A SYMMETRY-GROUP SEMANTICS FOR SHAPE GRAMMARS

Lattice Methods for the Classification of Strand-based Patterns

ERIC HORTOP

A THESIS
IN
THE DEPARTMENT
OF
MATHEMATICS AND STATISTICS

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF SCIENCE
CONCORDIA UNIVERSITY
MONTRÉAL, QUÉBEC, CANADA

AUGUST 2007

Your file Votre référence
ISBN: 978-0-494-34696-9
Our file Notre référence
ISBN: 978-0-494-34696-9

Abstract

A SYMMETRY-GROUP SEMANTICS FOR SHAPE GRAMMARS

Lattice Methods for the Classification of Strand-based Patterns

Eric Hortop

In ascribing information to designs, two of the purposes a researcher could have in mind are to *classify* designs, grouping them with similar designs and highlighting specific characteristics, and to *store* information for reproducing them. The notations and methods for these two tasks are likely to be different.

In the *Generative Design Project*, researchers have undertaken classification of two-dimensional repeating patterns using *symmetry groups*, and storage for reproduction using a restricted form of *shape grammars*. This thesis demonstrates an approach to ascribing symmetry groups to shape grammars, allowing a researcher to concentrate their efforts on representing an artefact or collection, while retaining the benefits of a standard classification.

This work defines a class of strand-based repeating patterns to which this approach is suited, outlines shape grammars in general and in a restricted context-free form, It also defines the symmetry information which is the output of the process starting with broad categories of isometric groups and working down to the information required to fully describe the symmetries of interest.

The thesis also provides set, scalar and vector lattices used in the approach, and provides a novel, consistent method for carrying out the assignment of groups to grammars using those lattices and additional tests to bound the symmetries possible in the language of a given grammar conforming to the restrictions discussed. By establishing a symmetry-group semantics for shape grammars, this thesis allows a pattern or family of patterns to be more fully described by its shape grammar without requiring separate rendering and classification work.

Acknowledgments

This thesis would not have happened if it weren't for help, support and inspiration from many people.

I'd like to thank Dr. Fred Szabo, my supervisor, who kept me on track, helped focus my work and keep my plans realistic, and has lent support in myriad ways throughout my time at Concordia. He encouraged me to become a co-op student in 2001, vouched for me on many occasions, involved me as a teaching assistant in his courses, and listened to various worries, problems and hare-brained schemes of mine. His help, encouragement and gentle but firