6)	1330	0.9962	0.3683	0.2121

whereas also both other examples have similar, but not equal, shapes on different modules. For the density we have example where the image has lower or higher densities than the preimage. In particular, we cannot assume that for a generic dense matrix its image under a highest weight representation will be much more sparse than the original matrix. This is an important information, since later on we have to solve linear systems, thus it might be interesting to know whether the systems are dense or not.

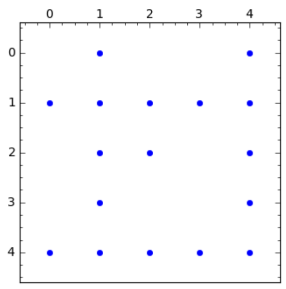
We now have a short look on the timings. As mentioned above we can compute each matrix representation at its own, or we can reuse precomputed lists of generators. We took timings for both cases (see Table 11.2). Parts of the precomputed lists depend only on the Lie algebra or the Lie algebra and the group element of which we compute the representation. Other parts depend also on the highest weight module. It turns out that the operations not depending on the module do not contribute much to the runtime in case of examples of highest weight modules of higher rank. As expected, the time grows with increasing rank of the module. This is valid in both cases. For small rank examples the dominant part is the decomposition of the group elements, whereas this part is less important for higher rank modules.

In the larger examples the runtime of the precomputation depends mainly on the time needed for the computation of the generator functions for the images of root vectors in the module. The other parts are negligible. For a single computation of the representation this is also the dominant part of the whole computation. If we use the precomputed lists it turns out that the most relevant part of each computation is not the decomposition of the given group element but the evaluation and multiplication of the factors in the computed decomposition, i.e. the very last part of the algorithm stated in SAGE Source Code B.10.

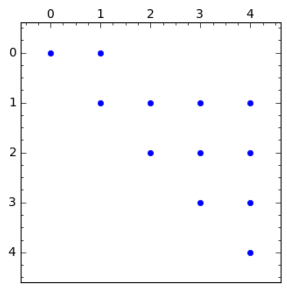
Figure 11.3.: *Shape of matrices given by* `getreprofgroupelement` *for matrices* `m1`,`m2` *and* `m3` *on different highest weight modules. Non-zero entries are indicated by dots.*



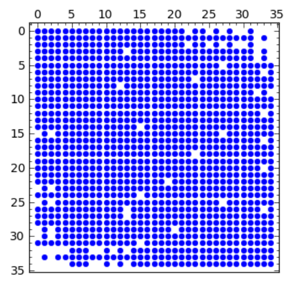
(a) *m1 on* $\mathbb{M}_{0,1}$



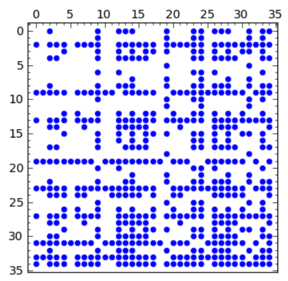
(b) *m2 on* $\mathbb{M}_{0,1}$



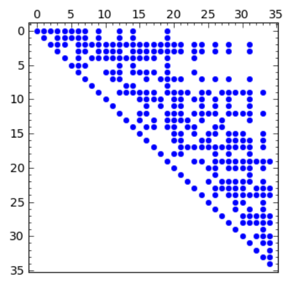
(c) *m3 on* $\mathbb{M}_{0,1}$



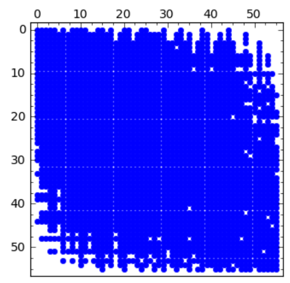
(d) *m1 on* $\mathbb{M}_{2,1}$



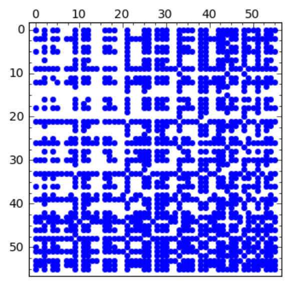
(e) *m2 on* $\mathbb{M}_{2,1}$



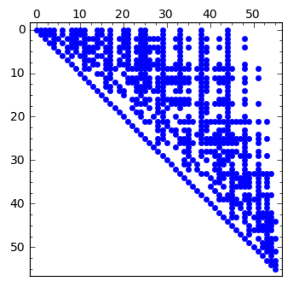
(f) *m3 on* $\mathbb{M}_{2,1}$



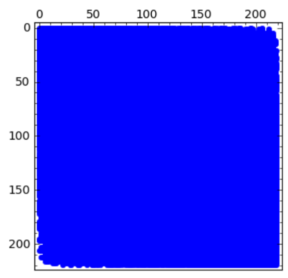
(g) *m1 on* $\mathbb{M}_{5,0}$



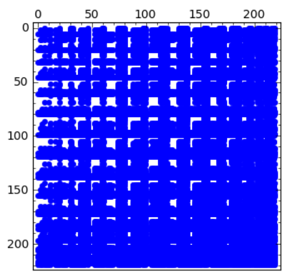
(h) *m2 on* $\mathbb{M}_{5,0}$



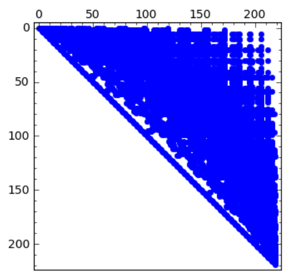
(i) *m3 on* $\mathbb{M}_{5,0}$



(j) *m1 on* $\mathbb{M}_{4,2}$



(k) *m2 on* $\mathbb{M}_{4,2}$



(l) *m3 on* $\mathbb{M}_{4,2}$

Table 11.2.: *Timings: Table (a) shows the time to compute the matrix representation for `m1`, `m2`, `m3` with and without precomputed lists of generators for different weights. In Table (b)we state the timings for the precompuation of the tables. The first block contains examples for the list corresponding to the highest weight module indicated in the first column. The second block consists of those timings, which only depend on the underlying Lie algebra, i.e. they do not change if we change the module. All timings are done on `theia` at the MPI for Mathematics and are the arithmetic mean of 5 runs of the program.*

		Time [s] without			**Time [s] with**		
Weight	**Dimension**	`m1`	`m2`	`m3`	`m1`	`m2`	`m3`
(1,0)	4	1.19	1.52	0.71	0.753	1.260	0.024
(0,1)	5	2.41	1.69	1.06	0.612	1.420	0.027
(2,1)	35	4.28	4.72	3.71	0.783	1.170	0.038
(5,0)	56	8.83	12.80	8.06	1.410	1.990	0.067
(4,1)	105	13.90	14.00	11.30	2.850	3.450	0.193
(4,2)	220	43.00	52.20	47.50	15.600	16.200	1.020
(4,4)	625	305.59	297.41	87.81	227.580	221.950	14.210
(5,4)	880	596.00	565.00	130.00	572.030	552.000	27.700

(a) *Computation of `getreprofgroupelement(m,M)`*

Depends on the module		
	Time [ms] for	
Weight	`M`[a]	**Gen. Funct.**[b]
(1,0)	13.6	357
(0,1)	15.0	397
(2,1)	16.0	3250
(5,0)	37.1	5120
(4,1)	17.8	10400
(4,2)	89.2	23600
(4,4)	390.0	72650
(5,4)	711.0	107000

Does not depends on the module	
Functions	**Time [ms]**
$(T \mapsto \mathbf{Y}_\alpha(T))_{\alpha\in\Phi^+}$[c]	188.0
$(T \mapsto \mathbf{X}_\alpha(T))_{\alpha\in\Phi^+}$[d]	187.0
$(\mathbf{Z}_\alpha)_{\alpha\in\Phi^+}$[e]	4.1
Lookup table for row operations[f]	2.3
$\mathfrak{U}(\mathrm{L})_\mathbb{Z}$[g]	9.3
`getdecomposition(m1,L)`	790.0
`getdecomposition(m2,L)`	1260.0
`getdecomposition(m3,L)`	387.0

(b) *Precomputations depending on modules and Lie algebras*

[a]Initialization of `M` using `L.HighestWeightModule(wt)`.
[b]Generating functions $T \mapsto \exp(T \cdot \rho_{\mathtt{M}}(\mathbf{n}_i))$ computed by the command `genericgroupgeneratorsfunctionsonHWM(M,...)`.
[c]Computed with `negativegenericgroupgeneratorsfunctions(L,...)`.
[d]Computed with `positivegenericgroupgeneratorsfunctions(L,...)`.
[e]Computed with `genericpermutations(L,....)`.
[f]Computed with `genericrowoperationdict(L,...)`.
[g]The list of the integral lattice basis elements in the universal enveloping algebra $\mathfrak{U}(\mathrm{L})$, computed with `latticegeneratorsinUEA(L)`.

11.2.4.2. The Special Linear Group $\mathrm{SL}_4(\mathbb{Z})$

Finally we give a short example which shows that the program works also for other groups than $\mathbf{Sp}_2(\mathbb{Z})$. We consider the group $\mathbf{SL}_4(\mathbb{Z}) \subset \mathbf{SL}_4(\mathbb{R})$. The associated Lie algebra is the simple Lie algebra $\mathfrak{sl}_4$ of type A_3.

```
sage: L = SimpleLieAlgebra ("A", 3); L
Algebra( Rationals, [ v.1, v.2, v.3, v.4, v.5, v.6, v.7, v.8,
  v.9, v.10, v.11, v.12, v.13, v.14, v.15 ] )
sage: V = L.HighestWeightModule([1,0,0]);
  # The standard representation
sage: V.Dimension()
    4
sage: M101 = HighestWeightModule([1,0,1],0); M.Dimension()
    15
sage: M111 = HighestWeightModule([1,1,1],0); M.Dimension()
    64
sage: m= randommatrixbyLiealgebra(L); m
    [ 2  1 -3  3]
    [ 0  1  1  0]
    [-1 -1  0 -1]
    [-1 -1  5 -4]
sage: M1 = getreprofgroupelement(m,M111); M1.trace()
    -6
sage: M2 = getreprofgroupelement(m,M101); M2
     # Control messages deleted
[ -4  -2  -2   6   2  -1  -1   9  -8  -1  -3  -3   3   3  -3]
[  0  -2   0  -2   0  -1   1  -1   0  -1   0   1  -1   0   0]
[-16  -8 -10  24  10  -5   1  39 -28  -2 -15 -15   6  15  -6]
[  2   2   1   0  -1   1   0  -2   3   1   1   0   0  -1   1]
[-10  -5  -6  15   8  -3   2  24 -17  -1  -9 -12   3  12  -3]
[  0  -8   0  -8   0  -5   5  -5   0  -2   0   5  -2   0   0]
[  0  -5   0  -5   0  -3   4  -3   0  -1   0   4  -1   0   0]
[  8   8   5   0  -5   5  -3  -8  10   2   5   0   0  -5   2]
[ 10  10   6 -10  -6   6  -3 -21  19   3   9   5  -5  -9   6]
[  0   5   0   5   0   3  -3   3   0   2   0  -3   2   0   0]
[  8   8   5 -40  -5   5  -3 -57  34   2  20  25 -10 -20   8]
[  5   5   3   0  -4   3  -3  -5   6   1   3   0   0  -4   1]
[ -5  -5  -3   0   3  -3   1   5  -7  -2  -3   0   0   3  -2]
[  5   5   3 -25  -4   3  -3 -35  21   1  12  20  -5 -16   4]
[ -5  -5  -3  25   3  -3   1  35 -22  -2 -12 -15  10  12  -8]
```

11.2.5. Computing Invariants of a Highest Weight Module

Let $\Gamma' \subset \Gamma$ be any subgroup of the Siegel modular group and $(\mathbb{M}, \rho)$ be some highest weight module. Then the invariants are by definition given by

$$\begin{aligned} \mathbb{M}^{\Gamma'} &= \left\{ \mathbf{m} \in \mathbb{M} : \rho(\gamma) \cdot \mathbf{m} = \mathbf{m} \quad \forall \gamma \in \Gamma' \right\} \\ &= \bigcap_{\gamma \in \Gamma'} \mathbf{ker}\left(\rho(\gamma) - \mathrm{Id}\right). \end{aligned} \tag{11.24}$$

We know from Sections 11.2.2 and 11.2.3 how to compute $\rho(\gamma)$. Therefore, we can easily compute the space of invariants. In SAGE we implemented the computation of invariants in the function `invariants`, which takes – beside some optional parameters – two parameters, the highest weight module $\mathbb{M}$ at the first place and the group Γ', a `MatrixGroup` (see SAGE Source Code B.17 in the Appendix). In the algorithm it is possible to replace the `MatrixGroup` by a set of matrices or matrix group elements. We can do some things to speed-up the computation of invariants, which are controlled by some optional parameters.

Use Generators All groups we are considering are finitely generated.[10] Let us assume that Γ' is generated by $\gamma_1, \dots, \gamma_s$. Then it is enough to compute the kernels for $\rho(\gamma_i) - \mathrm{Id}$ instead of those for all group elements to get the invariants. Thus, we get

$$\mathbb{M}^{\Gamma'} = \bigcap_{i=1,\dots,s} \mathbf{ker}\left(\rho(\gamma_i) - \mathrm{Id}\right). \tag{11.25}$$

Therefore, we can save lots of non necessary expensive computations of group actions, which partially involve quite large matrices, at the expense of finding the generators, which only requires a few multiplications of 4×4 matrices. One might expect that the reduction of the runtime for highest weight modules is roughly proportional to the factor between the order of the group and the number of its generators[11], because the runtime to compute the generators is independent of rank of the considered highest weight module and contribute therefore *asymptotically* only as a constant. In other words the gain of this is larger, the larger the highest weight modules are. In particular, for small modules the saved runtime should be competitively small.

We discuss the runtime more in detail in Section 11.2.6. The result there will be a little bit surprising. For highest weight modules of lower rank we find a factor between 1 and 5 also for groups with a large factor between the number of generators and the order of the group, i.e. a factor about 15. We observe that in most cases for larger rank modules the gain increases, but at least for the tested examples – some of them can be found in Table 11.3 – the gain is always much smaller as the expected factor. Nevertheless it is recommended to set the option `only_generators` of the function `invariants` to `True`, which is also the value by default.

[10] The group Γ is finitely generated (see Theorem 11.2.1) and all stabilizers of cells are even finite subgroups of Γ.

[11] Here it is the computed number of generators with the function `getgenerators` (see SAGE Source Code B.15 in the Appendix) and not the minimal number, because $-\mathrm{Id}$ is always in this set if it is in the group. This leads to a larger set of generators e.g. in case of some groups isomorphic to a cyclic group, as e.g. the stabilizers of cells of type DESARGUES, which get a set of two generators, whereas the group can be generated by only one element.

A further speed-up can be obtained if we precompute the generators of the occurring groups and pass the list of them instead of the group. This was one reason why we allow lists instead of matrix groups.

Use Precomputed Lists For the computation of `getreprofgroupelement` (see SAGE Source Code B.10) we reduce the runtime because we reuse some already computed results as e.g. the generators of the corresponding Lie algebra etc., which are passed in terms of lists to that function. The computation of invariants involves, according to equation (11.24), the computation of several actions of group elements on highest weight modules (see SAGE Source Code B.17). Therefore, the precomputed lists are essentially the same lists as in case of the function `getreprofgroupelement`. In our implementation of `invariants` we can switch between a computation without any precomputed data and different levels of usage of precomputed data simply by passing all or even only some of the lists. By default each of the lists is set to be equal to the empty list `[ ]`. All empty lists are then replaced by newly computed versions. This is the same procedure we use inside the function `getreprofgroupelement`. After that the iteration over the elements starts and we compute each matrix of the group action with the just computed full set of lists. Usually the iteration run over several elements. Therefore, we will save some time because we use the function `getreprofgroupelement` with all possible lists precomputed.

We obtain a further speed-up if we pass all list from the beginning to the function `invariants` instead to compute them inside the the function. Some comparisons between both versions (all list precomputed and no list precomputed) will be done in Section 11.2.6.

Use Early Break Loop Conditions In equation (11.24) the result will be the zero module if for one $\gamma \in \Gamma'$ we have that $\mathbf{ker}\,(\rho(\gamma) - \mathrm{Id}) = 0$. Therefore, we check before compute the intersection of the newly computed kernel with the intersection of the previous ones, whether the kernel equals zero. Furthermore, we check after we computed the intersection whether the result is zero. If one of these checks forces the result to be zero at the end we break the loop over the elements in Γ' (or over its set of generators) and return the zero module. Therefore, if the module of invariants is zero we will get the result probably faster. In practice we will see this in particular in case of odd weights (comp. Table 11.5).

Some examples including some discussion of their runtime for invariants under stabilizers of cells are given in Section 11.2.6. Some further examples on invariants under the full Siegel modular group Γ, i.e. global sections, are discussed in Section 12.4.

11.2.6. Examples and Runtime Analysis for Computing Invariants

In our program we can easily – not necessarily fast – compute invariants. The output is an abstract submodule of a generic free module module `V` of rank n over `base_ring`, where n equals to the rank of the highest weight module `M` over `base_ring`. Here we identify the standard basis of unit vectors in `V` with the elements in the PBW-basis of `M` in the order as they occur in our algorithm (comp. Section 11.2.2). Here an simple example of the usage:

```
sage: L = SimpleLieAlgebra("C",2, base_ring=QQ)
sage: M = HighestWeightModule(L,[4,2])
sage: G = RedSquare.stabilizer();
sage: invariants(M,G)
        # Control messages deleted
Vector space of degree 220 and dimension 16 over Rational Field
Basis matrix:
16 x 220 dense matrix over Rational Field
```

We should emphasize that the output of `invariants` is in particular not a submodule of `M`. To obtain again elements in `M` from a coordinate vector in the module `V` we can use the function `coordinatevector2HWMelement`, which converts a coordinate vector in `V` to an element in `M`. After we got the matrix representation by evaluating the action of the generators on the PBW-basis it is, from a computational point of view, neither necessary nor useful to do the computations explicitly in `M` itself. It is much faster to stay in an abstract setting with matrices and coordinate vectors. But for some other questions the representation of highest weight module elements in terms of the related PBW-basis are better suited. Therefore, we implemented that conversion function. Here an example:

```
sage: L = SimpleLieAlgebra("C",2)
sage: M = HighestWeightModule(L,[2,0])
sage: G = Pyramid.stabilizer();
sage: MG1 = invariants (M,G); MG1
# Control massages deleted
Vector space of degree 10 and dimension 6 over Rational Field
Basis matrix:
[ 1  0  0  0  0 -1  0  0  0  1]
[ 0  1  0  2  0  0 -1  0  0  0]
[ 0  0  1  0  0  2  0  0  0 -2]
[ 0  0  0  0  1 -2  0  0  0  2]
[ 0  0  0  0  0  0  0  1  0  0]
[ 0  0  0  0  0  0  0  0  1  2]
sage: for b in MG1.basis(): print coordinatevector2HWMelement(b,M1)
1*v0+-1*y4*v0+y4^(2)*v0
y1*v0+2*y1^(2)*v0+-1*y1*y4*v0
y3*v0+2*y4*v0+-2*y4^(2)*v0
y1*y3*v0+-2*y4*v0+2*y4^(2)*v0
y3^(2)*v0
y3*y4*v0+2*y4^(2)*v0
```

To illustrate the usage of our program routines we computed the invariants under the stabilizers of the 11 standard cells. Their dimensions are shown in Table 11.4. Further, we did some analysis of the runtime for the computation of the invariants. The computation of invariants is, as explained in Section 11.2.5, essentially based on the computation of the group action on a highest weight module. Therefore, the runtime is roughly given by:

- The time needed to compute the matrix representation of a single element of the group.

- The number of elements in the group for which we have to compute the action. These are at the best only a minimal set of generators of the considered group. In our implementation we compute by default we compute from the group a set of generators. We do not check the minimality but in praxi in most cases the size is equal to the minimal size (comp. Table 10.7). The minimal number of generators is known because we know the group structure and therefore also the minimal number of generators. To obtain some gain it is important to reduce the number of elements with as less effort as possible. For a further speed-up, e.g. for really large modules, one should try only to use sets of generators of minimal size.
- The time for the computation of kernels of matrices and their intersection with some space. As explained in Section 11.2.5 we compute first a kernel and intersect it then with the result of the previous step.

Compared to the first two points the third point is in most cases negligible for larger weights, since computing kernels can be done efficiently by standard libraries. Even for such large systems for which even the computation of a single matrix representation needs, roughly estimated, a day it can be done in a few minutes. However in those case where the result is zero, there might be a relevant speed-up because we break a loop earlier. In particular, this can be observed for highest weights modules whose weights are odd in its first component, see e.g. $\lambda = (5, 4)$ in Table 11.5. As we expect, according to the results stated later in Lemma 12.3.2 and (12.10), we get as result in this cases the zero module (see Table 11.4).

As already mentioned in Section 11.2.5 we reduce the number of objects which have to be computed in two ways. First, we compute the intersections in (11.24) only for a set of generators of the group. Second, we have the possibility to pass some precomputed lists of generators for one computation of the group action to the function `invariants`.

If we replace the group by its generators it is important that we spent – compared to the computation of a single group action – not too much time with the search for a system of generators. In praxi this means that for small modules it is enough to reduce the number in a significant way whereas for really large modules it becomes more important to have an as small as possible set. Since we are using our algorithm only for competitively small modules we took no effort to get a minimal set. Furthermore, there is an other important observation which leads to a not necessarily minimal set of generators. Namely it turns out that $-\mathrm{Id}$, even if it is not in a minimal set of generators, should be in the set of generators at the first handled position, because in some cases – in particular for weights with odd first component – this element forces the invariants to be zero, which leads to a significant gain of time. This is the more important the larger the modules are. So for small weights with odd first component, i.e. in Tables 11.5 and 11.4 the weights $(1, 0)$ and $(5, 0)$, this gain is not very large compared to the other operations which were done. All results in Tables 11.5 and 11.4 are computed with the default option `only_generators=True`. One can see in all cases with odd first component in Tables 11.5 that the runtime is smaller than the runtime for modules of about the same size or even of smaller size, whereas the runtime in the case without this trick should increase with the rank of the module times some factor only depending on the group. Note that the rank of a highest weight module with respect to its weight is in this case a cubic polynomial in the components of the weight (see equation (6.56) and Example 6.4.9).

Table 11.3.: *Comparison of runtimes for some highest weight modules between the computation of* `invariants` *with option* `only_generators` *equals* `True` *and* `False`*. Furthermore, we state in the last column the dimension of the space of invariants over $\mathbb{Q}$. All timings are done on* `theia` *at the MPI for Mathematics and are the arithmetic mean of 5 runs of the program.*

Cell $\widetilde{\sigma}$	Factor[a]	Weight	Time [s] `only_generators` `True`	`False`	Gain[b]	Dimension
		(2,1)	10.05	23.13	2.30	1
RedSquare	10.7	(4,1)	49.59	57.65	1.16	6
		(4,2)	42.64	229.56	5.38	16
		(2,1)	5.57	9.33	1.68	15
Square	2.00	(4,1)	15.91	19.97	1.26	49
		(4,2)	35.75	56.91	1.59	116
		(2,1)	33.90	182.01	5.37	1
Hexagon	14.4	(4,1)	21.61	142.63	6.60	2
		(4,2)	58.22	492.47	8.47	7
		(2,1)	28.88	31.05	1.08	0
Reye	16.0	(4,1)	17.83	111.00	6.23	2
		(4,2)	48.10	392.91	8.17	12
		(2,1)	5.19	10.17	1.96	7
Desargues	5.00	(4,1)	15.27	29.41	1.93	21
		(4,2)	36.51	100.26	2.75	44

[a]Factor = $\#\Gamma_{\widetilde{\sigma}}/\#$ generators.

[b]Gain = time with all elements/ time with only generators.

Table 11.4.: *Dimensions of invariants under stabilizers of the standard cells for some highest weight modules over $\mathbb{Q}$. In the first row we state the dimension of the modules over $\mathbb{Q}$.*

Weight	(1,0)	(0,1)	(2,1)	(5,0)	(6,0)	(4,1)	(4,2)	(4,4)	(5,4)
Dimension	4	5	35	56	84	105	220	625	880
RedSquare$_0$	0	0	1	0	6	6	16	46	0
Vertebra$_0$	0	1	3	0	10	9	18	53	0
Crystal$_0$	0	0	1	0	8	6	23	61	0
Pyramid$_0$	0	1	15	0	44	49	116	325	0
Triangle$_0$	0	5	35	0	84	105	220	625	0
Square$_0$	0	1	15	0	44	49	116	325	0
Hexagon$_0$	0	0	1	0	4	2	7	19	0
DR$_0$	0	5	35	0	84	105	220	625	0
RR$_0$	0	0	3	0	17	13	42	115	0
Reye$_0$	0	0	0	0	5	2	12	31	0
Desargues$_0$	0	1	7	0	16	21	44	125	0

In Table 11.3 we compare for the stabilizers of five cells and three modules the runtime for the two choices of the option `only_generators`. As expected we find a gain between the case where the option is set to `True` and the case where the option is set to `False`. Surprisingly the gain is not as high as one might expect. Naively one will expect that if we reduce the number of elements to $1/N$ times the number of the original set, we should see some factor of this size if we compare the runtime in both cases. This assumption should be plausible if in average the operation needs about the same time for each run with a different element and if the time to find the set of generators is negligible. This is partially wrong. The time to decompose elements into their generators coming from the generic generators of the Lie algebra differs as we have seen earlier (comp. Table 11.2). However also tests with different sets of generators of the same cardinality give us results of about the same size. Therefore, this gives no explanation for the difference to the expected runtime gain. The second assumption is true, since the time to get a list of generators is really negligible (see Table 11.6).

Thus, there we have still no explanation for this strange difference between expected and observed runtime gain. We suspect, after some consultation of experts in SAGE, Python, and GAP, that it is probably due to some side effects in the used libraries or a question of memory usage. They told me that similar effects happen in many cases also in MAGMA. The used library commands are plain C code, which is wrapped into GAP and/or SAGE. At least in case of SAGE wrapping leads to as fast code as if that part is executed completely in C and not included in Python via the Cython libraries.

The other possible explanation to look at the memory needs to check which kind of memory is accessed. It is probably known to all of us that at the time where we have not enough RAM to keep the computation in this kind of memory and we have to store at least some parts on the hard drive the runtime increases rapidly, because the access to data on hard drives is much slower than the access to data in the RAM. To be precise this happens much earlier since there are different levels of cache memory. there are some difference between the cache memory of the three used computers. The computer `theia` has much more memory than the other two. In addition `theia` is the only computer which has a quite large Level 2 cache *and* also some Level 3, which will be used before the usual RAM is used. This means that `theia` is much better suited for computation with extensive memory consumptions. These differences in memory can explain difference in runtime by different usage of memory. But this effect should go in the opposite direction since the computations with all elements needs more memory, whereas the computations with only generators should also be faster if they fit into the faster memory, which is not large enough to handle also the other case. Therefore, we believe that the different memory usage is not (the only) reason. This is not a satisfactory answer, but at the moment we have also no better one.

We assume that – maybe – the difference between expected and seen runtime might be less significant for much larger modules, i.e. of rank $\gg 1000$, because at least some side effects might be weaker. This test has to be done carefully because for really big modules – larger modules need larger memory – it becomes relevant that also the almost 400GB RAM of `theia` are not enough and the speed will decrease significantly is one has to use hard disc memory as mentioned above. For practical reasons it was not possible to check this up to now. It would be interesting to analyse this deeper later on.

Table 11.5.: *Time to take invariants of some highest weight modules under stabilizers of the standard cells. For each pair consisting of a cell and a highest weight module we took the time on the one hand for runs without (w/o) any precomputed data and on the other hand with (w/) precomputed data. All timings are done on* `theia` *at the MPI for Mathematics and are the arithmetic mean of 5 runs of the program.*

		Time [s]								
	Weight	**(1,0)**	**(0,1)**	**(2,1)**	**(5,0)**	**(6,0)**	**(4,1)**	**(4,2)**	**(4,4)**	**(5,4)**
RedSquare$_0$	w/o	1.49	2.30	5.38	6.51	11.70	14.70	43.1	269.0	209.0
	w/	0.42	1.32	1.71	0.70	3.22	4.05	15.3	177.0	90.0
Vertebra$_0$	w/o	1.13	1.50	4.10	6.05	9.74	13.30	32.1	157.0	197.0
	w/	0.50	0.54	0.70	0.67	1.44	1.98	7.3	83.6	91.3
Crystal$_0$	w/o	1.18	1.91	5.88	6.37	12.50	14.90	45.1	353.0	204.0
	w/	0.44	0.921	2.33	0.66	4.64	5.45	21.7	260.0	91.0
Pyramid$_0$	w/o	1.18	1.78	4.88	6.25	10.70	13.60	34.0	211.0	191.0
	w/	0.43	0.66	1.13	0.65	2.10	2.91	11.3	150.0	90.7
Triangle$_0$	w/o	1.21	1.23	4.29	6.16	9.73	12.10	28.6	118.0	202.0
	w/	0.44	0.48	0.56	0.60	0.87	1.15	4.0	42.5	89.0
Square$_0$	w/o	1.29	1.89	4.95	5.96	10.20	13.50	35.2	221.0	233.0
	w/	0.43	0.75	0.97	0.73	2.22	4.08	11.5	156.0	92.1
Hexagon$_0$	w/o	1.29	2.69	6.24	0.74	13.40	5.45	45.0	353.0	237.0
	w/	0.53	2.06	2.39	0.79	4.68	4.64	20.1	219.0	86.1
DR$_0$	w/o	1.27	1.34	4.24	6.04	9.20	11.90	29.5	121.0	206.0
	w/	0.42	0.44	0.54	0.52	0.88	1.09	3.8	41.2	89.0
RR$_0$	w/o	1.26	1.82	5.29	6.45	11.10	14.60	39.1	288.0	235.0
	w/	0.51	1.06	1.35	0.69	2.59	3.36	14.2	196.0	85.8
Reye$_0$	w/o	1.25	1.94	4.95	6.94	11.10	15.90	40.3	364.0	202.0
	w/	0.45	1.00	1.25	0.71	2.63	3.58	16.2	264.0	85.5
Desargues$_0$	w/o	1.33	1.37	4.50	6.34	9.80	12.80	31.8	213.0	202.0
	w/	0.46	0.57	0.86	0.67	1.58	2.14	9.2	101.0	94.7

Table 11.6.: *Time to compute a set of generators with the function* `getgenerators`*. All timings are done on* `theia` *at the MPI for Mathematics and are the arithmetic mean of 5 runs of the program.*

	RedSquare$_0$	Vertebra$_0$	Crystal$_0$	Pyramid$_0$	Triangle$_0$	Square$_0$	Hexagon$_0$	DR$_0$	RR$_0$	Reye$_0$	Desargues$_0$
Time [ms]	3.98	2.03	3.62	0.22	0.10	0.22	32.70	0.10	0.85	7.37	0.32

A further speed-up can be obtained by checking in the preprocessing of `invariants` whether $-\mathrm{Id}$ is in the set of elements we have to use and whether we have a highest weight module with odd first component. We decided not to implement this at the moment for two reasons. First, it works in this simple way only for highest weight modules and not for more complicated modules. Further, it is better to do this earlier in the computation of the cohomology since then according to Corollary 12.3.3 all cohomology groups vanish. In other words, in practice we would never ask for these invariants. Here we use the known result to validate our implementation of the algorithms. At the present state there is no predefined function to compute invariants for arbitrary representation modules. By complete reducibility one can decompose each such module into a direct sum of objects where we can apply the implemented methods.

For the analysis of the runtime with and without the usage of precomputed data we have to take into account Table 11.2 as well as Table 11.5. There are two different types of precomputed Data (see Table 11.2 (b)):

- Data *depending* on the module
  - The module $\mathbb{M}$
  - The list of generating function for the group action on the module $\mathbb{M}$
- Data *independent* of the module
  - The list of generators for the Lie algebra and the Lie group.
  - The decomposition of the group elements (This is actually *NOT* passed in any way to any function.)

For runtime considerations the computation of the generators for the Lie group and Lie algebra are completely negligible. For asymptotic runtime analysis, i.e. the runtime for modules of large rank, the generation of all data which does not depend on the module does not really contribute to the runtime. The only part which has to be computed in each run is the decomposition of the group element into its generators in the Lie group. We might later also implement a reusage of a once computed decomposition. However their contribution is really small even if we we take into account that some of them has to be computed several times (see Table 11.2 (b)). However the dominant part, at least for modules of larger rank, where also the total runtime is much longer, is the generation of the precomputed data which depends on the module. Here the initialization and generation of a highest weight module is the less important part (comp. Table 11.2 (b)).

For small modules the accumulated time for all precomputations – depending on the group element – is up to 80% of the total runtime of the computation of a single group action (see Table 11.2 (a)). For larger modules the ratio decreases in most cases down to less than 10% of the runtime of the evaluation of the function `getreprofgroupelement`. However there are cases, e.g. the example `m3` in Table 11.2, where also in larger modules one has gains of about 80%. In any case one saves at least some percent, if we look only at one group action and use precomputed data. We see that the behaviour depends strongly on the group elements and thus it is not that easy to say a priori how long the computation of `invariants` will need.

Since we use, as explained in Section 11.2.5, inside the function `invariants` the precomputed version of `getreprofgroupelement` we get also some speed-up compared to the usage of only the non-precomputed version of that function to compute the group action.

Let us now have a closer look at the examples of the evaluation of the function `invariants` for some highest weight modules and the stabilizers of all standard cells, which can be found in Table 11.5. For later use we need in addition also the invariants under intersections of stabilizers. For questions concerning runtime they behave in the same way as the considered groups. Anyway, according to the results stated in Table 10.12 and in `intersections.pdf` on the CD-ROM, which is included to the printed copy of this thesis, most intersections are isomorphic to C_2 and should Therefore, behave similar to the stabilizers of cells of type Triangle or DR. As expected we observe that the computation with precomputed data is faster than without. It is interesting to observe that we have, in contrast to the previous discussion, a gain which is also larger than the expected gain. Naively one might expect that the runtime of the computation with precomputed data is roughly the runtime without precomputed data minus the sum over all runtimes to produce the precomputed data. Let us consider the weight $\lambda = (4,4)$. In detail we have, according to Table 11.2 (b):

	0.39s	Computing $\mathbb{M}_{4,4}$
	72.65s	Computing generating functions for $\rho_{4,4}$
	0.19s	gen. from neg. roots
	0.19s	gen. from pos. roots
	<0.01s	Look-up table for row op.
	0.01s	Lattice generators of $\mathfrak{U}(\mathfrak{sp}_2)$
Sum	73.43s	

In addition to this we have to compute the decomposition of the group elements in the set of generators, i.e. we have to do this up to five times. Tests show that this needs between 250ms and 3s each. However this has to be done in both cases. Therefore, we can guess that the difference between both versions should be roughly about or below 70s. It is astonishing to compute these differences (see Table 11.7), because in many cases the gain is larger than expected. In one case, the invariants under the stabilizer of the cell Hexagon$_0$, we even have a gain twice as large as expected. For weight $(4,4)$ this happens quite often, for the other weights displayed in Table 11.7 we get in most cases the expected behaviour.

It is not easy to find out the reason for this. However probably it has similar reasons as the behaviour discussed earlier in context of doing the computation only for generators. Different to that situation, we have here in most cases the behaviour we expect and only in some cases we see a difference, which has to be explained somehow. In this situation some side effects are a much more plausible explanation than in the situation before. To get a better picture in which cases there are differences, we have to do more computations in particular with much larger modules, i.e. of rank $\gg 1000$, because here (not only in the displayed examples) the phenomenon occurs more often for modules of higher rank.

The computations are with the made enhancement not as fast as expected, but they are much faster than the unoptimized versions of the functions. Therefore, usage of previously computed data and only the generators is highly recommended. A further speed-up might be obtained if we can manage to avoid the computation of the decomposition of the matrices into generators and the image of them under the considered representation. However for this

Table 11.7.: *Differences of the runtime of the function `invariants` between the computations with and without precomputed data. The data was taken from Tables 11.2 (b) and 11.5.*

	(2,1)	(4,1)	(4,2)	(4,4)
Time for precomputation	4.65 s	11.80 s	25.07 s	73.43 s
RedSquare$_0$	3.67 s	10.65 s	27.80 s	92.00 s
Vertebra$_0$	3.40 s	11.32 s	24.80 s	73.40 s
Crystal$_0$	3.55 s	9.45 s	23.40 s	93.00 s
Pyramid$_0$	3.75 s	10.69 s	22.70 s	61.00 s
Triangle$_0$	3.73 s	10.95 s	24.60 s	75.50 s
Square$_0$	3.98 s	9.42 s	23.70 s	65.0 s
Hexagon$_0$	3.85 s	0.81 s	24.90 s	133.40 s
DR$_0$	3.70 s	10.81 s	25.70 s	80.00 s
RR$_0$	3.94 s	11.24 s	24.90 s	92.00 s
Reye$_0$	3.70 s	12.32 s	24.10 s	100.00 s
Desargues$_0$	3.64 s	10.66 s	22.60 s	112.00 s

additional work would be necessary to implement an efficient way for storing and accessing this data, whereas at least in the considered examples, i.e. stabilizers of cells, whose elements decompose into only a few factors, the estimated gain is not that high. Therefore, this is not included in our code at the moment.

12. Cohomology of Arithmetic Groups

There are several different accesses to the subject of cohomology of arithmetic groups. For example, if $\Gamma \subset \mathbf{G}$ is a *nice* arithmetic group of a real Lie group $\mathbf{G}$, $\mathbb{X} = \mathbf{G}/\mathbf{K}$ is the symmetric space associated to $\mathbf{G}$ and $\mathfrak{g}$ is the associated Lie algebra, then there are the following notions:

a) Group cohomology, i.e. $H^q(\Gamma, \mathbb{M})$ for some Γ-module $\mathbb{M}$ (see [Bro94]).

b) Cohomology of sheaves, i.e. $H^q(\mathbb{X}/\Gamma, \mathscr{F})$ for some sheaf $\mathscr{F}$ (see e.g. [Bre97, Ive86, Har08c]).

c) Cohomology of the topological space, i.e. $H^q(\mathbb{X}/\Gamma, A)$ for some abelian group A (see e.g. [Hat02, Mas91, Mun84, Mas78]).

d) Relative Lie algebra cohomology, i.e. $H^q(\mathfrak{g}, \mathbf{K}; C^\infty(\mathbf{G}/\Gamma) \otimes \mathbb{M})$ for some Γ-module $\mathbb{M}$ (see [BW00]).

There are some equivalences of these different approaches, which we do not want to treat exhaustively. Here we give only a sketch of some of these identifications:

- The group cohomology $H^q(\Gamma, \mathbb{M})$ is isomorphic to the sheaf cohomology with some sheaves $\tilde{\mathbb{M}}$, which are given by the Γ-module $\mathbb{M}$ (comp. Theorem 12.2.3).
- Relative Lie algebra cohomology $H^q(\mathfrak{g}, \mathbf{K}; C^\infty(\mathbf{G}/\Gamma) \otimes \mathbb{M})$ is isomorphic to group cohomology if the Γ-modules are nice (see e.g. [BW00; Chap. VII]).
- Sheaf cohomology $H^q(\mathbb{X}/\Gamma, \mathscr{F})$ equals $H^q(\mathbb{X}/\Gamma, A)$ if $\mathscr{F}$ is the constant sheaf with values in A and $\mathbb{X}/\Gamma$ is connected.

If the circumstances are "nice" enough, all four points of view will coincide. For us the main access will be the approach b) via sheaf cohomology (see Section 12.2), because this will give a much more detailed picture of the situation than group cohomology a) or the cohomology with constant coefficients c). Although relative Lie algebra cohomology provides lots of interesting insights into the structure of the cohomology groups (see e.g. [Fra98, FS98]), we will not consider this kind of cohomology in the following.

For sheaf cohomology we consider sheaves which are so called local systems if the group Γ is torsion free. If Γ is not torsion free, they are some generalization of local systems we will introduce, defined by some Γ-modules (see Section 12.1). In this context we will consider also the connection between group cohomology and the sheaf cohomology (see Theorem 12.2.3).

As reference for the cohomology of arithmetic groups we suggest the books of Günter Harder [Har08c, Har93], as well as his lecture notes [Har05, Har08a], available online, and two of his papers [Har87, Har08a]. For the theory of local systems ans its generalizations to orbifolds we refer to the already mentioned sources of Harder or the PhD thesis of Dan Yasaki [Yas05]. For the background on cohomology of sheaves there are the books of Birger Iversen and Glen

Bredon [Ive86, Bre97].

12.1. Orbilocal Systems

There is a standard way to define a sheaf which is associated to a given Γ'-module for an arbitrary arithmetic group Γ'. For simplicity in the notation we assume that the underlying group Γ' is an arithmetic subgroup of the Siegel modular group

$$\Gamma := \mathbf{Sp}_2(\mathbb{Z}) \subset \mathbf{Sp}_2(\mathbb{R}) =: \mathbf{G}$$

of finite index and that we consider the symmetric space $\mathbb{X} = \mathfrak{S}_2$. The definitions hold also if we replace $\mathbf{G}$ and $\mathbb{X}$ by a different Lie group with its associated symmetric space. But since we also want to work out the connection between sheaves on $\mathfrak{S}_2/\Gamma'$ and sheaves on $\mathbb{W}/\Gamma'$ it is more convenient to consider at this place only the special case.

Let $\mathbb{M} := (\rho_{\mathbb{M}}, \mathbb{M})$ be any Γ'-module and

$$\pi : \mathfrak{S}_2 \to \mathfrak{S}_2/\Gamma' \tag{12.1}$$

the canonical projection onto the quotient. Let furthermore

$$r : \mathfrak{S}_2 \to \mathbb{W} \tag{12.2}$$

be the restriction map from the Siegel upper half-space onto its retract constructed in Chapter 9. To define a sheaf on $\mathfrak{S}_2/\Gamma'$ we define for every open $U \subset \mathfrak{S}_2/\Gamma'$ the set of sections over U by

$$\widetilde{\mathbb{M}}(U) = \left\{ f : \pi^{-1}(U) \to \mathbb{M} : \substack{f \text{ is locally constant,} \\ f(R_\gamma(x)) = \rho_{\mathbb{M}}(\gamma)\cdot f(x),\ \forall \gamma \in \Gamma',\ x \in \pi^{-1}(U)} \right\}. \tag{12.3}$$

This forms, with the obvious restriction maps, a presheaf. We denote by $\widetilde{\mathbb{M}}$ also the sheafification of this presheaf and call it an **orbilocal system** on $\mathfrak{S}_2/\Gamma'$. Similarly we can define a sheaf $\widetilde{\mathbb{M}_{\mathbb{W}}}$ on $\mathbb{W}/\Gamma'$. We will often use $\mathcal{M}$ instead of $\widetilde{\mathbb{M}}$ and $\mathcal{M}_{\mathbb{W}}$ instead of $\widetilde{\mathbb{M}_{\mathbb{W}}}$. This construction defines a functor

$$\mathrm{sh}_{\Gamma'} : \mathbf{Mod}_{\Gamma'} \to \mathbf{Sheaves}_{\mathfrak{S}_2/\Gamma'} \tag{12.4}$$

from the category of Γ'-modules to the category of sheaves on $\mathfrak{S}_2/\Gamma$ via $\mathrm{sh}_{\Gamma'}(\mathbb{M}) = \widetilde{\mathbb{M}}$. In the same way we can define a functor for the sheaves on $\mathbb{W}/\Gamma'$ which we also denote by $\mathrm{sh}_{\Gamma'}$. One finds that

Lemma 12.1.1: *The functor* sh_Γ *maps injective* Γ*-modules to acyclic sheaves.*

The proof of this proposition needs some more work. Therefore, we refer for the proof to [Har05; pp. 55 f.].

There is a different point of view on the introduced sheaves in terms of flat fiber bundles, which is in some cases quite useful. Recall that the sections of sheaves can be viewed as sections of suitable flat bundles. We define the following quotients of bundles

$$\begin{aligned} \mathfrak{S}_2 \times_{\Gamma'} \mathbb{M} &:= \mathfrak{S}_2 \times \mathbb{M} / \left\{ (x,m) \sim (R_\gamma(x), \rho_{\mathbb{M}}(\gamma)\cdot m) : \gamma \in \Gamma' \right\} \\ \mathbb{W} \times_{\Gamma} \mathbb{M} &:= \mathbb{W} \times \mathbb{M} / \left\{ (x,m) \sim (R_\gamma(x), \rho_{\mathbb{M}}(\gamma)\cdot m) : \gamma \in \Gamma' \right\}. \end{aligned} \tag{12.5}$$

This yields, with the notations from above, the following diagram

$$\begin{tikzcd}
 & \mathbb{W} \times \mathbb{M} \arrow[r] \arrow[dl] \arrow[d] & \mathfrak{S}_2 \times \mathbb{M} \arrow[d] \arrow[dr] & \\
\mathbb{W} \times_{\Gamma'} \mathbb{M} \arrow[d] \arrow[rrr, bend left=60] & \mathbb{W} \arrow[r, hook, bend left, "\iota"] \arrow[dl, "\widetilde{\pi}"'] & \mathfrak{S}_2 \arrow[l, two heads, bend left, "r"] \arrow[dr, "\pi"] & \mathfrak{S}_2 \times_{\Gamma'} \mathbb{M} \arrow[d] \\
\mathbb{W}/\Gamma' \arrow[rrr, bend left=10, "\iota_{\Gamma'}"] & & & \mathfrak{S}_2/\Gamma' \arrow[lll, bend left=10, "r_{\Gamma'}"]
\end{tikzcd} \tag{12.6}$$

If there is no torsion in Γ' and thus $\mathfrak{S}_2/\Gamma'$ is a smooth manifold, all these bundles are flat fiber bundles over a manifold. But if there is torsion, the quotient $\mathfrak{S}_2/\Gamma'$ is only an orbifold. Therefore, $\mathfrak{S}_2 \times_{\Gamma'} \mathbb{M}$ is by construction a flat orbifold bundle over $\mathfrak{S}_2/\Gamma'$ of rank $\operatorname{rank}(\mathbb{M})$. In the same way $\mathbb{W} \times_{\Gamma'} \mathbb{M}$ is a flat orbifold bundle over $\mathbb{W}/\Gamma'$. For some additional facts on these bundles see e.g. [Yas05; Section 8.3].

One can use this point of view to prove the following proposition.

Proposition 12.1.2: *We have*

$$r_{\Gamma'}^* \mathcal{M} = \mathcal{M}_{\mathbb{W}},$$

where $r_{\Gamma'}^* \mathcal{M}$ is the push-forward of the sheaf $\mathcal{M}$ by the restriction map from (12.2).

The proof is identical to the proof of [Yas05; Prop. 8.3.1], which handles an analogous statement for different groups.

Before we proceed with a closer look on the functor $\mathrm{sh}_{\Gamma'}$ (12.4) we have to introduce a ring extension for a – finite or infinite – discrete group Γ'. We set

$$D_{\Gamma'} = \prod_{\substack{p \text{ prim},\\ \exists \Gamma_1 \subset \Gamma: \\ p \mid \# \Gamma_1}} p. \tag{12.7}$$

It is easy to see that if $\Gamma' \subset \Gamma$ is an arithmetic subgroup then $D_{\Gamma'}$ is the product of all primes which divide the order of a stabilizer of some point $x \in \mathfrak{S}_2$. For our standard example, the Siegel modular group Γ, we have

$$D_\Gamma = 2 \cdot 3 \cdot 5 = 30, \tag{12.8}$$

because we know by Corollary 6.3.7 that the only prime orders of elements in Γ are $2, 3$, and 5.

Definition 12.1.3: *Let $\mathbf{R}$ be a commutative ring with one and Γ' a discrete subgroup of Γ, then we define a new ring by $\mathbf{R}_{\Gamma'} = \mathbf{R}[1/D_{\Gamma'}]$.*

The most important example of such a ring is the ring $\mathbb{Z}_\Gamma = \mathbb{Z}[1/30]$, i.e. $\mathbf{R} = \mathbb{Z}$ and $\Gamma' = \Gamma$. Rings of this type are important for several reasons, which we explain later.

Let $\Gamma' \subset \Gamma$ be an arithmetic group and $\mathbb{M} \in \mathbf{Obj}\left(\mathbf{Mod}_{\Gamma',\mathbf{R}}\right)$ for a commutative ring with one $\mathbf{R}$. Then $\mathrm{sh}_{\Gamma'}(\mathbb{M})$ is a by definition a sheaf of Γ'-$\mathbf{R}$-modules on $\mathfrak{S}_2$. If the group Γ' acts torsion free on $\mathfrak{S}_2$, which is not the case for $\Gamma' = \Gamma$, the quotient $\mathfrak{S}_2/\Gamma'$ is a manifold and, independent of $\mathbb{M}$, the sheaf $\widetilde{\mathbb{M}}$ is a local system.

Definition 12.1.4 (Local System)**:** *A sheaf $\mathcal{M}$ on a topological space $\mathcal{X}$ is called* **locally constant sheaf** *or* **local system** *if for all $x \in \mathcal{X}$ there is a neighbourhood U containing x such that $\mathcal{M}|_U$ is a constant sheaf.*

Some authors do not differentiate between local systems and orbilocal systems.

We want to have a closer look at stalks and sections of orbilocal systems. Let us start with the global sections of the sheaf $\widetilde{\mathbb{M}}$. We find immediately by its definition that

$$\widetilde{\mathbb{M}}(\mathfrak{S}_2/\Gamma') = \mathbb{M}^{\Gamma'} \tag{12.9}$$

because a locally constant map $f : \mathfrak{S}_2 \to \mathbb{M}$ is necessarily constant, since $\mathfrak{S}_2$ is contractible, and has to be invariant under Γ'. Let us now consider the stalks of $\widetilde{\mathbb{M}}$ in a point $x \in \mathfrak{S}_2/\Gamma'$. Then we have

$$\widetilde{\mathbb{M}}_x = \mathbb{M}^{\Gamma'_{\widetilde{x}}}, \tag{12.10}$$

where $\widetilde{x} \in \pi^{-1}(x) \subset \mathfrak{S}_2$ is a chosen lift for x and $\Gamma'_{\widetilde{x}}$ is the stabilizer of this point in Γ'. Note that this identification depends on the choice of the lift $\widetilde{x}$, but the resulting invariants are isomorphic for different choices of $\widetilde{x}$. Now there is a difference between the cases where $\Gamma' \subset \Gamma$ acts with and without torsion on $\mathfrak{S}_2$. If Γ' acts torsion free on $\mathfrak{S}_2$ then we know that the stabilizer of each point $\widetilde{x} \in \mathfrak{S}_2$ is trivial, i.e. equal to $\Gamma'_{\widetilde{x}} = \langle \pm \mathrm{Id} \rangle$ for all $\widetilde{x} \in \mathfrak{S}_2$, and therefore we get for all $x \in \mathfrak{S}_2/\Gamma'$ that

$$\widetilde{\mathbb{M}}_x = \mathbb{M}^{\langle \pm \mathrm{Id} \rangle}. \tag{12.11}$$

In other words, all stalks are equal.[1] This shows also that in the torsion free case the sheaves $\widetilde{\mathbb{M}}$ are (locally) constant.

If there is torsion the situation is slightly different. The $\widetilde{\mathbb{M}}$ are still sheaves, but they are not locally constant any more. The stalks will change from point to point whenever the stabilizer of a lift of the points changes because of (12.10).

Definition 12.1.5: *A sheaf $\mathscr{F}$ on a topological space $\mathcal{X}$ is called* **weakly locally constant** *or* **w-locally constant** *if for each point $x \in \mathcal{X}$ there is a neighbourhood $U_x \subset \mathcal{X}$ of x such that the map*

$$f_{U_x,x} : \mathscr{F}(U_x) \to \mathscr{F}_x$$

given by taking the germ of a section is an isomorphism.

[1] This is what we expected since locally constant sheaves on connected spaces are constant.

It is not difficult to see that orbilocal systems as well as local systems are weakly locally constant, because Γ' acts properly discontinuously on $\mathfrak{S}_2$. We will consider this isomorphisms in more detail later in Section 13.1 on constructible sheaves.

Now it is easy to see that the functor $\mathrm{sh}_{\Gamma'}$ can in general not be exact on the category $\mathbf{Mod}_{\Gamma'}$. The functor is certainly exact if Γ' acts torsion free. If this is not the case we can introduce the (right) derived functors $\mathrm{R}^q\,\mathrm{sh}_{\Gamma'}$. The higher images of these functors $\mathrm{R}^q\,\mathrm{sh}_{\Gamma'}(\mathbb{M})$ are sheaves on $\mathfrak{S}_2/\Gamma'$ which are concentrated on the fixed point sets of Γ'. Moreover, we get (see e.g. [Har05; p. 54]):

Proposition 12.1.6: *The derived functor $\mathrm{R}^q\,\mathrm{sh}_{\Gamma'}(\mathbb{M})$ for $q > 0$ is annihilated by some power of the integer $D_{\Gamma'}$.*

We will see some consequences of this proposition in the next section.

12.2. Cohomology of Arithmetic Groups

Let $\Gamma' \subset \Gamma$ be an arithmetic group and $\mathbb{M}$ any Γ'-module (later we will restrict ourselves to Γ'-$\mathbb{Z}_{\Gamma'}$-modules). The cohomology groups related to Γ' we are interested in are, as mentioned in the introduction of this chapter, the sheaf cohomology groups

$$H^q\big(\mathfrak{S}_2/\Gamma', \mathrm{sh}_{\Gamma'}(\mathbb{M})\big) = H^q(\mathfrak{S}_2/\Gamma', \mathcal{M})$$

defined in Section 5.1, where the coefficient sheaf $\mathcal{M} = \mathrm{sh}_{\Gamma'}(\mathbb{M})$ is given by the orbilocal system associated to the module $\mathbb{M}$. If we speak of the cohomology of an arithmetic group we will mean in the following this cohomology group. This is justified by the equalities between group cohomology and cohomology of sheaves we present in this section (see Theorems 12.2.1 and 12.2.3). To understand this, we need some preparation.

We already observed in (12.9) that

$$H^0(\mathfrak{S}_2/\Gamma', \mathcal{M}) = \mathcal{M}(\mathfrak{S}_2/\Gamma') = \mathbb{M}^{\Gamma'}. \tag{12.12}$$

By Proposition 12.1.2 this descends to the sheaf $\mathcal{M}_{\mathbb{W}}$ on the retraction, i.e.

$$H^0(\mathbb{W}/\Gamma', \mathcal{M}_{\mathbb{W}}) = \mathcal{M}_{\mathbb{W}}(\mathbb{W}/\Gamma') = \mathbb{M}^{\Gamma'}. \tag{12.13}$$

Furthermore, we mentioned in the previous section that the functor $\mathrm{sh}_{\Gamma'}$ is not always exact, such that we have to consider the derived functors $\mathrm{R}^q\,\mathrm{sh}_{\Gamma'}(\mathbb{M})$ for $q > 0$, which are according to Proposition 12.1.6 annihilated by a power of $D_{\Gamma'}$. We have in this cases the well known spectral sequence (comp. [Gro57b, Gro57a; Chapitre V.])

$$H^p(\mathfrak{S}_2/\Gamma', \mathrm{R}^q\,\mathrm{sh}_{\Gamma'}(\mathbb{M})) \Rightarrow H^n(\Gamma', \mathbb{M}), \qquad n = p + q, \tag{12.14}$$

which is a useful tool for the study of cohomology and well defined in all its terms under the assumptions from above. Some further analysis of this spectral sequence leads to the following theorem, which uses the hypercohomology of sheaves (see [Har05; 2.9.3]):

Theorem 12.2.1: *Let $\mathbb{M}$ be a Γ'-module, then we have the following identity:*

$$H^n(\Gamma', \mathbb{M}) = \bigoplus_{p+q=n} H^p(\mathfrak{S}_2/\Gamma', \mathrm{R}^q \operatorname{sh}_{\Gamma'}(\mathbb{M})),$$

where $H^0\big(\operatorname{sh}_{\Gamma'}(\mathbb{M})\big) = \operatorname{sh}_{\Gamma'}(\mathbb{M})$ and the $H^q(\operatorname{sh}_{\Gamma'}(\mathbb{M}))$ for $q > 0$ are sheaves whose stalks $H^q(\operatorname{sh}_{\Gamma'}(\mathbb{M}))_x$ in $x \in \mathfrak{S}_2/\Gamma'$ are $H^q(\Gamma'_x, \mathbb{M})$.

This yields the following proposition:

Proposition 12.2.2: *Let $\mathbf{R}$ be a commutative ring with one. If the orders of stabilizers in Γ' are invertible in $\mathbf{R}$, i.e. $\mathbf{R} = \mathbf{R}_{\Gamma'}$, we have that for $q > 0$*

$$\mathrm{R}^q \operatorname{sh}_{\Gamma'}(\mathbb{M}) = 0.$$

Furthermore, we get from the edge homomorphism of (12.14) a natural homomorphism

$$\alpha_{\Gamma'} : H^q(\Gamma', \mathbb{M}) \to H^q(\mathfrak{S}_2/\Gamma', \mathcal{M}). \tag{12.15}$$

which is in general neither injective nor surjective (comp. [Har08a; p. 32] or [Har05; 2.9.1]). But we have:

Theorem 12.2.3: *Let $\Gamma' \subset \Gamma$ be an arithmetic subgroup and $D_{\Gamma'}$ the integers as above. Then $\alpha_{\Gamma'}$ is an isomorphism is an isomorphism if the map $m \mapsto D_{\Gamma'} \cdot m$ is an automorphism of the module $\mathbb{M}$. In the other cases $\mathbf{ker}\,(\alpha_{\Gamma'})$ and $\mathbf{coker}\,(\alpha_{\Gamma'})$ are annihilated by a power of $D_{\Gamma'}$.*

From this we can conclude immediately:

Corollary 12.2.4: *If $\mathbb{M} \in \mathbf{Obj}\left(\mathbf{Mod}_{\Gamma', \mathbf{R}_{\Gamma'}}\right)$ we have*

$$H^q(\Gamma', \mathbb{M}) = H^q(\mathfrak{S}_2/\Gamma', \mathcal{M}).$$

This is of course a weaker result than the statement (12.14) on spectral sequences or Theorem 12.2.1, but it is somehow easier to handle in general. This is one of those reasons we announced earlier to consider later on for the cohomology instead of Γ'-$\mathbb{Z}$-modules Γ'-$\mathbb{Z}_{\Gamma'}$-modules.

Finally we want to close this section with the famous lemma of Shapiro from 1961, which connects the cohomology of an arithmetic group with the cohomology of a subgroup. Although the result is know as Shapiro's Lemma it is originally stated by Benno Eckmann already in 1953. Let $\Gamma'' \subset \Gamma' \subset \Gamma$ and $\mathbb{M}$ a Γ''-module, then the induced module is defined to be

$$\begin{aligned} \operatorname{Ind}_{\Gamma''}^{\Gamma'}(\mathbb{M}) &:= \{f : \Gamma' \to \mathbb{M} : f(\gamma'' \cdot \gamma') = \gamma'' \cdot f(\gamma') \quad \forall \gamma' \in \Gamma', \gamma'' \in \Gamma''\} \\ &= \bigoplus_{\gamma \in \Gamma'/\Gamma''} \gamma \cdot \mathbb{M}, \end{aligned} \tag{12.16}$$

which is a then Γ'-module. Then we have (see e.g. [Har05; 2.9.5])

Lemma 12.2.5 (Shapiro's Lemma: A. Shapiro, 1961; B. Eckmann, 1953)**:**

$$H^q\left(\mathfrak{S}_2/\Gamma'', \operatorname{sh}_{\Gamma''}(\mathbb{M})\right) = H^q\left(\mathfrak{S}_2/\Gamma', \operatorname{sh}_{\Gamma'}\left(\operatorname{Ind}_{\Gamma''}^{\Gamma'}(\mathbb{M})\right)\right).$$

12.3. Vanishing Results

There are several reasons why vanishing results for the full cohomology or some parts are interesting. First of all, if we already know that something is zero we do not need to calculate the related groups. But mathematicians like to compute zero in ways as complicated as possible and one can use the comparison between the computed result and the expected one to check the used algorithms for their correctness (or the correctness of the implementation of the used algorithms).

Furthermore, we can get some information on the inside of the cohomology if we know that some parts, say the inner cohomology are vanishing, because in that cases we can conclude that the computed cohomology belongs to the complement.

Last but not least the vanishing of the cohomology by Theorem 12.3.4 was the motivation for the search for a retract in Part II. As in the previous section we state the results only in case of the symplectic group $\mathscr{G}/\mathbb{Q} = \mathbf{Sp}_2/\mathbb{Q}$.

12.3.1. Weights

12.3.1.1. Trivial Representation and Global Sections

Let $\mathbb{M}_0$ be the trivial representation of the symplectic group defined over the integers. By definition $\mathbb{M}_{0,0}$ is the highest weight representation of weight $0 = 0 \cdot \omega_1 + 0 \cdot \omega_2$. In particular, we know that $\mathbb{M}_0$ is an irreducible representation (for the algebraic group), therefore there are only two possible invariant subspaces of $\mathbb{M}_0 \otimes_{\mathbb{Z}} \mathbb{C}$ under $\mathbf{Sp}_2(\mathbb{C})$, namely $\mathbb{M}_0 \otimes_{\mathbb{Z}} \mathbb{C}$ itself or 0. Furthermore, the set of invariants $(\mathbb{M}_0 \otimes_{\mathbb{Z}} \mathbb{C})^{\mathbf{Sp}_2(\mathbb{C})}$ is known to be an invariant subspace. It is easy to check that

$$(\mathbb{M}_0 \otimes_{\mathbb{Z}} \mathbb{C})^{\mathbf{Sp}_2(\mathbb{C})} = \mathbb{M}_0 \otimes_{\mathbb{Z}} \mathbb{C} \tag{12.17}$$

and that for all weights $\lambda \neq 0$

$$(\mathbb{M}_\lambda \otimes_{\mathbb{Z}} \mathbb{C})^{\mathbf{Sp}_2(\mathbb{C})} = 0. \tag{12.18}$$

One immediately finds that from this it follows that

$$\mathbb{M}_\lambda^{\mathscr{G}} = \begin{cases} \mathbb{M}_0 & \text{if } \lambda = 0 \\ 0 & \text{if } \lambda \neq 0. \end{cases} \tag{12.19}$$

Since $\Gamma = \mathbf{Sp}_2(\mathbb{Z})$ is Zariski-dense in $\mathscr{G}$ it follows:

Theorem 12.3.1: *Let λ be an integral weight and $\mathbb{M}_\lambda$ its associated representation module defined over the integers, then*

$$H^0\left(\mathfrak{S}_2/\Gamma, \widetilde{\mathbb{M}_\lambda \otimes_{\mathbb{Z}} \mathbb{Z}_\Gamma}\right) = \begin{cases} (\mathbb{M}_0 \otimes_{\mathbb{Z}} \mathbb{Z}_\Gamma)^\Gamma & \textit{for } \lambda = 0 \\ 0 & \textit{for } \lambda \neq 0. \end{cases}$$

Note: The invariants of the module in Theorem 12.3.1 under the discrete subgroup Γ need not to be neither the full module nor the zero module, since it is in general only irreducible for the complex representation.

12.3.1.2. Odd Weights

It is well known that spaces of modular forms – Siegel modular forms as well as elliptic modular forms – with respect to an arithmetic group Γ' vanish if $-\mathrm{Id} \in \Gamma'$ acts trivially, whereas the matrix representation of $-\mathrm{Id}$ is not the identity. There is a similar phenomenon for the cohomology. According to the definition of an orbilocal system $\widetilde{\mathbb{M}_{m,n}}$ associated to a highest weight module $\mathbb{M}_{m,n} \in \mathbf{Obj}\left(\mathbf{Mod}_\Gamma\right)$ in (12.3) we have to look for locally constant functions f on $\pi^{-1}(U)$ for an open set $U \subset \mathfrak{S}_2/\Gamma$ with values on $\mathbb{M}_{m,n}$ which satisfy

$$f\left(R_\gamma(x)\right) = \rho'_{m,n}(\gamma) \cdot f(x). \tag{12.20}$$

We have always $R_{-\mathrm{Id}}(x) = x$ whereas

$$\rho'_{m,n}(-\mathrm{Id}) = \begin{cases} -1 & \text{if } m \equiv 1 \mod 2 \\ +1 & \text{if } m \equiv 0 \mod 2. \end{cases} \tag{12.21}$$

Therefore, we get

Lemma 12.3.2: *Let $\lambda = m \cdot \omega_1 + n \cdot \omega_2$, then we have*

$$\widetilde{\mathbb{M}_\lambda} = 0 \tag{12.22}$$

if $m \equiv 1 \mod 2$.

This leads immediately to

Corollary 12.3.3: *Let $\lambda = m \cdot \omega_1 + n \cdot \omega_2$, then we have*

$$H^q\left(\mathfrak{S}_2/\Gamma, \widetilde{\mathbb{M}_\lambda}\right) = 0 \tag{12.23}$$

for all q if $m \equiv 1 \mod 2$.

The same stays true if one replaces Γ by a subgroup containing $-\mathrm{Id}$ or if we replace the module $\mathbb{M}_\lambda$ by $\mathbb{M}_\lambda \otimes_{\mathbb{Z}} \mathbf{R}$ for some ring $\mathbf{R} \supseteq \mathbb{Z}$.

12.3.2. Virtual Cohomological Dimension

The main motivation to construct retracts of (Siegel) modular varieties for a given algebraic group $\mathscr{G}/\mathbb{Q}$ of lower dimension than $\dim \mathbb{X}_{\mathscr{G}}$ is, as already discussed in Section 7.2, to get an as simple as possible model for the variety and reduce the computational costs for e.g. the cohomology. The main result which lets us hope that this is possible is the vanishing of the cohomology groups above a certain limit, the so called **virtual cohomological dimension** (vcd). Recall that vcd was defined in (6.7) to be

$$\mathrm{vcd}\left(\Gamma\right) := \dim_{\mathbb{R}}\left(\mathbb{X}\right) - \mathrm{rank}_{\mathbb{Q}}\left(\Gamma\right).$$

There is an equivalent purely topological definition due to the following theorem of Armand Borel and Jean Pierre Serre [BS73; Theorem 11.4.4].

Table 12.1.: *The virtual cohomological dimension for subgoups of $\mathbf{Sp}_g(\mathbb{Z})$ for $n \leq 9$*

g	1	2	3	4	5	6	7	8	9
$\dim_{\mathbb{R}} \mathbb{X}_{\mathbf{Sp}_g}$	2	6	12	20	30	42	56	72	90
$\operatorname{vcd}\left(\mathbf{Sp}_g(\mathbb{Z})\right)$	1	4	9	16	25	36	49	64	81

Theorem 12.3.4 (Borel, Serre,1973)**:** *For an (not necessarily torsionfree) arithmetic subgroup $\Gamma \subseteq \mathscr{G}(\mathbb{Z})$ it is $H^i(\Gamma, \mathbb{M}) = 0$ for $i > \operatorname{vcd}(\Gamma)$ independent of $\mathbb{M}$.*

In other words, the vcd is the highest degree of cohomology where independent of the chosen coefficient system the cohomology is non-vanishing.

It is important to emphasize that the virtual cohomological dimension depends only on the underlying algebraic group $\mathscr{G}/\mathbb{Q}$ and not on the particular arithmetic group $\Gamma \subset \mathscr{G}(\mathbb{Z})$.

Example 12.3.5 (Symplectic Group)**.** In case of the symplectic group, i.e. $\mathscr{G}/\mathbb{Q} = \mathbf{Sp}_g/\mathbb{Q}$ we know that (see e.g. [BL04; 8.1])

$$\dim_{\mathbb{R}} \mathbb{X}_{\mathbf{Sp}_g} = g(g+1) \qquad \text{and} \qquad \operatorname{rank}_{\mathbb{Q}} \mathbf{Sp}_g = g.$$

Therefore, we get

$$\operatorname{vcd}\left(\mathbf{Sp}_g(\mathbb{Z})\right) = g^2.$$

12.4. Computing Global Sections

Computing global sections of the orbilocal sheaf $\widetilde{\mathbb{M}}$ associated to a highest weight module $\mathbb{M}$ (see equation 12.12) is essentially the same as the computation of invariants under the (finite) stabilizers of cells, which we discussed in Sections 11.2.5 and 11.2.6. But different to that we need to compute global sections to compute the invariants under the full group Γ, or equivalently the invariants under the (finitely many) generators of the group (comp. Theorem 11.2.1 and equation (11.25)). Here is a simple example of the usage of our program to compute $\mathbb{M}_{1,0}^{\Gamma}$, i.e. the invariants for the highest weight module of weight $\lambda = (1,0)$:

```
sage: L = SimpleLieAlgebra('C',2, base_ring=QQ)
sage: G = GeneratorsSymplecticGroup(2, symplectic_type=1)
sage: G = [symplectic2symplectic(g,type_from=1,type_to=0) for g in G]
sage: M = p.HighestWeightModule(L,[1,0])
sage: invariants(M,G, only_generators=False)
#Control messages deleted
Vector space of degree 4 and dimension 0 over Rational Field
Basis matrix:
[]
```

For all other weights we obtain the same result except for the weight $\lambda = (0,0)$. There we get:

```
...
sage: M = p.HighestWeightModule(L,[0,0])
sage: invariants(M,G, only_generators=False)
#Control messages deleted
Vector space of degree 1 and dimension 1 over Rational Field
Basis matrix:
[1]
```

This is exactly what we expected, namely that all invariants of highest weight modules of weight λ are equal to zero except for $\lambda = (0,0)$, where we have a one-dimensional space of invariants (comp. Section 12.3.1.1).

It is strongly recommended to use the option `only_generators=False`, because we know that the list of elements passed to the function is a set of generators and we can avoid the computation of generators. In the current version of the function `getgenerators` elements of infinite order are not supported. Therefore, we have to pass the list of generators of Γ and force the program not to try to find a set of generators, because in the best case the function `getgenerators` will send the input list (or a sublist of it) to the computation of kernels (comp. Section 11.2.5), in the worst case is can raise an exception.

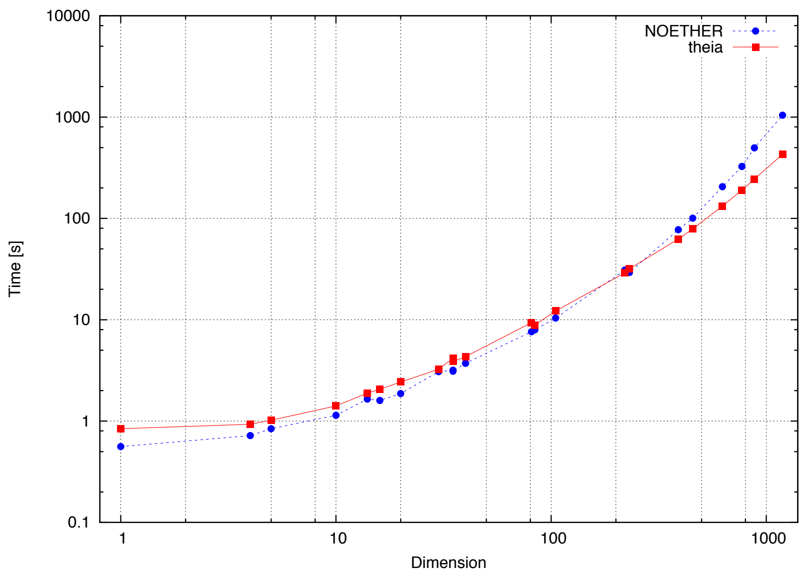
As in some other cases, we here have to pass the generators of the symplectic group to the function `invariants` in the right form, because the representation of symplectic group elements in context of highest weight modules uses a symplectic form which differs from the symplectic form used in the computation of stabilizers of cells (comp. Section 6.3.1.3). Since the function `Invariants` is implemented independent of both constructions, we have to translate the group from one world into the other, which is quite easy.

In Table 12.2 and Figure 12.1 we state the runtime for a collection of highest weight modules on the two machines `NOETHER` and `theia`. For small weights my local iMac at home, `NOETHER`, is the faster one. But at some point the computation becomes slower than on `theia`. This is probably due to the equipment with memory. `theia` has much more RAM (320GB compared to 64GB), much more L2 cache (6MB compared to 512kB) and an L3 cache, which `NOETHER` does not have (see Section 2.2). As one might expect we find that the larger the considered modules are the larger is the memory one needs to perform the operations. Other factors different from the memory consumption are only secondary effects. The clock rates of the two platforms are not that different to explain a difference. An other effect which is somehow difficult to estimate is the different structure of the arithmetic units in the CPUs. We have the feeling that – as long as the memory consumption is not relevant – in most operations which involve integer or rational (and not floating point) operations the iMac with its Intel processors is significantly faster than the AMD processors of `theia`. If we replace the default operations over the integers or the rational numbers by operations over the real numbers, i.e. we use then floating point arithmetic, the speed of `theia` increases. Probably the program uses or does not use integer arithmetic circuits on the CPU. In any case, for larger modules, which are the cases we are interested in (comp. Section 1.2), one should use `theia`.

Table 12.2.: *Computation of global sections. The computations are done without precomputation, since for a single computation this makes no difference. All timings are done on* `theia` *at the MPI for Mathematics and on* `NOETHER`*, my personal computer. They are each the arithmetic mean of 5 runs of the program.*

Weight	Dimension	Time[s]	
		`NOETHER`	`theia`
(0,0)	1	0.56	0.84
(1,0)	4	0.72	0.93
(0,1)	5	0.84	1.02
(2,0)	10	1.14	1.41
(0,2)	14	1.65	1.89
(1,1)	16	1.60	2.06
(3,0)	20	1.87	2.43
(0,3)	30	3.07	3.25
(2,1)	35	3.18	4.17
(4,0)	35	3.11	3.88
(1,2)	40	3.72	4.32
(2,2)	81	7.62	9.36
(6,0)	84	7.94	8.86
(4,1)	105	10.40	12.20
(4,2)	220	30.92	29.00
(6,1)	231	29.3	31.80
(4,3)	390	77.34	62.65
(6,2)	455	100.54	78.65
(4,4)	625	205.83	132.70
(6,3)	770	325.37	189.63
(5,4)	880	497.25	244.53
(6,4)	1190	1044.76	430.06

Figure 12.1.: *Plotted data from Table 12.2.*



13. Sheaf Cohomology of the Decomposition

Non est ad astra mollis e terris via.

(Seneca)

This chapter is dedicated to the introduction of a useful tool which one can use to compute the cohomology groups $H^q(\Gamma, \mathbb{M})$ for a Γ-$\mathbb{Z}_\Gamma$-module $\mathbb{M}$. Remember, $\mathbb{Z}_\Gamma$ was defined to be the smallest ring extension of $\mathbb{Z}$ such that all orders of elements in Γ are invertible (see Definition 12.1.3). In this chapter we look at the problem from a more general point of view.

It is the goal of this chapter to give an explicit way to compute the cohomology of a (orbi-)cell decomposition with coefficients coming from constant sheaves (or orbilocal systems).

The starting point is a nice[1] decomposition of a nice[2] topological space $\mathcal{X}$ together with the action of a discrete group $\mathbf{G}$ of automorphism on it. We want to emphasize that we do not assume that the group $\mathbf{G}$ acts torsion free on the topological space $\mathcal{X}$. This decomposition can be for example the decomposition, which we constructed in Chapter 9. Let us assume in this chapter that all occurring modules are defined over the integers, i.e. $\mathbb{M} \in \mathbf{Obj}\left(\mathbf{Mod}_{\mathbb{Z}}\right)$, and set

$$\mathbb{M}_{\mathbf{G}} := \mathbb{M} \otimes_{\mathbb{Z}} \mathbb{Z}_{\mathbf{G}}. \tag{13.1}$$

We have then that $\mathbb{M}_{\mathbf{G}} \in \mathbf{Obj}\left(\mathbf{Mod}_{\mathbf{G},\mathbb{Z}_{\mathbf{G}}}\right)$. Denote by $\mathcal{M}_{\mathbf{G}} = \widetilde{\mathbb{M}_{\mathbf{G}}}$ the corresponding orbilocal system (comp. Section 12.1). This assumption simplifies some things: For example, we have by Theorem 12.2.3 that[3]

$$H^q(\mathbf{G}, \mathbb{M}_{\mathbf{G}}) = H^q(\mathcal{X}/\mathbf{G}, \mathcal{M}_{\mathbf{G}}).$$

To compute cohomology groups as on the right side of the equation above we use a concept called constructible sheaves which allows to describe the sheaf in terms of local data. This construction is known in various contexts [AGV73, Bor84, KS90, Sch03, Dim04]. The rough idea of constructible sheaves is to glue together locally defined sheaves which are locally constant on some kind of partition to a sheaf on the global space. Good references for the ideas leading to our approach are in particular [KS90, Dim04] and in some cases [Sch03], which handles the topic in a more general framework than the two sources mentioned before.

The idea of constructible sheaves is due to Alexander Grothendieck, who introduced the notion in the context of étale cohomology (see e.g. [AGV23; exposé IX by M. Artin]). After their invention they became very popular because of their connection to holonomic $\mathcal{D}$-modules found by the Japanese mathematician Masaki Kashiwara [Kas75], and some years

[1] Nice means here in particular that the decomposition has to be locally finite, regular, compatible with the group action and fine enough. We come back to the details of this if we introduce w-constructible sheaves.

[2] Nice means here that the space should be a simply connected manifold.

[3] We stated the theorem there only for the symplectic group. However the theorem is also valid in a more general context.

later because of their connection to intersection cohomology indicated by Mark Goresky and Robert MacPherson [GM83]. These new developments linked the construction to lots of other areas and provided a new powerful tool to describe and construct sheaves in a simple way. It turned out that the different kinds of constructible sheaves have many good properties. The different notions differ in detail, but lead in most cases to the so called cohomological constructible sheaves [KS90; III.4]. Special cases of this are $\mathbb{C}$-constructible sheaves (see [KS90; VIII.5] and [Dim04; 4.1]), where the underlying space is complex analytic and the decomposition is a decomposition into complex analytic subspaces, and S-constructible sheaves [KS90; VIII.1] for a simplicial complex S. We adapt these two notions to our context.

In our case we have to keep in mind that first we have for $\mathbb{W}$ only a CW-complex and for $\mathbb{W}/\Gamma$ even only an orbicell decomposition and not a simplicial complex. Furthermore, the (orbi-)cells do not need to be complex analytic or Zariski open subspaces. Therefore, in Section 13.1 we start with the definition of constructible sheaves we want to use, before we extend the notion of constructible sheaves also to sheaves together with an additional action of a discrete group in Section 13.2. There is a different construction using $\mathbf{G}$-spaces and group action in the context of constructible sheaves, which is due to Alexander Grothedieck (see e.g. [Gro57a, Gro57b] or [Sch03; Chap. 3]). These sheaves are called equivariant constructible sheaves. However these constructions turned out to be not adaptable to our situation in detail. Therefore, we developed a new kind of sheaf which we will call an orbi-constructible sheaf, which is exactly the kind of sheaf we need during our computations (see e.g. Section 13.3). Since this tool seems to us quite useful, we introduce this tool in a slightly more general context.

After the introduction of (w-)constructible and orbi-(w-)constructible sheaves we construct some resolutions. This will be done in two steps:

Step 1 Construct an acyclic resolution for a constant sheaf on $\mathcal{X}$ and the associated orbilocal system on $\mathcal{X}/\mathbf{G}$ using two families of short exact sequences. See Section 13.3.1.

Step 2 Construct a further resolution of the one from Step 1 using long exact cohomology sequences from (5.4). See Section 13.3.5.

This will reduce the computation of cohomology groups to the computation of cokernels of maps between global sections of some sheaves on $\mathcal{X}$ or $\mathcal{X}/\mathbf{G}$ respectively. Essentially it is also possible to compute the cohomology using only the result of the first step, i.e. the acyclic resolution. However it turns out that it is better to do it in the way indicated above because the maps are easier to handle.

13.1. Constructible Sheaves

The definition of constructible sheaves on orbifolds of the form $\mathcal{X}/\mathbf{G}$, where $\mathcal{X}$ is a smooth Riemannian manifold and $\mathbf{G}$ is a finite discrete group of isometries of $\mathcal{X}$, will be done in two steps. In this section we first introduce constructible sheaves on a locally compact topological space $\mathcal{X}$ which has a locally finite cell decomposition. In Section 13.2 we extend the construction to quotients with an orbicell decomposition. We will later show that all sheaves occurring in the resolution mentioned already in the introduction of this chapter (see Section 13.3), which we need to compute the cohomology groups, are of this form (see Lemma 13.3.2).

13.1.1. Definition w-Constructible Sheaves

Let $\mathcal{X}$ be a locally compact topological space with a locally finite cell decomposition

$$\mathcal{X} = \bigsqcup_{i \in \mathcal{I}} \mathcal{X}_i = \bigsqcup_{\mathcal{X}_i \in \mathcal{P}} \mathcal{X}_i. \tag{13.2}$$

For simplicity we assume that the closure of each cell $\mathcal{X}_i$ in $\mathcal{P}$ is a union of $\mathcal{X}_i$ with finitely many lower dimensional cells in $\mathcal{P}$. Furthermore, we will assume that for each point $x \in \mathcal{X}$ there is a neighbourhood $U \ni x$, such that for all cells the intersection with U is either empty or simply connected.[4] In particular, if we skip the second condition, many of the proofs become much more complicated or one has to weaken the results somehow. It is always possible to replace the partition $\mathcal{P}$ by a suitable refinement for which this condition holds. Note that these two assumptions are fulfilled for the decomposition of $\mathbb{W}$ (comp. Section 9.1). That is not true for the decomposition of quotients of $\mathbb{W}$, but this is not the condition we need and therefore does not effect our construction at all.

Definition 13.1.1 (Constructible Sheaf)**:**

a) *A sheaf $\mathscr{F}$ of $\mathbb{Z}$-modules on $\mathcal{X}$ is called* **weakly constructible** *with respect to the partition $\mathcal{P}$ if the restriction $\mathscr{F}|_{\mathcal{X}_i}$ is a local system of $\mathbb{Z}$-modules. We will also say that $\mathscr{F}$ is* **weakly $\mathcal{P}$-constructible**, **w-$\mathcal{P}$-constructible** *or, if $\mathcal{P}$ is fixed, short* **w-constructible**.

b) *A sheaf $\mathscr{F}$ on $\mathcal{X}$ is called* **constructible** *with respect to $\mathcal{P}$ or* **$\mathcal{P}$-constructible** *if $\mathscr{F}$ is w-constructible and all stalks $\mathscr{F}_x$ are $\mathbb{Z}$-modules of finite type.*

The (weakly-)constructible sheaves on $\mathcal{X}$ form an abelian category, which is denoted by w-$\mathbf{Const}_{\mathcal{X}}$ resp. $\mathbf{Const}_{\mathcal{X}}$.

Later we will only consider sheaves whose stalks are $\mathbb{Z}$-modules of finite rank over $\mathbb{Z}$. Thus, in all these cases w-constructible sheaves are also constructible since the decomposition is locally finite. There are definitions of constructible sheaves in various contexts. These definitions differ depending on which kind of space the sheaves are defined on and for which kind of partition. For example, Grothendieck et al. considered in their famous Séminaire géométrie

[4]If this condition is not fulfilled, one has to add in the Definition 13.1.1 that the sheaf has to be weakly locally constant, since this cannot be concluded only from that definition.

algébrique (SGA) du Bois-Marie, which was held from 1963 to 1964, schemes with a locally closed decomposition into affine subschemes on which the sheaf is locally constant (see e.g. in SGA 4 [AGV23]), whereas others consider a locally compact topological space with a locally finite decomposition into analytic pieces or simplicial sets (see e.g. [Dim04]) or a filtration of that space (see e.g. [Bor84; p. 69] or [Sch03; p. 18]). In all cases the basic idea is the same. One fixes the stalks of the sheaves on a certain family of subsets. Hence, in nice cases most of the definitions coincide or are at least compatible in some sense because to be constructible or not depends only up to some level on the partition $\mathcal{P}$ (comp. Lemma 13.1.4).

The easiest example for constructible sheaves are locally constant sheaves of abelian groups (comp. Definition 12.1.4).

Example 13.1.2 (Locally Constant Sheaves). Let $\mathcal{X} = \bigsqcup_{i\in\mathcal{I}} \mathcal{X}_i$ be the decomposition of $\mathcal{X}$ into its connected components and $\mathbb{A}_i$ for $i \in \mathcal{I}$ a collection of $\mathbb{Z}$-modules, i.e. abelian groups. Then the associated **locally constant sheaf** or **local system** $\widetilde{\mathbb{A}_{\mathcal{X}}}$ has stalks

$$\widetilde{\mathbb{A}_{\mathcal{X}}}_x = \mathbb{A}_i \quad \text{for } x \in \mathcal{X}_i$$

and is therefore w-constructible. If all the $\mathbb{A}_i$ are of finite type $\widetilde{\mathbb{A}_{\mathcal{X}}}$ is constructible. In particular, the constant sheaves on a manifold are w-constructible.

The big advantage of using constructible sheaves over local systems is that this property is preserved under many operations of sheaves which is not valid for local systems.

Theorem 13.1.3 (Constructions):

a) Let $\mathscr{F}$ and $\mathscr{F}'$ be two (w-)constructible sheaves on $\mathcal{X}$ and $f \in \mathbf{Hom}\left(\mathscr{F}, \mathscr{F}'\right)$ a morphism of sheaves, then $\mathscr{F} \oplus \mathscr{F}'$, $\mathscr{F} \otimes \mathscr{F}'$, $\mathbf{ker}\,(f)$, $\mathbf{im}\,(f)$, and $\mathbf{coker}\,(f)$ are (w-)constructible.

b) If $0 \to \mathscr{F}' \to \mathscr{F} \to \mathscr{F}'' \to 0$ is an exact sequence of sheaves and two of the non-zero sheaves are (w-)constructible, then so is the third.

c) Let $\mathscr{F}$ be a (w-)$\mathcal{P}$-constructible sheaf on $\mathcal{X}$ and $f : \mathcal{X} \to \mathcal{Y}$ a continuous map which acts cellular, i.e. cells from the partition $\mathcal{P}$ of $\mathcal{X}$ are mapped onto cells from a partition $\mathcal{P}'$ of $\mathcal{Y}$, then $f_*\mathscr{F}$ is (w-)$\mathcal{P}'$-constructible. If $\mathcal{X}$ is a closed subset of $\mathcal{Y}$ and f an embedding one can skip the cellularity condition.[5]

d) Let $\mathscr{G}$ be a (w-)$\mathcal{P}$-constructible sheaf on $\mathcal{Y}$ and $f : \mathcal{X} \to \mathcal{Y}$ a continuous map then $f^*\mathscr{G}$ is (w-)$\mathcal{P}'$-constructible for some partition $\mathcal{P}'$ of $\mathcal{X}$. If f is an embedding and acts cellular, $f^*\mathscr{G}$ is also (w-)$\mathcal{P}$-constructible.

Proof. The proof of a) and b) is an easy application of the definition (comp. [Dim04; Ex. 4.1.3] or references there). The first part of c) follows immediately from the definitions. The second part can be found e.g. in [Dim04; Rem. 4.1.7. (ii)]. For a proof of the analogous statement to d) see [KS90; p. 347] and [Dim04; Rem. 4.1.7. (i)]. □

[5]In general $f_*\mathscr{F}$ is *not* a w-constructible sheaf, even if we assume that f is an embedding of a locally closed subspace. A counter example can be found in [Dim04; Rem. 4.1.7. (ii)]. The assumption of a cellular action is one way to ensure that $f_*\mathscr{F}$ is w-constructible but there are other reasonable assumptions on f and/or the spaces $\mathcal{X}$ and $\mathcal{Y}$ or their decompositions.

Lemma 13.1.4: *Assume we have a partition $\mathcal{P}$ for $\mathcal{X}$ and a refinement $\mathcal{P}'$ of this partition. If a sheaf $\mathscr{F}$ is (w-)$\mathcal{P}$-constructible then $\mathscr{F}$ is also (w-)$\mathcal{P}'$-constructible.*

This is more or less clear since if we decompose a cell into a union of smaller cells we do not change the stalks. Assume we have a constructible sheaf with respect to a partition $\mathcal{P}$. Then there will be a coarser partition $\mathcal{P}'$ on which the sheaf is already constructible, except $\mathcal{P}$ is maximal. This property is quite useful since we can assume that the partition is as good/regular as we want. If not, we can replace the partition always by a finer and sometimes also by a coarser partition.

Lemma 13.1.5 (Filtration)**:** *Let $\mathscr{F}$ be a (w-)$\mathcal{P}$-constructible sheaf on $\mathcal{X}$. Then there is for each $x \in \mathcal{X}$ an open neighbourhood U such that the restriction of $\mathscr{F}$ onto U has a finite filtration*

$$0 = \mathscr{F}_0 \subset \mathscr{F}_1 \subset \ldots \subset \mathscr{F}_m = \mathscr{F}\mid_U$$

where all sheaves $\mathscr{F}_k$ are (w-)$\mathcal{P}|_U$-constructible. All quotients $\mathscr{F}_{k+1}/\mathscr{F}_k$ are locally constant and of the form $i_*\mathscr{L}$ with $i : \sigma \cap U \hookrightarrow U$ the inclusion of the cell $\sigma \cap U$ into U and $\mathscr{L}$ is a local system on $\sigma \cap U$.

Proof. For details of the proof we refer to the proof of [Dim04; Cor.4.1.8], which is essentially the same. There we first have to replace the statement in Lemma 13.1.5 by the analogous statement for finite partitions, which is possible since the cellular decomposition of $\mathcal{X}$ is locally finite and thus the question is local. Further, we have to apply the results of Theorem 13.1.3 and a very similar construction to the one of the acyclic resolution in Section 13.3.1 to get the resolution in Lemma 13.1.5 and the claimed form of the sheaves there. □

The following theorem is the essential ingredient for our computations of constructible sheaves later on.

Theorem 13.1.6: *Let $\mathcal{X}$ be as above and $\mathscr{F}$ a w-constructible sheaf. Then for any $\sigma \in \mathcal{P}$ and any point $x_0 \in \sigma$ there is an open neighbourhood $U \subset \mathcal{X}$ of x_0 such that:*

a) the natural morphism $f_{U,x} : \mathscr{F}(U) \to \mathscr{F}_x$ induced by taking the germ of a section is an isomorphism for all $x \in \sigma \cap U$.

b) $H^q(U, \mathscr{F}\mid_U) = 0$ for all $q > 0$.

Proof. The proof is essentially the same as in [Dim04; Theo. 4.1.9] and [Sch03; Prop. 8.1.4]. □

If we remember the definition of w-locally constant sheaves (see Definition 12.1.5), we find immediately:

Corollary 13.1.7: *Any w-constructible sheaf is w-locally constant.*

Note: If we skip the additional assumption on the refinement of the decomposition from page 152 we have – as mentioned earlier –to include in the actual definition of w-constructibility of a sheaf (see Definition 13.1.1) that the sheaf has to be w-locally constant. Therefore, the condition to be w-constructible is stronger then only to be w-locally constant.

13.1.2. Construction of w-Constructible Sheaves

To give a method to construct w-constructible sheaves we first have to answer the following question:

Problem: Given a point $x \in \mathcal{X}$ and a w-constructible sheaf $\mathscr{F}$ on $\mathcal{X}$. How does a section $\mathscr{F}(U)$ over a given open neighbourhood $U \ni x$ and a stalk $\mathscr{F}_x$ of the the sheaf $\mathscr{F}$ in the point x look like?

We need to fix some additional notation. Let $\sigma \in \mathcal{P}$ be a cell. Then we set

$$\mathcal{U}(\sigma) := \left\{\sigma' \in \mathcal{P} : \overline{\sigma'} \cap \sigma \neq \emptyset\right\}. \tag{13.3}$$

Since the cell decomposition of $\mathcal{X}$ is locally finite we know that $\mathcal{U}(\sigma)$ is a finite union of cells. Further, $\mathcal{U}(\sigma)$ is the smallest open set in $\mathcal{X}$ which contains the cell σ and is a union of cells. Let $x_0 \in \mathcal{X}$, then there is a unique cell σ_0 such that $x_0 \in \sigma_0$. In this case we set

$$\mathcal{U}(x_0) = \mathcal{U}(\sigma_0). \tag{13.4}$$

If σ_0 is a top dimensional cell, i.e. σ_0 is open in $\mathcal{X}$, we find that

$$\mathcal{U}(\sigma_0) = \sigma_0. \tag{13.5}$$

We call $\mathcal{U}(\sigma)$ (or $\mathcal{U}(x_0)$) the (open) **star set** of σ (or x_0) with respect to the cellular decomposition $\mathcal{P}$. Using the definitions it is not difficult to see that:

Fact 13.1.8: *Let $\sigma, \tau \in \mathcal{P}$, then $\mathcal{U}(\sigma) \cap \tau \neq \emptyset$ if and only if $\overline{\tau} \cap \sigma \neq \emptyset$. In particular, if $\mathcal{U}(\sigma) \cap \tau \neq \emptyset$ then $\mathcal{U}(\sigma) \supset \tau$.*

and also

Fact 13.1.9: *Let $x \in \mathcal{X} = \bigcup \mathcal{X}_i$ be any point. Assume $\mathcal{U}(x) = \bigsqcup_{i=1}^k \mathcal{X}_i$, then for all open sets $U \subset \mathcal{X}$ with $x \in U$ we have $U \cap \mathcal{X}_i \neq \emptyset$ for all $i = 1, \ldots, k$.*

The stalks of a w-constructible sheaf are by definition constant on cells, therefore the only possibility to change the sections of such a sheaf over different open sets is that the set of cells which hit the open sets changes. Thus, we get

Lemma 13.1.10: *Let $U \subset U' \subset \mathcal{X}$ be two neighbourhoods of a point $x \in \mathcal{X}$ and $\mathscr{F}$ a w-constructible sheaf with respect to a partition $\mathcal{P}$, then*

$$\{\sigma \in \mathcal{P} : \sigma \cap U \neq \emptyset\} = \left\{\sigma \in \mathcal{P} : \sigma \cap U' \neq \emptyset\right\}$$

implies $\mathscr{F}(U) = \mathscr{F}(U')$.

Fix a cell $\sigma \in \mathcal{P}$ for the moment. We know by Theorem 13.1.6 and Corollary 13.1.7 that any w-constructible sheaf $\mathscr{F}$ is w-locally constant and that we can find for every point $x \in \sigma$ an open neighbourhood $U \subset \mathcal{X}$ of x and an isomorphism $f_{U,x} : \mathscr{F}(U) \to \mathscr{F}_x$. Assume U_x is maximal under all open neighbourhoods of x with this property.

Corollary 13.1.11: *We have $\mathcal{U}(x) := \mathcal{U}(\sigma) \subset U_x$ and $\mathscr{F}(U_x) = \mathscr{F}(\mathcal{U}(x))$.*

Proof. According to Fact 13.1.9 we know that every cell contained in $\mathcal{U}(x)$ has to intersect every neighbourhood of x and thus also U_x. By definition, $\mathcal{U}(x)$ is the smallest open neighbourhood of x which can be formed by gluing cells together. In particular, $U_x \cap \mathcal{U}(x)$ is non empty and open. We can now apply Lemma 13.1.10 twice. First, we take $U = U_x \cap \mathcal{U}(x)$ and $U' = U_x$. Then we get $\mathscr{F}(U_x \cap \mathcal{U}(x)) = \mathscr{F}(U_x)$. Now we take $U' = \mathcal{U}(x)$, then we conclude, since the cells which intersect U and U' are the same, that $\mathscr{F}(U_x \cap \mathcal{U}(x)) = \mathscr{F}(\mathcal{U}(x))$. Thus, $\mathscr{F}(U_x) = \mathscr{F}(\mathcal{U}(x))$ and so $\mathcal{U}(x) \subset U_x$, since we assumed that U_x is maximal. □

Remark: In most non-trivial examples, i.e. sheaves whose stalks are not constant over unions of different cells, we even have $U_x = \mathcal{U}(x)$.

We can directly conclude from the previous statements that

Corollary 13.1.12: *The restriction map* $\mathrm{res}_U^{U_x} = \mathrm{Id}$ *for any* $U \subset U_x$ *with* $x \in U$.

and further that

Corollary 13.1.13: *Let* $\sigma \in \mathcal{P}$ *be a cell,* $\mathscr{F}$ *be a* w*-constructible sheaf with respect to* $\mathcal{P}$ *and* $U \subset \mathcal{U}(\sigma)$ *an open set with* $U \cap \sigma \neq \emptyset$ *then* $\mathscr{F}(U) = \mathscr{F}(\mathcal{U}(\sigma))$.

We now have a closer look to the maps $f_{U,x}$.

Lemma 13.1.14: *Let* $f_{U,x} : \mathscr{F}(U) \to \mathscr{F}_x$ *with* $x \in U \subset U_x$ *be the isomorphism defined in Theorem 13.1.6. Let* $V \subset U_x$ *be an other neighbourhood of* x*, then* $f_{U,x} = f_{V,x} = f_{U_x,x}$.

Proof. We have the following commutative diagram

$$\begin{tikzcd}
\mathscr{F}(U) \arrow[dr, "f_{U,x}"] & \\
\mathscr{F}(U_x) \arrow[u, "\mathrm{res}_U^{U_x}"] \arrow[r, "f_{U_x,x}"] \arrow[d, "\mathrm{res}_U^{U_x}"'] & \mathscr{F}_x \\
\mathscr{F}(V) \arrow[ur, "f_{V,x}"'] &
\end{tikzcd} \tag{13.6}$$

where due to Corollary 13.1.12 $\mathrm{res}_U^{U_x} = \mathrm{res}_V^{U_x} = \mathrm{Id}$ and the other maps are isomorphisms. This shows the lemma. □

From now on we write for short $f_x := f_{U_x,x}$ if there is no need to specify the set.

Let now $y \in U_x$, then there is also an open neighbourhood $U_y \subset \mathcal{X}$ of y and an isomorphism $f_y : \mathscr{F}(U_y) = \mathscr{F}(U_x \cap U_y) \to \mathscr{F}_y$ (see Corollary 13.1.12). There is no reason that U_y has to have a non-trivial intersection with the cell which contains x. Then there is a natural map

$$\begin{tikzcd}
\mathscr{F}_x \arrow[r, "f_x^{-1}", "\sim"'] \arrow[rrr, bend right, "\Phi_{x,y}"'] & \mathscr{F}(U_x) \arrow[r, "\mathrm{res}^{U_x}_{U_x \cap U_y}"] & \mathscr{F}(U_y \cap U_x) = \mathscr{F}(U_y) \arrow[r, "f_y", "\sim"'] & \mathscr{F}_y
\end{tikzcd} \tag{13.7}$$

where $\mathrm{res}^{U_x}_{U_x \cap U_y}$ is the restriction map which we have since $\mathscr{F}$ is a sheaf and which is defined by $\mathrm{res}^{U_x}_{U_x \cap U_y} \mathscr{F}(U) = \mathscr{F}(U \cap U_x \cap U_y)$ for any open subset $U \subset U_x \subset \mathcal{X}$. We know that this

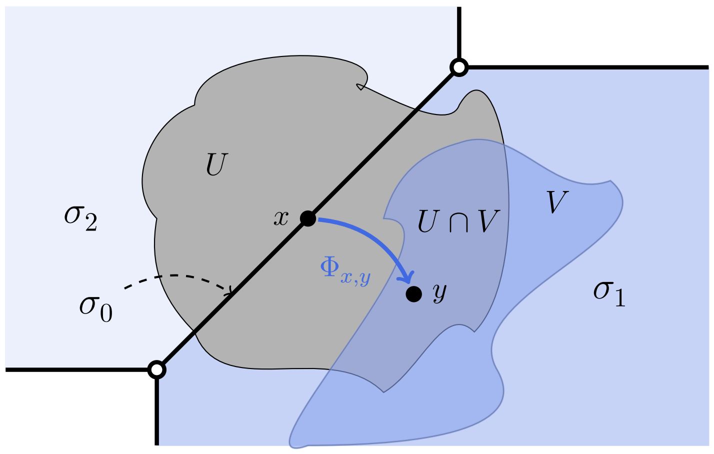


Figure 13.1.: *Schematic picture of the local situation in a cell decomposition: There are three labelled cells, one 1-cell σ_0, and two 2-cells σ_1 and σ_2. Further, we have two points $x \in \sigma_0$ and $y \in \sigma_1$ together with open neighbourhoods $y \in V \subset U \ni x$. We have here $U \subset U_x$ and $V \subset U_y$. The map $\Phi_{x,y}$ from the stalk at x to the stalk at y is shown as blue arrow.*

map exists if $U_x \cap U_y \neq \emptyset$. It is not necessary that also $\mathcal{U}(x) \cap \mathcal{U}(y) \neq \emptyset$ (comp. Fact 13.1.8). We now collect some properties of these maps.

Remark: In most examples we even have $U_y \subset U_x$.

Theorem 13.1.15 (Properties): *Let $\mathscr{F}$ be a w-constructible sheaf with respect to a partition $\mathcal{P}$ and $x, y, z \in \mathcal{X}$ with maximal open sets $U_x, U_y, U_z \subset \mathcal{X}$, then we have:*

a) $\Phi_{x,y} = \mathrm{Id}$, *if there is a $\sigma \in \mathcal{P}$ such that $x, y \in \sigma$.*

b) $\Phi_{x,y} = \Phi_{x,z}$, *if there is a $\sigma \in \mathcal{P}$ such that $y, z \in \sigma$, i.e. the family of maps $\Phi_{x,\_}$ is constant on cells.*

c) $\Phi_{x,y} = \Phi_{z,y}$, *if there is a $\sigma \in \mathcal{P}$ such that $x, z \in \sigma$, i.e. the family of maps $\Phi_{\_,y}$ is constant on cells.*

d) $\Phi_{y,z} \circ \Phi_{x,y} = \Phi_{x,z}$.

Proof.

a) If $x = y$ we have $U_x = U_y$ and the restriction map for all sheaves is the identity map.

b) If $y, z \in \sigma \in \mathcal{P}$, then we have $\mathscr{F}_y = \mathscr{F}_z$. Thus, the domain where the map is defined is equal for $\Phi_{x,y}$ and $\Phi_{x,z}$. The claim follows now from the fact that $\mathcal{U}(y) = \mathcal{U}(z)$, together with Corollary 13.1.11 and Corollary 13.1.12.

c) Follows in the same way as the previous claim.

d) We can assume that $y, z \in U_x$, and $z \in U_y$, because otherwise the maps are not defined. Without loss of generality we can assume that $U_x \supset U_y$ and $U_y \supset U_z$. Otherwise we can replace U_y by $U'_y := U_x \cap U_y$ and U_z by $U'_z := U_z \cap U'_y$ which does not change those maps at all by Corollary 13.1.12 and Lemma 13.1.10. We get the following commutative diagram

$$\begin{tikzcd}
\mathscr{F}_x \arrow[r, "f^{-1}_{U_x,x}"] \arrow[d, "\Phi_{x,z}"] \arrow[rrrd, bend left, "\Phi_{x,y}"] & \mathscr{F}(U_x) \arrow[d, "\mathrm{res}^{U_x}_{U_z}"] \arrow[rd, "\mathrm{res}^{U_x}_{U_y}"] & & \\
& & \mathscr{F}(U_y) \arrow[r, shift left, "f_{U_y,y}"] \arrow[ld, "\mathrm{res}^{U_y}_{U_z}"] & \mathscr{F}_y \arrow[l, shift left, "f^{-1}_{U_y,y}"] \arrow[llld, bend left, "\Phi_{y,z}"] \\
\mathscr{F}_z & \mathscr{F}(U_z) \arrow[l, "f_{U_z,z}"] & &
\end{tikzcd}$$

which proves the claim. □

Since the maps $\Phi_{x,y}$ are constant in both indices on cells we set

$$\Phi_{i,j} := \Phi_{x,y} \tag{13.8}$$

if $x \in \mathcal{X}_i$ and $y \in \mathcal{X}_j$. We call a pair (i,j) a **valid pair** if the map $\Phi_{i,j}$ is defined. Denote by $\mathbf{val}_{\mathcal{P}}$ the set of all valid pairs with respect to $\mathcal{P}$. Let moreover

$$\mathbf{val}^i_{\mathcal{P}} = \left\{ l : (k,l) \in \mathbf{val}_{\mathcal{P}} \text{ with } k = i \right\}.$$

Theorem 13.1.16: *Let $\mathcal{X} = \bigsqcup_{\mathcal{X}_i \in \mathcal{P}} \mathcal{X}_i$ be a decomposition as above and $\mathscr{F}$ a sheaf on $\mathcal{X}$. Then $\mathscr{F}$ is w-constructible if and only if the following three conditions hold:*

a) There is a countable system of modules $\{\mathbb{F}_{\mathcal{X}_i} : \mathcal{X}_i \in \mathcal{P}\}$ such that $\mathscr{F}_x = \mathbb{F}_i$ for all $x \in \mathcal{X}_i$, where $\mathbb{F}_i := \mathbb{F}_{\mathcal{X}_i} \in$ $\mathbf{Obj}\,(\mathbf{Mod}_{\mathbb{Z}})$.

b) For every point $x \in \mathcal{X}_i \subset \mathcal{X}$ there is for all $y \in \mathcal{X}_j$ with $\mathcal{X}_i \subset \overline{\mathcal{X}_j}$ a map $\Phi_{x,y} \in \mathbf{Hom}_{\mathbb{Z}}\left(\mathscr{F}_x, \mathscr{F}_y\right)$ which satisfies point b) and c) of Theorem 13.1.15.

c) If $x \in \mathcal{X}_i$, $y \in \mathcal{X}_j$, and $z \in \mathcal{X}_k$ with $\mathcal{X}_i \subset \overline{\mathcal{X}_j}$ and $\mathcal{X}_j \subset \overline{\mathcal{X}_k}$ then $\Phi_{y,z} \circ \Phi_{x,y} = \Phi_{x,z}$.

A w-constructible sheaf is uniquely determined by a collection of modules satisfying the points a)–c).

We have, as already mentioned in context of the construction of the cell decomposition for $\mathfrak{S}_2/\Gamma$ in Chapter 7, that a cell decomposition induces a partial ordering on the set of cells. This ordering is given for a pair of cells by checking whether one cell lies in the closure of the other. These inclusion relations of our cell decomposition together with a w-constructible sheaf define via Theorem 13.1.15 in an obvious way a directed set, which is encoded in the set of valid pairs. We call such a directed set $\{(\mathbb{F}_i, \{\Phi_{i,j}\}_{j \in \mathbf{val}^i_{\mathcal{P}}})_{i \in \mathcal{I}}\}$ associated to a w-constructible sheaf a **w-constructible system** with respect to the partition $\mathcal{P}$ and denote it by $\mathfrak{F}_{\mathcal{P}}$. By Lemma 13.1.4 one can replace $\mathcal{P}$ by a suitable refinement $\mathcal{P}'$ without changing the constructibility property of the sheaf. Therefore, we identify all w-constructible systems which can be obtained by a refinement of an underlying partition. If 'the' partition is fixed we omit the index and use simply $\mathfrak{F}$. We can form the direct limit, which is by definition given by

$$\varinjlim_i \mathbb{F}_i = \left\{ (\mathbf{m}_i)_{\mathcal{X}_i \in \mathcal{P}} \in \prod_{\mathcal{X}_i \in \mathcal{P}} \mathbb{F}_i : \Phi_{i,j}(\mathbf{m}_i) = \mathbf{m}_j \quad \forall (i,j) \in \mathbf{val}_{\mathcal{P}} \right\}. \tag{13.9}$$

By construction we have that $\varinjlim_i \mathbb{F}_i$ is a submodule of $\prod_{\mathcal{X}_i \in \mathcal{P}} \mathbb{F}_i$ with an equivalence relation induced by the maps $\Phi_{i,j}$ for $(i,j) \in \mathbf{val}_{\mathcal{P}}$ (comp. (13.9)), i.e. we get

$$\varinjlim_{i} \mathbb{F}_i = \prod_{\mathcal{X}_i \in \mathcal{P}} \mathbb{F}_i \Big/ \sim \; \subseteq \prod_{\mathcal{X}_i \in \mathcal{P}} \mathbb{F}_i, \tag{13.10}$$

where $\mathbf{m}_i \sim \mathbf{m}_j$ for $\mathbf{m}_i \in \mathbb{F}_i$ and $\mathbf{m}_j \in \mathbb{F}_j$ if there is a $k \in \mathcal{I}$ such that $(i,k), (j,k) \in \mathbf{val}_{\mathcal{P}}$ and $\Phi_{i,k}(\mathbf{m}_i) = \Phi_{j,k}(\mathbf{m}_j)$.

Denote by $\mathbf{Conn}_i(U)$ the set of connected components of the intersection $U \cap \mathcal{X}_i$ of a cell $\mathcal{X}_i$ with an open set $U \subset \mathcal{X}$.

Lemma 13.1.17: *A w-constructible system $\mathfrak{F}_{\mathcal{P}}$ defines a w-constructible sheaf $\mathscr{F}$ on $\mathcal{X}$ by defining its sections to be given by*

$$\mathscr{F}(U) = \left\{ \Big((\mathbf{m}_i)_{\sigma \in \mathbf{Conn}_i(U)} \Big)_{\mathcal{X}_i \in \mathcal{P}} \in \prod_{\mathcal{X}_i \in \mathcal{P}} \prod_{\mathbf{Conn}_i(U)} \mathbb{F}_i : \Phi_{i,j}(\mathbf{m}_i) = \mathbf{m}_j \; \forall (i,j) \in \mathbf{val}_{\mathcal{P}} \right\} \tag{$*$}$$

for any open set $U \subset \mathcal{X}$ together with the obvious restriction maps.

Note: For a general U the connected components of the intersection of U with a cell can and will look dreadful. Thus, in the general case the number of elements in $\mathbf{Conn}_i(U)$ need not to be finite or even countable. This is a general phenomenon in the theory of sheaves. Therefore, most properties use stalks rather than sections of sheaves, which behave much better in most cases. We use this construction only in two cases: In case of global sections (see Corollary 13.1.18), so that thus there is only one connected component for each cell, or in case of sections of arbitrary small neighbourhoods of points, where there are only finitely many connected components since the decomposition is locally finite.

Proof. We have to verify two things: First, that $(*)$ defines indeed a sheaf. Second, that this sheaf restricted to each cell $\mathcal{X}_i$ is locally constant. The first is obvious by construction. By $(*)$ we have

$$\mathscr{F}\mid_{\mathcal{X}_i}(U) = \mathscr{F}(U \cap \mathcal{X}_i) = \prod_{\mathbf{Conn}_i(U)} \mathbb{F}_i \tag{13.11}$$

for any open set $U \subset \mathcal{X}$, where the modules $\mathbb{F}_i$ for different i are connected via the maps $\Phi_{i,j}$ under each other. Hence, $\mathscr{F}$ restricted to a cell $\mathcal{X}_i$ is locally constant. □

If all $\mathbb{F}_i$ are of finite rank, the sheaf $\mathscr{F}$ is even a constructible sheaf with respect to $\mathcal{P}$. In fact, also a suitable subset of finite rank modules forces, via the $\Phi_{x,y}$, the other modules to be also of finite rank.

Proof of Theorem 13.1.16.

$\Longrightarrow$ If the sheaf $\mathscr{F}$ is w-constructible, we know that all stalks over points in the same cell $\mathcal{X}_i$ are constant by Corollary 13.1.7. Furthermore, we can choose the map $\Phi_{x,y}$ as the map defined in (13.7). Then this direction results from Theorem 13.1.15.

$\Longleftarrow$ Follows from Lemma 13.1.17.

To show the uniqueness statement it is enough to mention that two sheaves whose stalks are equal in every point have to be equal, which is here the case. □

From Lemma 13.1.17 and Theorem 13.1.16 we get

Corollary 13.1.18 (Global sections)**:** *Let $\mathscr{F}$ be a w-constructible sheaf on $\mathcal{X}$ with respect to some partition $\mathcal{P}$ on $\mathcal{X}$ defined by a w-constructible system $\mathfrak{F}_{\mathcal{P}}$ as in Lemma 13.1.17. Then the global sections $\mathscr{F}(\mathcal{X})$ are given by*

$$\mathscr{F}(\mathcal{X}) = \varinjlim_{i} \mathbb{F}_i.$$

13.2. Orbi-Constructible Sheaves

We introduced in the previous section the notion of constructible sheaves with respect to a given partition $\mathcal{P}$ and studied some of their properties. Now we include the action of a group into this framework and extend the notion of constructible sheaves to orbifolds. There has been – as already mentioned in the introduction – no notion like this before except perhaps the so called equivariant constructible sheaves, which differs in many details, such that we will give here a description of this useful tool which is a little bit more detailed than necessary for our purpose.

Let $\mathcal{X}$ be a smooth Riemannian manifold and $\mathcal{P}$, as before, a locally finite decomposition of $\mathcal{X}$. Let $\mathbf{G}$ be a discrete group of automorphisms of $\mathcal{X}$ acting (from the right) cellularly on $\mathcal{X}$, i.e. $\mathbf{G}$ maps cells onto cells in the decomposition, with only *finitely many orbits*. The restriction on the number of orbits is in many cases not necessary for the general construction but simplifies a lot of things.[6] Furthermore, it is always fulfilled for decompositions in the sense of Theorem 9.2.11. The quotient $\mathcal{X}/\mathbf{G}$ is in general an orbifold (comp. Section 5.2).

Let $\mathcal{P}_{\mathbf{G}}$ be a system of representatives of the cells in $\mathcal{P}$ under the action of $\mathbf{G}$. Let $\mathcal{I}_{\mathbf{G}} \subset \mathcal{I}$ be the index set of those cells. Since we have only finitely many orbits, $\mathcal{P}_{\mathbf{G}}$ is finite. Thus, we have

$$\mathcal{X} = \bigsqcup_{i\in\mathcal{I}} \mathcal{X}_i = \bigsqcup_{\mathcal{X}_i\in\mathcal{P}} \mathcal{X}_i = \bigsqcup_{i\in\mathcal{I}_{\mathbf{G}}} \bigcup_{\mathbf{g}\in G} \mathcal{X}_i \cdot \mathbf{g} = \bigsqcup_{\mathcal{X}_i\in\mathcal{P}_{\mathbf{G}}} \underbrace{\bigcup_{\mathbf{g}\in G} \mathcal{X}_i \cdot \mathbf{g}}_{=\mathcal{X}_i\cdot\mathbf{G}}. \tag{13.12}$$

We do not assume that the action is free, therefore it might happen that an element $\mathbf{g} \in \mathbf{G}$ maps some cells onto themselves but not necessarily point-wise. Hence, we get a decomposition

$$\mathcal{X} = \bigsqcup_{i\in\mathcal{I}_{\mathbf{G}}} \bigsqcup_{\mathbf{g}\in\mathbf{G}_i\backslash\mathbf{G}} \mathcal{X}_i \cdot \mathbf{g} \tag{13.13}$$

where $\mathbf{G}_i$ is the stabilizer of $\mathcal{X}_i$ in $\mathbf{G}$. To extend the notion of w-constructible sheaves to a space $\mathcal{X}$ with a group action we have to ensure that the group action on the level of cells and the group action on level of the modules, i.e. the stalks of the sheaves, are compatible. Therefore, we have to introduce some additional compatibility maps coming from the group action. The basic idea is that we do not longer consider w-constructible systems of arbitrary $\mathbb{Z}$-modules, but a system which associates to each cell $\mathcal{X}_i$ a $\mathbf{G}_i$-$\mathbb{Z}_{\mathbf{G}}$-module, such

[6] In the classical definitions of constructible sheaves by Grothedieck one needs some kind of finiteness condition. Mostly one assumes that the decomposition is finite.

that we know how $\mathbf{G}_i$ acts on the module $\mathbb{F}_i$. A priori, there is no need for the restriction to $\mathbb{Z}_\mathbf{G}$-modules. However, this simplifies some proofs later on and we will only apply the results to such modules. We make this more precise in the following.

Let $\mathcal{X}_i, \mathcal{X}_j \in \mathcal{P}$. If $\mathcal{X}_i$ and $\mathcal{X}_j$ lie in the same orbit under $\mathbf{G}$ then there is an element $\mathbf{g}_{i,j} \in \mathbf{G}$ such that $\mathcal{X}_i \cdot \mathbf{g}_{i,j} = \mathcal{X}_j$. Since the action is cellular, each element $\mathbf{g} \in \mathbf{G}$ can be realized as a unique permutation on the set of indices and we will write $\mathbf{g}_{i,j}(i) = j$. On the contrary, the element $\mathbf{g}_{i,j}$ itself is not unique. It is only unique up to action of $\mathbf{G}_j$ from the right and $\mathbf{G}_i$ from the left. Now we will focus on the associated modules. Let

$$\mathfrak{F}_\mathcal{P} = \left\{ \left(\mathbb{F}_i, (\Phi_{i,j})_{j \in \mathbf{val}^i_\mathcal{P}}\right)_{i \in \mathcal{I}} \right\} \tag{13.14}$$

be a w-constructible system where $\mathbb{F}_i \in \mathbf{Obj}\left(\mathbf{Mod}_{\mathbf{G}_i, \mathbb{Z}_\mathbf{G}}\right)$. To ensure the compatibility with the action of $\mathbf{G}$ it is essential that the $\mathbb{F}_i \in \mathbf{Obj}\left(\mathbf{Mod}_{\mathbf{G}_i, \mathbb{Z}}\right)$. The additional condition on the underlying ring to be $\mathbb{Z}_\mathbf{G}$ is useful to avoid nasty problems while taking invariants under stabilizers. It is possible, as in case of the resolution of sheaves (comp. Section 13.3), to consider arbitrary $\mathbb{Z}$-modules to the cost that at some places one has to replace objects by higher derived images of them, which we want to avoid for the moment.

For $(i,j) = \left(i, \mathbf{g}_{i,j}(i)\right)$ is a trivial fact that

$$\mathbf{G}_i = \mathbf{g}_{i,j} \cdot \mathbf{G}_j \cdot \mathbf{g}_{i,j}^{-1}. \tag{13.15}$$

Therefore, we can identify the category $\mathbf{Mod}_{\mathbf{G}_i}$ with $\mathbf{Mod}_{\mathbf{G}_j}$ via the isomorphism coming from (13.15). In the following, we consider any $\mathbf{G}_j$-module as a $\mathbf{G}_i$-module via (13.15) if necessary. To construct a sheaf on $\mathcal{X}$ which is compatible with the action of $\mathbf{G}$, we need to compare the modules $\mathbb{F}_i$ and $\mathbb{F}_j$ in some way, because they are associated to the same orbit of cells. For each such pair $(i,j) = (i, \mathbf{g}(i))$ for some $\mathbf{g} \in \mathbf{G}$ we define an isomorphism $\Psi_{i,\mathbf{g}} \in \mathbf{Hom}_{\mathbf{G}_i, \mathbb{Z}_\mathbf{G}}\left(\mathbb{F}_i, \mathbb{F}_{\mathbf{g}(i)}\right)$, which transfers the action on the level of cells to the level of modules in an appropriate way. This means that:

a) For all $\mathbf{g} \in \mathbf{G}_i \subseteq \mathbf{G}$, $\mathbf{g}_1 \in \mathbf{G}$, and $\mathbf{m} \in \mathbb{F}_i$ we have

$$\Psi_{i,\mathbf{g}_1}(\mathbf{m} \cdot \mathbf{g}) = \Psi_{i,\mathbf{g}_1}(\mathbf{m}) \cdot \mathbf{g}_1^{-1} \cdot \mathbf{g} \cdot \mathbf{g}_1,$$

i.e. $\Psi_{i,\mathbf{g}_1}$ is compatible with (13.15) and realizes the $\mathbf{Mod}_{\mathbf{G}_i}$ structure on $\mathbb{F}_j$.

b) For all $\mathbf{g}_1, \mathbf{g}_2 \in \mathbf{G}$ we have

$$\begin{tikzcd}
\mathbb{F}_i \arrow[d, "\Psi_{i,\mathbf{g}_1}"'] \arrow[dr, "\Psi_{i,\mathbf{g}_1 \cdot \mathbf{g}_2}"] & \\
\mathbb{F}_{\mathbf{g}_1(i)} \arrow[r, "\Psi_{\mathbf{g}_1(i),\mathbf{g}_2}"'] & \mathbb{F}_{\mathbf{g}_2(\mathbf{g}_1(i))}.
\end{tikzcd}$$

c) If $\mathbf{g} \in \mathbf{G}_i$, then $\Psi_{i,\mathbf{g}}(\mathbf{m}) = \mathbf{m} \cdot \mathbf{g}$. for all $\mathbf{m} \in \mathbb{F}_i$.

By condition b) we immediately get

$$\Psi_{i,\mathbf{g}}^{-1} = \Psi_{\mathbf{g}(i),\mathbf{g}^{-1}}. \tag{13.16}$$

A collection of such maps transfers the group action to the level of modules. We use these maps to attach modules to orbicells in $\mathcal{X}/\mathbf{G}$. Therefore, we want to have a closer look to the collection of modules together with their associated family of isomorphisms. For this we need to make some choices. We fix a set of representatives for the cosets $\mathbf{G}_i\backslash\mathbf{G}$ for each $i \in \mathcal{I}$ and a set of representatives

$$\mathcal{P}_{\mathbf{G}} = \left\{\mathcal{X}_i : i \in \mathcal{I}_{\mathbf{G}}\right\}$$

of orbits of cells together with a family of isomorphisms $\Psi_{i,\mathbf{g}}$ for $\mathbf{g} \in \mathbf{G}_i\backslash\mathbf{G}$. Moreover, we can assume without loss of generality that for all $i \in \mathcal{I}_{\mathbf{G}}$

$$\mathbb{F}_{\mathbf{g}(j)} = \Psi_{i,\mathbf{g}}(\mathbb{F}_i). \tag{13.17}$$

The global sections of the sheaf associated to a w-constructible system $\mathfrak{F}_{\mathcal{P}}$ are a submodule of $\prod_{i\in\mathcal{I}} \mathbb{F}_i$ (comp. 13.10). We get, using (13.13) and the definition of the induced representation,

$$\prod_{i\in\mathcal{I}} \mathbb{F}_i = \bigoplus_{i\in\mathcal{I}_{\mathbf{G}}} \underbrace{\prod_{\mathbf{g}\in\mathbf{G}_i\backslash\mathbf{G}} \Psi_{i,\mathbf{g}}(\mathbb{F}_i)}_{=:\mathrm{Ind}_{\mathbf{G}_i}^{\mathbf{G}}(\mathbb{F}_i)} = \bigoplus_{i\in\mathcal{I}_{\mathbf{G}}} \mathrm{Ind}_{\mathbf{G}_i}^{\mathbf{G}}\left(\mathbb{F}_i\right). \tag{13.18}$$

This representation depends only on the choice of the collection of maps and an order for the representatives in $\mathcal{I}_{\mathbf{G}}$ and $\mathbf{G}_i\backslash\mathbf{G}$. If we replace $i \in \mathcal{I}_{\mathbf{G}}$ by an other element $\mathbf{g}(i)$ for some $\mathbf{g} \in \mathbf{G}$, the corresponding map

$$\Psi_{\mathbf{g}(i),\mathbf{g}'} = \Psi_{i,\mathbf{g}\cdot\mathbf{g}'} \circ \Psi_{i,\mathbf{g}}^{-1} = \Psi_{i,\mathbf{g}\cdot\mathbf{g}'} \circ \Psi_{\mathbf{g}(i),\mathbf{g}^{-1}} \tag{13.19}$$

for all $\mathbf{g}' \in \mathbf{G}$. Analogously, each change of a representative $\mathbf{g} \in \mathbf{G}_i\backslash\mathbf{G}$ can be realized by a relation of the form (13.17).

By this construction the $\mathbb{Z}_{\mathbf{G}}$-module $\prod_{i\in\mathcal{I}} \mathbb{F}_i$ gets the structure of a $\mathbf{G}$-$\mathbb{Z}_{\mathbf{G}}$-module. We have for all $i \in \mathcal{I}$ a decomposition

$$\mathbf{G} = \mathbf{G}_i \times \mathbf{G}_i\backslash\mathbf{G}. \tag{13.20}$$

Note that $\mathbf{G}_i\backslash\mathbf{G}$ is in general only a coset and only in some special cases a group, because not all $\mathbf{G}_i$ are normal in $\mathbf{G}$. This decomposition can be used together with (13.18) to compute the invariants under $\mathbf{G}$:

$$\begin{aligned}\left(\prod_{i\in\mathcal{I}} \mathbb{F}_i\right)^{\mathbf{G}} &= \left(\bigoplus_{i\in\mathcal{I}_{\mathbf{G}}} \mathrm{Ind}_{\mathbf{G}_i}^{\mathbf{G}}\left(\mathbb{F}_i\right)\right)^{\mathbf{G}} = \bigoplus_{i\in\mathcal{I}_{\mathbf{G}}} \left(\mathrm{Ind}_{\mathbf{G}_i}^{\mathbf{G}}\left(\mathbb{F}_i\right)\right)^{\mathbf{G}} \\ &= \bigoplus_{i\in\mathcal{I}_{\mathbf{G}}} \left(\mathrm{Ind}_{\mathbf{G}_i}^{\mathbf{G}}\left(\mathbb{F}_i\right)^{\mathbf{G}_i\backslash\mathbf{G}}\right)^{\mathbf{G}_i} = \prod_{i\in\mathcal{I}_{\mathbf{G}}} \mathbb{F}_i^{\mathbf{G}_i}\end{aligned} \tag{13.21}$$

As before, also this identification depends on the choice of the lifts of orbicells in $\mathcal{X}/\mathbf{G}$ to $\mathcal{X}$ or equivalently on the choice of an index set $\mathcal{I}_{\mathbf{G}}$. Analogue to the considerations in the context of global sections of the sheaf associated to a w-constructible system, the global sections of the associated sheaf in the quotient is contained in $\mathbf{G}$-invariants of the module in that case (see Corollary 13.1.18 together with (13.10) and (13.18) for the sheaf on $\mathcal{X}$ and for Theorem 13.2.2 and Lemma 13.2.6 for the sheaf on $\mathcal{X}/\mathbf{G}$).

To obtain the information on the sections themselves, it remains to include into our considerations the compatibility conditions given by the $\Phi_{i,j}$ from (13.7), which ensure that the local data glue to a sheaf on $\mathcal{X}$ (see Lemma 13.1.17).

For this we need to make some additional assumptions. Let $\mathbb{F}_i \in \mathbf{Obj}\left(\mathbf{Mod}_{\mathbf{G}_i,\mathbb{Z}_{\mathbf{G}}}\right)$ and $\mathbb{F}_j \in \mathbf{Obj}\left(\mathbf{Mod}_{\mathbf{G}_j,\mathbb{Z}_{\mathbf{G}}}\right)$ with $(i,j) \in \mathbf{val}_{\mathcal{P}}$. Note that

$$\mathbf{Mod}_{\mathbf{G}_i,\mathbb{Z}_{\mathbf{G}}} \subset \mathbf{Mod}_{\mathbf{G}_i \cap \mathbf{G}_j,\mathbb{Z}_{\mathbf{G}}} \supset \mathbf{Mod}_{\mathbf{G}_j,\mathbb{Z}_{\mathbf{G}}}.$$

Moreover, we need, as before, homomorphisms

$$\Phi_{i,j} : \mathbb{F}_i \to \mathbb{F}_j,$$

which have to be, in addition to the previously made assumptions (see Theorem 13.1.15), compatible with the group action. This means two things: First, the homomorphisms $\Phi_{i,j}$ have to respect the action of elements which belong to both associated stabilizers, i.e. $\Phi_{i,j} \in \mathbf{Hom}_{\mathbf{G}_i \cap \mathbf{G}_j}\left(\mathbb{F}_i, \mathbb{F}_j\right)$, and secondly that the diagram

$$\begin{tikzcd}
\mathbb{F}_i \arrow[r, "\Phi_{i,j}"] \arrow[d, "\Psi_{i,\mathbf{g}}"'] & \mathbb{F}_j \arrow[d, "\Psi_{j,\mathbf{g}'}"] \\
\mathbb{F}_{\mathbf{g}(i)} \arrow[r, "\Phi_{\mathbf{g}(i),\mathbf{g}'(j)}"'] & \mathbb{F}_{\mathbf{g}'(j)}.
\end{tikzcd} \tag{13.22}$$

commutes for all $\mathbf{g}, \mathbf{g}' \in \mathbf{G}$ and all $(i,j) \in \mathbf{val}_{\mathcal{P}}$.

Definition 13.2.1: *Let $\mathfrak{F}_{\mathcal{P}}$ be a w-constructible system on $\mathcal{X}$ such that:*

- $\mathbb{F}_i \in \mathbf{Obj}\left(\mathbf{Mod}_{\mathbf{G}_i,\mathbb{Z}_{\mathbf{G}}}\right)$ *for all $i \in \mathcal{I}$ and*
- $\Phi_{i,j} \in \mathbf{Hom}_{\mathbf{G}_i \cap \mathbf{G}_j}\left(\mathbb{F}_i, \mathbb{F}_j\right)$.

If there is a family of $\Psi_{i,\mathbf{g}} \in \mathbf{Hom}_{\mathbf{G}_i,\mathbb{Z}_{\mathbf{G}}}\left(\mathbb{F}_i, \mathbb{F}_{\mathbf{g}(i)}\right)$ for $i \in \mathcal{I}_{\mathbf{G}}$ and $\mathbf{g} \in \mathbf{G}$ which satisfy conditions a) – c) on page 161 and are compatible with the $\Phi_{i,j}$ according to (13.22)*, then we call the w-constructible system $\mathfrak{F}_{\mathcal{P}}$ a* **G-w-constructible system** *with respect to* $\mathbf{G}$ *and $\mathcal{P}$. We denote such a system by $\mathfrak{F}_{\mathcal{P},\mathbf{G}}$, where, as before, we omit in some cases the partition $\mathcal{P}$. If $\mathfrak{F}_{\mathcal{P}}$ is a constructible system we call $\mathfrak{F}_{\mathcal{P},\mathbf{G}}$ a* **G-constructible system**. *Again we identify* $\mathbf{G}$*-(w-)constructible systems which can be obtained by a refinement of the underlying partition.*

As before (comp. Theorem 13.1.16), we can use $\mathbf{G}$-w-constructible systems to describe certain sheaves. First, they define a sheaf on $\mathcal{X}$ and secondly they define also a sheaf on the quotient $\mathcal{X}/\mathbf{G}$.

Theorem 13.2.2: *Let $\mathfrak{F}_{\mathcal{P},\mathbf{G}}$ be a $\mathbf{G}$-w-constructible system on $\mathcal{X} = \bigcup_{i\in\mathcal{I}} \mathcal{X}_i$ as above, then:*

a) *There is a w-constructible sheaf $\mathscr{F}$ on $\mathcal{X}$ associated to $\mathfrak{F}_{\mathcal{P},\mathbf{G}}$, which is defined by $(*)$ in Lemma 13.1.17. This is a sheaf of $\mathbf{G}$-$\mathbb{Z}_{\mathbf{G}}$-modules.*

b) *The global sections of $\mathscr{F}$ are given by*

$$\mathscr{F}(\mathcal{X}) = \varinjlim_{i} \mathbb{F}_i = \bigoplus_{i\in\mathcal{I}_{\mathbf{G}}} \operatorname{Ind}_{\mathbf{G}_i}^{\mathbf{G}}(\mathbb{F}_i) \Big/_{\sim} \subseteq \bigoplus_{i\in\mathcal{I}_{\mathbf{G}}} \operatorname{Ind}_{\mathbf{G}_i}^{\mathbf{G}}(\mathbb{F}_i),$$

where the equivalence relation $\sim$ is the same as in (13.10).

Proof.

a) The first part is known from Lemma 13.1.17. The additional assumptions are no restriction. The second part follows from the considerations on page 162.

b) This a direct consequence of (13.10), (13.18) and Corollary 13.1.18. □

Let $\pi : \mathcal{X} \to \mathcal{X}/\mathbf{G}$ be the canonical projection of the cell complex $\mathcal{X}$ onto its quotient $\mathcal{X}/\mathbf{G}$, which sends each cell $\mathcal{X}_i \subset \mathcal{X}$ onto the orbicell $\mathcal{X}_i/\mathbf{G}_i \subset \mathcal{X}/\mathbf{G}$.

Definition 13.2.3: *Let $\mathscr{F}$ be the sheaf given by a $\mathbf{G}$-w-constructible system $\mathfrak{F}_{\mathcal{P},\mathbf{G}}$ as in Theorem 13.2.2. Then the sheaf of abelian groups*

$$\widehat{\mathscr{F}} = \left(\pi_*\left(\mathscr{F}\right)\right)^{\mathbf{G}} \in \mathbf{Obj}\left(\mathbf{Ab}_{\mathcal{X}/\mathbf{G}}\right)$$

on $\mathcal{X}/\mathbf{G}$ is called an **orbi-w-constructible sheaf** *on $\mathcal{X}/\mathbf{G}$ with respect to $\mathbf{G}$ and $\mathcal{P}$. If $\mathscr{F}$ is a constructible sheaf we call $\widehat{\mathscr{F}}$ an* **orbi-constructible sheaf** *with respect to $\mathbf{G}$ and $\mathcal{P}$. If $\mathbf{G}$ and $\mathcal{P}$ are known we may omit them in the notation.*

Example 13.2.4. Orbilocal sheaves (of finite rank) are orbi-constructible. The $\mathbf{G}$-w-constructible system $\mathfrak{F}_{\mathcal{P},\mathbf{G}}$ consists of a collection of modules, where all modules are equal to a given representation module $\mathbb{M}_i$ as long as they belong to the same connected component $\mathcal{Y}_i$ of $\mathcal{X}$, thus the compatibility maps $\Phi_{k,l} \equiv \operatorname{Id}$ for al valid pairs $(k,l) \in \mathbf{val}_{\mathcal{P}}$. The maps $\Psi_{k,\mathbf{g}}$ are not uniquely determined. The easiest cases is that $\Psi_{k,\mathbf{g}} = \rho(\mathbf{g})$ where ρ is the representation of $\mathbf{G}$ associated to the $\mathbf{G}$-module $\mathbb{M}_i$ if the cell $\mathcal{X}_k \subseteq \mathcal{Y}_i$. The restriction of the representation ρ onto a stabilizer is in particular a representation for the stabilizer. There are infinitely many choices for collections of $\Psi_{i,\mathbf{g}}$, since we can concatenate all maps in the family of maps $(\Psi_{i,\mathbf{g}})_{\mathbf{g}\in\mathbf{G}}$ by one arbitrary isomorphism on $\mathbb{F}_i$. However all these sheaves for different actions are isomorphic.

Note: As before, the definition of an orbi-w-constructible sheaf depends only up to some level on the choice of the partition $\mathcal{P}$ because we can replace $\mathcal{P}$ by a refinement $\mathcal{P}'$ (comp. Lemma 13.1.4), which defines an equivalent $\mathbf{G}$-w-constructible system.

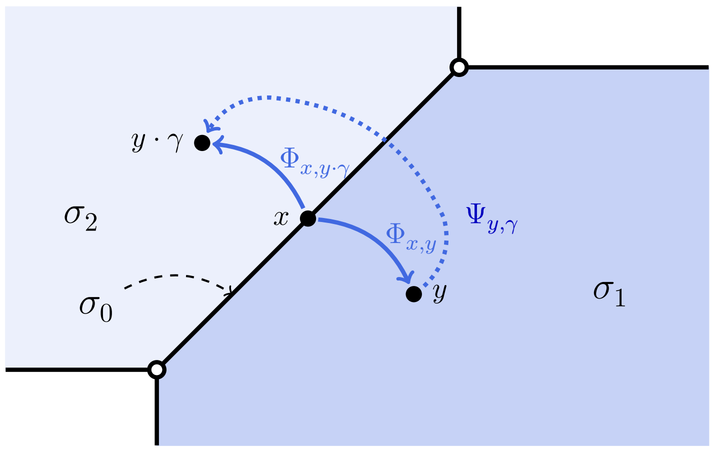


Figure 13.2.: *Schematic picture of the local situation in a cell decomposition.*

From the viewpoint of an abstract sheaf most choices during the construction of $\widehat{\mathscr{F}}$ do not change the sheaf $\widehat{\mathscr{F}}$ itself, but they change the representation of the sheaf. From a computational viewpoint there are several ways to define the very same sheaf that differ in:

- The choice of $\mathfrak{F}_{\mathcal{P},\mathbf{G}}$ in a class of refinements of the defining partition $\mathcal{P}$,
- the order of elements in the collection of cells, i.e. the index set of the collection,
- the choice of the collection of modules defining $\mathfrak{F}_{\mathcal{P},\mathbf{G}}$, in particular of the bases of the modules,
- the representation of the families of maps $(\Phi_{i,j})$ and $(\Psi_{i,\gamma})$, which can easily be changed by concatenation with some isomorphisms, and
- the choice of representatives in an orbit under the group $\mathbf{G}$ or equivalently of lifts of (orbi-)cells from $\mathcal{X}/\mathbf{G}$ to $\mathcal{X}$.

In particular, the last point is somehow tricky in practice, since in more complicated cell decompositions for some reasons one need to choose more than one representative per orbit under $\mathbf{G}$, because $\mathbf{G}$ acts not transitive on the cells and thus we can have several orbits under the stabilizer $\mathbf{G}_i$ of some cell $\mathcal{X}_i$ (comp. Sections 10.2, 10.7.3, and 13.3). Beside the above mentioned choices which induce lots of isomorphisms of the way how to write down the underlying $\mathbf{G}$-w-constructible system, the sheaf is – as one can easily check from the definition – given uniquely by a system $\mathfrak{F}_{\mathcal{P},\mathbf{G}}$. Thus, we have:

Lemma 13.2.5 (Uniqueness): *Any orbi-w-constructible sheaf $\widehat{\mathscr{F}}$ (with respect to $\mathbf{G}$ and $\mathcal{P}$) is (up to isomorphism) uniquely determined by its underlying $\mathbf{G}$-w-constructible system $\mathfrak{F}_{\mathcal{P},\mathbf{G}}$.*

In the following lemma we collect some useful properties of orbi-constructible sheaves.

Lemma 13.2.6: *Let $\widehat{\mathscr{F}}$ be an orbi-w-constructible sheaf with respect to a $\mathbf{G}$-w-constructible system $\mathfrak{F}_{\mathcal{P},\mathbf{G}}$.*

a) *Let $U \subset \mathcal{X}/\mathbf{G}$, then the sections of $\widehat{\mathscr{F}}$ are given by*

$$\widehat{\mathscr{F}}(U) \cong \left\{ \left((\mathbf{m}_i)_{\sigma \in \mathbf{Conn}_i\left(\pi^{-1}(U)\right)} \right)_{i \in \mathcal{I}_\mathbf{G}} \in \bigoplus_{i \in \mathcal{I}_\mathbf{G}} \prod_{\mathbf{Conn}_i(\pi^{-1}(U))} \mathbb{F}_i : \\ \Phi_{i,j}(\mathbf{m}_i) = \mathbf{m}_j \, \forall (i,j) \in \mathbf{val}_\mathcal{P} \right\}^{\mathbf{G}}, \quad (13.23)$$

where the isomorphism depends only on the choice of $\mathcal{I}_\mathbf{G} \subset \mathcal{I}$, i.e. the set of lifts, and can be expressed in terms of the maps $\Psi_{i,\gamma}$. Moreover, the global sections can be stated in the following form:

$$\widehat{\mathscr{F}}(\mathcal{X}/\mathbf{G}) = H^0\left(\mathcal{X}/\mathbf{G}, \widehat{\mathscr{F}}\right) \cong \varinjlim_i \mathbb{F}_i^{\mathbf{G}_i} = \bigoplus_{i \in \mathcal{I}_\mathbf{G}} \mathbb{F}_i^{\mathbf{G}_i} \Big/ _\sim \subseteq \bigoplus_{i \in \mathcal{I}_\mathbf{G}} \mathbb{F}_i^{\mathbf{G}_i},$$

where the isomorphism is, as before, and $\sim$ the equivalence relation defined in (13.10).

b) *Let $x \in \mathcal{X}/\mathbf{G}$ with $x = \pi(\widetilde{x})$, where $\widetilde{x} \in \mathcal{X}_i \subset \mathcal{X}$, then*

$$\widehat{\mathscr{F}_x} \cong \mathbb{F}_i^{\mathbf{G}_{\widetilde{x}}},$$

where $\mathbf{G}_{\widetilde{x}}$ is the stabilizer of the point $\widetilde{x}$ in $\mathbf{G}$. The isomorphism depends on the choice of a point $\widetilde{x} \in \pi^{-1}(x)$.

c) *Any orbi-w-constructible sheaf is w-locally constant.*

Proof.

a) The first part follows from Definition 13.2.3 of $\widehat{\mathscr{F}}$ together with Lemma 13.1.17. Analogously, the second part can be obtained from the Definition of $\widehat{\mathscr{F}}$ and (13.21) with (13.10) and Corollary 13.1.18.

b) Apply Definition 13.2.3 together with Theorem 13.1.16 and Lemma 13.1.17.

c) For each point $x \in \mathcal{X}/\mathbf{G}$ there is a small neighbourhood $U_x \subset \mathcal{X}/\mathbf{G}$ such that for a chosen lift $\widetilde{x} \in \pi^{-1}(x)$ there is one connected component of $\widetilde{U}_{\widetilde{x}} \subset \pi^{-1}(U_x)$ such that the stabilizer $\mathbf{G}_{\widetilde{U}_{\widetilde{x}}}$ of the set $\widetilde{U}_{\widetilde{x}}$ is a subgroup of the stabilizer $\mathbf{G}_{\widetilde{x}}$ of the lift $\widetilde{x}$, i.e. $\mathbf{G}_{\widetilde{U}_{\widetilde{x}}} \subseteq \mathbf{G}_{\widetilde{x}}$. We skip the proof of that, since it is lengthy and not very illuminating. Now the claim follows essentially from Definition 13.2.3 by point b) of this lemma together with Corollary 13.1.7. □

Since $\mathbf{G}$ acts cellular as automorphism on $\mathcal{X}$, we know that the relative order (one cell is in the boundary of an other one) is preserved by the action of $\mathbf{G}$ on $\mathcal{X}$. Therefore, we can, if necessary, extend the map $\Phi_{i,j}$ for $(i,j) \in \mathbf{val}_\mathcal{P}$ to a map on the stalks in the quotient in a natural way. We skip the details at this place.

To simplify some proofs later on we transfer the results of Theorem 13.1.3 from w-constructible sheaves to orbi-w-constructible sheaves with respect to the group $\mathbf{G}$.

Theorem 13.2.7 (Constructions)**:**

a) *Let $\mathscr{F}$ and $\mathscr{F}'$ be two orbi-(w-)constructible sheaves on $\mathcal{X}/\mathbf{G}$ and $f \in \mathbf{Hom}\left(\mathscr{F}, \mathscr{F}'\right)$ a morphism of sheaves, then $\mathscr{F} \oplus \mathscr{F}'$, $\mathscr{F} \otimes \mathscr{F}'$, $\mathbf{ker}\,(f)$, $\mathbf{im}\, f$, and $\mathbf{coker}\, f$ are orbi-(w-)constructible.*

b) *If $0 \to \mathscr{F}' \to \mathscr{F} \to \mathscr{F}'' \to 0$ is an exact sequence of sheaves and two of the non-zero sheaves are orbi-(w-)constructible, then so is the third.*

c) *Let $\mathscr{F}$ be an orbi-(w-)$\mathcal{P}$-constructible sheaf on $\mathcal{X}$ and $f : \mathcal{X} \to \mathcal{Y}$ a continuous map which acts cellular, i.e. cells are mapped onto cells, then $f_*\mathscr{F}$ is orbi-(w-)$\mathcal{P}'$-constructible for some partition $\mathcal{P}'$ of $\mathcal{Y}$. If $\mathcal{X}$ is a closed subset of $\mathcal{Y}$ and f an embedding, one can skip the cellularity condition.*

d) *Let $\mathscr{G}$ be an orbi-(w-)$\mathcal{P}$-constructible sheaf on $\mathcal{Y}$ and $f : \mathcal{X} \to \mathcal{Y}$ a continuous map then $f^*\mathscr{G}$ is orbi-(w-)$\mathcal{P}'$-constructible for some partition $\mathcal{P}'$ of $\mathcal{X}$. If f is injective and acts cellular, $f^*\mathscr{G}$ is also orbi-(w-)$\mathcal{P}$-constructible.*

Proof. The proof is essentially an application of Theorem 13.1.3 (or its proof) to Definition 13.2.3 together with the fact that the considered modules are $\mathbb{Z}_\Gamma$ modules and thus taking invariants is exact. □

13.3. Resolutions for (Orbi-)local Systems

13.3.1. The First Resolution

Let $\mathcal{X} = \bigsqcup_{i\in\mathcal{I}} \mathcal{X}_i$ as in Section 13.2 be a cell decomposition of a nice Riemannian manifold of (real) dimension d on which $\mathbf{G}$ acts cellular with finitely many orbits. Then $\mathcal{X}/\mathbf{G}$ is an orbifold with an induced orbicell decomposition

$$\mathcal{X}/\mathbf{G} = \bigsqcup_{i\in\mathcal{I}_\mathbf{G}} \mathcal{X}_i/\mathbf{G}_i, \tag{13.24}$$

where $\mathbf{G}_i \subset \mathbf{G}$ is the stabilizer of the cell $\mathcal{X}_i$ and $\mathcal{I} \subset \mathcal{I}_\mathbf{G}$. We defined in (5.9) the set of k-(orbi-)cells $\mathbf{C}_k(\mathcal{X})$. By definition of the k-skeleton in (5.10) we have

$$\mathbf{C}_k(\mathcal{X}) = \mathbf{S}_k(\mathcal{X}) - \mathbf{S}_{k-1}(\mathcal{X}) \tag{13.25}$$

and analogously

$$\mathbf{C}_k(\mathcal{X}/\mathbf{G}) = \mathbf{S}_k(\mathcal{X}/\mathbf{G}) - \mathbf{S}_{k-1}(\mathcal{X}/\mathbf{G}). \tag{13.26}$$

Let

$$\widetilde{i}_k : \mathbf{C}_k(\mathcal{X}) \hookrightarrow \mathcal{X} \tag{13.27}$$

be the natural inclusion of (relative open) k-cells into the full cell complex $\mathcal{X}$ and

$$i_k : \mathbf{C}_k(\mathcal{X}/\mathbf{G}) \hookrightarrow \mathcal{X}/\mathbf{G} \tag{13.28}$$

be the inclusion of the (relative open) k-orbicells into the quotient $\mathcal{X}/\mathbf{G}$ induced by $\widetilde{i}_k$ under the canonical projection.

We assume in the following that all modules $\mathbb{M}$ are defined over $\mathbb{Z}$ and of finite rank. For simplicity we formulate all results, as before, only for modules

$$\mathbb{M}_{\mathbf{G}} := \mathbb{M} \otimes_{\mathbb{Z}} \mathbb{Z}_{\mathbf{G}}. \tag{13.29}$$

In some cases the results stay also true over the integers. We get, as a direct consequence of the definition of an orbilocal system associated to a module $\mathbb{M}_{\mathbf{G}}$ in Section 12.1:

Proposition 13.3.1: *Any orbilocal system $\mathcal{M}_{\mathbf{G}} = \widetilde{\mathbb{M}}_{\mathbf{G}}$ associated to a $\mathbf{G}$-module $\mathbb{M}$ can be obtained from the corresponding constant sheaf $\underline{\mathbb{M}}_{\mathbf{G}}$ by*

$$\mathcal{M}_{\mathbf{G}} = \left(\pi_*(\underline{\mathbb{M}}_{\mathbf{G}})\right)^{\mathbf{G}}.$$

Recall that this is exactly the construction we used in Definition 13.2.3 to obtain an orbi-constructructible sheaf with respect to $\mathbf{G}$. Let us now construct the building blocks of our resolution. We set

$$\widetilde{\mathcal{S}}_d := \underline{\mathbb{M}}_{\mathbf{G}} \qquad\qquad \mathcal{S}_d := \mathcal{M}_{\mathbf{G}} := \widetilde{\mathbb{M}}_{\mathbf{G}} \tag{13.30}$$

and

$$\widetilde{\mathcal{A}}_d := \widetilde{i}_{d*}\widetilde{i}_d^{\,*}\widetilde{\mathcal{S}}_d \qquad\qquad \mathcal{A}_d := i_{d*}i_d^{\,*}\mathcal{S}_d. \tag{13.31}$$

Then there are two maps between sheaves:

$$\widetilde{j}_d : \widetilde{\mathcal{S}}_d \to \widetilde{\mathcal{A}}_d \qquad\qquad j_d : \mathcal{S}_d \to \mathcal{A}_d. \tag{13.32}$$

The maps $\widetilde{j}_d$ and j_d are injective (see Corollary 13.3.5). For the moment we postpone the proof of this. Since the maps $\widetilde{j}_d$ and j_d are injective we get the two exact sequences (horizontal sequences)

$$\begin{tikzcd} 0 \arrow[r] & \widetilde{\mathcal{S}}_d \arrow[r, "\widetilde{j}_d"] \arrow[d] & \widetilde{\mathcal{A}}_d \arrow[r, "\widetilde{p}_d"] \arrow[d] & \widetilde{\mathcal{S}}_{d-1} \arrow[r] \arrow[d] & 0 \\ 0 \arrow[r] & \mathcal{S}_d \arrow[r, "j_d"] & \mathcal{A}_d \arrow[r, "p_d"] & \mathcal{S}_{d-1} \arrow[r] & 0, \end{tikzcd} \tag{13.33}$$

where the maps from top to bottom are given by the functor

$$\begin{aligned} \left(\pi_*(\_)\right)^{\mathbf{G}} : \mathbf{Sheaves}_{\mathcal{X}} &\to \mathbf{Sheaves}_{\mathcal{X}/\mathbf{G}} \\ \mathscr{F} &\mapsto \widehat{\mathscr{F}} = (\pi_*\mathscr{F})^{\mathbf{G}}, \end{aligned} \tag{13.34}$$

which is in general only left exact on the category of sheaves. But under our additional assumptions this functor will also be exact on the category of (w-)constructible sheaves. The sheaves $\widetilde{\mathcal{S}}_{d-1}$ and $\mathcal{S}_{d-1}$ are defined to be the quotient sheaves

$$\widetilde{\mathcal{S}}_{d-1} := \widetilde{\mathcal{A}}_d \Big/ \widetilde{j}_d\Big(\widetilde{\mathcal{S}}_d\Big) \qquad\qquad \mathcal{S}_{d-1} := \mathcal{A}_d / j_d(\mathcal{S}_d). \tag{13.35}$$

It is not difficult to check that the diagram (13.33) commutes, since the maps are cellular and taking invariants under stabilizers is exact in the category of $\mathbb{Z}_{\mathbf{G}}$-modules (comp. [Gro57b; Chapitre V.]).

One finds that the sheaves $\widetilde{\mathcal{S}}_{d-1}$ and $\mathcal{S}_{d-1}$ have only support in the $(d-1)$-skeleton $\mathbf{S}_{d-1}(\mathcal{X})$ and $\mathbf{S}_{d-1}(\mathcal{X}/\mathbf{G})$ respectively (see Lemma 13.3.3). Under the assumption that we have again injective maps $\widetilde{j_{k'}}$ for all $k \leq k' \leq d$ for some k with $1 \leq k \geq d$, we can now proceed to define successively further (exact) sequences

$$\begin{tikzcd}
0 \arrow[r] & \widetilde{\mathcal{S}}_{k'} \arrow[r, "\widetilde{j}_{k'}"] \arrow[d] & \widetilde{i}_{k'*}\widetilde{i}_{k'}^{\,*}\widetilde{\mathcal{S}}_{k'} \arrow[r, "\widetilde{p}_{k'}"] \arrow[d] & \widetilde{\mathcal{S}}_{k'-1} \arrow[r] \arrow[d] & 0 \\
0 \arrow[r] & \mathcal{S}_{k'} \arrow[r, "j_{k'}"] & i_{k'*}i_{k}^{\,*}\mathcal{S}_{k'} \arrow[r, "p_{k'}"] & \mathcal{S}_{k'-1} \arrow[r] & 0,
\end{tikzcd} \tag{13.36}$$

for $d \geq k' \geq k$. This process terminates after $d+1$ steps if the iteratively defined quotient sheaves

$$\widetilde{\mathcal{S}}_{k-1} := \widetilde{\mathcal{A}}_k/\widetilde{j}_k\Big(\widetilde{\mathcal{S}}_k\Big) \qquad \mathcal{S}_{k-1} := \mathcal{A}_k/j_k\big(\mathcal{S}_k\big) \tag{13.37}$$

have only support in decreasing dimensions, i.e. we have in each step that

$$\dim\Big(\mathbf{supp}\,\widetilde{\mathcal{S}}_k\Big) < \dim\Big(\mathbf{supp}\,\widetilde{\mathcal{S}}_{k+1}\Big), \qquad \dim\Big(\mathbf{supp}\,\mathcal{S}_k\Big) < \dim\Big(\mathbf{supp}\,\mathcal{S}_{k+1}\Big) \tag{13.38}$$

holds. We show that this is true if all $\widetilde{j_{k'}}$ are injective for $k \leq k' \leq d$ (see Lemma 13.3.3). Then $\widetilde{\mathcal{A}}_k$ and $\mathcal{A}_k$ can be defined by (comp. Equation (13.31))

$$\widetilde{\mathcal{A}}_k := \widetilde{i}_{k*}\widetilde{i}_k^{\,*}\widetilde{\mathcal{S}}_k \qquad \mathcal{A}_k := i_{k*}i_k^{\,*}\mathcal{S}_k. \tag{13.39}$$

Note that for $k < 0$ these sheaves are equal to the zero-sheaf. As in the case of (13.33), one finds that the diagram (13.36) commutes. Therefore, we get the following identities

$$\Big(\pi_*\widetilde{\mathcal{A}}_k\Big)^{\mathbf{G}} = \mathcal{A}_k \qquad \Big(\pi_*\widetilde{\mathcal{S}}_k\Big)^{\mathbf{G}} = \mathcal{S}_k. \tag{13.40}$$

There is one special case in this construction, namely the step 0. It is easy to see that in this last non-trivial case the diagram (13.36) is somehow simpler, because the 0-skeleton coincides with the set of 0-cells, thus we have:

$$\begin{tikzcd}
 & & & 0 \arrow[d, equal] & \\
0 \arrow[r] & \widetilde{\mathcal{S}}_0 \arrow[r, hook, "\widetilde{j}_0"] \arrow[d] & \widetilde{i}_{0*}\widetilde{i}_0^{\,*}\widetilde{\mathcal{S}}_0 \arrow[r, "\widetilde{p}_0"] \arrow[d] & \widetilde{\mathcal{S}}_{-1} \arrow[r] \arrow[d] & 0 \\
0 \arrow[r] & \mathcal{S}_0 \arrow[r, hook, "j_0"] & i_{0*}i_0^{\,*}\mathcal{S}_0 \arrow[r, "p_0"] & \mathcal{S}_{-1} \arrow[r] \arrow[d, equal] & 0 \\
 & & & 0. &
\end{tikzcd} \tag{13.41}$$

In this case we can easily see that the maps $\widetilde{j}_0$ and j_0 are injective (see (13.41)). In fact, they are isomorphisms. We call the minimal k with $k \leq d$ such that for all k' with $k \leq k' \leq d$ the map $\widetilde{j_{k'}}$ is injective the **level of injectivity** (with respect to $\mathcal{P}$). We denote the this minimum by k_{inj}. By the above observation we find that if $k_{\text{inj}} < 1$ then $k_{\text{inj}} = -\infty$, since for zero we have always an isomorphism and for indices below zero the spaces are even zero. Thus, the only map is the zero map, which is in these cases an isomorphism. We conjecture that, for the example we are interested in, we always have that $k_{\text{inj}} = -\infty$ (see Conjecture 1 on page 195).

In Section 13.1.2 we introduced a so called star set $\mathcal{U}(\sigma)$ associated to a cell σ. Now we set

$$\mathcal{U}_\sigma^k = \mathbf{C}_k(\mathcal{X}) \cap \mathcal{U}(\sigma) \tag{13.42}$$

for a cell $\sigma \subset \mathcal{X}$ and denote by $\mathbf{Conn}_\sigma^k$ the set of connected components of $\mathcal{U}_\sigma^k$. For $\sigma = \mathcal{X}_i$ we use $\mathcal{U}_i^k := \mathcal{U}_{\mathcal{X}_i}^k$ and $\mathbf{Conn}_i^k := \mathbf{Conn}_{\mathcal{X}_i}^k$ instead. Note that $\mathbf{Conn}_\sigma^k$ is a set which contains only k-cells. Now we can state a collection of fundamental lemmata, which allow us to deduce essentially two different ways to compute the cohomology groups we are looking for.

Lemma 13.3.2 ((Orbi-)Constructibility): *Assume the sheaves $\widetilde{\mathcal{S}}_\ell$ are w-constructible*[7] *for all ℓ with $k_{inj} \leq k < \ell \leq d$, i.e. $\widetilde{j_{k'}}$ is injective for all k' with $k \leq k' \leq d$, then:*

a) The $\widetilde{\mathcal{S}}_k$ and all sheaves $\widetilde{\mathcal{A}}_{k'}$, and $\widetilde{\mathcal{A}}_{k-1}$ are w-constructible. If in addition $\mathbb{M}$ is a $\mathbb{Z}$-module of finite rank, then these sheaves are even constructible.

b) The sheaves $\mathcal{S}_{k'}$, $\mathcal{A}_{k'}$, and $\mathcal{A}_{k-1}$ are orbi-w-constructible. If $\mathbb{M}$ is a $\mathbb{Z}$-module of finite rank, then these sheaves are even orbi-constructible.

Note: By (13.30) we have that $\widetilde{\mathcal{S}}_d = \underline{\mathbb{M}}_{\mathbf{G}}$, which is a constant sheaf, which is w-constructible by Example 13.1.2. Therefore, there is a least one non-trivial case for this lemma.

Proof.

a) It is sufficient to prove the first part, because the second part then follows immediately from the definition of a constructible sheaf. By assumption, we know that $\widetilde{j_{k+1}}$ is injective. Hence, we get that the sequence

$$\begin{tikzcd} 0 \arrow[r] & \widetilde{\mathcal{S}}_{k+1} \arrow[r, "\widetilde{j}_{k+1}"] & \widetilde{i}_{k+1*}\widetilde{i}_{k+1}{}^*\widetilde{\mathcal{S}}_{k+1} \arrow[r, "\widetilde{p}_{k+1}"] & \widetilde{\mathcal{S}}_k \arrow[r] & 0 \end{tikzcd}$$

is exact and the sheaf $\widetilde{\mathcal{S}}_k$ is given by (13.37). This shows in particular that the sheaves $\widetilde{\mathcal{S}}_m$ and $\widetilde{\mathcal{A}}_m$ are well defined for $k \leq m \leq d$. We have, by Theorem 13.1.3 c) and d), that $\widetilde{\mathcal{A}}_{k+1} := \widetilde{i}_{k+1*}\widetilde{i}_{k+1}{}^*\widetilde{\mathcal{S}}_{k+1}$ has to be w-constructible, since we know that $\widetilde{\mathcal{S}}_{k+1}$ is w-constructible. According to Theorem 13.1.3 b), we get that if all non-zero sheaves but one in the short exact sequence above are w-constructible that also the third sheaf has to be w-constructible. This proves the first part.

b) This follows analogously to the previous case by Theorem 13.2.7 together with the fact that the functor $\mathscr{F} \mapsto \widehat{\mathscr{F}}$ is exact since all modules are defined over $\mathbb{Z}_{\mathbf{G}}$ (comp. the discussion of (13.34)). $\square$

[7]This implicitly means that the sheaves $\widetilde{\mathcal{A}}_m$ for m with $k+1 < m \leq d$ are also w-constructible and that all sheaves with larger indices are well defined.

Lemma 13.3.3 (Support): *The sheaf $\widetilde{\mathcal{S}}_k$ has – if it is defined – support in the k-skeleton $\mathbf{S}_k(\mathcal{X})$ of $\mathcal{X}$ and the sheaf $\mathcal{S}_k$ has support in the k-skeleton $\mathbf{S}_k(\mathcal{X}/\mathbf{G})$ of $\mathcal{X}/\mathbf{G}$.*

Proof. By equation (13.30) the sheaf $\widetilde{S}_d$ is defined to be equal to $\underline{\mathbb{M}}_{\mathbf{G}}$ and therefore has its support in the d-skeleton $\mathbb{S}_d(\mathcal{X})$, i.e. the full space. Suppose $\mathbf{supp}\left(\widetilde{\mathcal{S}}_{k'}\right) \subseteq \mathbf{S}_{k'}(\mathcal{X})$ for all $k' \geq k$. Then we can restrict the sheaf $\widetilde{\mathcal{S}}_k$ to $\mathbf{S}_k(\mathcal{X})$, which is an isomorphism. We know by sheaf theory that $\widetilde{i}_{k'}{}^*$ is an exact functor for all k'. Therefore, we get from (13.36) that the sequence

$$\begin{tikzcd} 0 \arrow[r] & \widetilde{i}_k{}^*\left(\widetilde{\mathcal{S}}_k\right) \arrow[r, "\widetilde{i}_k{}^*\widetilde{j}_k"] & \widetilde{i}_k{}^*\widetilde{i}_{k*}\widetilde{i}_k{}^*\left(\widetilde{\mathcal{S}}_k\right) \arrow[r, "\widetilde{i}_k{}^*\widetilde{p}_k"] & \widetilde{i}_k{}^*\left(\widetilde{\mathcal{S}}_{k-1}\right) \arrow[r] & 0 \end{tikzcd}$$

is also exact. Since $\widetilde{i}_k{}^*\widetilde{i}_{k*} = \mathrm{Id}$ and $\widetilde{i}_k{}^*\widetilde{j}_k$ is injective, because $\widetilde{j}_k$ is injective by induction hypothesis, we have that $\widetilde{i}_k{}^*\widetilde{j}_k$ is an isomorphism. Hence, by exactness of this short sequence of sheaves we know that $\widetilde{i}_k{}^*\widetilde{\mathcal{S}}_{k-1} = 0$ and thus the sheaf $\widetilde{\mathcal{S}}_{k-1}$ has only support in the complement of $\mathbf{im}\left(\widetilde{i}_k\right)$ in the k-skeleton $\mathbf{S}_k(\mathcal{X})$ of $\mathcal{X}$. In other words, we have

$$\mathbf{supp}\left(\widetilde{\mathcal{S}}_{k-1}\right) \subseteq \mathbf{S}_k(\mathcal{X}) - \mathbf{im}\left(\widetilde{i}_k\right) = \mathbf{S}_k(\mathcal{X}) - \mathbf{C}_k(\mathcal{X}) = \mathbf{S}_{k-1}(\mathcal{X}).$$

The claim for the sheaf $\mathcal{S}_{k-1}$ follows completely analogously. □

Next we have a closer look to stalks and the maps between them, which define, according to Lemma 13.1.17 and 13.2.5, the structure of a constructible sheave. We start with the stalks.

Lemma 13.3.4 (Stalks): *Let $k_{inj} \leq k \leq d$ and $d_i = \dim_{\mathbb{R}} \mathcal{X}_i$. Let further $\widetilde{x} \in \mathcal{X}_i$ be any point in $\mathcal{X}$ and $\left(\widetilde{x}_\sigma\right)_{\sigma\in\mathcal{P}}$ be a collection of arbitrarily chosen points $\widetilde{x}_\sigma \in \mathcal{X}$ such that $\widetilde{x}_\sigma \in \sigma$.*

Then we have that the sheaves $\widetilde{\mathcal{S}}_k$ and $\widetilde{\mathcal{A}}_k$ are w-constructible and the stalks of $\widetilde{\mathcal{S}}_k$ and $\widetilde{\mathcal{A}}_k$ in x are given by

$$\left(\widetilde{\mathcal{A}}_k\right)_{\widetilde{x}} = \bigoplus_{\sigma\in\mathbf{Conn}_i^k} \left(\widetilde{\mathcal{S}}_k\right)_{\widetilde{x}_\sigma} \qquad \left(\widetilde{\mathcal{S}}_k\right)_{\widetilde{x}} = \begin{cases} \mathbb{M}_{\mathbf{G}} & \text{for } k = d \\ 0 & 0 \leq k < d_i \leq d \\ \widetilde{\mathcal{S}}(i,k,\widetilde{x}) & 0 \leq d_i \leq k < d \\ 0 & k < 0, \end{cases}$$

where

$$\widetilde{\mathcal{S}}(i,k,\widetilde{x}) = \left(\widetilde{\mathcal{A}}_{k+1}\right)_{\widetilde{x}} \Big/ \left(\widetilde{j}_{k+1}(\widetilde{\mathcal{S}}_{k+1})\right)_{\widetilde{x}}.$$

The induced map on the stalks is given by

$$\left(\widetilde{j}_k\right)_{\widetilde{x}} = \bigoplus_{\sigma\in\mathbf{Conn}_i^k} \Phi^k_{\widetilde{x},\widetilde{x}_\sigma},$$

where the $\Phi^k_{-,-}$ are the compatibility maps associated to the w-constructible sheaf $\widetilde{\mathcal{S}}_k$ defined in (13.7), *and*

$$\widetilde{j}_k\big(\widetilde{\mathcal{S}}_k\big)_{\widetilde{x}} = \left\{ \mathbf{m} = (\mathbf{m}_\sigma) \in \bigoplus_{\sigma \in \mathbf{Conn}_i^k} \left(\widetilde{\mathcal{S}}_k\right)_{\widetilde{x}_\sigma} : \exists m \in \big(\widetilde{\mathcal{S}}_k\big)_{\widetilde{x}} : \forall \sigma \in \mathbf{Conn}_i^k : \mathbf{m}_\sigma = \Phi^k_{\widetilde{x},\widetilde{x}_\sigma}(m) \right\}.$$

Proof. The first part follows immediately from Lemma 13.3.2. For $k = d$ we have

$$\left(\widetilde{\mathcal{S}}_d\right)_{\widetilde{x}} = \mathbb{M}_{\mathbf{G}}$$

since $\widetilde{\mathcal{S}}_d$ is the constant sheaf with values in $\mathbb{M}_{\mathbf{G}}$. Let us now assume we know $\left(\widetilde{\mathcal{S}}_{k'}\right)_{\widetilde{x}}$ for all $k < k' \leq d$ and for each point $\widetilde{x} \in \mathcal{X}_i$. Then we have by Corollaries 13.1.11 and 13.1.13

$$\begin{aligned} \big(\widetilde{\mathcal{A}}_k\big)_{\widetilde{x}} &= \varinjlim_{U \ni \widetilde{x}} \widetilde{\mathcal{A}}_k(U) = \widetilde{\mathcal{A}}_k\big(\mathcal{U}(\widetilde{x})\big) = \widetilde{\mathcal{A}}_k\big(\mathcal{U}(\mathcal{X}_i)\big) \\ &= \widetilde{i}_{k*}\widetilde{i}_k^{\,*}\widetilde{\mathcal{S}}_k\big(\mathcal{U}(\mathcal{X}_i)\big) = \widetilde{i}_k^{\,*}\widetilde{\mathcal{S}}_k\Big(\widetilde{i}_k^{\,-1}\big(\mathcal{U}(\mathcal{X}_i)\big)\Big) \\ &= \widetilde{i}_k^{\,*}\widetilde{\mathcal{S}}_k\Big(\bigsqcup_{\sigma \in \mathbf{Conn}_i^k} \sigma\Big). \end{aligned}$$

Using the Mayer-Vietoris sequence (see e.g. [Ive86; 5.10]) for the space $\widetilde{i}_k^{\,-1}\big(\mathcal{U}(\mathcal{X}_i)\big) = \bigsqcup_{\sigma \in \mathbf{Conn}_i^k} \sigma$ and the sheaf $\widetilde{i}_k^{\,*}\widetilde{\mathcal{S}}_k$ on that space leads to

$$\big(\widetilde{\mathcal{A}}_k\big)_{\widetilde{x}} = \bigoplus_{\sigma \in \mathbf{Conn}_i^k} \widetilde{i}_k^{\,*}\widetilde{\mathcal{S}}_k(\sigma).$$

We can assume, by induction hypothesis, that for $k' > k$ the sheaf $\widetilde{\mathcal{S}}_{k'}$ and therefore by Theorem 13.1.3 d) also $\widetilde{i}_{k'}^{\,*}\widetilde{\mathcal{S}}_{k'}$ is (w-)constructible. We know by the definition of a (w-)constructible sheaf (see Definition 13.1.1) that then

$$\left(\widetilde{i}_k^{\,*}\widetilde{\mathcal{S}}_k\right)(\sigma) = \left(\widetilde{i}_k^{\,*}\widetilde{\mathcal{S}}_k\right)_{\widetilde{x}_\sigma} = \left(\widetilde{\mathcal{S}}_k\right)_{\widetilde{x}_\sigma},$$

where this equality does not depend of the choice of the points $\widetilde{x}_\sigma$. Hence, we get

$$\big(\widetilde{\mathcal{A}}_k\big)_{\widetilde{x}} = \bigoplus_{\sigma \in \mathbf{Conn}_i^k} \left(\widetilde{\mathcal{S}}_k\right)_{\widetilde{x}_\sigma}. \tag{13.43}$$

For all $k \leq d$, the map $\widetilde{j}_k$ induces, by the universal property of the direct limit, for each point $\widetilde{x} \in \mathcal{X}$ a map $(\widetilde{j}_k)_{\widetilde{x}}$. The above made observation on stalks of the sheaves $\widetilde{\mathcal{A}}_k$ gives us also a description of the induced maps $(\widetilde{j}_k)_{\widetilde{x}}$ using the $\Phi^k_{\_,\_}$. Let $\widetilde{x} \in \mathcal{X}_i$, then $\Phi^k_{\widetilde{y},\widetilde{x}}$ is a map from the stalk of $\widetilde{\mathcal{S}}_k$ in $\widetilde{y}$ to the stalk in $\widetilde{x}$. By (13.43) we can write the image of $\Phi^k_{\widetilde{y},\widetilde{x}}$ as direct sum over the k-cells lying around $\mathcal{X}_i$. For each of these k-cells $\sigma \in \mathbf{Conn}_i^k$ the map $\Phi^k_{\widetilde{x},\widetilde{x}_\sigma}$ is defined. One easily checks that the induced map is just the direct sum of all these maps. Hence, we have

$$\big(\widetilde{j}_k\big)_{\widetilde{x}} = \bigoplus_{\sigma \in \mathbf{Conn}_i^k} \Phi^k_{\widetilde{x},\widetilde{x}_\sigma}. \tag{13.44}$$

Therefore, we have

$$\widetilde{j}_k\big(\widetilde{\mathcal{S}}_k\big)_{\widetilde{x}} = \left\{ \mathbf{m} = (\mathbf{m}_\sigma) \in \bigoplus_{\sigma \in \mathbf{Conn}_i^k} \Big(\widetilde{\mathcal{S}}_k\Big)_{\widetilde{x}_\sigma} : \exists m \in \big(\widetilde{\mathcal{S}}_k\big)_{\widetilde{x}} : \forall \sigma \in \mathbf{Conn}_i^k : \mathbf{m}_\sigma = \Phi^k_{\widetilde{x},\widetilde{x}_\sigma}(m) \right\}. \tag{13.45}$$

By definition the sheaf $\widetilde{\mathcal{S}}_{k-1}$ is given by

$$\widetilde{\mathcal{S}}_{k-1} = \widetilde{\mathcal{A}}_k \Big/ \widetilde{j}_k\Big(\widetilde{\mathcal{S}}_k\Big).$$

Therefore, its stalk in a point $\widetilde{x} \in \mathcal{X}$ is given by

$$\Big(\widetilde{\mathcal{S}}_{k-1}\Big)_{\widetilde{x}} = \Big(\widetilde{\mathcal{A}}_k\Big)_{\widetilde{x}} \Big/ \Big(\widetilde{j}_k\big(\widetilde{\mathcal{S}}_k\big)\Big)_{\widetilde{x}}.$$

We already computed the stalks of $\widetilde{\mathcal{A}}_k$ in (13.43) as well as the stalks of $\widetilde{j}_k\big(\widetilde{\mathcal{S}}_k\big)_{\widetilde{x}}$ in (13.45). □

Corollary 13.3.5: *The maps $\widetilde{j}_d$ and j_d are injective.*

Proof. For $k = d$ we get immediately from the previous Lemma that

$$\big(\widetilde{\mathcal{A}}_d\big)_{\widetilde{x}} = \bigoplus_{\sigma \in \mathbf{Conn}_i^k} (\underline{\mathbb{M}_\mathbf{G}})_{\widetilde{x}_\sigma} = \bigoplus_{\sigma \in \mathbf{Conn}_i^k} \mathbb{M}_\mathbf{G}.$$

Since the compatibility maps are in this case all equal to the identity, the induced map $\widetilde{j}_d$ on the stalks is given by

$$\begin{aligned} \big(\widetilde{j}_d\big)_{\widetilde{x}} : \big(\widetilde{\mathcal{S}}_d\big)_{\widetilde{x}} &\to \big(\widetilde{\mathcal{A}}_d\big)_{\widetilde{x}} \\ \mathbf{m} &\mapsto (\mathbf{m})_{\sigma \in \mathbf{Conn}_i^k}, \end{aligned}$$

which is obviously injective. Therefore, the upper sequence in (13.33) is exact, or, to be more precise, the first two non-zero terms can be completed with $\widetilde{\mathcal{S}}_{d-1}$ as defined in (13.35) to an exact sequence. The vertical maps in (13.33) are given by the functor defined in (13.34), which is left exact. Therefore, also j_d is injective. □

The question whether the other maps $\widetilde{j}_k$ and j_k are injective for $k < d$ is more difficult than for $k = d$. We conjecture that all of them are injective (see Conjecture 1 on page 195), but we cannot prove this in general.

Let $\widetilde{x} \in \mathcal{X}_i$ and $\widetilde{y} \in \mathcal{X}_j$ such that (i, j) is a valid pair for the underlying partition of the sheaves $\widetilde{\mathcal{S}}_k$ and $\widetilde{\mathcal{A}}_k$, which are known to be w-constructible sheaves by Lemma 13.3.2. Therefore, there is, according to Theorem 13.1.6, a collection of compatibility maps for each of them, which we denote by $\Phi^{\widetilde{\mathcal{S}}_k}_{\widetilde{x},\widetilde{y}}$ and $\Phi^{\widetilde{\mathcal{A}}_k}_{\widetilde{x},\widetilde{y}}$. To shorten the notation we set

$$\Phi^k_{i,j} := \Phi^{\widetilde{\mathcal{S}}_k}_{\widetilde{x},\widetilde{y}} \qquad\qquad \underline{\Phi^k_{i,j}} := \Phi^{\widetilde{\mathcal{A}}_k}_{\widetilde{x},\widetilde{y}}.$$

It is not difficult to check with the already made observations that the compatibility maps $\underline{\Phi^k_{i,j}}$ associated to the sheaf $\widetilde{\mathcal{A}}_k$ are given by

$$\begin{aligned} \underline{\Phi^k_{i,j}} : \big(\widetilde{\mathcal{A}}_k\big)_{\widetilde{x}} &\to \big(\widetilde{\mathcal{A}}_k\big)_{\widetilde{y}} \\ (\mathbf{m}_\sigma)_{\sigma\in \mathbf{Conn}^k_i} &\mapsto (\mathbf{m}_\tau)_{\tau\in \mathbf{Conn}^k_j}. \end{aligned} \tag{13.46}$$

Note that by definition of $\mathcal{U}(\mathcal{X}_i)$ and $\mathcal{U}(\mathcal{X}_j)$ in (13.3) and the definition of the set $\mathbf{Conn}^k_i$ we know that for each valid pair (i, j)

$$\mathbf{Conn}^k_j \subseteq \mathbf{Conn}^k_i \tag{13.47}$$

holds. Hence, the map $\underline{\Phi^k_{i,j}}$ is an epimorphism, i.e. it is surjective.

For $k = d$ we have that $\Phi^d_{i,j} = \mathrm{Id}$ holds for all valid pairs, since $\widetilde{\mathcal{S}}_d = \underline{\mathbb{M}}_{\mathbf{G}}$ (see (13.30)). The compatibility maps $\Phi^k_{i,j}$ for the sheaf $\widetilde{\mathcal{S}}_k$ for $k < d$ can be defined inductively. By the universal property of the direct limit we get from the upper row in (13.36) induced maps $(\widetilde{p_k})_{\widetilde{x}}$ and $(\widetilde{p_k})_{\widetilde{y}}$ on the stalks in $\widetilde{x}$ and $\widetilde{y}$ respectively, which are surjective by definition. We define $\Phi^{k-1}_{i,j}$ by the condition to complete the following diagram

$$\begin{tikzcd} \big(\widetilde{\mathcal{A}}_k\big)_{\widetilde{x}} \arrow[r, "\underline{\Phi^k_{i,j}}"] \arrow[d, dashed, "(\widetilde{p}_k)_{\widetilde{x}}"'] & \big(\widetilde{\mathcal{A}}_k\big)_{\widetilde{y}} \arrow[d, dashed, "(\widetilde{p}_k)_{\widetilde{y}}"] \\ \big(\widetilde{\mathcal{S}}_{k-1}\big)_{\widetilde{x}} \arrow[r, "\Phi^{k-1}_{i,j}"] & \big(\widetilde{\mathcal{S}}_{k-1}\big)_{\widetilde{y}} \end{tikzcd} \tag{13.48}$$

to a commutative diagram. Since $(\widetilde{p_k})_{\widetilde{x}}$ and $(\widetilde{p_k})_{\widetilde{y}}$ are surjective we find for each point $\widetilde{x} \in \mathcal{X}$ a preimage in $\big(\widetilde{\mathcal{A}}_k\big)_{\widetilde{x}}$ under $(\widetilde{p}_k)_{\widetilde{x}}$. Let $\mathbf{m}$ and $\mathbf{m}'$ be two different preimages, then there is an element $\mathbf{n} \in \big(\widetilde{\mathcal{S}}_k\big)_{\widetilde{x}}$ such that

$$\mathbf{m}' = \mathbf{m} + \widetilde{j}_k(\mathbf{n}).$$

To prove that the map $\Phi^k_{i,j}$ is uniquely determined by the above diagram we have to show that $\mathbf{m}$ and $\mathbf{m}'$ are mapped to the same image under $(\widetilde{p}_k)_{\widetilde{y}} \circ \underline{\Phi^k_{i,j}}$. Since all maps are linear it is sufficient to show that $(\widetilde{j}_k)_{\widetilde{x}}(\mathbf{n})$ is mapped to $0 \in \big(\widetilde{\mathcal{S}}_{k-1}\big)_{\widetilde{y}}$. We have by (13.44) that

$$(\widetilde{j}_k)_{\widetilde{x}}(\mathbf{n}) = \bigoplus_{\sigma\in\mathbf{Conn}^k_i} \Phi^k_{\widetilde{x},\widetilde{x}_\sigma}(\mathbf{n}).$$

But by the description of $\underline{\Phi^k_{i,j}}$ in (13.46) we get

$$\underline{\Phi^k_{i,j}}\big((\widetilde{j}_k)_{\widetilde{x}}(\mathbf{n})\big) = \bigoplus_{\sigma\in\mathbf{Conn}^k_j} \Phi^k_{\widetilde{x},\widetilde{x}_\sigma}(\mathbf{n}).$$

We know by (13.44) together with Theorem 13.1.15 d)

$$\bigoplus_{\sigma\in\mathbf{Conn}^k_j} \Phi^k_{\widetilde{x},\widetilde{x}_\sigma}(\mathbf{n}) = \bigoplus_{\sigma\in\mathbf{Conn}^k_j} \Phi^k_{\widetilde{y},\widetilde{x}_\sigma}(\mathbf{n}')$$

for $\mathbf{n}' = \Phi^k_{\widetilde{x},\widetilde{y}}(\mathbf{n}) \in \left(\widetilde{\mathcal{A}}_k\right)_{\widetilde{y}}$,which lies in $\left(\widetilde{j}_k\widetilde{\mathcal{S}}_k\right)_{\widetilde{y}}$ and is therefore mapped onto 0 under $\left(\widetilde{p}_k\right)_{\widetilde{y}}$. Thus, the map $\Phi^{k-1}_{i,j}$ is well defined and, therefore all $\Phi^{k'}_{i,j}$ for $k_0 - 1 \leq k' \leq d$.

To get the structure of a $\mathbf{G}$-w-constructible system related to the sheaves $\widetilde{\mathcal{S}}_k$ and $\widetilde{\mathcal{A}}_k$ we have to describe an additional family of compatibility maps (comp. Definition 13.2.1), i.e. we need to define

$$\Psi^k_{i,\mathbf{g}} := \Psi^{\widetilde{\mathcal{S}}_k}_{\widetilde{x},\mathbf{g}} \qquad\qquad \underline{\Psi^k_{i,\mathbf{g}}} := \Psi^{\widetilde{\mathcal{A}}_k}_{\widetilde{x},\mathbf{g}}, \tag{13.49}$$

where $\widetilde{x} \in \mathcal{X}_i$. Let us – similar to the previously discussed maps – consider the following diagram for $k_0 \leq k \leq d$

$$\begin{tikzcd} 0 \arrow[r] & \left(\widetilde{\mathcal{S}}_k\right)_{\widetilde{x}} \arrow[r, "(\widetilde{j}_k)_{\widetilde{x}}"] \arrow[d, "\Psi^k_{i,\mathbf{g}}"'] & \left(\widetilde{\mathcal{A}}_k\right)_{\widetilde{x}} \arrow[r, "(\widetilde{p}_k)_{\widetilde{x}}"] \arrow[d, "\underline{\Psi^k_{i,\mathbf{g}}}"'] & \left(\widetilde{\mathcal{S}}_{k-1}\right)_{\widetilde{x}} \arrow[r] \arrow[d, "\Psi^{k-1}_{i,\mathbf{g}}"'] & 0 \\ 0 \arrow[r] & \left(\widetilde{\mathcal{S}}_k\right)_{\widetilde{x}} \arrow[r, "(\widetilde{j}_k)_{\widetilde{x}\cdot\mathbf{g}}"'] & \left(\widetilde{\mathcal{A}}_k\right)_{\widetilde{x}} \arrow[r, "(\widetilde{j}_k)_{\widetilde{x}\cdot\mathbf{g}}"'] & \left(\widetilde{\mathcal{A}}_{k-1}\right)_{\widetilde{x}} \arrow[r] & 0. \end{tikzcd} \tag{13.50}$$

Since $\widetilde{S}_d = \underline{\mathbb{M}}_{\mathbf{G}}$ we have

$$\Psi^d_{i,\mathbf{g}} = \mathrm{Id}_{(\widetilde{\mathcal{S}}_k)_{\widetilde{x}}}. \tag{13.51}$$

We can now write down the remaining maps inductively by (13.50). It is not difficult to check that for $\mathbf{m} \in \left(\widetilde{\mathcal{A}}_k\right)_{\widetilde{x}}$

$$\underline{\Psi^k_{i,\mathbf{g}}}(\mathbf{m}) = \bigoplus_{\sigma\in\mathbf{Conn}^k_j} \Psi_{x_\sigma,\mathbf{g}} \tag{13.52}$$

holds, where the $(x_\sigma)_{\sigma\in\mathcal{P}}$ are some arbitrarily chosen points such that $x_\sigma \in \sigma$ (comp. Lemma 13.3.4). Since according to the definition of w-constructible sheaves the stalks are constant on cells, we know that the map does not depend on the choice of these points. Now we can proceed, as before, and define the map $\Psi^{k-1}_{i,\mathbf{g}}$ by the condition that it has to make (13.50) into a commutative diagram. Analogously to the last case, one shows by application of (13.22) together with the description of $(\widetilde{j}_k)_{\widetilde{x}}$ from Lemma 13.3.4 that this definition does not depend on the choice of the representative of an element from $\left(\widetilde{\mathcal{S}}_{k-1}\right)_{\widetilde{x}}$ in $\left(\widetilde{\mathcal{A}}_k\right)_{\widetilde{x}}$.

The map $\Psi^k_{i,\mathbf{g}}$ induces the needed structure of a $\mathbf{G}_i$-module on the stalk $\left(\widetilde{\mathcal{S}}_k\right)_{\widetilde{x}}$. Analogously, $\underline{\Psi^k_{i,\mathbf{g}}}$ defines the $\mathbf{G}_i$-module structure on $\left(\widetilde{\mathcal{A}}_k\right)_{\widetilde{x}}$.

As an easy consequence of Lemma 13.3.3 on the support of the sheaves $\mathcal{S}_k$ and $\widetilde{\mathcal{S}}_k$ we get:

Corollary 13.3.6: *We have*

$$H^q\left(\mathcal{X}/\mathbf{G}, \mathcal{S}_k\right) = 0 \qquad \textit{and} \qquad H^q\left(\mathcal{X}, \widetilde{\mathcal{S}}_k\right) = 0$$

for $q > k \geq k_{inj}$.

Theorem 13.3.7: *The sheaves $\mathcal{A}_k$ and $\widetilde{\mathcal{A}}_k$ are acyclic for $k_{inj} \leq k \leq d$.*

Proof. It is sufficient to prove the claim only for the sheaf $\widetilde{\mathcal{A}}_k$, because according to (13.40) we have $(\pi_*\widetilde{\mathcal{A}}_k)^{\mathbf{G}} = \mathcal{A}_k$. According to Definition 5.1.2 we have to show

$$H^q(\mathcal{X}, \widetilde{\mathcal{A}}_k) = \begin{cases} \widetilde{\mathcal{A}}_k\,(\mathcal{X}) & \text{if } q = 0 \\ 0 & \text{if } q > 0. \end{cases}$$

It is clear by definition that for $q = 0$ the cohomology equals the global sections. It remains to show that higher cohomology groups vanish. The map $\widetilde{i}_k$ is an inclusion and we have $\mathbf{im}\left(\widetilde{i}_k\right) = \mathbf{C}_k(\mathcal{X})$, which is open in its closure. We know from Lemma 13.3.3 that $\mathbf{supp}\,\widetilde{\mathcal{A}}_k \subseteq \mathbf{S}_k(\mathcal{X})$ and have by assumption that

$$\mathbf{C}_k(\mathcal{X}) = \bigsqcup_{\substack{i\in\mathcal{I} \\ \dim \mathcal{X}_i = k}} \mathcal{X}_i = \bigsqcup_{i\in\mathcal{I}^k} \mathcal{X}_i,$$

where $\mathcal{I}^k \subseteq \mathcal{I}$ is countable since the decomposition is locally finite. Hence, we get:

$$\begin{aligned} H^q(\mathcal{X}, \widetilde{\mathcal{A}}_k) &= H^q\Big(\mathcal{X}, \widetilde{i}_{k*}\widetilde{i}_k^{\,*}\widetilde{\mathcal{S}}_k\Big) = H^q\Big(\mathbf{C}_k(\mathcal{X}), \widetilde{i}_k^{\,*}\widetilde{\mathcal{S}}_k\Big) \\ &= H^q\Big(\bigsqcup_{i\in\mathcal{I}^k} \mathcal{X}_i, \widetilde{i}_k^{\,*}\widetilde{\mathcal{S}}_k\Big) = \bigoplus_{i\in\mathcal{I}^k} H^q\Big(\mathcal{X}_i, \widetilde{i}_k^{\,*}\widetilde{\mathcal{S}}_k\Big). \end{aligned}$$

The $\mathcal{X}_i$ are (open) k-cells, i.e. they are homeomorphic to an open k-ball $\mathbf{B}_k$. In other words, it is enough to know the cohomology of $\mathbf{B}_k$ for any given sheaf $\mathscr{F}$ on $\mathbf{B}_k$, which is known to be

$$H^q(\mathbf{B}_k, \mathscr{F}) = \begin{cases} \mathscr{F}(\mathbf{B}_k) & \text{if } q = 0 \\ 0 & \text{if } q > 0. \end{cases}$$

Therefore, we get for $q > 0$

$$H^q\Big(\mathcal{X}, \widetilde{\mathcal{A}}_k\Big) = \bigoplus_{i\in\mathcal{I}^k} H^q\Big(\mathcal{X}_i, \widetilde{i}_k^{\,*}\widetilde{\mathcal{S}}_k\Big) = \bigoplus_{i\in\mathcal{I}^k} H^q\Big(\mathbf{B}_k, \widetilde{i}_k^{\,*}\widetilde{cS}_k\Big) = 0.$$

This proves the claim for the sheaf $\widetilde{\mathcal{A}}_k$. □

We can now define two families of maps $\widetilde{\delta}_k$ and δ_k for $d \geq k \geq k_{\text{inj}}$ by

$$\widetilde{\delta}_k = \widetilde{j}_{k-1} \circ \widetilde{p}_k \qquad\qquad \delta_k = j_{k-1} \circ p_k. \tag{13.53}$$

If $k_{\text{inj}} \leq 0$ we get from (13.41) that $\widetilde{p}_0 = \widetilde{\delta}_0 = 0$ and $p_0 = \delta_0 = 0$.

We can combine the exact sequences from the rows in (13.36) to two commutative diagrams (see the two diagrams shown in Figure 13.3 on page 177).

Then the sequences in Figure 13.3 will have maximal length and the most simple form if $k_{\text{inj}} \leq 0$. For indices below zero all sheaves are zero and the maps are the zero map. Therefore, they are not necessary for our computations.

Figure 13.3.

$$\begin{tikzcd}
 & 0 \arrow[d] & & & & & & & & \\
 & \mathcal{M}_{\mathbb{W}} \arrow[d, hook, "j_d"'] & & & & & & & & \\
 & \mathcal{A}_d \arrow[d, two heads, "p_d"] \arrow[dr, "\delta_d"] & & 0 \arrow[d] & & & & & & \\
0 \arrow[r] & \mathcal{S}_{d-1} \arrow[r, hook, "j_{d-1}"'] \arrow[d] & \mathcal{A}_{d-1} \arrow[r, two heads, "p_{d-1}"] \arrow[dr, "\delta_{d-1}"'] & \mathcal{S}_{d-2} \arrow[r] \arrow[d, hook, "j_{d-2}"] & 0 & & & & & \\
 & 0 & & \mathcal{A}_{d-2} \arrow[d, two heads, "p_{d-2}"'] \arrow[dr, "\delta_{d-2}"] & & 0 \arrow[d] & & & & \\
 & 0 \arrow[d] & 0 \arrow[r] & \mathcal{S}_{d-3} \arrow[r, hook, "j_{d-3}"'] \arrow[d] & \mathcal{A}_{d-3} \arrow[r, two heads, "p_{d-3}"] \arrow[dr, "\delta_{d-3}"'] & \mathcal{S}_{d-4} \arrow[r] \arrow[d, "j_{d-4}"] & 0 & & & \\
 & \underline{\mathbb{M}}_{\mathbf{G}} \arrow[d, hook, "\widetilde{j}_d"'] & & 0 & & \ddots \arrow[dr, "\delta_{k_{\mathrm{inj}}+2}"'] & \vdots \arrow[d, hook, "j_{k_{\mathrm{inj}}+1}"] & & & \\
 & \widetilde{\mathcal{A}}_d \arrow[d, two heads, "\widetilde{p_d}"] \arrow[dr, "\widetilde{\delta}_d"] & & 0 \arrow[d] & & & \mathcal{A}_{k_{\mathrm{inj}}+1} \arrow[d, two heads, "p_{k_{\mathrm{inj}}+1}"'] \arrow[dr, "\delta_{k_{\mathrm{inj}}+1}"] & & & \\
0 \arrow[r] & \widetilde{\mathcal{S}}_{d-1} \arrow[r, hook, "\widetilde{j}_{d-1}"'] \arrow[d] & \widetilde{\mathcal{A}}_{d-1} \arrow[r, two heads, "\widetilde{p}_{d-1}"] \arrow[dr, "\widetilde{\delta}_{d-1}"'] & \widetilde{\mathcal{S}}_{d-2} \arrow[r] \arrow[d, hook, "\widetilde{j}_{d-2}"] & 0 & 0 \arrow[r] & \mathcal{S}_{k_{\mathrm{inj}}} \arrow[r, hook, "j_{k_{\mathrm{inj}}}"'] \arrow[d] & \mathcal{A}_{k_{\mathrm{inj}}} \arrow[r, two heads, "p_{k_{\mathrm{inj}}}"] & \mathcal{S}_{k_{\mathrm{inj}}-1} \arrow[r] & 0 \\
 & 0 & & \widetilde{\mathcal{A}}_{d-2} \arrow[d, two heads, "\widetilde{p}_{d-2}"'] \arrow[dr, "\widetilde{\delta}_{d-2}"] & & 0 \arrow[d] & 0 & & & \\
 & & 0 \arrow[r] & \widetilde{\mathcal{S}}_{d-3} \arrow[r, hook, "\widetilde{j}_{d-3}"'] \arrow[d] & \widetilde{\mathcal{A}}_{d-3} \arrow[r, two heads, "\widetilde{p}_{d-3}"] \arrow[dr, "\widetilde{\delta}_{d-3}"'] & \widetilde{\mathcal{S}}_{d-4} \arrow[r] \arrow[d, "\widetilde{j}_{d-4}"] & 0 & & & \\
 & & & 0 & & \ddots \arrow[dr, "\widetilde{\delta}_{k_{\mathrm{inj}}+2}"'] & \vdots \arrow[d, hook, "\widetilde{j}_{k_{\mathrm{inj}}+1}"] & & & \\
 & & & & & & \widetilde{\mathcal{A}}_{k_{\mathrm{inj}}+1} \arrow[d, two heads, "\widetilde{p}_{k_{\mathrm{inj}}+1}"'] \arrow[dr, "\widetilde{\delta}_{k_{\mathrm{inj}}+1}"] & & & \\
 & & & & & 0 \arrow[r] & \widetilde{\mathcal{S}}_{k_{\mathrm{inj}}} \arrow[r, hook, "\widetilde{j}_{k_{\mathrm{inj}}}"'] \arrow[d] & \widetilde{\mathcal{A}_{k_{\mathrm{inj}}}} \arrow[r, two heads, "\widetilde{p}_{k_{\mathrm{inj}}}"] & \widetilde{\mathcal{S}}_{k_{\mathrm{inj}}-1} \arrow[r] & 0 \\
 & & & & & & 0 & & &
\end{tikzcd}$$

If $k_{\text{inj}} > 0$ the corresponding sequences will stop earlier as shown in the sequences in Figure 13.3 since the sheaves $\widetilde{\mathcal{S}}_k$ and $\mathcal{S}_k$ are not defined for $k < k_{\text{inj}} - 1$. In these cases we can try to replace for them the corresponding short exact sequences by longer ones and in principle we could proceed analogously to the method we describe now.

But we assume for the moment that $k_{\text{inj}} \leq 0$. Then we can extract from each of the two sequences in Figure 13.3 a sequence (taking the bold blue arrows):

$$\begin{tikzcd}
0 \arrow[r] & \underline{\mathbb{M}}_{\mathbf{G}} \arrow[r, hook, "\widetilde{j}_d"] \arrow[d] & \widetilde{\mathcal{A}}_d \arrow[r, "\widetilde{\delta}_d"] \arrow[d] & \widetilde{\mathcal{A}}_{d-1} \arrow[r, "\widetilde{\delta}_{d-1}"] \arrow[d] & \ldots \arrow[r, "\widetilde{\delta}_3"] & \widetilde{\mathcal{A}}_2 \arrow[r, "\widetilde{\delta}_2"] \arrow[d] & \widetilde{\mathcal{A}}_1 \arrow[r, two heads, "\widetilde{\delta}_1"] \arrow[d] & \widetilde{\mathcal{A}}_0 \arrow[r, two heads, "\widetilde{\delta}_0"] \arrow[d] & 0 \\
0 \arrow[r] & \mathcal{M}_{\mathbf{G}} \arrow[r, hook, "j_d"] & \mathcal{A}_d \arrow[r, "\delta_d"] & \mathcal{A}_{d-1} \arrow[r, "\delta_{d-1}"] & \ldots \arrow[r, "\delta_3"] & \mathcal{A}_2 \arrow[r, "\delta_2"] & \mathcal{A}_1 \arrow[r, two heads, "\delta_1"] & \mathcal{A}_0 \arrow[r, two heads, "\delta_0"] & 0.
\end{tikzcd} \tag{13.54}$$

Theorem 13.3.8: *Let $k_{inj} \leq 0$. Then*

a) The sequence of sheaves $0 \to \underline{\mathbb{M}}_{\mathbf{G}} \to \widetilde{\mathcal{A}}_\bullet$ forms an acyclic resolution for the sheaf $\underline{\mathbb{M}}_{\mathbf{G}}$ on $\mathcal{X}$.

b) The sequence of sheaves $0 \to \mathcal{M}_{\mathbf{G}} \to \mathcal{A}_\bullet$ forms an acyclic resolution for the sheaf $\mathcal{M}_{\mathbf{G}}$ on $\mathcal{X}/\mathbf{G}$.

Proof. The sheaves $\widetilde{\mathcal{A}}_k$ and $\mathcal{A}_k$ are acyclic by Theorem 13.3.7. It remains to show that the sequences are exact. The upper horizontal sequence in (13.54) is exact in $\underline{\mathbb{M}}_{\mathbf{G}}$ since (13.33) is exact in the first row, i.e. $\widetilde{j}_d$ is in particular injective (comp. e.g. Corollary 13.3.5). Next we have to show that $\mathbf{im}\,\widetilde{j}_d = \mathbf{ker}\,\widetilde{\delta}_d$. By definition we have $\widetilde{\delta}_d = \widetilde{j}_{d-1} \circ \widetilde{p}_d$, where, because of exactness of the first vertical and the first horizontal sequence in Diagram 13.3 on page 177, $\widetilde{p}_d$ is surjective and $\widetilde{j}_{d-1}$ is injective because $d > k_{\text{inj}}$ by Corollary 13.3.5. Since $\widetilde{j}_{d-1}$ is injective we conclude

$$\mathbf{ker}\,\widetilde{\delta}_d = \mathbf{ker}\,\widetilde{j}_{d-1} \circ \widetilde{p}_d = \mathbf{ker}\,\widetilde{p}_d.$$

But by exactness of the upper row of (13.33) we know already that $\mathbf{im}\,\widetilde{j}_d = \mathbf{ker}\,\widetilde{p}_d$ which proves exactness at $\widetilde{\mathcal{A}}_d$. We have to show $\mathbf{ker}\,\widetilde{\delta}_{k-1} = \mathbf{im}\,\widetilde{\delta}_k$ for $k = d, \ldots, 1$. On the one hand we know that the the maps $\widetilde{j}_k$ are injective and get Therefore, by exactness of (13.36) for $k-1$

$$\mathbf{ker}\,\widetilde{\delta}_k = \mathbf{ker}\,\widetilde{j}_{k-2} \circ \widetilde{p}_{k-1} = \mathbf{ker}\,\widetilde{p}_{k-1} = \mathbf{im}\,\widetilde{j}_{k-1}.$$

On the other hand we have

$$\mathbf{im}\,\widetilde{\delta}_{k-1} = \mathbf{im}\,\widetilde{j}_{k-1} \circ \widetilde{p}_k = \mathbf{im}\,\widetilde{j}_{k-1}.$$

This shows exactness in $\widetilde{\mathcal{A}}_k$ for the remaining non-zero cases. The exactness for the sequence in the quotient can either be obtained by repeating the procedure but with the tildes removed or one applies (13.40), which also immediately proves the claim. $\square$

We can now apply the general principle from Theorem 5.1.4 and get immediately

Theorem 13.3.9: *Let $k_{inj} \leq 0$ then we have*

$$H^q(\mathcal{X}, \underline{\mathbb{M}}_{\mathbf{G}}) = H^q(H^0(\mathcal{X}, \widetilde{\mathcal{A}}_\bullet))$$

and

$$H^q(\mathcal{X}/\mathbf{G}, \mathcal{M}_{\mathbf{G}}) = H^q(H^0(\mathcal{X}/\mathbf{G}, \mathcal{A}_\bullet))$$

for $0 \leq q \leq d$.

13.3.2. Interpretation of the Terms in the Resolutions

Recall that we have by Lemma 13.3.4 that

$$(\widetilde{\mathcal{A}}_k)_{\widetilde{x}} = \bigoplus_{\sigma \in \mathbf{Conn}_i^k} (\widetilde{\mathcal{S}}_k)_{\widetilde{x}_\sigma}.$$

We reinterpret this equality with respect to the action of group elements. There is a natural action of the stabilizer $\mathbf{G}_i$ of the central cell on the set $\mathbf{Conn}_i^k$. Denote by

$$\Delta_i^k = \mathbf{Conn}_i^k / \mathbf{G}_i \tag{13.55}$$

the set of equivalence classes under this action. In general this action is not transitive. Each class $\delta \in \Delta_i^k$ contains at least one cell. Let $\sigma_\delta \in \delta$ be an arbitrarily chosen representative, i.e. σ_δ is a k-cell in $\mathcal{X}$. Let $\widetilde{x}_\delta := \widetilde{x}_{\sigma_\delta}$ be the point in σ_δ which occurs in the set of points in Lemma 13.3.4. Let $\mathbf{G}_\delta := \mathbf{G}_{\sigma_\delta}$. We observe that $\sigma_\delta = \sigma_\delta \cdot (\mathbf{G}_\delta \cap \mathbf{G}_i)$.

Therefore, we get from the above equality

$$(\widetilde{\mathcal{A}}_k)_{\widetilde{x}} = \bigoplus_{\delta \in \Delta_i^k} \underbrace{\bigoplus_{\mathbf{g} \in \mathbf{G}_i / \mathbf{G}_\delta \cap \mathbf{G}_i} \Psi^k_{\widetilde{x}_\delta, \mathbf{g}} \left((\widetilde{\mathcal{S}}_k)_{\widetilde{x}_\delta} \right)}_{\overset{(13.18)}{=} \mathrm{Ind}^{\mathbf{G}_i}_{\mathbf{G}_\delta \cap \mathbf{G}_i} \left((\widetilde{\mathcal{S}}_k)_{\widetilde{x}_\delta} \right)} = \bigoplus_{\delta \in \Delta_i^k} \mathrm{Ind}^{\mathbf{G}_i}_{\mathbf{G}_\delta \cap \mathbf{G}_i} \left((\widetilde{\mathcal{S}}_k)_{\widetilde{x}_\delta} \right). \tag{13.56}$$

One easily checks with condition b) on page 161 that the representation does (up to isomorphism) not depend on the choice of the representatives of the classes in Δ_i^k. We get:

Proposition 13.3.10: *Let $k \geq k_0$ and $\widetilde{x} \in \mathcal{X}_i$. Then*

$$(\widetilde{\mathcal{A}}_k)_{\widetilde{x}} = \bigoplus_{\delta \in \Delta_i^k} \mathrm{Ind}^{\mathbf{G}_i}_{\mathbf{G}_\delta \cap \mathbf{G}_i} \left((\widetilde{\mathcal{S}}_k)_{\widetilde{x}_\delta} \right)$$

holds.

We find for $\widetilde{x} \in \mathcal{X}_i$ and $\mathbf{m} \in (\widetilde{\mathcal{S}}_k)_{\widetilde{x}}$ that we can rewrite the map $\widetilde{\jmath}_k$ in the following form:

$$\begin{aligned}\widetilde{\jmath}_k(\mathbf{m}) &= \bigoplus_{\sigma\in\mathbf{Conn}_i^k} \Phi_{\widetilde{x},\widetilde{x}_\sigma}(\mathbf{m}) = \bigoplus_{\delta\in\Delta_i^k} \bigoplus_{\mathbf{g}\in\mathbf{G}_i/\mathbf{G}_\delta\cap\mathbf{G}_i} \Psi^k_{\widetilde{x}_\delta,\mathbf{g}}\left(\Phi_{\widetilde{x},\widetilde{x}_\delta}(\mathbf{m})\right) \\ &= \bigoplus_{\delta\in\Delta_i^k} \operatorname{Ind}_{\mathbf{G}_\delta\cap\mathbf{G}_i}^{\mathbf{G}_i}\left(\Phi_{\widetilde{x},\widetilde{x}_\delta}(\mathbf{m})\right)\end{aligned} \tag{13.57}$$

Since by Lemma 13.3.2 we have that $\widetilde{\mathcal{A}}_k$ is constructible for $k \geq k_{\mathrm{inj}}$ we can apply Theorem 13.2.2 together with Proposition 13.3.10 and get

$$\begin{aligned}H^0(\mathcal{X},\widetilde{\mathcal{A}}_k) &= \varinjlim_{i\in\mathcal{I}} \mathbb{F}_i = \varinjlim_{\mathcal{X}_i\in\mathcal{P}} (\widetilde{\mathcal{A}}_k)_{\widetilde{x}_{\mathcal{X}_i}} = \varinjlim_{i\in\mathcal{I}} \bigoplus_{\delta\in\Delta_i^k} \operatorname{Ind}_{\mathbf{G}_\delta\cap\mathbf{G}_i}^{\mathbf{G}_i}\left((\widetilde{\mathcal{S}}_k)_{\widetilde{x}_\delta}\right) \\ &= \bigoplus_{i\in\mathcal{I}} \bigoplus_{\delta\in\Delta_i^k} \operatorname{Ind}_{\mathbf{G}_\delta\cap\mathbf{G}_i}^{\mathbf{G}_i}\left((\widetilde{\mathcal{S}}_k)_{\widetilde{x}_\delta}\right) \Big/_{\sim},\end{aligned}$$

where $\sim$ is the equivalence relation on the factors from (13.10). By (13.13) we get

$$\begin{aligned}H^0(\mathcal{X},\widetilde{\mathcal{A}}_k) &= \bigoplus_{i\in\mathcal{I}_{\mathbf{G}}} \prod_{\mathbf{g}\in\mathbf{G}/\mathbf{G}_i} \Psi^k_{i,\mathbf{g}}\left(\bigoplus_{\delta\in\Delta_i^k} \operatorname{Ind}_{\mathbf{G}_\delta\cap\mathbf{G}_i}^{\mathbf{G}_i}\left((\widetilde{\mathcal{S}}_k)_{\widetilde{x}_\delta}\right)\right) \Big/_{\sim} \\ &= \bigoplus_{i\in\mathcal{I}_{\mathbf{G}}} \bigoplus_{\delta\in\Delta_i^k} \underbrace{\prod_{\mathbf{g}\in\mathbf{G}/\mathbf{G}_i} \Psi^k_{i,\mathbf{g}}\left(\operatorname{Ind}_{\mathbf{G}_\delta\cap\mathbf{G}_i}^{\mathbf{G}_i}\left((\widetilde{\mathcal{S}}_k)_{\widetilde{x}_\delta}\right)\right)}_{\overset{(13.18)}{=}\operatorname{Ind}_{\mathbf{G}_i}^{\mathbf{G}}\left(\operatorname{Ind}_{\mathbf{G}_\delta\cap\mathbf{G}_i}^{\mathbf{G}_i}\left((\widetilde{\mathcal{S}}_k)_{\widetilde{x}_\delta}\right)\right)} \Big/_{\sim} \\ &= \bigoplus_{i\in\mathcal{I}_{\mathbf{G}}} \bigoplus_{\delta\in\Delta_i^k} \operatorname{Ind}_{\mathbf{G}_i}^{\mathbf{G}}\left(\operatorname{Ind}_{\mathbf{G}_\delta\cap\mathbf{G}_i}^{\mathbf{G}_i}\left((\widetilde{\mathcal{S}}_k)_{\widetilde{x}_\delta}\right)\right) \Big/_{\sim}\end{aligned}$$

Recall that the map $\underline{\Phi^k_{i,j}}$ defined in (13.46), which induces the equivalence relation $\sim$, is given by

$$\begin{aligned}\underline{\Phi^k_{i,j}} : (\widetilde{\mathcal{A}}_k)_{\widetilde{x}} = \bigoplus_{\sigma\in\mathbf{Conn}_i^k} (\widetilde{\mathcal{S}}_k)_{\widetilde{x}_\sigma} &\longrightarrow \bigoplus_{\tau\in\mathbf{Conn}_j^k} (\widetilde{\mathcal{S}}_k)_{\widetilde{x}_\tau} = (\widetilde{\mathcal{A}}_k)_{\widetilde{y}} \\ (\mathbf{m}_\sigma)_{\sigma\in\mathbf{Conn}_i^k} &\longmapsto (\mathbf{m}_\tau)_{\tau\in\mathbf{Conn}_j^k}\end{aligned}$$

for $\widetilde{x} \in \mathcal{X}_i$ and $\widetilde{y} \in \mathcal{X}_j$. As before, we can decompose this representation under the action of the stabilizer of the related central cell $\mathcal{X}_i$ or $\mathcal{X}_j$, respectively, and get a representation compatible with the representation of the module stated above. This representation differs for each valid pair $(i,j) \in \mathbf{val}_{\mathcal{P}}$. Note that some of the maps $\Psi^k_{i,k}$ can be collected and interpreted according to (13.52) as the map $\underline{\Psi^k_{i,k}}$, which commutes by (13.22) in some way[8]

[8]If we exchange the order of the maps, the indices will change.

with the $\underline{\Phi}^k_{i,j}$. Keeping this in mind it is not difficult to compute the equivalences for concrete examples, if needed. Thus, we skip the further details here.

We summarize:

Lemma 13.3.11: *Let $k \geq k_0$. Then we have that*

$$H^0(\mathcal{X}, \widetilde{\mathcal{A}}_k) = \bigoplus_{i \in \mathcal{I}_{\mathbf{G}}} \bigoplus_{\delta \in \Delta_i^k} \operatorname{Ind}_{\mathbf{G}_i}^{\mathbf{G}} \left(\operatorname{Ind}_{\mathbf{G}_\delta \cap \mathbf{G}_i}^{\mathbf{G}_i} \left((\widetilde{\mathcal{S}}_k)_{\widetilde{x}_\delta} \right) \right) \Big/ _{\sim},$$

where $\sim$ is the equivalence introduced in (13.10).

It remains to have a closer look on the stalks and global sections in the quotient, i.e. we have to compute the invariants of the occurring modules. We skip here the discussion of maps on the quotients induced by the maps $\Phi^k_{i,j}$ and $\underline{\Phi}^k_{i,j}$, since for practical things it is more convenient to write them down in the original space and use the compatibility condition between them and the $\Psi^k_{i,j}$ and $\underline{\Psi}^k_{i,j}$ from (13.22) to ensure that they behave in the right way under the projection to the quotient.

Let us now start with the stalks of the sheaf $\mathcal{A}_k$ for $k_0 \leq k \leq d$ in $x = \pi(\widetilde{x})$ for a fixed lift of $\widetilde{x} \in \mathcal{X}_i$ and $i \in \mathcal{I}_{\mathbf{G}}$. We know from Proposition 13.3.10 that

$$(\widetilde{\mathcal{A}}_k)_{\widetilde{x}} = \bigoplus_{\delta \in \Delta_i^k} \operatorname{Ind}_{\mathbf{G}_\delta \cap \mathbf{G}_i}^{\mathbf{G}_i} \left((\widetilde{\mathcal{S}}_k)_{\widetilde{x}_\delta} \right). \tag{13.58}$$

Therefore, we have by Lemma 13.2.6 b) that

$$(\mathcal{A}_k)_x = \left(\bigoplus_{\delta \in \Delta_i^k} \operatorname{Ind}_{\mathbf{G}_\delta \cap \mathbf{G}_i}^{\mathbf{G}_i} \left((\widetilde{\mathcal{S}}_k)_{\widetilde{x}_\delta} \right) \right)^{\mathbf{G}_{\widetilde{x}}}. \tag{13.59}$$

One knows by the usual properties of the induced representation

$$(\mathcal{A}_k)_x = \bigoplus_{\delta \in \Delta_i^k} \left((\widetilde{\mathcal{S}}_k)_{\widetilde{x}_\delta} \right)^{\mathbf{G}_{\widetilde{x}} \cap \mathbf{G}_\delta \cap \mathbf{G}_i} \tag{13.60}$$

holds. Since $\widetilde{x} \in \mathcal{X}_i$ and the action is cellular (see Section 9.1.2) we know that $\mathbf{G}_{\widetilde{x}} \subseteq \mathbf{G}_i$ and thus

$$(\mathcal{A}_k)_x = \bigoplus_{\delta \in \Delta_i^k} \left((\widetilde{\mathcal{S}}_k)_{\widetilde{x}_\delta} \right)^{\mathbf{G}_{\widetilde{x}} \cap \mathbf{G}_\delta}. \tag{13.61}$$

The change of the chosen lifts leads to an isomorphic description, where the isomorphism can be given in terms of the maps of type $\Psi^k_{i,\mathbf{g}}$. We can easily check with the properties a) – c) of them stated on page 161 together with (13.22) that one choice can be transferred to an other one. We do not distinguish between these isomorphic models.

Analogously to Lemma 13.3.11, we can deduce from this together with Lemma 13.2.6 a):

Lemma 13.3.12: *Let $k \geq k_0$. Then we have that*

$$H^0(\mathcal{X}, \mathcal{A}_k) = \bigoplus_{i \in \mathcal{I}_{\mathbf{G}}} \bigoplus_{\delta \in \Delta_i^k} \left((\widetilde{\mathcal{S}}_k)_{\widetilde{x}_\delta} \right)^{\mathbf{G}_{\widetilde{x}} \cap \mathbf{G}_\delta} \Big/_{\sim},$$

where $\sim$ is the equivalence introduced in (13.10).

Let $\widetilde{x} \in \mathcal{X}_i$ and $\mathbf{m} \in \left(\widetilde{\mathcal{S}}_k\right)_{\widetilde{x}}$, then by Lemma 13.3.4 we have

$$\left(\widetilde{j}_k\right)_{\widetilde{x}} = \bigoplus_{\sigma \in \mathbf{Conn}_i^k} \Phi_{\widetilde{x}, \widetilde{x}_\sigma}(\mathbf{m})$$

and rewrite this as above into

$$\begin{aligned} \left(\widetilde{j}_k\right)_{\widetilde{x}} &= \bigoplus_{\delta \in \Delta_i^k} \bigoplus_{\mathbf{G}_i / \mathbf{G}_i \cap \mathbf{G}_\delta} \Psi^k_{\widetilde{x}_\delta, \mathbf{g}} \Phi_{\widetilde{x}, \widetilde{x}_\delta}(\mathbf{m}) = \bigoplus_{\delta \in \Delta_i^k} \mathrm{Ind}_{\mathbf{G}_i \cap \mathbf{G}_\delta}^{\mathbf{G}_i} \left(\Phi_{\widetilde{x}, \widetilde{x}_\delta}\right)(\mathbf{m}) \\ &= \bigoplus_{\delta \in \Delta_i^k} \mathrm{Ind}_{\mathbf{G}_i \cap \mathbf{G}_\delta}^{\mathbf{G}_i} \left(\Phi_{\widetilde{x}, \widetilde{x}_\delta}\right)(\mathbf{m}) \end{aligned} \tag{13.62}$$

Therefore, we get

$$\begin{aligned} \left(\mathcal{S}_k\right)_x &= \left(\mathcal{A}_{k+1}\right)_x \Big/ \left(\widetilde{j}_k\right)_x \left(\left(\mathcal{S}_{k+1}\right)_x \right) = \left(\widetilde{\mathcal{A}}_{k+1}\right)_{\widetilde{x}}^{\mathbf{G}_{\widetilde{x}}} \Big/ \left(\left(\widetilde{j}_k\right)_{\widetilde{x}} \left(\left(\widetilde{\mathcal{S}}_{k+1}\right)_{\widetilde{x}} \right) \right)^{\mathbf{G}_{\widetilde{x}}} \\ &= \bigoplus_{\delta \in \Delta_i^{k+1}} \left(\left(\widetilde{\mathcal{S}}_{k+1}\right)_{\widetilde{x}_\delta} \right)^{\mathbf{G}_{\widetilde{x}} \cap \mathbf{G}_\delta} \Big/ \bigoplus_{\delta \in \Delta_i^{k+1}} \left(\Phi_{\widetilde{x}, \widetilde{x}_\delta} \left(\left(\widetilde{\mathcal{S}}_{k+1}\right)_{\widetilde{x}} \right) \right)^{\mathbf{G}_{\widetilde{x}} \cap \mathbf{G}_\delta}. \end{aligned} \tag{13.63}$$

Therefore, we have a more handy description of the terms in the resolution Theorem 13.3.9, which allows us to compute them, since we are able to compute the necessary modules, groups, and invariants (see Chapters 10 and 11). In the following section we discuss more practical questions of the computation.

13.3.3. Computing the Terms with the SNF

In this section we give a more computational description of the stalks and some of the maps introduced in Section 13.3.1 and 13.3.2. Computational means that we write them down in terms of some chosen bases and matrices in these bases. The main tool here is the so called Smith-normal-form [Smi61], which can also be used to compute the (co-)homology from a sequence of (co-)boundary maps. The computations described in this section are quite similar but much simpler than the computation in case of (co-)homology since we have only a dependency on a single map. In case of the computation of (co-)homology groups one has to compute as well the image of one as the kernel of an other map simultaneously in – in the best case – one common basis. We give some more information and references related to the computation of cohomology groups in Section 13.3.4.

In this section we identify the maps on stalks with their matrix representation in suitable bases.

Let us start here with the description of the building blocks we need to describe the cohomology. One gets via a simple computation from Lemma 13.3.4 and the following description of the compatibility maps:

Lemma 13.3.13: *Let the notions be as in Lemma 13.3.4, $r_{d,\widetilde{x}} =: r_d$ be the rank of $\left(\mathcal{S}_d\right)_{\widetilde{x}}$ over $\mathbb{Z}_{\mathbf{G}}$, and $c_d = \#\mathbf{Conn}_i^d$, then the morphism*

$$(\widetilde{j}_d)_{\widetilde{x}} = \bigoplus_{\sigma\in\mathbf{Conn}_i^d} \Phi_{\widetilde{x},\widetilde{x}_\sigma} = \bigoplus_{\sigma\in\mathbf{Conn}_i^d} \mathrm{Id}_{r_d}$$

has Smith-normal-form (SNF)

$$\mathrm{SNF}\Big((\widetilde{j}_d)_{\widetilde{x}}\Big) = \begin{pmatrix} \mathrm{Id}_{r_d} \\ 0_{r_d(c_d-1),r_d} \end{pmatrix},$$

where $0_{m,n}$ is the $m\times n$ zero-matrix. The SNF can be obtained using the following transformation:

$$\mathrm{SNF}\Big((\widetilde{j}_d)_{\widetilde{x}}\Big) = \underbrace{\begin{pmatrix} \mathrm{Id}_{r_d} & & & \\ -\mathrm{Id}_{r_d} & \mathrm{Id}_{r_d} & & \\ & \ddots & \ddots & \\ & & -\mathrm{Id}_{r_d} & \mathrm{Id}_{r_d} \end{pmatrix}}_{=:\mathbf{T}_{i,d}^{SNF}} \cdot (\widetilde{j}_d)_{\widetilde{x}}.$$

One easily checks that

$$\left(\mathbf{T}_{i,d}^{\mathrm{SNF}}\right)^{-1} = \begin{pmatrix} \mathrm{Id}_{r_d} & & \\ \vdots & \ddots & \\ \mathrm{Id}_{r_d} & \cdots & \mathrm{Id}_{r_d} \end{pmatrix}. \tag{13.64}$$

Corollary 13.3.14: *Let $r_d = \mathrm{rank}_{\mathbb{Z}_{\mathbf{G}}}\big(\widetilde{\mathcal{S}}_d\big)_{\widetilde{x}}$, $\underline{r_d} = \mathrm{rank}_{\mathbb{Z}_{\mathbf{G}}}\big(\widetilde{\mathcal{A}}_d\big)_{\widetilde{x}}$ and $c_d = \#\mathbf{Conn}_i^d$. Further, we use the notions from Lemma 13.3.13.*

Let $\{\mathbf{e}_{j,\widetilde{x}}\}_{j=1,\dots,r_d}$ be the basis of $\big(\widetilde{\mathcal{S}}_d\big)_{\widetilde{x}}$ and let $\{\mathbf{f}_{j,\widetilde{x}}\}_{j=1,\dots,\underline{r_d}}$ be the basis of $\big(\widetilde{\mathcal{A}}_d\big)_{\widetilde{x}}$ (each as as $\mathbb{Z}_{\mathbf{G}}$-module). Let further $\{\mathbf{f}'_{j,\widetilde{x}}\}_{j=1,\dots,\underline{r_d}}$ be the basis of $\big(\widetilde{\mathcal{A}}_d\big)_{\widetilde{x}}$ such that

$$\begin{tikzcd}
 & \Big(\big(\widetilde{\mathcal{A}}_d\big)_{\widetilde{x}}, \{\mathbf{f}_{j,\widetilde{x}}\}_{j=1,\dots,\underline{r_d}}\Big) \arrow[dd, "\mathbf{T}_{i,d}^{SNF}\cdot -"] \\
\Big(\big(\widetilde{\mathcal{S}}_d\big)_{\widetilde{x}}, \{\mathbf{e}_{j,\widetilde{x}}\}_{j=1,\dots,r_d}\Big) \arrow[ur, "(\widetilde{j}_d)_{\widetilde{x}}\cdot -"] \arrow[dr, "\mathrm{SNF}\left(\widetilde{j}_d\right)\cdot -"'] & \\
 & \Big(\big(\widetilde{\mathcal{A}}_d\big)_{\widetilde{x}}, \{\mathbf{f}'_{j,\widetilde{x}}\}_{j=1,\dots,\underline{r_d}}\Big)
\end{tikzcd}$$

holds. Note here are the maps represented by their matrix representation in the bases indicated in the diagram.

If $\mathbb{M}$ is a free module of rank c_d over $\mathbb{Z}_{\mathbf{G}}$, then the stalks of $\widetilde{\mathcal{S}}_{d-1}$ are in each point $\widetilde{x} \in \mathcal{X}_i \subset \mathcal{X}$ a free module of rank $\underline{r_d} - r_d = r_d(c_d - 1)$ over $\mathbb{Z}_{\mathbf{G}}$. More precise we have:

$$\big(\widetilde{\mathcal{S}}_{d-1}\big)_{\widetilde{x}} = \bigoplus_{j=r_d+1}^{\underline{r_d}} \mathbb{Z}_{\mathbf{G}} \cdot \mathbf{f}_{j,\widetilde{x}} \cong \mathbb{Z}_{\mathbf{G}}^{r_d(c_d-1)}.$$

Proof. According to Lemma 13.3.4 the stalks $\widetilde{\mathcal{S}}_{d-1}$ in $\widetilde{x} \in \mathcal{X}_i$ are given by

$$\big(\widetilde{\mathcal{S}}_{d-1}\big)_{\widetilde{x}} = \big(\widetilde{\mathcal{A}}_d\big)_{\widetilde{x}} \Big/ \big(\widetilde{j}_d\big)_{\widetilde{x}} \cdot \Big(\big(\widetilde{\mathcal{S}}_d\big)_{\widetilde{x}}\Big).$$

Therefore, we get using the elementary divisor theorem

$$\begin{aligned}
\big(\widetilde{\mathcal{S}}_{d-1}\big)_{\widetilde{x}} &= \big(\widetilde{\mathcal{A}}_d\big)_{\widetilde{x}} \Big/ \big(\widetilde{j}_d\big)_{\widetilde{x}} \cdot \Big(\big(\widetilde{\mathcal{S}}_d\big)_{\widetilde{x}}\Big) \\
&= \bigoplus_{j=1}^{\underline{r_d}} \mathbb{Z}_{\mathbf{G}} \cdot \mathbf{f}_{j,\widetilde{x}} \Big/ \big(\widetilde{j}_d\big)_{\widetilde{x}} \cdot \left(\bigoplus_{j=1}^{r_d} \mathbb{Z}_{\mathbf{G}} \cdot \mathbf{e}_{j,\widetilde{x}}\right) \\
&= \bigoplus_{j=1}^{\underline{r_d}} \mathbb{Z}_{\mathbf{G}} \cdot \mathbf{f}_{j,\widetilde{x}} \Big/ \bigoplus_{j=1}^{r_d} \mathbb{Z}_{\mathbf{G}} \cdot \big(\widetilde{j}_d\big)_{\widetilde{x}} \cdot \mathbf{e}_{j,\widetilde{x}} \\
&= \bigoplus_{j=1}^{\underline{r_d}} \mathbb{Z}_{\mathbf{G}} \cdot \mathbf{f}_{j,\widetilde{x}} \Big/ \bigoplus_{j=1}^{r_d} \mathbb{Z}_{\mathbf{G}} \cdot \big(\mathbf{T}_{i,d}^{\mathrm{SNF}}\big)^{-1} \cdot \mathrm{SNF}\Big(\big(\widetilde{j}_d\big)_{\widetilde{x}}\Big) \cdot \mathbf{e}_{j,\widetilde{x}}.
\end{aligned}$$

By Lemma 13.3.13 we get

$$\big(\widetilde{\mathcal{S}}_{d-1}\big)_{\widetilde{x}} = \bigoplus_{j=1}^{\underline{r_d}} \mathbb{Z}_{\mathbf{G}} \cdot \mathbf{f}_{j,\widetilde{x}} \Big/ \bigoplus_{j=1}^{r_d} \mathbb{Z}_{\mathbf{G}} \cdot \underbrace{\big(\mathbf{T}_{i,d}^{\mathrm{SNF}}\big)^{-1} \cdot \mathbf{f}'_{j,\widetilde{x}}}_{=\mathbf{f}_{j,\widetilde{x}}} = \bigoplus_{j=r_d+1}^{\underline{r_d}} \mathbb{Z}_{\mathbf{G}} \cdot \mathbf{f}_{j,\widetilde{x}}.$$

By construction we have $\underline{r_d} = r_d c_d$ and the remaining parts of the corollary follow immediately. □

Let

$$k_0 = \max(k_{\mathrm{inj}}, 1). \tag{13.65}$$

Let for $k \geq k_0 - 1$

$$r_{k,\widetilde{x}} = \operatorname{rank}_{\mathbb{Z}_{\mathbf{G}}} \big(\widetilde{\mathcal{S}}_k\big)_{\widetilde{x}} \qquad\qquad \underline{r_{k,\widetilde{x}}} = \operatorname{rank}_{\mathbb{Z}_{\mathbf{G}}} \big(\widetilde{\mathcal{A}}_k\big)_{\widetilde{x}}. \tag{13.66}$$

Then there are bases $\{\mathbf{e}^k_{j,\widetilde{x}} : j = 1, \dots, r_{k,\widetilde{x}}\}$ of $\big(\widetilde{\mathcal{S}}_k\big)_{\widetilde{x}}$ and $\{\mathbf{f}^k_{j,\widetilde{x}} : j = 1, \dots, \underline{r_{k,\widetilde{x}}}\}$ of $\big(\widetilde{\mathcal{A}}_k\big)_{\widetilde{x}}$ such that that $\widetilde{j}_k$ has the SNF. In general the SNF has the following form

$$\mathrm{SNF}\Big(\big(\widetilde{j}_k\big)_{\widetilde{x}}\Big) = \left(\begin{array}{ccc|c|c} \lambda^{(1)}_{k,\widetilde{x}} & & & & \\ & \ddots & & & \\ & & \lambda^{(k_{\mathrm{tor},\widetilde{x}})}_{k,\widetilde{x}} & & \\ \hline & & & \mathrm{Id}_{k_{\mathrm{null},\widetilde{x}}} & \\ \hline \multicolumn{4}{c|}{0_{k_{\mathrm{free},\widetilde{x}},k_{\mathrm{null},\widetilde{x}}+k_{\mathrm{tor},\widetilde{x}}}} & 0_{k_{\mathrm{free},\widetilde{x}},r_{k,\widetilde{x}}-k_{\mathrm{tor},\widetilde{x}}-k_{\mathrm{null},\widetilde{x}}} \end{array}\right), \tag{13.67}$$

which is a (not necessary square) block diagonal matrix, where $\lambda^{(l+1)}_{k,\widetilde{x}} | \lambda^{(l)}_{k,\widetilde{x}}$. Then

$$\operatorname{rank}_{\mathbb{Z}_{\mathbf{G}}} \widetilde{j}_d = k_{\mathrm{tor},\widetilde{x}} + k_{\mathrm{null},\widetilde{x}}. \tag{13.68}$$

The basis elements corresponding to the upper left block span the torsion part of $(\widetilde{\mathcal{S}}_{k-1})_{\widetilde{x}}$ whereas the elements corresponding to the lower right block span the free part of $(\widetilde{\mathcal{S}}_{k-1})_{\widetilde{x}}$ interpreted as submodule of $(\widetilde{\mathcal{A}}_k)_{\widetilde{x}}$. A priori there is no reason that the torsion part is zero but since our modules are defined over $\mathbb{Z}_{\mathbf{G}}$ we might expect that in most cases (depending on the the module $\mathbb{M}$) it vanishes.

Analogously to the discussion of the case $k = d$ in Corollary 13.3.14 we can deduce from the SNF of $\widetilde{j}_k$ for $k \geq k_0$ a representation of $(\widetilde{\mathcal{S}}_{k-1})_{\widetilde{x}}$.

Theorem 13.3.15: *Let $k \geq k_0$ and*

$$\begin{tikzcd} \Big((\widetilde{\mathcal{S}}_k)_{\widetilde{x}}, \{\mathbf{e}^k_{j,\widetilde{x}}\}_{j=1,\dots,r_{k,\widetilde{x}}}\Big) \arrow[r, "(\widetilde{j}_d)_{\widetilde{x}}"] \arrow[d, "\mathbf{S}^{SNF}_{i,k,\widetilde{x}} \cdot -"'] & \Big((\widetilde{\mathcal{A}}_k)_{\widetilde{x}}, \{\mathbf{f}^k_{j,\widetilde{x}}\}_{j=1,\dots,\underline{r_{k,\widetilde{x}}}}\Big) \arrow[d, "\mathbf{T}^{SNF}_{i,k,\widetilde{x}} \cdot -"] \\ \Big((\widetilde{\mathcal{S}}_k)_{\widetilde{x}}, \{\mathbf{e'}^k_{j,\widetilde{x}}\}_{j=1,\dots,r_{k,\widetilde{x}}}\Big) \arrow[r, "\mathrm{SNF}\left((\widetilde{j}_d)_{\widetilde{x}}\right) \cdot -"'] & \Big((\widetilde{\mathcal{A}}_k)_{\widetilde{x}}, \{\mathbf{f'}^k_{j,\widetilde{x}}\}_{j=1,\dots,\underline{r_{k,\widetilde{x}}}}\Big), \end{tikzcd}$$

where $\mathbf{S}^{SNF}_{i,k,\widetilde{x}} \cdot \mathbf{e}^k_{j,\widetilde{x}} = \mathbf{e'}^k_{j,\widetilde{x}}$ and $\mathbf{T}^{SNF}_{i,k,\widetilde{x}} \cdot \mathbf{f}^k_{j,\widetilde{x}} = \mathbf{f'}^k_{j,\widetilde{x}}$ are mapping the bases from the upper row onto the corresponding bases in the bottom row.

Then we have that

$$\Big(\widetilde{\mathcal{S}}_{k-1}\Big)_{\widetilde{x}} = \underbrace{\bigoplus_{j=1}^{k_{tor,\widetilde{x}}} \Big(\mathbb{Z}_{\mathbf{G}} / \lambda^{(j)}_{k,\widetilde{x}} \cdot \mathbb{Z}_{\mathbf{G}}\Big) \cdot \mathbf{f}^k_{j,\widetilde{x}}}_{\textit{torsion part}} \oplus \underbrace{\bigoplus_{j=k_{null,\widetilde{x}}+1}^{\underline{r_{k,\widetilde{x}}}} \mathbb{Z}_{\mathbf{G}} \cdot \mathbf{f}^k_{j,\widetilde{x}}}_{\textit{free part}}.$$

Proof. By Lemma 13.3.4 we have that

$$(\widetilde{\mathcal{S}}_{k-1})_{\widetilde{x}} = (\widetilde{\mathcal{A}}_k)_{\widetilde{x}} \Big/ (\widetilde{j}_k)_{\widetilde{x}} \Big((\widetilde{\mathcal{S}}_k)_{\widetilde{x}}\Big).$$

As before in the proof of Corollary 13.3.14, we can rewrite this as

$$\begin{aligned}
\left(\widetilde{\mathcal{S}}_{k-1}\right)_{\widetilde{x}} &= \bigoplus_{j=1}^{r_{k,\widetilde{x}}} \mathbb{Z}_{\mathbf{G}} \cdot \mathbf{f}^k_{j,\widetilde{x}} \Big/ \left(\widetilde{\jmath}_k\right)_{\widetilde{x}} \left(\bigoplus_{j=1}^{r_{k,\widetilde{x}}} \mathbb{Z}_{\mathbf{G}} \cdot \mathbf{e}^k_{j,\widetilde{x}} \right) \\
&= \bigoplus_{j=1}^{r_{k,\widetilde{x}}} \mathbb{Z}_{\mathbf{G}} \cdot \mathbf{f}^k_{j,\widetilde{x}} \Big/ \bigoplus_{j=1}^{r_{k,\widetilde{x}}} \mathbb{Z}_{\mathbf{G}} \cdot \left(\widetilde{\jmath}_k\right)_{\widetilde{x}} \cdot \mathbf{e}^k_{j,\widetilde{x}} \\
&= \bigoplus_{j=1}^{r_{k,\widetilde{x}}} \mathbb{Z}_{\mathbf{G}} \cdot \mathbf{f}^k_{j,\widetilde{x}} \Big/ \bigoplus_{j=1}^{r_{k,\widetilde{x}}} \mathbb{Z}_{\mathbf{G}} \cdot \left(\mathbf{T}^{\mathrm{SNF}}_{i,k,\widetilde{x}}\right)^{-1} \cdot \mathrm{SNF}\left(\left(\widetilde{\jmath}_d\right)_{\widetilde{x}}\right) \cdot \mathbf{S}^{\mathrm{SNF}}_{i,k,\widetilde{x}} \cdot \mathbf{e}^k_{j,\widetilde{x}} \\
&= \bigoplus_{j=1}^{r_{k,\widetilde{x}}} \mathbb{Z}_{\mathbf{G}} \cdot \mathbf{f}^k_{j,\widetilde{x}} \Big/ \bigoplus_{j=1}^{r_{k,\widetilde{x}}} \mathbb{Z}_{\mathbf{G}} \cdot \left(\mathbf{T}^{\mathrm{SNF}}_{i,k,\widetilde{x}}\right)^{-1} \cdot \mathrm{SNF}\left(\left(\widetilde{\jmath}_d\right)_{\widetilde{x}}\right) \cdot \mathbf{e}'^k_{j,\widetilde{x}} \\
&= \bigoplus_{j=1}^{r_{k,\widetilde{x}}} \mathbb{Z}_{\mathbf{G}} \cdot \mathbf{f}^k_{j,\widetilde{x}} \Big/ \Bigg[\bigoplus_{j=1}^{k_{\mathrm{tor},\widetilde{x}}} \left(\mathbf{T}^{\mathrm{SNF}}_{i,k,\widetilde{x}}\right)^{-1} \cdot \left(\lambda^{(j)}_{k,\widetilde{x}} \cdot \mathbb{Z}_{\mathbf{G}} \cdot \mathbf{f}'^k_{j,\widetilde{x}}\right) \oplus \bigoplus_{j=k_{\mathrm{tor},\widetilde{x}}+1}^{k_{\mathrm{tor},\widetilde{x}}+k_{\mathrm{null},\widetilde{x}}} \left(\mathbf{T}^{\mathrm{SNF}}_{i,k,\widetilde{x}}\right)^{-1} \cdot \left(\mathbb{Z}_{\mathbf{G}} \cdot \mathbf{f}'^k_{j,\widetilde{x}}\right) \\
&\qquad \oplus \bigoplus_{j=k_{\mathrm{tor},\widetilde{x}}+k_{\mathrm{null},\widetilde{x}}+1}^{r_{k,\widetilde{x}}} \left(\mathbf{T}^{\mathrm{SNF}}_{i,k,\widetilde{x}}\right)^{-1} \cdot \left(0 \cdot \mathbf{f}'^k_{j,\widetilde{x}}\right) \Bigg] \\
&= \bigoplus_{j=1}^{r_{k,\widetilde{x}}} \mathbb{Z}_{\mathbf{G}} \cdot \mathbf{f}^k_{j,\widetilde{x}} \Big/ \left[\bigoplus_{j=1}^{k_{\mathrm{tor},\widetilde{x}}} \lambda^{(j)}_{k,\widetilde{x}} \cdot \mathbb{Z}_{\mathbf{G}} \cdot \underbrace{\left(\mathbf{T}^{\mathrm{SNF}}_{i,k,\widetilde{x}}\right)^{-1} \cdot \mathbf{f}'^k_{j,\widetilde{x}}}_{=\mathbf{f}^k_{j,\widetilde{x}}} \oplus \bigoplus_{j=k_{\mathrm{tor},\widetilde{x}}+1}^{k_{\mathrm{tor},\widetilde{x}}+k_{\mathrm{null},\widetilde{x}}} \mathbb{Z}_{\mathbf{G}} \cdot \underbrace{\left(\mathbf{T}^{\mathrm{SNF}}_{i,k,\widetilde{x}}\right)^{-1} \cdot \mathbf{f}'^k_{j,\widetilde{x}}}_{=\mathbf{f}^k_{j,\widetilde{x}}} \right] \\
&= \bigoplus_{j=1}^{k_{\mathrm{tor},\widetilde{x}}} \left(\mathbb{Z}_{\mathbf{G}} / \lambda^{(j)}_{k,\widetilde{x}} \cdot \mathbb{Z}_{\mathbf{G}}\right) \cdot \mathbf{f}^k_{j,\widetilde{x}} \oplus \bigoplus_{j=k_{\mathrm{tor},\widetilde{x}}+k_{\mathrm{null},\widetilde{x}}+1}^{r_{k,\widetilde{x}}} \mathbb{Z}_{\mathbf{G}} \cdot \mathbf{f}^k_{j,\widetilde{x}}. \qquad \square
\end{aligned}$$

We know now how to write down the quotients explicitly. To complete the picture we need to write down also the maps $\Phi^k_{i,j}$. In general one obtains them as described via (13.46) and (13.48). This is not that difficult on one hand but rather technical and and not very useful in this abstract setting, therefore we skip their description in this case at this place.

13.3.4. Computing Cohomology from a Complex

In this section we collect some references and give an overview on the question how to obtain the cohomology from a collection of cochain maps, which we have due to Theorem 13.3.9.

There are several examples of algorithms which compute generators for a representation of the (co-)homology from a (co-)chain complex of a simplicial or cubical complex using the so called Smith normal form (SNF) (see e.g. [KMM04; Chap. 3,4 and 7], [DHSW03, DSV00, DSV01, PAFL06], [Cai61; Chap. 5-7]). From an algorithmic point of view it is the same to compute homology and cohomology, up to a change in the indices of the (co-)boundary maps. Some of the algorithms are able to compute the whole (co-)homology, some only the reduced part, i.e. the cohomology modulo torsion. There has been some progress on this topic during the last years since cohomology (over $\mathbb{F}_{2^r}$) is used in some image processing and image recognition algorithms (see e.g. [KMM04; Chap. 8] or [PAFL06]).

In particular, the results of Jean-Guillaume Dumas et al. [DHSW03, DSV00, DSV01] and Samuel Peltier et al. [PAFL06] are a good reference for practical computations. The former results were also used for an implementation in GAP in case of complexes coming from a simplicial complex. For us it is not important where the complex comes from since in any case the following procedure is nearly identical.

The general strategy for this task can be formulated as follows:

a) Compute a cochain complex $(C^\bullet, d^\bullet)$ (comp. Theorem 13.3.9).

b) Compute the Smith-normal-form $\mathrm{SNF}(d^k)$ (see e.g. [Smi61]) to all the coboundary maps d^k together with base change maps

$$(\mathbf{T}^k)^{-1} \cdot \mathrm{SNF}(d^k) \cdot \mathbf{S}^k = d^k,$$

where

$$\mathrm{SNF}(d^k) = \left(\begin{array}{ccc|c|c} \lambda_k^{(1)} & & & & \\ & \ddots & & & \\ & & \lambda_k^{(k_{\mathrm{tor}})} & & \\ \hline & & & \mathrm{Id}_{k_{\mathrm{free}}} & \\ \hline \multicolumn{4}{c|}{0_{k^1_{\mathrm{null}},k_{\mathrm{tor}}+k_{\mathrm{free}}}} & 0_{k^1_{\mathrm{null}},k^2_{\mathrm{null}}} \end{array}\right).$$

c) Identify the kernel and image of the coboundary maps in SNF.

d) Compute the quotient

$$H^k(C^\bullet, d^\bullet) = \mathbf{ker}\left(d^k\right) \Big/ \mathbf{im}\left(d^{k-1}\right).$$

The last step is trivial, if it is possible to Smith normalize the maps d^{k-1} and d^k simultaneously, since one can easily read off the base elements corresponding to image and kernel. Usually that is not possible, but there are several modifications, such as a modified Smith-normal-from (mSNF), which has some better properties in this context (see e.g. [DSV01, DHSW03, PAFL06]). The authors also give therein some useful hints how to speed up or even avoid computations and analyse the runtime in detail. For our context we have to be careful with the runtime analysis stated there, since they consider only cases where the coboundary maps are sparse, which is plausible in most practical examples coming from usual simplicial complexes. In our cases it seems quite optimistic to hope that the occurring matrices are sparse. The additional group action makes the things more complicated.

We did some tests to check the program which is able to compute the building blocks (comp. Section 11.2) which occur in our construction in this chapter. Further, we know by the results in Chapter 10 enough on the structure of cells and their relations among each other that we expect a kind of block triangular structure of dense subblocks, since on one side e.g. both 0-cells occur in almost all higher dimensional cells in the boundary, whereas 2-cells occur only in the boundary of 3- and 4-cells and on the other side the neighbours behave in the opposite way. In our construction both play their role because we have to sum over representatives of equivalence classes of cells in the k-skeletons and for each of them over the surrounding cells. The non-zero blocks in this construction seem to be neither sparse nor very dense (comp. e.g. Table 11.1 and Figure 11.3), if we assume that the few computed cases of low weights, i.e. small rank of the module, are typical examples. However there are some maps which consist only of blocks equal to the identity matrix or the zero matrix and thus are sparse (e.g. $\widetilde{j}_d$ in (13.3) together with Lemma 13.3.13). As a consequence one should assume that we have to handle dense matrices and/or do some more tests to estimate the density and explore the real structure before doing the real computation of the SNFs or mSNFs, respectively.

We do not go more into the details because we found an alternative method (see Section 13.3.5) to compute the cohomology without doing the things sketched in this section.

13.3.5. The Second Resolution

The method which we state at the end of Section 13.3.1 has one major difficulty, namely that we need the injectivity of $\widetilde{j}_k$ at least for $0 \leq k \leq d$ to define the full acyclic resolutions $\widetilde{\mathcal{A}}_\bullet$ and $\mathcal{A}_\bullet$, which we need a priori to compute the cohomology groups we are looking for. So far the theory. In practice we know that also in that situation one need only some of them to obtain a particular cohomology group. In this section we give a step by step construction for the cohomology, which, if all maps were injective, will give a description for all cohomology groups and, if only all up to some level, it will give only of the groups up to some degree depending on how many maps we know to be injective. We will deduce this from the construction described above as kind of a refinement of that method, which is more handy for practical computations and avoids the explicit knowledge of the injectivity of $\widetilde{j}_k$ for all k. To do this we will construct some further sequences which can be used to compute the cohomology groups $H^q(\ldots)$ of degree q for $q = 1, ..., d+1-k_0$, where

$$k_0 = \max(k_{\text{inj}}, 1). \tag{13.69}$$

The case $q = 0$ is only the computation of invariants, which is not a problem at all. We obtain a method to get at least some cohomology groups of higher degrees.

To simplify the notation we use for a sheaf $\widetilde{\mathscr{F}}$ on $\mathcal{X}$

$$H^q\left(\widetilde{\mathscr{F}}\right) := H^q\left(\mathcal{X}, \widetilde{\mathscr{F}}\right) \qquad \text{and} \qquad H^q(\mathscr{F}) := H^q(\mathcal{X}/\mathbf{G}, \mathscr{F}) \tag{13.70}$$

for a sheaf $\mathscr{F}$ on $\mathcal{X}/\mathbf{G}$. We start with the two families of short exact sequences from (13.36). Each of these short exact sequences of sheaves defines by (5.4) a long exact sequences in cohomology. This leads to two huge diagrams displayed in Figure 13.4 on page 189 and 13.5 on page 190.

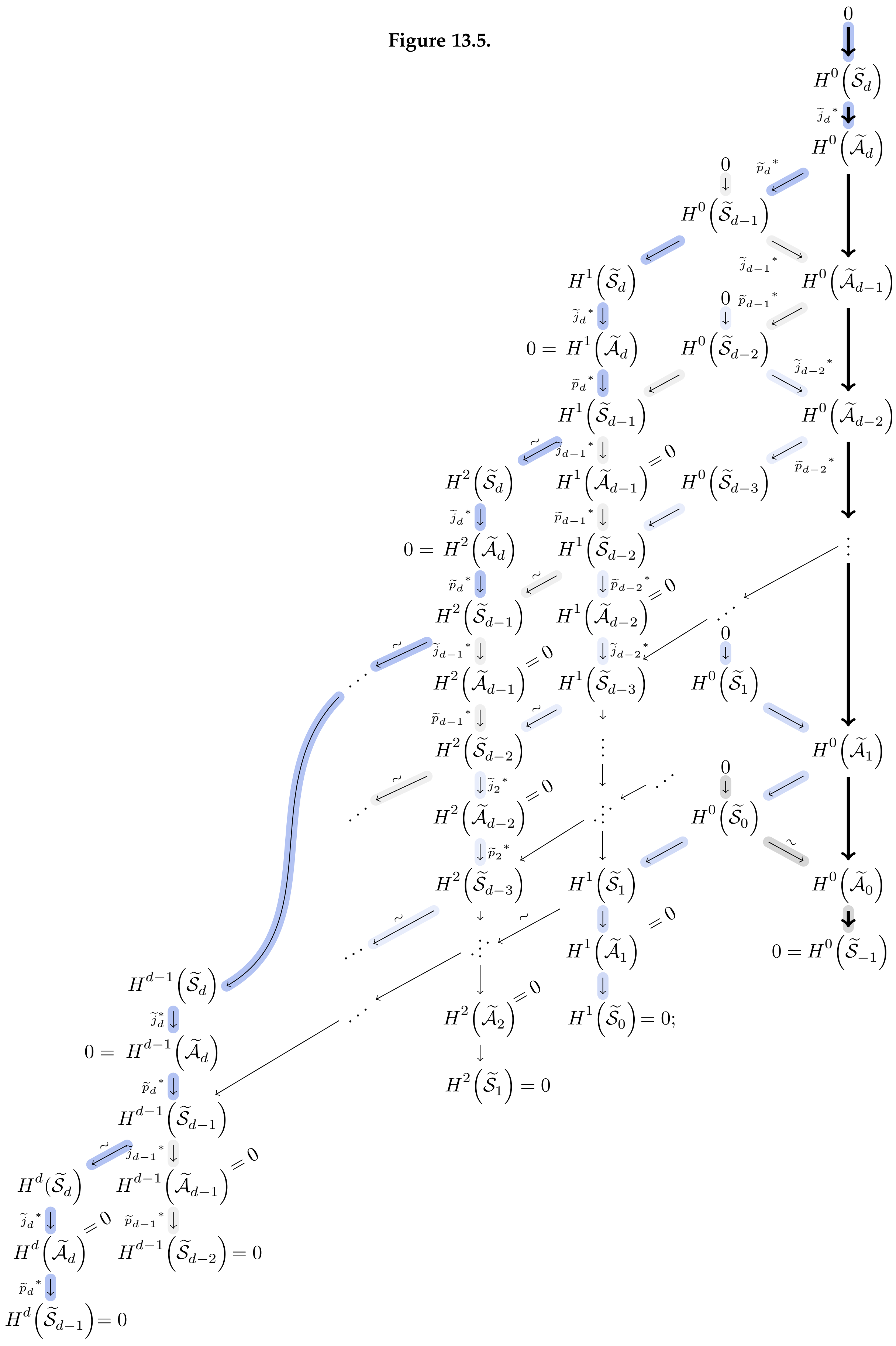
In these two diagrams the exact sequences in cohomology are indicated by different coloured shades of the maps, depending on the short exact sequence they are coming from. The entries in the diagrams are sorted such that cohomology groups of the same degree are placed in the same column. The rightmost column is the exact sequence of the acyclic resolution from Theorem 13.3.8.

By Theorem 13.3.7 we know that the sheaves $\widetilde{\mathcal{A}}_k$ and $\mathcal{A}_k$ are acyclic for $k \geq k_{\text{inj}}$. Therefore, the higher cohomology groups $H^q(\widetilde{\mathcal{A}}_k)$ and $H^q(\mathcal{A}_k)$ for $q > 0$ and $k \geq k_{\text{inj}}$ vanish. Furthermore,

Figure 13.4.

$$\begin{tikzcd}
&&&&&&&&&&&& 0 \arrow[d] \\
&&&&&&&&&&&& H^0(\mathcal{S}_d) \arrow[d, "j_d^*"'] \\
&&&&&&&&&& 0 \arrow[d] && H^0(\mathcal{A}_d) \arrow[dll, "p_d^*"'] \arrow[dd] \\
&&&&&&&&&& H^0(\mathcal{S}_{d-1}) \arrow[dll] \arrow[drr, "j_{d-1}^*"'] && \\
&&&&&&&& H^1(\mathcal{S}_d) \arrow[d, "j_d^*"'] && 0 \arrow[d] && H^0(\mathcal{A}_{d-1}) \arrow[dll, "p_{d-1}^*"'] \arrow[dd] \\
&&&&&&&& 0 = H^1(\mathcal{A}_d) \arrow[d, "p_d^*"'] && H^0(\mathcal{S}_{d-2}) \arrow[dll] \arrow[drr, "j_{d-2}^*"] && \\
&&&&&&&& H^1(\mathcal{S}_{d-1}) \arrow[dll, "\sim"'] \arrow[d, "j_{d-1}^*"'] &&&& H^0(\mathcal{A}_{d-2}) \arrow[dll, "p_{d-2}^*"] \arrow[dd] \\
&&&&&& H^2(\mathcal{S}_d) \arrow[d, "j_d^*"'] && H^1(\mathcal{A}_{d-1}) = 0 \arrow[d, "p_{d-1}^*"'] && H^0(\mathcal{S}_{d-3}) \arrow[dll] && \\
&&&&&& 0 = H^2(\mathcal{A}_d) \arrow[d, "p_d^*"'] && H^1(\mathcal{S}_{d-2}) \arrow[dll, "\sim"'] \arrow[d, "p_{d-2}^*"] &&&& \vdots \arrow[dll] \arrow[dddd] \\
&&&&&& H^2(\mathcal{S}_{d-1}) \arrow[ddll, "\sim"'] \arrow[dd, "j_{d-1}^*"'] && H^1(\mathcal{A}_{d-2}) = 0 \arrow[dd, "j_{d-2}^*"] && \iddots \arrow[ddll] && \\
&&&&&&&&&& 0 \arrow[d] && \\
&&&& \iddots \arrow[dddddll, equal, bend right] && H^2(\mathcal{A}_{d-1}) = 0 \arrow[d, "p_{d-1}^*"'] && H^1(\mathcal{S}_{d-3}) \arrow[dll, "\sim"'] \arrow[d] && H^0(\mathcal{S}_1) \arrow[drr] && \\
&&&&&& H^2(\mathcal{S}_{d-2}) \arrow[dll, "\sim"'] \arrow[d, "j_2^*"] && \vdots \arrow[d] & \iddots \arrow[dl] & 0 \arrow[d] && H^0(\mathcal{A}_1) \arrow[dll] \arrow[dd] \\
&&&& \iddots && H^2(\mathcal{A}_{d-2}) = 0 \arrow[d, "p_2^*"] && \iddots \arrow[dll] \arrow[d] && H^0(\mathcal{S}_0) \arrow[dll] \arrow[drr, "\sim"] && \\
&&&&&& H^2(\mathcal{S}_{d-3}) \arrow[dll, "\sim"'] \arrow[d] && H^1(\mathcal{S}_1) \arrow[dll, "\sim"'] \arrow[d] &&&& H^0(\mathcal{A}_0) \arrow[d] \\
&&&& \iddots && \iddots \arrow[ddll] \arrow[dd] && H^1(\mathcal{A}_1) = 0 \arrow[dd] &&&& 0 = H^0(\mathcal{S}_{-1}) \\
&& H^{d-1}(\mathcal{S}_d) \arrow[dd, "j_d^*"'] &&&&&&&&&& \\
&&&& \iddots \arrow[dddll] && H^2(\mathcal{A}_2) = 0 \arrow[dd] && H^1(\mathcal{S}_0) = 0 &&&& \\
&& 0 = H^{d-1}(\mathcal{A}_d) \arrow[dd, "p_d^*"'] &&&&&&&&&& \\
&&&&&& H^2(\mathcal{S}_1) = 0 &&&&&& \\
&& H^{d-1}(\mathcal{S}_{d-1}) \arrow[dll, "\sim"'] \arrow[d, "j_{d-1}^*"] &&&&&&&&&& \\
H^d(\mathcal{S}_d) \arrow[d, "j_d^*"'] && H^{d-1}(\mathcal{A}_{d-1}) = 0 \arrow[d, "p_{d-1}^*"'] &&&&&&&&&& \\
H^d(\mathcal{A}_d) = 0 \arrow[d, "p_d^*"'] && H^{d-1}(\mathcal{S}_{d-2}) = 0 &&&&&&&&&& \\
H^d(\mathcal{S}_{d-1}) = 0 &&&&&&&&&&&&
\end{tikzcd}$$

Figure 13.5.



we know by Corollary 13.3.6 that

$$H^q(\mathcal{S}_k) = H^q(\widetilde{\mathcal{S}}_k) = 0 \quad \text{for } q > k \geq k_{\text{inj}} \tag{13.71}$$

because $\mathcal{S}_k$, which is defined for $k \geq k_{\text{inj}}$, has only support in cells of dimension less or equal to k.

This has as an important consequence that we get by exactness in the huge diagrams shown in Figure 13.4 and 13.5 from above several isomorphisms. Thus, we get for the cohomology groups we are interested in the following identities for the sheaves on $\mathcal{X}$:

$$\begin{aligned} &H^2\Big(\widetilde{\mathcal{S}}_d\Big) \cong H^1\Big(\widetilde{\mathcal{S}}_{d-1}\Big) \\ &H^3\Big(\widetilde{\mathcal{S}}_d\Big) \cong H^2\Big(\widetilde{\mathcal{S}}_{d-1}\Big) \cong H^1\Big(\widetilde{\mathcal{S}}_{d-2}\Big) \\ &\qquad\vdots \\ &H^d\Big(\widetilde{\mathcal{S}}_d\Big) \cong H^{d-1}\Big(\widetilde{\mathcal{S}}_{d-1}\Big) \cong \ldots \cong H^{k_0+1}\Big(\widetilde{\mathcal{S}}_{k_0+1}\Big) \cong H^{k_0}\Big(\widetilde{\mathcal{S}}_{k_0}\Big). \end{aligned} \tag{13.72}$$

In the same way we get the analogous identities for the sheaves on $\mathcal{X}/\mathbf{G}$

$$\begin{aligned} &H^2\Big(\widetilde{\mathcal{S}}_d\Big) \cong H^1\Big(\widetilde{\mathcal{S}}_{d-1}\Big) \\ &H^3\Big(\widetilde{\mathcal{S}}_d\Big) \cong H^2\Big(\widetilde{\mathcal{S}}_{d-1}\Big) \cong H^1\Big(\widetilde{\mathcal{S}}_{d-2}\Big) \\ &\qquad\vdots \\ &H^d\Big(\widetilde{\mathcal{S}}_d\Big) \cong H^{d-1}\Big(\widetilde{\mathcal{S}}_{d-1}\Big) \cong \ldots \cong H^{k_0+1}\Big(\widetilde{\mathcal{S}}_{k_0+1}\Big) \cong H^{k_0}\Big(\widetilde{\mathcal{S}}_{k_0}\Big). \end{aligned} \tag{13.73}$$

This reduces the computation of the cohomology groups $H^q(\widetilde{\mathcal{S}}_d)$, or $H^q(\mathcal{S}_d)$ respectively, for for q in $0 \leq q \leq d+1-k_0$ to the computation of $H^1(\widetilde{\mathcal{S}}_k)$ or $H^1(\mathcal{S}_k)$ for $d \geq k \geq k_0$ and the computation of global sections, i.e. invariants of some modules under the group $\mathbf{G}$. However also $H^1(\widetilde{\mathcal{S}}_k)$ and $H^1(\mathcal{S}_k)$ can be computed by knowing only global sections, because we find in both the huge diagrams (see Figure 13.4 and 13.5) also for $d \geq k \geq k_0$ the following two families of exact sequences:

$$\begin{aligned} &0 \longrightarrow H^0\Big(\widetilde{\mathcal{S}}_k\Big) \overset{\widetilde{j}_k^*}{\longrightarrow} H^0\Big(\widetilde{\mathcal{A}}_k\Big) \overset{\widetilde{p}_k^*}{\longrightarrow} H^0\Big(\widetilde{\mathcal{S}}_{k-1}\Big) \longrightarrow H^1\Big(\widetilde{\mathcal{S}}_k\Big) \longrightarrow 0 \\ &0 \longrightarrow H^0\Big(\mathcal{S}_k\Big) \overset{j_k^*}{\longrightarrow} H^0\Big(\mathcal{A}_k\Big) \overset{p_k^*}{\longrightarrow} H^0\Big(\mathcal{S}_{k-1}\Big) \longrightarrow H^1\Big(\mathcal{S}_k\Big) \longrightarrow 0 \end{aligned} \tag{13.74}$$

We rewrite the isomorphism in (13.72) and (13.73) as

$$\begin{aligned} H^q\Big(\widetilde{\mathcal{S}}_d\Big) &\cong H^{q-k}\Big(\widetilde{\mathcal{S}}_{d-k}\Big) \cong H^1\Big(\widetilde{\mathcal{S}}_{d+1-q}\Big) \\ H^q\Big(\mathcal{S}_d\Big) &\cong H^{q-k}\Big(\mathcal{S}_{d-k}\Big) \cong H^1\Big(\mathcal{S}_{d+1-q}\Big) \end{aligned} \tag{13.75}$$

where $2 \leq q \leq d+1-k_0$ and $0 \leq k < q$. We can now replace $H^1(\widetilde{\mathcal{S}}_k)$ or $H^1(\mathcal{S}_k)$ in (13.74) by $H^q(\widetilde{\mathcal{S}}_d)$ or $H^q(\mathcal{S}_d)$, respectively, using (13.72) and (13.73) or equivalently (13.75). Therefore, we get from the exactness of (13.74) together with (13.72) and (13.73) immediately:

Theorem 13.3.16: *For $k = d, \dots, k_0$ we have*

$$H^{d+1-k}(\widetilde{\mathcal{S}}_d) = H^0(\widetilde{\mathcal{S}}_k)/\mathbf{im}\, \widetilde{p}_k^*\Big|_{H^0\left(\widetilde{\mathcal{A}}_k\right)} = \mathbf{coker}\left(\widetilde{p}_k^*\Big|_{H^0\left(\widetilde{\mathcal{A}}_k\right)}\right)$$

and

$$H^{d+1-k}(\mathcal{S}_d) = H^0(\mathcal{S}_k)/\mathbf{im}\, p_k^*\Big|_{H^0\left(\mathcal{A}_k\right)} = \mathbf{coker}\left(p_k^*\Big|_{H^0\left(\mathcal{A}_k\right)}\right).$$

It is an easy exercise in linear algebra to compute the cokernel of a map, if the map is explicitly known, e.g. as a matrix. Then the cokernel can be computed as the kernel of the transposed matrix and for that there are fast algorithms known. Thus, in the next step we need to get a description of the global sections and the connecting maps occurring in the previous theorem which is as explicit as possible.

13.4. Application to the Symplectic Group

In this section we indicate how the previously introduced constructions can be used to compute the cohomology of the symplectic group. There are two main tools, namely the orbicell decomposition, which we constructed in Section 9.2 and the constructible sheaves together with the two resolutions introduced in this chapter.

13.4.1. How to Apply the Resolutions

Let $\mathbb{M}$ be any given Γ-module $\mathbb{M} \in \mathbf{Obj}\,(\mathbf{Mod}_\Gamma)$ and $\mathbb{M}_\Gamma = \mathbb{M} \otimes_{\mathbb{Z}} \mathbb{Z}_\Gamma$ as before. We can use Proposition 12.1.2 to describe the connection between the cohomology of $\mathfrak{S}_2/\Gamma$ and the cohomology of the retraction $\mathbb{W}/\Gamma$ (see also (12.6) and (12.4)).

Theorem 13.4.1: $H^q(\mathfrak{S}_2/\Gamma, \mathcal{M}_\Gamma) \cong H^q(\mathbb{W}/\Gamma, \mathcal{M}_{\mathbb{W}})$ *where* $\mathcal{M}_\Gamma = \mathrm{sh}_\Gamma(\mathbb{M}_\Gamma)$ *and* $\mathcal{M}_{\mathbb{W}} = r_\Gamma^* \mathcal{M}_\Gamma$.

For a proof we refer to the proof of the analogous statement [Yas05; Theo.9.4.1]. We know by the Theorems 12.2.3 and 13.4.1 that

$$H^q(\Gamma, \mathbb{M}_\Gamma) \cong H^q(\mathfrak{S}_2/\Gamma, \mathcal{M}_\Gamma) \cong H^q(\mathbb{W}/\Gamma, \mathcal{M}_{\mathbb{W}})$$

Due to Theorem 9.2.11 we have two decompositions

$$\mathbb{W} = \bigsqcup_{X \in \mathfrak{W}} \mathscr{C}(X) \qquad\qquad \mathbb{W}/\Gamma = \bigsqcup_{X \in \Sigma} \mathscr{C}_\Gamma(X)$$

into disjoint components, where Σ the set of standard elements defined in Theorem 9.1.3 and with $\mathfrak{W} = \Gamma\Sigma$. If we apply one of the two methods stated in Theorems 13.3.9 and 13.3.16 to compute the cohomology $H^q(\Gamma, \mathbb{M}_\Gamma)$ of the symplectic group we have to ensure that the necessary conditions are fulfilled. This means that:

a) The retract $\mathbb{W}$ is a simply connected manifold. This holds because $\mathbb{W}$ is by Proposition 9.2.10 a strong deformation retract of the Siegel upper half-space, which has this property.

b) The retract $\mathbb{W}$ is locally finite (seeTheorem 9.1.10).

c) The retract $\mathbb{W}$ is a regular CW-complex (see Theorem 9.1.10).

d) The group Γ acts cellular on $\mathbb{W}$ (see Prop. 9.2.2).

e) The retract $\mathbb{W}$ has only finitely many orbits under Γ (see Prop. 9.2.3).

f) The quotient $\mathbb{W}/\Gamma$ has a finite orbicell decomposition (see Lemma 9.2.9).

g) The module $\mathbb{M}_\Gamma \in \mathbf{Obj}\left(\mathbf{Mod}_{\Gamma,\mathbb{Z}_\Gamma}\right)$ by assumption.

h) The decomposition of $\mathbb{W}$ is fine enough in the sense of the beginning of Section 13.1.1. One get this from the computed lists of cells (see Chapter 10), but one has to check this property at least once per type of cells. This can be done easily but not very fast because for each type one has to look for the cells in the closure of one chosen representative and then for the closures of their neighbours to ensure that there is no neighbourhood of a point which contains more then one connected component of the same cell. This are a lot of lists of cells which one has to compute and compare under each other.

Hence, we can apply the whole machinery of (orbi-)constructible sheaves. In this example we have the case that $d = 4$. We use in the following the notions from the last few sections on constructible sheaves in general. This means that we do not indicate here, as before, the underlying partition $\mathcal{P} = \mathfrak{W}$, since it does not change. To compute all maps and modules which occur in our construction we need to know for each cell, how to compute their boundary and to obtain their skeleton and the related embedding maps (see Sections 10.4 and 10.7.2). Since the support of the cells is concentrated in $\mathbf{S}_k(\mathbb{W})$ according to Lemma 13.3.3, we only need to compute the stalks in the non-zero cases. The computation of skeletons, closures, etc. was extensively discussed in Section 10.4. In addition we presented the results we need in this context in Sections 10.7.2. The related lists of cells are included on the CD-ROM attached to the printed copy of this thesis. This allows us to compute the embedding maps $\widetilde{i}_k$ for all $4 \geq k \geq 0$.

The modules $\mathbb{M}$ are highest weight modules and therefore in particular of finite rank over $\mathbb{Z}$. Hence, the modules $\mathbb{M}_\Gamma$ are of finite rank and therefore all w-constructible sheaves in the construction are even constructible (see Definition 13.1.1). As we already mentioned in Chapter 11, we are able to compute bases for highest weight modules and the action of group elements on the modules spanned by these basis elements. With standard SAGE and GAP functions we can also compute direct sums of highest weight modules. It turned out that not in any case the standard methods are the best choice since they are slow, in particular for large weights, and do not consider the additional group action we need. In practice it seems to be better to store direct sums as lists together with a special addition and group operation function. We need the direct sum operation essentially at two points during the construction:

- The computation of the stalks of the sheaves $\widetilde{\mathcal{A}}_k$ (see (13.39) for their definition and Proposition 13.3.10) involves a direct sum over a set of cells.
- To obtain the global sections (see e.g. Lemma 13.3.11 and 13.3.12) we need to sum with respect to the two families of compatibility maps the stalks over all types of cells.

This is the major missing part to get the original goal of this thesis. This part turned out to be much more complicated and time consuming than we thought at the beginning. There are several problems. We only stress the most important one.

There is no canonical way to describe the bases in terms of bases of the stalks. This means that if we fixed a basis of $\mathbb{M}_\Gamma$ we get a derived basis of any stalk of $\widetilde{\mathcal{A}}_4$ in a point $\widetilde{x} \in \mathbb{W}$, which are from an abstract point of view pairs $(\sigma, \mathbf{e}_i)$, where $\mathbf{e}_i$ is one base element from $\mathbb{M}_\Gamma$ and σ a cell in the collection of 4-cells lying around the cell, which contains $\widetilde{x}$.[9] We have that for a valid pair $(i, j) \in \mathbf{val}_\mathcal{P}$ the index sets over which we sum fulfil

$$\mathbf{Conn}_i^k \subseteq \mathbf{Conn}_j^k.$$

This is enough for abstract computations. In practice, we get the elements in both sets not necessarily in the same order, since there is no canonical order for the cells around another cell. Hence, one has to search for each cell in one list for the corresponding cell in the larger list to write down the compatibility maps according to (13.46). Furthermore, it is difficult to check whether two sets of representatives of cosets for the Δ_i^k are equal up to equivalence even if the corresponding sets $\mathbf{Conn}_i^k$ consist of the same elements in the same order, because the resulting set Δ_i^k depends also on the order of the elements in the group. All together this is no problem if we want only to compute a single stalk, but we have to compute the stalks for points in a bunch of cells and for all of them we need to known their compatibility maps, which depend on these sets. It is not impossible to do this, but makes the things much more complicated than only to put the object in a list, because we have to identify the identical elements occurring in the different sets, which can for example be done by a suitable indexing or the usage of a list of all known cells. Remember that due to the normalized representation of cells we are able to do this comparisons quite fast (comp Section 10.3.1).

Essentially this whole discussion is a question of fixing an orientation. Recall that an orientation is nothing else than to fix an order of cells around other cells. In this sense it is not really surprising that questions like this arise during the computation.

Let us assume for the moment that this problem is solved, then we can compute induced modules with group action and a suitable basis, using the function `quotientsystem`, which we implemented in SAGE to compute left cosets of two groups as list of representatives. It works fine for finite matrix groups and raises an exception otherwise. The algorithm is the most simple one can think of. One starts with the larger group, pick one element, computes the orbit in the larger group under the smaller group and removes all of the elements. Then one proceeds with the next not removed element an repeats the procedure until no element is left. Our function is faster than the build-in function in SAGE and works also for non-normal subgroups. In SAGE by default the output is a group object which is much more than we need.

In some cases we need to compute the intersection of matrix groups or, to be precise, of finite matrix groups, which occur as stabilizers of cells. This can be done with the function `intersectionofgroups` (see Section 10.7.5). In many cases the intersection is trivial, i.e. equal to $\{\pm \mathrm{Id}\}$ (see Table 10.12 and on the CD-ROM included in the printed copy of this thesis), which simplifies the computation.

All in all we are then able to compute all needed terms in the sequences, which can be used

[9] If the point $\widetilde{x}$ lies in a top dimensional cell, the collection consists only of that single cell.

to compute the cohomology according to Theorem 13.3.9 or 13.3.16. To apply these results in all its depth it remains to show:

Conjecture 1. Let $\mathbb{M}$ be a highest weight module and $\mathbb{M}_\Gamma := \mathbb{M} \otimes_{\mathbb{Z}} \mathbb{Z}_\Gamma$ as above. Let $\widetilde{j}_k$ be the map introduced in (13.36) with respect to $\mathbb{W}$ together with its cell decomposition, then $\widetilde{j}_k$ is injective for $k \leq 4$.

We have some indication that this should hold, but were not able to prove it in the abstract setting. However we believe that it should be possible to prove it via step by step construction of maps during the computation. Let us assume that Conjecture 1 holds. Then we can now have a closer look on the terms which occur in Theorems 13.3.9 and 13.3.16 respectively. Independent of Conjecture 1 we have by Corollary 13.3.5 that $k_{\text{inj}} \leq 4$. If Conjecture 1 holds, we know that the level of injectivity $k_{\text{inj}} = -\infty$ and thus $k_0 = 1$.

Hence, the acyclic resolution according to Theorem 5.1.4 exists and is in particular finite. Therefore, one can easily deduce the result of Raghunathan [Rag67, Rag68a, Rag68b] in case of the symplectic group.[10]

Theorem 13.4.2 (Raghunathan, 1967)**:** *If $\mathbf{R}$ is a commutative ring with identity and $\mathbb{M}$ a finitely generated $\mathbf{R}$-Γ-module, then the total cohomology*

$$\bigoplus_{q \geq 0} H^q\left(\mathbb{W}/\Gamma, \mathrm{sh}_\Gamma(\mathbb{M} \otimes_{\mathbf{R}} \mathbf{R}_\Gamma)\right)$$

is a finitely generated $\mathbf{R}_\Gamma$-module.

By this we know that the cohomology groups we want to compute are finitely generated abelian groups and have therefore by the classification theorem for finitely generated abelian groups (see e.g. [Mun84; §4]), the form

$$H^q(\Gamma, \mathbb{M} \otimes_{\mathbb{Z}} \mathbb{Z}_\Gamma) \cong \mathbb{Z}_\Gamma^b \oplus \left(\mathbb{Z}_\Gamma/a_1\mathbb{Z}_\Gamma \oplus \ldots \oplus \mathbb{Z}_\Gamma/a_k\mathbb{Z}_\Gamma\right), \tag{13.76}$$

where all a_i are prime powers and unique up to reordering. We discussed in Section 13.3.4 the question how to compute the cohomology in this form from the cochain complex induced by the acyclic resolution from Theorem 5.1.4.

For practical considerations it is better to use Theorem 13.3.16 rather than Theorem 13.3.9. Therefore, we rewrite the diagram in Figure 13.4 and get the diagram shown in Figure 13.6.

The rightmost column is the exact sequence of the acylic resolution from Theorem 13.3.8. Analogously we can specialize the Diagram in Figure 13.5 to our situation. According to (13.73) we get for the cohomology groups of the sheaves on the quotient:

$$\begin{aligned} H^2(\mathcal{M}_{\mathbb{W}}) &\cong H^1(\mathcal{S}_3) \\ H^3(\mathcal{M}_{\mathbb{W}}) &\cong H^2(\mathcal{S}_3) \cong H^1(\mathcal{S}_2) \\ H^4(\mathcal{M}_{\mathbb{W}}) &\cong H^3(\mathcal{S}_3) \cong H^2(\mathcal{S}_2) \cong H^1(\mathcal{S}_1). \end{aligned} \tag{13.77}$$

Recall that this leads to the following family of exact sequences

$$0 \to H^0(\mathcal{S}_k) \xrightarrow{j_k^*} H^0(\mathcal{A}_k) \xrightarrow{p_k^*} H^0(\mathcal{S}_{k-1}) \to H^1(\mathcal{S}_k) \to 0 \tag{13.78}$$

[10] Raghunathan proved this result in a more general context for arithmetic groups.

Figure 13.6.

$$\begin{tikzcd}
& & & & & 0 \arrow[d] \\
& & & & & H^0(\mathcal{M}_{\mathbb{W}}) \arrow[d, "j_4^*"'] \\
& & & & 0 \arrow[d] & H^0(\mathcal{A}_4) \arrow[dl, "p_4^*"'] \arrow[dd] \\
& & & & H^0(\mathcal{S}_3) \arrow[dl] \arrow[dr, "j_3^*"'] & \\
& & & H^1(\mathcal{M}_{\mathbb{W}}) \arrow[d, "j_4^*"'] & 0 \arrow[d] & H^0(\mathcal{A}_3) \arrow[dl, "p_3^*"'] \arrow[dd] \\
& & & 0 = H^1(\mathcal{A}_4) \arrow[d, "p_4^*"'] & H^0(\mathcal{S}_2) \arrow[dl] \arrow[dr, "j_2^*"] & \\
& & & H^1(\mathcal{S}_3) \arrow[dl, "\sim"'] \arrow[d, "j_3^*"'] & 0 \arrow[d] & H^0(\mathcal{A}_2) \arrow[dl, "p_2^*"] \arrow[dd] \\
& & H^2(\mathcal{M}_{\mathbb{W}}) \arrow[d, "j_4^*"'] & H^1(\mathcal{A}_3) = 0 \arrow[d, "p_3^*"'] & H^0(\mathcal{S}_1) \arrow[dl] \arrow[dr] & \\
& & 0 = H^2(\mathcal{A}_4) \arrow[d, "p_4^*"'] & H^1(\mathcal{S}_2) \arrow[dl, "\sim"'] \arrow[d, "p_2^*"] & 0 \arrow[d] & H^0(\mathcal{A}_1) \arrow[dl] \arrow[dd] \\
& & H^2(\mathcal{S}_3) \arrow[dl, "\sim"'] \arrow[d, "j_3^*"'] & H^1(\mathcal{A}_2) = 0 \arrow[d, "j_2^*"] & H^0(\mathcal{S}_0) \arrow[dl] \arrow[dr, "\sim"] & \\
& H^3(\mathcal{M}_{\mathbb{W}}) \arrow[d, "j_4^*"'] & H^2(\mathcal{A}_3) = 0 \arrow[d, "p_3^*"'] & H^1(\mathcal{S}_1) \arrow[dl, "\sim"'] \arrow[d] & & H^0(\mathcal{A}_0) \arrow[d] \\
& 0 = H^3(\mathcal{A}_4) \arrow[d, "p_4^*"'] & H^2(\mathcal{S}_2) \arrow[dl, "\sim"'] \arrow[d, "j_2^*"] & H^1(\mathcal{A}_1) = 0 \arrow[d] & & 0 = H^0(\mathcal{S}_{-1}) \\
& H^3(\mathcal{S}_3) \arrow[dl, "\sim"'] \arrow[d, "j_3^*"'] & H^2(\mathcal{A}_2) = 0 \arrow[d, "p_2^*"] & H^1(\mathcal{S}_0) = 0 & & \\
H^4(\mathcal{M}_{\mathbb{W}}) \arrow[d, "j_4^*"'] & H^3(\mathcal{A}_3) = 0 \arrow[d, "p_3^*"'] & H^2(\mathcal{S}_1) = 0 & & & \\
0 = H^4(\mathcal{A}_4) \arrow[d, "p_4^*"'] & H^3(\mathcal{S}_2) = 0 & & & & \\
H^4(\mathcal{S}_3) = 0 & & & & &
\end{tikzcd}$$

and thus to (see Theorem 13.3.16)

Theorem 13.4.3: *For $k = 4, \dots, k_0$ we have*

$$H^{5-k}(\widetilde{\mathcal{S}}_4) = H^0(\widetilde{\mathcal{S}}_k)/\mathbf{im}\,\widetilde{p}_k^*\Big|_{H^0\left(\widetilde{\mathcal{A}}_k\right)} = \mathbf{coker}\left(\widetilde{p}_k^*\Big|_{H^0\left(\widetilde{\mathcal{A}}_k\right)}\right)$$

and

$$H^{5-k}(\mathcal{S}_4) = H^0(\mathcal{S}_k)/\mathbf{im}\,p_k^*\Big|_{H^0\left(\mathcal{A}_k\right)} = \mathbf{coker}\left(p_k^*\Big|_{H^0\left(\mathcal{A}_k\right)}\right).$$

13.4.2. Trivial and Non-Trivial Weights

We restrict ourselves to such $\mathbb{M}$ which are highest weight modules. This makes some things easier, since we know a little bit more about the invariants of such modules. In particular, not all of them behave in the same way.

The Trivial Weight: Among these modules there is one exceptional case, namely the case where the sheaf is associated to the trivial representation, i.e. the underlying highest weight module $\mathbb{M}$ equals $\mathbb{M}_\lambda$, where $\lambda = 0$ (see Theorem 12.3.1).

Lemma 13.4.4 (Trivial weight)**:** *Let $\mathcal{M}$ be the sheaf associated to a trivial representation $\mathbb{M}_0 \otimes \mathbb{Z}_\Gamma$ and $\mathcal{M}_{\mathbb{W}}$ its restriction to the retract as in Theorem 13.4.1, then we have*

$$\mathbb{Z}_\Gamma = H^0(\mathcal{M}_{\mathbb{W}}) = H^0(\mathcal{A}_4) \qquad \textit{and} \qquad H^0(\mathcal{S}_3) \cong H^1(\mathcal{M}_{\mathbb{W}}).$$

Proof. Consider the exact sequence (13.78) for $k = 4$:

$$0 \longrightarrow H^0(\mathcal{M}_{\mathbb{W}}) \xrightarrow{j_4^*} H^0(\mathcal{A}_4) \xrightarrow{p_4^*} H^0(\mathcal{S}_3) \longrightarrow H^1(\mathcal{M}_{\mathbb{W}}) \longrightarrow 0.$$

The trivial representation $\mathbb{M}_0 \otimes \mathbb{Z}_\Gamma$ is one dimensional and the all elements act by the identity. Hence, we have for trivial representation

$$H^0(\mathcal{M}_{\mathbb{W}}) = (\mathbb{M}_0 \otimes \mathbb{Z}_\Gamma)^\Gamma = \mathbb{Z}_\Gamma.$$

We get analogously

$$H^0(\mathcal{A}^4) = H^0(\mathbf{C}_4, i_4^*\mathcal{M}_{\mathbb{W}}) = (\mathbb{M}_0 \otimes \mathbb{Z}_\Gamma)^{\Gamma_{\text{RedSquare}}} = \mathbb{Z}_\Gamma.$$

Since (13.78) is exact we know that j_4^* is injective and thus an isomorphism onto its image, which is the full space. Furthermore, we get by this observation that $p_4^* = 0$ because (13.78) is exact. Therefore, the third map has to be an isomorphism, which proves the lemma. □

Non-Trivial Weights: Let us now consider the case where the underlying highest weight module is of highest weight $\lambda \neq 0$. Then we know by Theorem 12.3.1 that

$$H^0(\mathcal{M}_{\mathbb{W}}) = 0. \tag{13.79}$$

This leads to

Lemma 13.4.5 (Non-trivial Weight)**:** *Let $\mathcal{M}$ be the sheaf associated to a representation $\mathbb{M}_\lambda \otimes \mathbb{Z}_\Gamma$ with $\lambda \neq 0$ and $\mathcal{M}_{\mathbb{W}}$ its restriction to the retract as in Theorem 13.4.1, then we have*

$$\begin{aligned} H^0(\mathcal{M}_{\mathbb{W}}) &= 0 \\ H^0(\mathcal{A}_4) &= (\mathbb{M}_\lambda \otimes_{\mathbb{Z}} \mathbb{Z}_\Gamma)^{\Gamma_{\text{RedSquare}}} \\ H^1(\mathcal{M}_{\mathbb{W}}) &\cong H^0(\mathcal{S}_3)/(\mathbb{M}_\lambda \otimes_{\mathbb{Z}} \mathbb{Z}_\Gamma)^{\Gamma_{\text{RedSquare}}}. \end{aligned}$$

Proof. The first part is Theorem 13.4.1. The second part was already used several times (see eg. proof of Lemma 13.4.4). For the last part we look again at the exact sequence (13.74) for $k = 4$, which reduces together with previous results to

$$0 \longrightarrow 0 \overset{j_4^*}{\hookrightarrow} (\mathbb{M}_\lambda \otimes_{\mathbb{Z}} \mathbb{Z}_\Gamma)^{\Gamma_{\text{RedSquare}}} \overset{p_4^*}{\hookrightarrow} H^0(\mathcal{S}_3) \twoheadrightarrow H^1(\mathcal{M}_{\mathbb{W}}) \longrightarrow 0.$$

Thus, the last part follows immediately from exactness of this sequence. □

13.4.3. Further Steps

We can in addition apply the interpretation of the occurring terms in the representation of the stalks in terms of induced modules as we did in Section 13.3.2. We are interested in the cohomology of the quotient, thus the induced modules e.g. in Proposition 13.3.10 are for us only a tool, which tell us how the group action descends to the quotient. The relation between a stalk in a point $\widetilde{x} \in \mathbb{W}$ and the stalk of the associated sheaf in $x = \pi(\widetilde{x})$ is giving by taking invariants (see Lemma 13.2.6 b)). If a stalk in $\widetilde{x}$ is given by an induced module one needs essentially two things to compute the associated stalk in x, namely to select one summand and to compute invariants of that summand. A priori there is no exceptional elements among the summands in the induced module. Hence, it is somehow arbitrary which of them to choose for the computation of invariants. It is more or less canonical to take the summand corresponding to the neutral element. For practical computations this has much less consequences than the representation of the related maps in SNF, which allows to compute efficiently a basis representation of the stalks of $\mathcal{S}_{k-1}$ if $\mathcal{S}_k$ and $\mathcal{A}_k$ are known. Unfortunately this leads to serious problems if we describe the group action on stalks of $\mathcal{S}_{k-1}$, because the base change is not compatible with the group action. Not to use SNFs leads to other problems. According to Lemma 13.3.4 the stalks of the $\widetilde{\mathcal{A}}_k$ are the direct sum of stalks of $\widetilde{\mathcal{S}}_k$. If we now try to write down the maps and modules as in our definition of them in Section 13.3.1 as the direct sum of previously defined maps or modules, respectively, and we write down the maps and modules for $\widetilde{\mathcal{S}}_{k-1}$ by taking representatives in $\widetilde{\mathcal{A}}_k$, in each step the size of the objects with which we have to perform our operations increases. Since we apply this recursively up to four times also the complexity of all following operations increases, because one has to rewrite all following objects and operations in terms of $\widetilde{\mathcal{A}}_4$ or $\widetilde{\mathcal{S}}_4$

respectively. This procedure makes the computations rather slow and we cannot recommend this. Thus, we have to decide whether we want to have an easy description of bases with a complicated computation of group actions or an easy description of the group action with a not very handy description of the modules.

We assume that it is much better to use SNFs even if one then has to implement a complicated group action. Here it might be useful to compute the function which computes the group action in a similar way than we already did to compute simple group actions on a single module (comp. e.g. Section 11.2.3).

We discussed in this last section of this thesis the issues which cause the main problems during the coming computations. To attack these problems is not difficult, but needs a significant amount of time. Thus, regrettably we have to restrict ourselves only to the presentation of the methods and to postpone the computation of the cohomology groups, which was the original goal of this thesis, to a follow up project.

14. Résumé et Perspectives

One never notices what has been done; one can only see what remains to be done.

(Marie Curie)

At last we give a condensed overview on what we have done in this thesis and what is left to be done to complete the computation of the cohomology of the symplectic group with the methods we introduced in this thesis. We will keep it short and refer to the sections where we already discussed several problems which occurred in this project.

14.1. Solved Problems

This thesis consists essentially of three parts.

- Computing a topological model.
- Implementing of the group action.
- Computing the cohomology groups with respect to the obtained topological model.

The first two points were solved completely during the work on this thesis. The last point could not be solved completely but only in parts. The main results we obtained are:

Extension to the Torsion Case We extended the construction of a deformation retract of Siegel modular varieties of McConnell and MacPherson [MM93, MM89], which in their work only holds for neat aritmetic subgroups of the symplectic group, to arbitrary arithmetic subgroups of finite index and thus in particular to the full group $\Gamma = \mathbf{Sp}_2(\mathbb{Z})$, which has torsion. To be able to handle torsion, we introduced the new notion of an orbicell decomposition, which is an orbifold equivalent to a classical cell decomposition. We obtain a deformation retract of $\mathfrak{S}_2/\Gamma$ together with an orbicell decomposition. The main result on this can be found in Theorem 9.2.11.

Computation of a Cell Decomposition We applied this construction of the nice[1] topological model to obtain a local description of the decomposition as well in the quotient $\mathbb{W}/\Gamma$ as in its lift $\mathbb{W}$. We can compute for each object in the decomposition of $\mathbb{W}$ its closure, boundary, neighbours, and stabilizer, which are the essential building blocks for the later constructions. The knowledge of this allows us to get a picture of the corresponding elements in $\mathbb{W}/\Gamma$. In Chapter 10 we discussed first how to compute efficiently all these derived informations and then we gave a detailed description of the computed

[1]Nice means in particular locally finite, regular, compatible with the group action and fine enough.

results. Parts of them can be found there or in the appendix of this thesis. More results are contained in the pdf-documents included in the CD-ROM, which is attached to the printed copy of this thesis.

Computation of the Group Action The implementation of the action of elements of the symplectic group on associated highest weight modules, which we described in Chapter 11, was much more challenging than expected. Our method returns the matrix representation in a fixed basis, which is also computed. To get both we had to apply various results from the theory of root systems, Lie algebras, and their representations. In addition one had to decompose each given group element in a special set of generators (see Section 11.2.3). At last we got some functions which are not only applicable to the symplectic group but also to some other groups (see e.g. Section 11.2.4.2). To reduce the runtime we use for example several precomputed partial results, such that it is possible to run that program also for highest weight modules beyond small ranks (see e.g. Section 6.5). We need this e.g. to compute the invariants of highest weight modules (see Section 11.2.5), i.e. the cohomology group of degree zero, which equals the global sections of the corresponding sheaf.

(Orbi-)Constructible Sheaves To get an efficient method to compute the cohomology, we are interested in in terms of the cohomology of sheaves, we extended the notion of constructible sheaves, which is originally due to Grothendieck and his collaborators, to an orbicell decomposition. This extension turned out to be quite powerful for the computation of cohomology of (orbi-)local systems on a space with a nice (orbi-)cell decomposition. Therefore, we introduced in Chapter 13 a new version of constructible sheaves on spaces like $\mathbb{W}$ together with some deduced sheaves on quotients of spaces, which we called orbi-constructible sheaves since they live on orbifolds, in a slightly more general context, which immediately applies to our situation (comp. Section 13.4). Furthermore, we derived some basic facts on them in Sections 13.1 and 13.2.

Resolutions The last step is the construction of two kinds of resolutions for (orbi-)constructible sheaves which are the most important tools beside the construction of the decomposition for computing the cohomology of the symplectic group. The first resolution uses embeddings of (orbi-)cells of fixed dimension to get an acyclic resolution (see Section 13.3.1 and therein Theorem 13.3.9). The second resolution is obtained by application of the long exact cohomology sequence from (5.4) to several short exact sequences used in the construction of the acyclic resolution. The malus point here is that we obtain a method which works only for some of the cohomology groups of degree less or equal to a given bound depending on the injectivity of maps $\widetilde{j}_k$ (level of injectivity). We conjecture that this bound is large enough to compute all cohomology groups.

14.2. Open Questions

First of all, we could not complete the computation of all cohomology groups, which was the original goal of this thesis, due to the fact that many parts of this project needed longer than estimated. Thus, we restricted this thesis more on the presentation of the tools, which can be used for the computation than on the computation of examples. Although we could at our current state obviously perform lots of computations by more or less step by step application of the functions included in our SAGE program, we decided not to include most of these partial results to this thesis. This is due to the fact that two necessary points are missing:

- One has to show that the level of injectivity $k_{\text{inj}} < 1$, such that the construction from Section 13.3.1 works for cohomology groups of *all* degrees $0 \leq q \leq d$ (d is the dimension of the space, i.e. in our example 4). We know already that $k_{\text{inj}} < d$. There are two ways to show that. One might try either to get an abstract proof in the general framework, which we did not find up to now, or one tries to attack this question directly with the example and computes the necessary maps (comp. the discussion in Section 13.4)
- One has to implement a suitable realization of the direct sum of stalks together with a group action on the sum of stalks before one starts the computation of *all* the comparison maps, which is not possible at the moment (comp. Section 13.4.3). It is only possible to plug the maps together from their building blocks by hand in some special cases, but not in general.

We assume that these missing parts are not a real problem. We prepared the things such that a possible follow-up study will be able to go the path we directed to the end and get all cohomology groups. This is the first part of the larger project whose final goal is the analysis of Hecke operators and their eigenvalues to check Harder's conjecture (comp. Section 1.2).

Appendix

A. List of Available Commands in Our Sage Program

In this chapter we give a list of the most important of the available commands grouped by their modules in our SAGE program package. The less important auxiliary functions are not stated here. In addition we state the method available in the Class `Cell`. For a more detailed description we refer to the comments in the source code. According to the SAGE programming guide lines [Tea11a], our source code has always a documentation of the function at begin of the code of the related function with a description on the usage.

Table A.1.: *Main methods from the module `io`.*

Command	Description
`saveobj2file`	saves a SAGE object to disk.
`savelist2file`	saves a list of SAGE objects to disk.
`loadobj`	loads a SAGE object which is stored on disk.

Table A.2.: *Main methods from the module `matrix_operations`.*

Command	Description
`is_diagonal_matrix`	checks whether a matrix is a diagonal matrix.
`expmat`	computes the exponential map for nilpotent matrices. This is more efficient than the build in method.

Table A.3.: *Main methods from the module `symplectic`.*

Command	Description
`symplectic_J`	returns for a given integer g and a type either J_g or J'_g (see Section 6.3.1.3)
`is_symplectic`	returns `True` if the input, a `Matrix` or a `MatrixGroupElement`, is symplectic and `False` otherwise.
`is_GSp`	is similar to the previous function, but checks the input to be in in the general symplectic group.

Table A.3.: *Main methods from the module `symplectic`.*

Command	Description
`symplectic2symplecticMatrix`	returns the matrix T, such that $T^t \cdot J_g \cdot T = J_g'$ (comp. Lemma 6.3.4 and (11.12)).
`symplectic2symplectic`	transfers a symplectic matrix with respect to one sympletic form to a symplectic matrix with respect to an other symplectic form using the matrix `symplectic2symplecticMatrix`.

Table A.4.: *Main methods from the module `symplecticgrp`.*

Command	Description
`GeneratorsSymplecticGroup`	computes for an integer g the generators of the symplectic group $\mathbf{Sp}_g(\mathbb{Z})$ as list of matrices.
`SymplecticGroup`	returns the symplectic group $\mathbf{Sp}_g(\mathbb{Z})$ for a given integer g as a SAGE `MatrixGroup`.
`makefinitegrp`	reduces a (finite) `MatrixGroup` over $\mathbb{Z}$ to a `MatrixGroup` over $\mathbb{Z}/N\mathbb{Z}$. By default we take $N = 7$, such that for a finite subgroup of $\mathbf{Sp}_2(\mathbb{Z})$ the reduction is an isomorphism .
`getgroupstructure`	returns the structure description of a finite `MatrixGroup` over a finite field using the GAP `SmallGroups` Library [BEO02] (comp. Section 10.5).
`getgenerators`	returns a list of elements generating a finite matrix group. It works over any ring and not as the build in method in SAGE only over finite fields.
`intersectionofgroups`	computes the intersection of two finite `MatrixGroup` objects.
`quotientsystem`	returns a set of representative s of the quotient of two groups.

Table A.5.: *Main methods from the module `cells`.*

Command	Description
`Cell`	the constructor for the class `Cell`. See Table A.8 for its attributes and Table A.9 for its methods.
`StandardCell`	the constructor for the class `StandardCell`, which is a sub class of `Cell`.
`is_Cell`	checks whether an object is an object in the class `Cell` (comp. Section 10.3.2).

Table A.5.: *Main methods from the module* `cells`*.*

Command	Description
`getmap2standard`	returns a matrix which maps a given cell to a standard cell (see Section 10.4.2).
`getstandardform`	returns to a given collection of column vectors stored as matrix the normalized form according to Section 10.3.1.
`getstabilizer`	computes the stabilizer of a cell as `MatrixGroup` object in SAGE (see Section 10.5).
`getclosure`	computes the closure of a cell as list of cells (see Section 10.4).
`getneighbours`	computes the neighbours of a cell (see Section 10.6)
`listofsymplecticmatrices` `fromcolumns`	returns a list of symplectic 4×4 matrices generated from columns of a given matrix by permutation and multiplying with some sign (see Section 10.4.1).
`listofGSpmatricesfromcolumns`	as above but for $\mathbf{GSp}_2$.
`decomposeintoorbits`	returns a set of representatives to a given list of cells and a group acting on that list.
`RedSquare`, `Vertebra`, `Crystal`,`Pyramid`, `Hexagon`, `Square`, `Triangle`, `DR`,`RR`, `Desargues`, `Reye`	is the standard cell of the corresponding type.
`StandardElementList`	is the list of standard elements.
`CellList`	list of cells in the closure of the standard cell of type RedSquare.
`VertebraList`, `CrystalList`, `PyramidList`, `HexagonList`, `SquareList`, `TriangleList`, `DRList`, `RRList`, `DesarguesList`, `ReyeList`	list of cells of the corresponding type in `CellList`.

Table A.6.: *Main methods from the module* `cellsio`*.*

Command	Description
`celloutput`	returns an object of type `Cell` with different formatting options (`screen`, `latex`, `plot2d`, `plot3d`, `skeleton`, and `graph`)
`savecell2file`	saves an object of type `Cell` with different formatting options (`screen`, `latex`, `plot2d`, `plot3d`, `skeleton`, and `graph`) to disk.
`savecelllist2file`	saves a list of objects of type `Cell` with `savecell2file` to disk.

Table A.7.: *Main methods from the module* `hwmodule`.

Command	Description
`SimpleLieAlgebra`	Constructor for a simple Lie algebra (see Section 11.2.1).
`LieType`	returns the type of a simple Lie algebra as string.
`HighestWeightModule`	Constructor for a highest weigh module to a given simple Lie algebra and a highest weight (see Section 11.2).
`randomgroupelemenetbyLiealgebra`	returns a random element in the Lie group associated to a Lie algebra.
`getdecomposition`	returns (essentially) the list which represents the decomposition of a group element into its generic generators (see Theorem 11.1.1).
`getdecompositionoflist`	does the same as the previous method but for a list of elements.
`getreprofgroupelement`	computes the matrix representation of a group element on a given highest weigh module (see Theorem 11.1.1).
`coordinatevector2HWMelement`	converts a coordinate vector into an element in a highest weight module.
`invariants`	computes the invariants of a highest weight module under a group element, a group or a list of group elements.

Table A.8.: *Attributes of the class* `Cells`.

Attribute	Description
`_matrix`	contains the defining set of the cell.
`_rank`	contains the rank of the cell (see Table 9.1.
`_name`	contains the string which is prompted on screen if the cell is returned.
`_latex_`	contains the LaTeX string which is returned by the method `latex()`.
`_cell_type`	contains the cell type as string.
`_bd_type`	is equal to `False` for all cells we consider.
`_transv`	contains the information on transversality according to Table 9.1.
`_map2standard`	contains, if known, the matrix which maps the cell onto one of the standard elements.
`_stab`	contains, if known, the stabilizer of the cell.
`_closure`	contains, if known, the closure of the cell.
`_nb`	contains, if known, the set of neighbours of the cell.

Table A.9.: *Methods of the class `Cells`.*

Method	Description
`__init__`	is the constructor for the class.
`_repr_`	returns the name of the cell stored in the attribute `_name`. All SAGE objects have to have this method.
`name`	returns the name of the cell stored in the attribute `_name`.
`_latex_`	returns the defining set stored in the attribute `_latex_`.
`latex`	is the same as above but has a different way to execute the command.
`_matrix_`	returns the defining set stored in the attribute `_matrix`.
`matrix`	is the same as above but has a different way to call it.
`matrix_latex`	returns the latex string to represent the defining set.
`map2standard`	returns the attribute `_map2standard`, if known, and computes the map otherwise before it is returned.
`rename`	is a method to set the attribute `_name` to a given value.
`rename_latex`	is a method to set the attribute `_latex_` to a given value.
`set_map2standard`	is a method to set the attribute `_map2standard` to a given value.
`set_closure`	is a method to set the attribute `_closure` to a given value.
`set_stabilizer`	is a method to set the attribute `_stab` to a given value.
`cell_type`	returns the attribute `_cell_type`.
`is_boundary_type`	returns the attribute `_bd_type`.
`is_transverse`	returns the attribute `_transv`.
`nrows`	returns the number of rows of the matrix representing the cell.
`ncols`	returns the number of columns of the matrix representing the cell.
`rank`	returns the attribute `_rank`.
`dim`	returns the dimension of the cell (see equation 9.10).
`__eq__`	is the method to evaluate expressions of cells with with ==.
`__ne__`	is the method to evaluate expressions of cells with with `!=`.
`__le__`	is the method to evaluate expressions with <=, i.e the partial ordering from Corollary 9.2.6.
`__ge__`	is the method to evaluate expressions with >=, i.e the partial ordering from Corollary 9.2.6.

Table A.9.: *Methods of the class* `Cells`.

Method	Description
`__cmp__`	is the method to evaluate comparisons of cells with respect to the partial ordering from Corollary 9.2.6.
`__mul__`	returns a new cell, which is the image of the cell under a given matrix or group element. This method implements left and right multiplication by a group element. (see Section 10.3.2.2)
`__rmul__`	see above.
`__apply__`	is essentially the same as `__mul__`, but allows to set in addition the attributes `_name` and `_latex_`.
`deleteclosure`	sets the content of the attribute `_closure` to `None`.
`closure`	returns the attribute `_closure`, if it is different from `None`, and otherwise computes the closure, sets the attribute to the computed result and returns it.
`boundary`	returns the boundary of the cell. If the closure is not known it computes the closure first.
`skeleton`	returns to a given integer k the set of cells in the k-skeleton.
`is_in_closure_of`	returns `True` if the cell is in the closure of an other cell and `False` otherwise (see Section 10.3.2.3).
`is_in_boundary_of`	returns `True` if the cell is in the boundary of an other cell and `False` otherwise (see Section 10.3.2.3).
`is_in_direct_boundary_of`	returns `True` if the cell is in the boundary of an other cell and their dimension differ by one and `False` otherwise (see Section 10.3.2.3).
`deleteneighbours`	sets the content of the attribute `_nb` to `None`.
`neighbours`	returns the neighbours of the cell. If no additional parameter is set, is computes all types, otherwise only those of some type.
`deletestabilizer`	set the content of the attribute `_stab` to `None`.
`stabilizer`	returns the stabilizer of the cell.
`orbit`	returns the orbit of the cell under the group generated by a given group element. The computation stops by default after 500 terms. There are at least orbits of length 12 if the element has finite order (comp. Section 6.3.3).

B. Sage Source Code

B.1. Sage Source Code from Section 10.3

Sage Source Code B.1: `getstandardform`.

```
def getstandardform (matr):
    if not is_Matrix(matr):
        raise TypeError, "matr has to be a matrix.";
    _matr_cols = matr.columns();
    _normed_col = [];
    _n = matr.nrows();
    _m = matr.ncols();
    #Normalizing columns
    for col in _matr_cols:
        for i in range(_n):
            if col[i]<0:
                _normed_col.append((-vector(col)).list());
                break;
            elif col[i]==0:
                continue;
            else:
                _normed_col.append(col.list());
                break;
    #Sorting
    _sort_key = lambda x : tuple ([x.count(0)]+[x[i] for i in range(_n)]);
    _sorted_list = sorted(_normed_col, key = _sort_key,reverse=True);
    _out_list=[];
    for vec in _sorted_list:
        _out_list+=vec;
    _new_matr=matrix(_m,_out_list).transpose();
    return _new_matr;
```

Sage Source Code B.2: *The method* `__mul__` *in the class* `Cell`.

```
def __mul__ (self, other):
    if is_Matrix(other):
        __matrix = other
    elif is_MatrixGroupElement(other):
        __matrix=other.matrix()
    else:
        try :
            __matrix = matrix(other)
        except:
            raise TypeError, "One Factor has to be a matrix or a matrix group
                element";
    __out = getstandardform((__matrix.inverse())*(self.matrix()));
    if self._map2standard == None:
        __out_map2std = None
    else:
        __out_map2std = self.map2standard()*__matrix
    if self._stab == None:
        __out_stab =None
    else:
        _stablst = [__matrix.inverse()*g*__matrix for g in self._stab]
        __out_stab = MatrixGroup(_stablst)
    if self._closure == None:
        __out_closure = None
    else:
        __out_closure =[__matrix*c for c in self._closure]
    __name = getname(self)
    __latex_name = getlatexname (self)
    return Cell (__out, name = __name, latex_name = __latex_name, cell_rank =
        self._rank, cell_type = self._cell_type, transv = self._transv, bd_type
         = self._bd_type, map2standard = __out_map2std, stabilizer = __out_stab
        , closure = __out_closure)
```

Sage Source Code B.3: *The class method* `is_in_closure_of`*. in the class* `Cell`*.*

```
def is_in_closure_of (self, other):
    """
    Returns True if self is in the closure of other
    """
    return all([column in self.matrix().columns() for column in other.matrix()
        .columns()]);
```

Sage Source Code B.4: *The class method* `__cmp__`*. in the class* `Cell`*.*

```
def __cmp__ (self, other):
    if self == other:
        return 0;
    elif self.is_in_closure_of(other):
        return -1;
    elif other.is_in_closure_of(self):
        return 1;
    else:
        return None;
```

B.2. Sage Source Code from Section 10.4

Sage Source Code B.5: *Function* `getclosure`*: Here a shortened version for the case that the cell is in the boundary of the standard 4-cell is stated. So we will skip the part where the cell is transformed to a standard cell and back again. We skip also those parts where we check whether there are precomputed results, where the results are loaded and stored and replace them by a comment what is done.*

```
def getclosure (cell_in, basedir = None, dir = 'output'):
    #1. Attribute set?
    if cell_in._closure != None:
        return cell_in._closure;
    #2. Load and return if a precomputed version possible.
    #3. If cell_in is in boundary of C(RedSq) and the closure for C(RedSq) is
        stored. Take that list and filterit for cells in closure of cell_in,
        set the attributes, and store them.
    #4. If cell_in not in the closure of C(RedSq) use getmap2standard to
        transform it to a standard cell
    #5. Start with the computation in case of a standard cell
    _n_rows = cell_in.nrows()
    _sgn = [];
    for sgn in tuples([-1,1],_n_rows):
        d=diagonal_matrix(sgn)
        if (d not in _sgn) and ( -d not in _sgn):
            _sgn.append(d);
    _sgn_mat_symp = [m for m in _sgn if is_symplectic(m)]
    _sgn_mat_non_symp = [m for m in _sgn if not is_symplectic(m)]
    _out=[ ]
    for cell in StandardElementList:
        if cell.dim() > cell_in.dim():
            continue
        print "* Elements of type %s" %cell.cell_type();
        _i = getindex(cell)
        cell_list = [ cell._matrix.matrix_from_columns(arr) for arr in
            arrangements(range(cell.ncols()),_n_rows)];
        cell_list = [ m for m in cell_list if m.det().abs()==1]
        cell_list0 = cell_list[:]
        cell_list_s =[cell_list.pop(cell_list.index(m)) for m in cell_list0 if
            is_symplectic(m)]
        cell_list_n = cell_list[:]
        _cell_matrix_list =[]
        print " - Matrices in Sp: Part 1 (symp)";
        for m in cell_list_s:
            _new_cell = m.inverse()*cell._matrix
            for d in _sgn_mat_symp:
                _new_cell_d = getstandardform(d*_new_cell)
                if all([column in _new_cell_d.columns() for column in cell_in.
                    _matrix.columns()]):
                    if _new_cell_d not in _cell_matrix_list :
                        _cell_matrix_list.append(_new_cell_d)
                        __out_map2std = m*d
                        _NC = Cell (_new_cell_d, name = cell._cell_type+str(_i), latex_name = cell._latex_+'_{%s}'%_i, cell_rank = cell._rank, cell_type = cell._cell_type, transv = cell._transv, bd_type = cell._bd_type, map2standard = __out_map2std)
                        _out.append(_NC)
                        _i += 1
        print " - Matrices in Sp: Part 2 (non symp.)";
        for m in cell_list_n:
            _new_cell = m.inverse()*cell._matrix
            for d in _sgn_mat_non_symp:
                _new_cell_d = getstandardform(d*_new_cell)
                __out_map2std = m*d
                if is_symplectic(__out_map2std):
                    if all([column in _new_cell_d.columns() for column in cell_in._matrix.columns()]):
                        if _new_cell_d not in _cell_matrix_list:
                            _cell_matrix_list.append(_new_cell_d)
                            _NC = Cell (_new_cell_d, name = cell._cell_type+str(_i), latex_name = cell._latex_+'_{%s}'%_i, cell_rank = cell._rank, cell_type = cell._cell_type, transv = cell._transv, bd_type = cell._bd_type, map2standard = __out_map2std)
                            _out.append(_NC)
                            _i += 1
    # Postprocessing: If the cell was not in the closure of C(RedSq) transform back. Save to disk, set attributes
    return cell_in._closure;
```

´

B.3. Sage Source Code from Section 10.6

Sage Source Code B.6: *Function* *listofsymplecticmatricesfromcolumns: ...*

```
def listofsymplecticmatricesfromcolumns (mat,symplectic_type=0):
    """
    Returns a list of symplectic matrice which are build out of column vectors of mat (up to sign)
    """

    _n_rows = mat.nrows();

    sgn_mat_list = [diagonal_matrix(d) for d in tuples([-1,1],_n_rows)]
    _sgn = sgn_mat_list[:]
    sgn_s = [sgn_mat_list.pop(sgn_mat_list.index(m)) for m in _sgn if
        is_symplectic(m, symplectic_type = symplectic_type)]
    sgn_ns = sgn_mat_list[:]

    print "Computing list of matrices for possible surrounding the top
        dimensional cell\n"
    #print " This might take some time. Please be patient-";
    _tmp = [mat.matrix_from_columns(ar) for ar in arrangements(range(mat.ncols
        ()),_n_rows)]
    print " * All %sx%s matrices from columns %10s" % (_n_rows,_n_rows,len(
        _tmp))
    _tmp = [m for m in _tmp if m.rank()==_n_rows]
    print " * All %sx%s matrices of full rank from columns %10s" % (_n_rows,
        _n_rows,len(_tmp))
    print " --> split list in symplectic and non symplectic";
    _tmp_0 = _tmp[:]
    list_s =[_tmp.pop(_tmp.index(m)) for m in _tmp_0 if is_symplectic(m,
        symplectic_type = symplectic_type)];
    list_n = _tmp[:];
    del _tmp;
    print " --> done";
    print "Add symplectic products to list"
    out = [d*m for m in list_s for d in sgn_s]
    print " --> done\n";
    print "Start to check symplecticity for the remaining cases.\n";
    for d in sgn_ns:
        for m in list_n:
            md= d*m
            if is_symplectic(md, symplectic_type = symplectic_type):
                out.append(md)

    print " Done\n"
    print " * Found number of symplectic %sx%s matrices from columns is %10s\n
        " % (_n_rows,_n_rows,len(out))

    return out
```

Sage Source Code B.7: *Function* `getneighbours`*: We abbreviate the part of the function, which is related to load and return already computed lists of neighbours.*

```
def getneighbours(cellin, type=None, basedir = BASEPATH, dir = 'output/cells/
    neighbours'):

    if not is_Cell(cellin):
        raise ValueError, "%s has to be a cell" %cellin

#1. Attribute set?
```

```
    if cellin._nb!=None:
        if type == None:
                        return cellin._nb
        else:
            out = [c for c in cellin._nb if c.cell_type()==type]
            return out
#2. Load and return if a precomputed version possible.
#3.Start with the computation
    if type != None and cellin.cell_type()==type:
        return []
    elif type == None:
        print "=======================================================
        print "| Compute List of symplectic matrices from columns "
        sympMats = [m.inverse() for m in listofsymplecticmatricesfromcolumns(
            cellin.matrix())]
        print "| --->done"
        print "=======================================================\n"
        out =[]
        print "======================================================"
        print "| * Start loop over cells"
        for c in CellList:
            if cellin.dim() >= c.dim():
                continue

            for m in sympMats:
                _m = c.apply(m)
                if cellin.is_in_closure_of(_m):
                    if _m not in out:
                        out.append(_m)
        print "\n";
    else:# different types
        stdC = [c for c in StandardElementList if c.cell_type()== type][0]
        d = stdC.dim();
        print "d=%s" %d;
        if cellin.dim() >= d:
            print "E"
            out = []
        # Compute List of symplectic matrices from columns
        sympMats = listofsymplecticmatricesfromcolumns(cellin.matrix())
        print "| --->done"

        # Start loop over cells"
        out = []
        for m in sympMats:
            _m = stdC.apply(m)
            if cellin.is_in_closure_of(_m):
                if _m not in out:
                    out.append(_m)
        print "| --> Done";
        print "====================================================="

    # Write elements

    #Set attributes
    if type== None:
        setattr(cellin, '_nb', out)

        # Save neighbours for later use
    saveobj2file (out, cellin.name()+'neighbours-'+'%s'%type , basedir =
        basedir, dir = dir)
    print '---> Done\n'

    return out
```

B.4. Sage Source Code from Section 11.2

Sage Source Code B.8: `negative_identity`.

```
def negative_identity(L, step=0, sparse=True, permutation_list=[],
    negative_list=[], positive_list=[], base_ring=QQ):
    if permutation_list == []:
        if negative_list == []:
            negative_list = negativegenericgroupgeneratorsfunctions(L,base_ring
                =base_ring)
        if positive_list == []:
            positive_list = positivegenericgroupgeneratorsfunctions(L,base_ring
                =base_ring)
        permutation_list = genericpermutations(L,negative_list=negative_list,
            positive_list=positive_list)

    _ncols = permutation_list[0][1].ncols()

    _tmp = [[[i,j],g1*g2] for i,g1,p1 in permutation_list for j,g2,p2 in
        permutation_list]
    for i in range(len(_tmp)):
        if all([_tmp[i][1][j,j]==1 for j in range(step)]) and _tmp[i][1][step,
            step]==-1:

            return _tmp[i][1]
    _tmp =[(idx1+idx2, g1*g2) for idx1,g1 in _tmp for idx2,g2 in _tmp if
        is_diagonal_matrix(g1*g2)]
    for i in range(len(_tmp)):
        if all([_tmp[i][1][j,j]==1 for j in range(step)]) and _tmp[i][1][step,
            step]==-1:
```

```

            return _tmp[i][1]
```

Sage Source Code B.9: *actionmatrixofUEAelement.*

```
def actionmatrixofUEAelement(UEAel,HWM, base_ring=QQ, sparse=True):

    entries = dict();
    _B = [gap.ExtRepOfObj(b) for b in HWM.Basis()]
    _dim = len(_B)
    _Null = (0*_B[0]).ExtRepOfObj()
    j=0
    for b in _B:
        z = gap.POW(UEAel,b)
        z_ = z.ExtRepOfObj()

        if z_!=_Null:
            i = 1
            _len = len(z_)
            while i < _len:
                entries[(int(z_[i,1])-1,j)] = int(z_[i+1])
                i+=2
        j+=1
    return matrix(base_ring, _dim, entries, sparse=sparse)
```

Sage Source Code B.10: *getreprofgroupelement.*

```
def getreprofgroupelement(el, HWM, negative_list=[], positive_list=[],
    HWMactiongenerators=[], permutation_list=[], row_position_list=[],
    UEA_generators=[], base_ring=QQ, sparse=True):
    if el==el.parent().one_element():
        _dim = HWM.Dimension()
        return identity_matrix(base_ring, _dim, sparse=sparse)

    L=HWM.LeftActingAlgebra()

    if negative_list == []:
        negative_list = negativegenericgroupgeneratorsfunctions(L,base_ring=
            base_ring)
    if positive_list == []:
        positive_list = positivegenericgroupgeneratorsfunctions(L,base_ring=
            base_ring)
    if permutation_list == []:
        permutation_list = genericpermutations(L,negative_list=negative_list,
            positive_list=positive_list)
    if row_position_list == []:
        row_position_list = genericrowoperationdict(L, negative_list=
            negative_list)
    if UEA_generators == []:
        UEA_generators = latticegeneratorsinUEA(L)
    if HWMactiongenerators == []:
        HWMactiongenerators = genericgroupgeneratorsfunctionsonHWM (HWM,
            base_ring=base_ring, sparse=sparse)
    _dec = getdecomposition(el,L,collection=[], negative_list = negative_list,
         positive_list = positive_list, permutation_list = permutation_list,
        row_position_list = row_position_list, base_ring=base_ring, sparse=
        sparse)
    _mat = prod([HWMactiongenerators[ref[0]](ref[1]) for ref, mat in _dec])
    return _mat
```

Sage Source Code B.11: *rowpermutation*.

```
def rowpermutation(mat,L, step=0, collection=[], negative_list=[],
    positive_list=[],permutation_list=[], base_ring=QQ, sparse=True):
    _out=mat
    # Generate possibly non existing lists
    if permutation_list == []:
        if negative_list == []:
            negative_list = negativegenericgroupgeneratorsfunctions(L,base_ring
                =base_ring)
        if positive_list == []:
            positive_list = positivegenericgroupgeneratorsfunctions(L,base_ring
                =base_ring)
        permutation_list = genericpermutations(L,negative_list=negative_list,
            positive_list=positive_list)
    # Permutations
    _collect = collection[:]
    _len = len(permutation_list)
    _perm_dict = dict([[p_j,i] for i,m,p in permutation_list for p_j in p])
    _col = mat.column(step)
    _min = 0
    _cntr = 0
    _cntr_min = 0
    for c_i in _col[step:]:
        if c_i!= 0:
            if _min==0 or abs(c_i)<abs(_min):
                _min = c_i
                _cntr_min = _cntr
        _cntr+=1
    _p = (step,step+_cntr_min)
    if _p == (step,step):
        if _out[step,step] < 0:
            _out = negative_identity(L, step=step, sparse=sparse,
                permutation_list=permutation_list, negative_list=negative_list,
                positive_list=positive_list, base_ring=base_ring)*_out
            append_negative_identity(L,_collect, step=step, sparse=sparse,
                permutation_list=permutation_list,negative_list=negative_list,
                positive_list=positive_list,base_ring=base_ring)
        return [_out, _collect]
    _i = _perm_dict[_p]
    _perm_mat = permutation_list[_i][1]
    _out = _perm_mat * _out

    _collect.append([[_i, -1], negative_list[_i](-1)])
    _collect.append([[_i+_len, 1], positive_list[_i](1)])
    _collect.append([[_i, -1], negative_list[_i](-1)])
    #Give the right sign
    if _out[step,step] < 0:
        _out = negative_identity(L, step=step, sparse=sparse, permutation_list=
            permutation_list, negative_list=negative_list, positive_list=
            positive_list, base_ring=base_ring)*_out
        append_negative_identity(L,_collect, step=step, sparse=sparse,
            permutation_list=permutation_list,negative_list=negative_list,
            positive_list=positive_list,base_ring=base_ring)
    return [_out, _collect]
```

Sage Source Code B.12: *div_min.*

```
def div_min(div,n):
    try:
        div=int(div);
        n=int(n)
    except:
        try:
            div=float(div);
            n=float(n);
        except:
            raise ValueError, "arguments of div_min has to be integers or
                floats"
    if div== 0:
        raise ZeroDivisionError, "Division by zero"
    if n==0:
        return [0,0]
    if abs(n)<abs(div):
        raise ValueError, "div>n";
    _out= divmod(n,div)
    _out = [-1*_out[0],_out[1]]

    if abs(_out[1] )> abs(div/2):
        _out = [_out[0]-1,_out[1]-div];

    return _out
```

Sage Source Code B.13: *rowtransformation. To abbreviate the code we only give the version for* $\mathbf{R} = \mathbb{Z}$*, which raises an error in some cases, otherwise the variable _sgn has to be handled in a different way.*

```
def rowtransformation (mat,L, step=0, collection=[], negative_list=[],
    positive_list=[], permutation_list=[],row_position_list=[], base_ring=QQ,
    sparse=True):
    # Generate possibly non existing lists
    if negative_list == []:
        negative_list = negativegenericgroupgeneratorsfunctions(L,base_ring=
            base_ring)
    if positive_list == []:
        positive_list = positivegenericgroupgeneratorsfunctions(L,base_ring=
            base_ring)
    if permutation_list == []:
        permutation_list = genericpermutations(L,negative_list=negative_list,
            positive_list=positive_list)
    if row_position_list == []:
        row_position_list = genericrowoperationdict(L, negative_list=
            negative_list)

    _collect = collection[:]
    _out_mat = mat
    _len = mat.nrows()
    _len_n = len(negative_list)

    for i in range(step+1,_len ):
        _a = _out_mat[step,step]
        _n = _out_mat[i,step]

        if _n == 0:
            continue;
        _sgn = row_position_list[i][(i,step)][1]
        if abs(_sgn)!=1:
            raise ValueError, "Entries in generators different from +-1";
        _cntr = _sgn*( div_min(_a,_n)[0])
        _i = row_position_list[i][(i,step)][0]
        _row_mat = negative_list[_i](_cntr)
        _out_mat = _row_mat*_out_mat
        _collect.append([[_i,-_cntr],negative_list[_i](-_cntr)])
        _out = [_out_mat,_collect]
        # Permute Elements
        _out = rowpermutation (_out[0], L,step=step, collection=_out[1],
            negative_list=negative_list, positive_list=positive_list,
            permutation_list=permutation_list, base_ring = base_ring, sparse=
            sparse)
        _collect = _out[1]
        _out_mat = _out[0]
    return _out
```

Sage Source Code B.14: *The function `getdecomposition`, which computes the whole decomposition of a given matrix `mat` with respect to generic Lie algebra generators coming from the Lie algebra `L`.*

```
def getdecomposition (mat, L, collection=[], negative_list=[], positive_list
    =[], permutation_list=[],row_position_list=[], base_ring=QQ, sparse=True):
    if not is_Matrix(mat):
        if not is_MatrixGroupElement(mat):
                raise TypeError, "Object to decompose has to be a matrix or a
                    matrix group element."
        else:
                mat = mat.matrix();
    #Trivial Case
    if mat == mat.parent().one_element():
        return collection
    # Generate non existing lists
    if negative_list == []:
        negative_list = negativegenericgroupgeneratorsfunctions(L,base_ring=
            base_ring)
    if positive_list == []:
        positive_list = positivegenericgroupgeneratorsfunctions(L,base_ring=
            base_ring)
    if permutation_list == []:
        permutation_list = genericpermutations(L,negative_list=negative_list,
            positive_list=positive_list)
    if row_position_list == []:
        row_position_list = genericrowoperationdict(L, negative_list=
            negative_list)

    _len = len(negative_list)
    _collect = collection[:]
    _out = [mat, _collect]
    print 'Transforming to upper triangular matrix:\n Please wait!\n\n';
    for step in range(LieType(L)[1]):
        _out = rowpermutation(_out[0], L, step=step, collection=_out[1],
            negative_list=negative_list, positive_list=positive_list,
            permutation_list=permutation_list, base_ring=base_ring, sparse=
            sparse)
        while not is_zeroed(_out[0], step=step):
            _out = rowtransformation(_out[0], L, step=step, collection=_out[1],
                 negative_list=negative_list, positive_list=positive_list,
                permutation_list=permutation_list, row_position_list=
                row_position_list, base_ring=base_ring, sparse=sparse)

    _mat_out = _out[0]
    _collect = _out[1]
    print "Eliminating elements in upper triangular matrix:\n Please wait!\n\n
        "
    for m in row_position_list:
        _keys = m.keys();
        if len(_keys)>0:
                for k in _keys:
                        _entry = _mat_out[k[1],k[0]];
                        if _entry !=0:
                            _idx=m[k][0]
                            _sgn=m[k][1]
                            if abs(_sgn) != 1:
                                raise ValueError, "Entries in generators
                                    different from +-1";
                            _mat_out = positive_list[_idx](-_sgn*_entry)*_mat_out
                            _collect.append([[_idx+_len, _sgn*_entry],
                                positive_list [_idx] (_sgn*_entry)])
    if _mat_out!= _mat_out.parent().one_element():
        print _mat_out;
        print 'ERROR'
        return None
    print 'Decomposition succesful';
    return _collect
```

Sage Source Code B.15: *The function* `getgenerators`

```
def getgenerators (G, check=False):

    if check:
        try:
            _H = G[:]
            for _g1 in G:
                for _g2 in G:
                    _g12 = _g1*_g2
                    _H.remove(_g12)
                    _H.append(_g12)
        except:
            print "Set is not closed under multiplication\n Therefore, the
                output is the set itself and NOT a set of generators"
            return G

    __G = G[:]

    _id = __G[0].parent().one()
    _gen = []
    tmp = []
    if _id in __G:
        tmp.append(_id)
        __G.remove(_id)
    if -_id in __G:
        _gen.append(-_id)
        tmp.append(-_id)
        __G.remove(-_id)

    while __G!=[]:
        #take 1st element
        _g = __G.pop()
        _gen.append(_g)
        tmp.append(_g)
        _gprod =_g
        if __G==[]:
            return _gen
        # remove powers of th element
        cnt = 0
        while _gprod != _id:
            if cnt > max_cnt:
                print "Computation of generators failed."
                raise ValueError, "Maximal recursion depth exceeded. Pobable
                    there is an element of infinite order"
            cnt=cnt+1
            _gprod = _gprod * _g
            if _gprod in __G:
                __G.remove(_gprod)
            if _gprod not in tmp:
                tmp.append(_gprod)
            if __G==[]:
                return _gen

        for _h1 in tmp:
            for _h2 in tmp:
                _h12 = _h1*_h2
                if _h12!=_id:
                    if _h12 not in tmp:
                        tmp.append(_h12)
                        if _h12 in __G:
                            __G.remove(_h12)
                            if __G==[]:
                                return _gen
    return _gen
```

Sage Source Code B.16: *The function* `coordinatevector2HWMelement`

```
def coordinatevector2HWMelement (v, M):
    _bas = M.Basis()
    _v = 0*_bas[1]
    _cnt =1
    for vi in v:
        _v=_v+vi*_bas[_cnt]
        _cnt = _cnt+1
    return _v
```

Sage Source Code B.17: *The function* `Invriants`*, which computes the invariants of a given highest weight modules and a group.*

```
def invariants(HWM, Grp, only_generators= True, in_type=0, negative_list=[],
    positive_list=[], HWMactiongenerators=[], permutation_list=[],
    row_position_list=[], UEA_generators=[], base_ring=QQ, sparse=True):

    print "* Check Input and convert to list of matrices"
    V = VectorSpace(base_ring,HWM.Dimension())
    if isinstance(Grp, list):
        if Grp==[]:
            return V
        elif is_Matrix(Grp[0]):
            pass
            _G = Grp[:]
        elif is_MatrixGroupElement(Grp[0]):
            _G = [g.matrix() for g in Grp]
        else:
            raise TypeError, "Elements in the list has to be a matrix or a
                matrix group element"
    elif is_MatrixGroup(Grp):
        if not Grp.is_finite():
            raise NotImplementedError, "Is only implemented for finite Groups"
        else:
            _G = [g.matrix() for g in Grp]
    else:
        raise TypeError, "Input has to be either a MatrixGroup or a list of
            matrices or matrix group elements."

    if only_generators:
        print "* Compute List of Generators"
        #check has to be false since some of the sets are not invariant under
            the group.
        _G = getgenerators(_G,check=False)
        print " --> Done"
    if in_type == 0:
        print "* Coverting matrices to the right type."
        _G = [symplectic2symplectic(g, type_from=0,type_to=1) for g in _G]
        print " Number of generators%s"% len(_G)
        print " -> Done"

    print "* Precompute list related to HWMs if not known"

    L=HWM.LeftActingAlgebra()
    if negative_list == []:
        negative_list = negativegenericgroupgeneratorsfunctions(L,base_ring=
            base_ring)
    if positive_list == []:
        positive_list = positivegenericgroupgeneratorsfunctions(L,base_ring=
            base_ring)
    if permutation_list == []:
        permutation_list = genericpermutations(L,negative_list=negative_list,
            positive_list=positive_list)
    if row_position_list == []:
        row_position_list = genericrowoperationdict(L, negative_list=
            negative_list)
    if UEA_generators == []:
        UEA_generators = latticegeneratorsinUEA(L)
    if HWMactiongenerators == []:
        HWMactiongenerators = genericgroupgeneratorsfunctionsonHWM (HWM,
            base_ring=base_ring, sparse=sparse, UEA_generators=UEA_generators)

    print " Have to handle %s elements" %len(_G)
    cnt =1
    for g0 in _G:
        print " * Start with element %s" %cnt
        g = getreprofgroupelement(g0, HWM, negative_list = negative_list,
            positive_list = positive_list, HWMactiongenerators =
            HWMactiongenerators, permutation_list = permutation_list,
            row_position_list = row_position_list, UEA_generators=[], base_ring
             = base_ring, sparse = sparse)

        id = g.parent().one_element()
        m = g-id
        mk = m.right_kernel()
        if mk == V.zero_submodule():
            return V.zero_submodule()
        V = V.intersection(mk)

        if V==V.zero_submodule():
            return V
        cnt=cnt+1


    print ' --> Done'

    return V
```

C. List of Files on the CD-ROM

We included in the printed copy of this thesis a CD-ROM with the complete source code of the used SAGE functions and objects and some additional tables with computational results, which we did not include in the printed version due to their size.

In the folder `sagesource` on finds the following files:

- `__init__.py`: Has to be there by Python development rules and controls the import of our package.
- `all.py`: Lists all SAGE commands which are available in the package without referring to the module, i.e. file, in which they are contained. Commands listed in `all.py` are accessible directly, those which are not listed via `filename.command`.
- `cells.py`: This is the most important file for the computations related to cells, their closures, neighbours, stabilizers etc.. It contains in particular the definition of the class `Cell`, which is used for the description of cells in the program.
- `hwmodule.py`: This module bundles all things related to the computations of highest weight modules and group actions on them (see also Section 11.2).
- `cellsio.py`: This module contains the functions which are used to save cells or lists of cells to a file as well as the function which defines several output forms for cells like `matrix` or `graph`.
- `io.py`: Here general function related to the reading and writing of objects from/to disk are defined. These functions are for example used to load precomputed results.
- `matrix_operations.py`: Contains the two functions `is_diagonal_matrix` and `expmat`.
- `symplectic.py`: This module contains the functions which check whether matrices are symplectic and the function which converts different realisations of symplectic groups into another realisation (see Section 6.3.1.3).
- `symplecticgrp.py`: This module contains the functions `GeneratorsSymplecticGroup` and `SymplecticGroup` as well as some additional function for groups and their structure.

In the folder `data` there are the following files:

- `cells_in_closure_of_RedSq.pdf` (52 pages): A list of all cells in the closure of the standard 4-cell RedSquare. We list these cells grouped by their type. In addition we included for each cell the defining set, i.e. the collection of column vectors which determines the cell according to Sections 8.4 and 9.1, and a matrix which maps the cell to a standard cell in Σ.

- `stabilizers.pdf` (137 pages): Here one finds for each cell in the closure of the standard cell RedSquare its stabilizer. The stabilizers are stated as the list of their elements which are represented by 4×4 matrices.
- `intersections.pdf` (20 pages):
- `neighbours.pdf` (10 pages): This file contains for all standard cells in Σ, grouped by the type, the cells in the neighbourhood. The neighboured cells are represented by their defining sets.
- `modules.pdf` (66 pages): We list the PBW bases for all highest weight modules of weight $\lambda = (\lambda_1, \lambda_2)$ for the symplectic group for $\lambda_1 \leq 5$ and a collection of $\lambda_2 \geq 0$.

List of Figures

List of Tables

Bibliography

[AGM02] Avner Ash, Paul E. Gunnells, and Mark McConnell, **Cohomology of Congruence Subgroups of** $\mathrm{SL}_4(\mathbb{Z})$, J. Number Theory **94** (2002), 181—-212.

[AGM08] ______, **Cohomology of Congruence Subgroups of** $\mathrm{SL}_4(\mathbb{Z})$ **II**, J. Number Theory **128** (2008), no. 8, 2263—-2274.

[AGM10] ______, **Cohomology of Congruence Subgroups of** $\mathrm{SL}_4(\mathbb{Z})$**. III**, Math. Comp. **79** (2010), no. 271, 1811—-1831, arXiv:0903.3201v1 [math.NT].

[AGM11] ______, **Torsion in the cohomology of congruence subgroups of** $\mathrm{SL}_4(\mathbb{Z})$ **and galois representations**, J. Algebra **325** (2011), no. 1, 404—-415, arXiv:1002.3385 math.NT.

[AGV73] Michael Artin, Alexander Grothendieck, and Jean-Louis Verdier (eds.), **Sém. géométrie algébrique du Bois-Marie (1963-1964) (SGA4) - Théorie des Topos et Cohomologie Étale des Schémas**, Lecture Notes in Mathematics (LNM), Springer-Verlag, 1972-73.

[AGV23] ______, **Sém. géométrie algébrique du Bois-Marie (1963-1964) (SGA4) - Théorie des Topos et Cohomologie Étale des schémas Tome 1 - Exposés IX à XIX**, Springer-Verlag, 1972/3.

[AM97] Avner Ash and Mark McConnell, **Cohomology at Infinity and the Well-Rounded Retract For the General Linear Groups**, Duke Math. J. **90** (1997), no. 3, 549—-576.

[AMRT75] Avner Ash, David Mumford, Michael Rapoport, and Yung Sheng Tai, **Smooth compactification of locally symmetric varieties**, Lie Groups: History, Frontiers and Applications, vol. IV, Math Sci. Press, 1975.

[Ash77] Avner Ash, **Deformation retracts with lowest posible dimension of arithmetic quotients of self adjoint homogenous cones**, Math. Ann. **225** (1977), no. 1, 69—-76.

[Ash80] ______, **Cohomology of congruence subgroups of** $\mathrm{SL}_n(\mathbb{Z})$, Math. Ann. **249** (1980), no. 1, 55—-73.

[Ash84] ______, **Small-dimensional classifying spaces for subgroups of general linear groups**, Duke Math. J. **51** (1984), 459—-468.

[Bar57] E.S. Barnes, **The complete enumeration of extreme senary forms**, Philos. Trans. Roy. Soc. London. Ser. A. **249** (1957), no. 969, 461—-506.

[BEO02] Hans Ulrich Besche, Bettina Eick, and Eamonn O'Brien, **GAP - SmallGroups Library**, `http://www-public.tu-bs.de:8080/~beick/soft/small/small.html`, January 2002.

[BFvdG08] Jonas Bergström, Carel Faber, and Gerard van der Geer, **Siegel Modular Forms of Genus 2 and Level 2 Cohomological Computations and Conjectures**, Int. Math. Res. Not. IMRN **2008** (2008), 1—-20, arXiv:0803.0917v2 [math.AG].

[BFvdG11] ______, **Siegel modular forms of degree three and the cohomology of local systems**, August 2011.

[BGL99a] Christina Birkenhake, Víctor González-Aguilera, and Herbert Lange, **Automorphism groups of 3-dimensional complex tori**, J. Reine Angew. Math. (Crelles Journal) **508** (1999), 99—-125.

[BGL99b] ______, **Automorphisms of 3-dimensional abelian varieties**, Complex geometry of groups (Olmué, 1998) (Providence, R.I.), Contemp. Math., no. 240, American Mathematical Society, 1999.

[BHC61] Armand Borel and Harish-Chandra, **Arithmetic subgroups of algebraic groups**, Ann. of Math. **75** (1961), no. 3, 485—-535.

[Bir37] George David Birkhoff, **Representability of Lie algebras and Lie groups by matrices**, Ann. of Math. **38** (1937), no. 2, 526—-532.

[BL92] Alan Brownstein and Ronnie Lee, **Cohomology of the Symplectic Group** $\mathrm{Sp}_4(\mathbb{Z})$ **Part I: The Odd torsion Case**, Trans. Amer. Math. Soc. **334** (1992), no. 2, 575—-596.

[BL94a] Christina Birkenhake and Herbert Lange, **Fixed-point free automorphisms of abelian varieties**, Geom. Dedicata (1994), no. 51, 201—-213.

[BL94b] Alan Brownstein and Ronnie Lee, **Cohomology of the Symplectic Group** $\mathrm{Sp}_4(\mathbb{Z})$ **Part II: Computations at the Prime 2**, Michigan Math. J. **41** (1994), no. 1, 181—-208.

[BL04] Christina Birkenhake and Herbert Lange, **Complex abelian varieties**, 2nd ed., Grundlehren der Mathematischen Wissenschaften, no. 302, Springer-Verlag, 2004.

[BLS11] Reinier Broker, Kristin Lauter, and Marco Streng, **Abelian surfaces admitting an (l,l)-endomorphism**, arXiv:1106.1884v1 [math.AG].

[Bor84] Armand Borel, **Sheaf Theoretic Intersection Theory**, Intersection Cohomology (Armand Borel et al., eds.), Progress in Mathematics, vol. 50, Birkhäuser Verlag, 1984, pp. 47—-182.

[Bre97] Glen E. Bredon, **Sheaf theory**, 2nd ed., Graduate Texts in Mathematics (GTM), no. 170, Springer-Verlag, 1997.

[Bro94] Kenneth S. Brown, **Cohomology of groups**, Graduate Texts in Mathematics (GTM), no. 87, Springer-Verlag, 1994.

[BS73] Armand Borel and Jean-Pierre Serre, **Corners and arithmetic groups**, Comment. Math. Helv. **48** (1973), 436—-491, avec un appendice: Arrondissement des variétés à coins, par A. Douady et L. Hérault.

[Bum04] Daniel Bump, **Lie Groups**, Graduate Texts in Mathematics (GTM), no. 225, Springer-Verlag, 2004.

[Bür79] Balz Christoph Bürgisser, **Gruppen virtuell endlicher Dimension und Periodizität der Cohomologie**, Ph.D. thesis, ETH Zürich, Diss ETH 6425, 1979.

[Bür82] ______, **Elements of finite order in symplectic groups**, Arch.Math. **39** (1982), 501—-509.

[BW00] Armand Borel and Nolan Wallach, **Continuous cohomology, discrete subgroups, and representations of reductive groups**, 2nd ed., Mathematical Surveys and Monographs, vol. 67, American Mathematical Society, Providence, R.I., 2000.

[Cai61] Stewart Scott Cairns, **Introductory topology**, The Ronald Press Company, New York, 1961.

[CS88] John H. Conway and Neil J. A. Sloane, **Low-Dimensional Lattices III: Perfect Forms**, Proc. Royal Soc. London, Series A **418** (1988), 43—-80.

[CS99] John H. Conway and Neil J. A. Sloan, **Sphere packings, lattices and groups**, 3rd ed., Grundlehren der Mathematischen Wissenschaften, no. 290, Springer-Verlag, 1999.

[de 02] Willem A. de Graaf, **QuaGroup, Reference, Version 1.3**, The GAP Group, 2002.

[DHSW03] Jean-Guillaume Dumas, Frank Heckenbach, David Saunders, and Volkmar Welker, **Computing simplicial homology based on efficient Smith normal form algorithms**, Algebra, Geometry, and Software Systems (Michael Joswig and Nobuki Takayama, eds.), Springer-Verlag, March 2003, pp. 177—-206.

[Dim04] Alexandru Dimca, **Sheaves in topology**, Universitext, Springer-Verlag, 2004.

[DSV00] Jean-Guillaume Dumas, David Saunders, and Gilles Villard, **Integer Smith Form via the Valence: Experience with Large Sparse Matrices from Homology**, Algorithms and Computation 11th International Conference, ISAAC 2000 Taipei, Taiwan, December 18–20, 2000Algorithms and Computation , ISAAC 2000 Taipei, Taiwan, December 18–20, 2000 (Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, D. T. Lee, and Shang-Hua Teng, eds.), Lecture Notes in Computer Science (LNCS), vol. 1969, 2000.

[DSV01] ______, **On Efficient Sparse Integers Matrix Smith Normal Form Computations**, Journal of Symbolic Computation **32** (2001), no. 1-2, 71—-99.

[Dum11] Neil Dummingan, **A $U(2,2)$-Analogue of Harder's Conjecture**, http://neil-dummigan.staff.shef.ac.uk/papers.html, November 2011.

[EVGS02] Philippe Elbaz-Vincent, Herbert Gangl, and Christophe Soulé, **Quelques calculs de la cohomologie de $\mathrm{GL}_n(\mathbb{Z})$ et de la K-théorie de $\mathbb{Z}$**, CRAS Acad. Sc. Paris **335** (2002), 321—-324.

[FH91] William Fulton and Joe Harris, **Representation theory - a first course**, Graduate Texts in Mathematics (GTM), no. 129, Springer-Verlag, 1991.

[Fra98] Jens Franke, **Harmonic Analysis in Weighted L_2-Spaces**, Annales scientifiques de l'École Norm. Sup. S. 4^e série **31** (1998), no. 2, 181—-279.

[Fre83] Eberhard Freitag, **Siegelsche Modulfunktionen**, Grundlehren der mathematischen Wissenschaften, no. 254, Springer-Verlag, 1983.

[Fre91] ______, **Singular modular forms and theta relations**, Lecture Notes in Mathematics (LNM), vol. 1487, Springer-Verlag, 1991.

[FS98] Jens Franke and Joachim Schwermer, **A Decomposition of Spaces of Automorphic Forms, and Eisenstein Cohomology of Arithmetic Groups**, Math. Ann. **311** (1998), no. 4, 765—-790.

[Fuj88] Akira Fujiki, **Finite automorphism groups of complex tori of dimension two**, Publ. R.I.M.S. Kyoto Univ. **24** (1988), no. 1, 1–97.

[FvdG04a] Carel Faber and Gerard van der Geer, **Sur la Cohomologie des Systémes Locaux sur les Espaces des Modules des Modules des Courbes de Genus 2 è des Surfaces Abéliennes I**, Comptes Rendus Mathematique Acad. Sci. Paris **338** (2004), no. 5, 381—-384.

[FvdG04b] ______, **Sur la Cohomologie des Systémes Locaux sur les Espaces des Modules des Modules des Courbes de Genus 2 è des Surfaces Abéliennes II**, Comptes Rendus Mathematique Acad. Sci. Paris **338** (2004), no. 6, 467—-470.

[Gau01] Carl Friedrich Gauß, **Disquisitiones arithmeticae**, Gerhard Fleischer, Lipsiae (Leipzig), 1801.

[GM83] Mark Goresky and Robert MacPherson, **Intersection homology II**, Invent. Math. **71** (1983), no. no. 1, 77—-129.

[GMG01] Víctor González-Aguilera, Jose M. Muñoz-Porras, and Alexis García-Zamora, **On irreducible components of the singular locus A_g**, J. Algebra **240** (2001), 230—-250.

[GMG04] ______, **Some recent results on the irreducible components of the singular locus of A_g**, The geometry of Riemann surfaces and abelian varieties (Providence, R.I.), Contemp. Math., no. 397, American Mathematical Society, 2004, pp. 119–125.

[GMG05] ______, **On the 0-dimensional irreducible components of the singular locus of A_g**, Arch.Math. (2005), 298—-303.

[Gon09] Víctor González-Aguilera, **On automorphisms of principally polarized abelian varieties**, Geom. Dedicata **139** (2009), 249—-258.

[Got59] Erhard Gottschling, **Explizite Bestimmung der Randflächen des Fundamentalbereiches der Modulgruppe zweiten Grades**, Math. Ann. (1959), no. 138, 103—-124.

[GR99] Víctor González-Aguilera and Rubí E. Rodríguez, **Families of irreducible principally polarized abelian varieties isomorphic to a product of elliptic curves**, Proc. Amer. Math. Soc. **128** (1999), no. 3, 629—-636.

[Gro57a] Alexander Grothendieck, **Sur quelques points d'algèbre homologique i**, Tôhoku Math. J. **9** (1957), no. 2, 119—-183.

[Gro57b] ______, **Sur quelques points d'algèbre homologique ii**, Tôhoku Math. J. **9** (1957), no. 3, 185–221.

[GRS12] Alexandru Ghitza, Nathan C. Ryan, and David Sulon, **Further verification of Harder's conjecture**, http://arxiv.org/abs/1203.5611, March 2012.

[Gru07] Peter M. Gruber, **Convex and discrete geometry**, Grundlehren der Mathematischen Wissenschaften, no. 336, Springer-Verlag, 2007.

[Gru09] ______, **Geometry of cone of positive quadratic forms**, Forum Math. **21** (2009), no. 1, 147–166.

[Gun00] Paul E. Gunnells, **Symplectic modular symbols**, Duke Math. J. **102** (2000), no. 2, 329—-350.

[Gun06] ______, **Robert MacPherson and Arithmetic Groups**, Pure and Applied Math. Quarterly **2** (2006), no. 4, 1015—-1052.

[Gun07] ______, **Computing in Higher Rank - Appendix in Modular Forms a Computational Approach by W. Stein**, Graduate Studies in Mathematics, vol. 79, ch. Appendix, pp. 203—-252, American Mathematical Society, Providence, R.I., 2007.

[Gun09] ______, **On the Cohomology of Congruence Subgroups of $\mathbf{SL}_4(\mathbb{Z})$**, Proceedings of 2009 RIMS Conference Automorphic representations, automorphic L-functions and arith- metic, 2009, arXiv:0905.0401v1 [math.NT].

[GW98] Roe Goodman and Nolan R. Wallach, **Representations and Invariants of the Classical Groups**, Encyclopedia of Mathematics and its Applications, vol. 68, Cambridge University Press, 1998.

[GW09] ______, **Symmetry, representation, and invariants**, Graduate Texts in Mathematics (GTM), no. 255, Springer-Verlag, 2009.

[Har75] Günter Harder, **On the cohomology of discrete arithmetically defined groups**, Discrete subgroups of Lie groups and applications to moduli (Internat. Colloq., Bombay, 1973) (Bombay), Oxford Univ. Press, 1975, pp. 129—-160.

[Har87] ______, **Eisenstein cohomology of arithmetic groups. the case $\mathbf{Gl}_2$**, Invent. Math. **89** (1987), no. 1, 37—-118.

[Har93] ______, **Eisensteinkohomologie und die Konstruktion gemischter Motive**, Lecture Notes in Mathematics (LNM), vol. 1562, Springer-Verlag, 1993.

[Har03] ______, **A Congruence between Siegel and Elliptic Modular Forms**, Colloquium in Bonn, February 2003.

[Har05] ______, **Kohomologie Arithmetischer Gruppe**, Vorlesungsskript Universität Bonn, 2005.

[Har08a] ———, **Cohomology of Arithmetic Groups**, Lecture notes for the lecture held in summer term 2008, available at `http://www.math.uni-bonn.de/people/harder/`, June 2008.

[Har08b] ———, **A congruence between a siegel and an elliptic modular form**, in Ranestad [Ran08], pp. 247—-262.

[Har08c] ———, **Lectures on Algebraic Geometry I - Sheaves, Cohomology of Sheaves, and Applications to Riemann Surfaces**, 1st ed., Aspects of Mathematics, no. E 35, Vieweg, 2008.

[Hat02] Allen Hatcher, **Algebraic topology**, Cambridge University Press, 2002.

[Hau06] Felix Hausdorff, **Die symbolische Exponentialformel in der Gruppentheorie**, Ber. über die Verhandlungen der Königl. Sächs. Ges. der Wiss. zu Leipzig. Math.-phys. Klasse (1906), no. 58, 19—-48.

[Hel78] Sigurdur Helgason, **Differential Geometry, Lie Groups, and Symmetric Spaces**, Academic Press, 1978.

[HN65] Tsuyoshi Hayashida and Mieo Nishi, **Existence of curves of genus two on product of two elliptic curves**, J. Math. Soc. Japan **Vol. 17** (1965), no. 1, 1–16.

[Hum78] James E. Humphreys, **Introduction to Lie Algebras and Representation Theory**, 2nd ed., Graduate Texts in Mathematics (GTM), no. 9, Springer-Verlag, 1978.

[HW98] J. William Hoffman and Steven H. Weintraub, **Cohomology of Siegel Modular Group of Degree Two and Level Four**, Mem. Amer. Math. Soc. **133** (1998), no. 631, 59—-75.

[HW01] ———, **The siegel modular variety of degree two and level three**, Trans. Amer. Math. Soc. **353** (2001), no. 8, 3267—-3305.

[HW03] ———, **Cohomology of the boundary of Siegel modular varieties of degree two, with applications**, Fund. Math. **178** (2003), no. 1, 1—-47.

[Ibu08] Tomoyoshi Ibukiyama, **A conjecture on a Shimura type correspondence for Siegel modular forms, and Harder's conjecture on congruences**, Modular Forms on Schiermonnikoog (Bas Edixhoven, Gerard van der Geer, and Ben Moonen, eds.), Cambridge University Press, 2008.

[Ive86] Birger Iversen, **Cohomology of sheaves**, Universitext, Springer-Verlag, 1986.

[Kas75] Masaki Kashiwara, **On the maximally overdetermined systems of linear differential equastions I.**, Publ. R.I.M.S. Kyoto Univ. (1975), no. 40, 563—-579.

[KK07] Max Koecher and Aloys Krieg, **Elliptische Funktionen und Modulformen**, 2nd ed., Springer-Verlag, 2007.

[Kli90] Helmut Klingen, **Introductory Lectures on Siegel Modular Forms**, Cambridge Studies in Advanced Mathematics, no. 20, Cambridge University Press, 1990.

[KMM04] Tomasz Kaczynski, Konstantin Mischaikow, and Marian Mrozek, **Computational**

Homology, Applied Mathematical Science, vol. 157, Springer-Verlag, 2004.

[Kop00] Helmut Kopka, **LaTeX- Einführung Band 1**, Addison Wesley, 2000.

[Kop02] ———, **LaTeX- Einführung Band 2: Ergänzungen**, Addison Wesley, 2002.

[Kos66] Bertram Kostant, **Groups over Z**, Proc. of the Symp. in Pure Mathematics of the American Mathematical Society, held at the University of Colorado, Boulder, Colorado, July 5–August 6 1965 (Providence, R.I.) (Armand Borel and George D. Mostow, eds.), Proc. of Symp. Pure. Math., vol. IX, American Mathematical Society, 1966, pp. 90—-98.

[KS90] Masaki Kashiwara and Pierre Schapira, **Sheaves on manifolds**, Grundlehren der mathematischen Wissenschaften, no. 292, Springer-Verlag, 1990.

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra Jr., and László Lovász, **Factoring polynomials with rational coefficients**, Math. Ann. **261** (1982), no. 4, 515—-534.

[LR04a] Herbert Lange and Sevin Recillas, **Abelian Varieties with Group Action**, J. Reine Angew. Math. (Crelles Journal) **575** (2004), 135—-155.

[LR04b] ———, **Poincaré's reducibility theorem with g-action**, Bol. Soc. Mat. Mexicana (3) **10** (2004), no. 1, 43—-48.

[LS78] Ronnie Lee and Robert Henry Szczarba, **On the torsion in $K_4(\mathbb{Z})$ and $K_5(\mathbb{Z})$**, Duke Math. J. **45** (1978), no. 1, 101—-129.

[LS09] Jian-Shu Li and Joachim Schwermer, **On the cuspidal cohomology of arithmetic groups**, Amer. J. Math. **131** (2009), 1431—-1464.

[LW85] Ronnie Lee and Steven H. Weintraub, **Cohomology of $\mathrm{Sp}_4(\mathbb{Z})$ and related groups and spaces**, Topology **24** (1985), no. 4, 391—-410.

[LW98] ———, **The Siegel modular variety of degree two and level four**, Mem. Amer. Math. Soc. **133** (1998), no. 631, 1—-58.

[Mac] **The mactutor history of mathematics archive**, `http://www-history.mcs.st-andrews.ac.uk/`.

[Mac98] Saunders MacLane, **Categories for the working mathematician**, 2nd ed., Graduate Texts in Mathematics (GTM), no. 5, Springer-Verlag, 1998.

[Mas78] William S. Massey, **Homology and Cohomology Theory**, Pure and Applied Mathematics, vol. 46, Mercel Dekker, Inc., 1978.

[Mas91] ———, **A Basic Course in Algebraic Topology**, Graduate Texts in Mathematics (GTM), no. 127, Springer-Verlag, 1991.

[McC91] Mark McConnell, **Classical projective geometry and arithmetic groups**, Math. Ann. **290** (1991), 441—-462.

[McC98] ———, **Generalisations of Voronoï I to Symmetric Spaces and Arithmetic Groups**, Voronoï's Impact on Modern Science - Book I (P. Engel and H. Syta, eds.), Mathematics and its Application, Proceedings of the Institute of Mathematics of the National Academy of Science of Ukraine, vol. 21, National Academy of Science of Ukraine Institute of Mathematics, Kyiv, Ukraine, 1998.

[McC99] ______, **Cell Decompositions of Satake Compactifications**, Publ. Inst. Math. (Beograd) (N.S.) **66** (1999), no. 80, 46—-90.

[Min05] Hermann Minkowski, **Diskontinuitätsbereich für arithmetische Äquivalenz**, J. Reine Angew. Math. (Crelles Journal) **129** (1905), 220—-274.

[MM89] Robert MacPherson and Mark McConnell, **Classical projective geometry and modular varieties**, Algebraic Analysis, Geometry, and Number Theory - Proceedings of the JAMI Inaugural Conference, Baltimore 1988 (Baltimore) (Jun-Ichi Igusa, ed.), Johns Hopkins Univ. Press, 1989, pp. 237—-290.

[MM93] ______, **Explicit Reduction Theory for Siegel Modular Threefolds**, Invent. Math. **111** (1993), 575—-625.

[Mum70] David Mumford, **Abelian varieties**, Oxford Univ. Press, 1970.

[Mun84] James R. Munkres, **Elements of algebraic topology**, Addison Wesley, 1984.

[Nam80] Yukihiko Namikawa, **Toroidal Compactification of Siegel Spaces**, Lecture Notes in Mathematics (LNM), vol. 812, Springer-Verlag, 1980.

[OS90] Takayuki Oda and Joachim Schwermer, **Mixed hodge structures and automorphic forms for siegel modular varieties of degree two**, Math. Ann. **286** (1990), no. 1-3, 481–509.

[PAFL06] Samuel Peltier, Sylvie Alayrangues, Laurent Fuchs, and Jacques-Olivier Lachaud, **Computation of homology groups and generators**, Computers & Graphics **30** (2006), no. 1, 62—-69.

[Poi00] Henri Poincarè, **Sur les groupes continus**, Trans. Cambr. Philos. Soc. **18** (1900), 220—-225.

[Pyt] **Python**, `http://www.python.org`.

[Rag67] Madabusi S. Raghunathan, **Cohomology of Arithmetic Subgroups of Algebraic Groups: I**, Ann. of Math. **86** (1967), no. 3, 409—-424.

[Rag68a] ______, **Cohomology of Arithmetic Subgroups of Algebraic Groups: II**, Ann. of Math. **87** (1968), no. 2, 279—-304.

[Rag68b] ______, **A Note on Quotients of Real Algebraic Groups by Arithmetic Subgroups**, Invent. Math. **4** (1968), 318—-335.

[Ram16] Srinivasa Ramanujan, **On certain arithmetical functions**, Trans. Cambridge Philos. Soc. **22** (1916), no. 9, 159–184.

[Ran08] Kristian Ranestad (ed.), **1-2-3 Modular Forms - Lectures at a Summer School in Nordfjordeid, Norway**, Universitext, Springer-Verlag, 2008.

[Roa90] Shi-Shyr Roan, **Complex 3-Tori with an Order 7 Special Automorphism**, Unpublished Preprint, 1990.

[Sat60a] Ichirō Satake, **On Compactifications of the Quotients Spaces for Arithmetically Defined Discontinuous Groups**, Ann. of Math. **72** (1960), 555—-580.

[Sat60b] ______, **On representations and compactifications of symmetric Riemannian spaces**, Ann. of Math. **71** (1960), 77—-110.

[Sch83] Joachim Schwermer, **Kohomologie arithmetisch definierter Gruppen und Eisensteinreihen**, Lecture Notes in Mathematics (LNM), vol. 988, Springer-Verlag, 1983.

[Sch86] ______, **On arithmetic quotients of the Siegel upper half space of degree two**, Compos. Math. (1986), no. 58, 233—-258.

[Sch94] ______, **Eisenstein series and the cohomology of arithmetic groups: The generic case**, Invent. Math. **116** (1994), 481—-511.

[Sch97] Dieter Schmidt, **Automorphismen 2- und 3-dimensionaler abelscher Varietäten**, Ph.D. thesis, Friedrich-Alexander Universität Erlangen-Nürnberg, 1997.

[Sch03] Jörg Schürmann, **Topology of singular spaces and constructible sheaves**, Monografie Matematycze, New Series, vol. 63, Birkhäuser Verlag, 2003.

[Sch07] Achill Schürmann, **Computational geometry of positive definite quadratic forms, theory, algorithms, applications (Habilitationschrift)**, August 2007.

[Sch09] ______, **Computational geometry of positive definite quadratic forms: Polyhedral reduction theories, algorithms, and applications**, University Lecture Series, vol. 48, American Mathematical Society, Providence, R.I., 2009.

[Sch10] Joachim Schwermer, **Geometric Cycles, Arithmetic Groups and their Cohomology**, Bull. Amer. Math. Soc. (N.S.) **47** (2010), no. 2, 187—-279.

[Ser71] Jean-Pierre Serre, **Cohomologie des groupes discrete**, Prospects in Math. Ann. Math. Study (1971), no. 70, 77—-169.

[Ser73] ______, **A course in arithmetic**, Graduate Texts in Mathematics (GTM), no. 7, Springer-Verlag, 1973.

[Ser79] ______, **Arithmetic groups**, Homological Group Theory (C.T.C. Wall, ed.), LMS Lecture Notes Series, no. 36, Cambridge University Press, 1979, pp. 105—-136.

[Shi63] Goro Shimura, **An analytic families of polarized abelian varieties and automorphic functions**, Ann. of Math. **78** (1963), no. 1, 149—-192.

[Shi71] ______, **On the Zeta-Function of Abelian Varieties with Complex Multiplication**, Ann. of Math. **94** (1971), no. 3, 504—-533.

[Shi98] ______, **Abelian varieties with complex multiplication and modular functions**, Princeton University Press, 1998.

[Sie43] Carl Ludwig Siegel, **Symplectic Geometry**, Amer. J. Math. **65** (1943), no. 1, 1—-86.

[Sie55] ______, **Zur Theorie der Modulfunktionen n-ten Grades**, Comm. Pure and

Appl. Math. (1955), no. 8, 677—-681.

[SL01] Joachim Schwermer and Jian-Shu Li, **Automorphic Representations and Cohomology of Arithmetic Groups**, Challanges for the 21st Century - Papers from the international conference on fundamental science: mathematics and theoretical physics (ICFS 2000), Singapore, March 13–17 2000 (Louis H.Y. Chen et al., eds.), World Scientific, 2001, pp. 102—-137.

[Smi61] Henry J. Stephen Smith, **On systems of linear indeterminate equations and congruences**, Phil. Trans. R. Soc. Lond. **151** (1861), no. 1, 293——326.

[Sou78] Christophe Soulé, **Cohomologie de** $\mathrm{SL}_3(\mathbb{Z})$, Topology **17** (1978), 1—-22.

[Sou07] ———, **An introduction to arithmetic groups**, Frontiers in Number Theory, Physics and Geometry II - On Conformal Field Theories Discrete Groups and Renormalisation (Pierre Cartier, Bernard Julia, Pierre Moussa, and Pierre Vanhove, eds.), Springer-Verlag, 2007, arXiv:math.GR/0403390v1.

[Spr08] Tonny Albert Springer, **Linear algebraic groups**, reprint of the 1998 sencond edition ed., Modern Birkhäuser Classics, Birkhäuser Verlag, 2008.

[Ste07] William Stein, **Modular forms a computational approache**, Graduate Studies in Mathematics, vol. 79, American Mathematical Society, Providence, R.I., 2007.

[Ste11] ———, **Sage: Open Source Mathematical Software (Version 4.6.1)**, The Sage Group, 2011, `http://www.sagemath.org`.

[Str69] Volker Strassen, **Gaussian Elimination is not Optimal**, Numerische Mathematik (1969), no. 13, 354—-356.

[Str10] Marco Streng, **Complex multiplication of abelian surfaces**, Ph.D. thesis, Universiteit Leiden, 2010.

[Tea11a] The Sage Development Team, **Sage developer guide - release 4.7**, June 2011, `http://www.sagemath.org`.

[Tea11b] ———, **Sage reference manual - release 4.7**, June 2011, `http://www.sagemath.org`.

[The08] The GAP Group, **GAP - Reference Manual**, release 4.4.12 ed., December 2008.

[Vas07] Oleg Nikolaevich Vasilenko, **Number-theoretic algorithms in cryptography**, Translations of Mathematical Monographs, vol. 232, American Mathematical Society, Providence, R.I., 2007.

[vdG08] Gerard van der Geer, **Siegel modular forms and their applications**, in Ranestad [Ran08], pp. 181—-246.

[vG03] Joachim von zur Gathen and Jürgen Gerhard, **Modern computer algebra**, 2nd ed., Cambridge University Press, 2003.

[Vor08a] Georges Voronoï, **Nouvelles applications des paramètres continus à la théorie de formes quadratique - deuxième mémoire - recherches sur les paralléloèdres**

primitif - premier partie, J. Reine Angew. Math. (Crelles Journal) **134** (1908), 198—-287.

[Vor08b] ———, **Nouvelles applications des paramètres continus à la théorie de formes quadratique - premier mémoir - sur quelques propriétés des form quadratiques positives parfaites**, J. Reine Angew. Math. (Crelles Journal) **133** (1908), 97—-178.

[Vor09] ———, **Nouvelles applications des paramètres continus à la théorie de formes quadratique - deuxième mémoire - recherches sur les paralléloèdres primitif - seconde partie - domaines de formes quadratiques correspondant aux différents types de paralléloèdres primitifs,**, J. Reine Angew. Math. (Crelles Journal) **136** (1909), no. 2, 67—-181.

[Š74] Mikhail I. Štogrin, **Locally quasi-densest lattice packings of spheres**, Sov. Math. Dokl. **15** (1974), 1288—-1292.

[Wei88] Rainer Weissauer, **On the cohomology of Siegel modular threefolds**, Arithmetic of Complex Manifolds - Proceedings of a Conference Held in Erlangen, FRG, May 27-31, 1988 (Wolf-P. Barth and Herbert Lange, eds.), Lecture Notes in Mathematics (LNM), vol. 1399, Springer-Verlag, 1988, pp. 155—-170.

[Wei08] Michael Weigend, **Python**, 4th ed., GE-PACKT Referenzen, mitp, 2008.

[Wey46] Hermann Weyl, **The Classical Groups - Their Invariants and Representations**, 2nd ed., Princeton University Press, 1946.

[Wit37] Ernst Witt, **Treue Darstellung Liescher Ringe**, J. Reine Angew. Math. (Crelles Journal) **177** (1937), 152—-160.

[Yas05] Dan Yasaki, **On the existence of spines for $\mathbb{Q}$-rank 1 groups**, Ph.D. thesis, Department of Mathematics Duke University, 2005.

[Yas07a] ———, **An explicit spine for the Picard modular group over the gaussian integers**, J. Number Theory **128** (2007), no. 1, 207—-234.

[Yas07b] ———, **On the existence of spines for $\mathbb{Q}$-rank 1 groups**, Selecta Math. (N.S.) **12** (2007), no. 3–4, 541—-564.

[Yas10] ———, **Integral cohomology of certain Picard modular surfaces**, submitted (2010).

Symbols

Index of Persons

Index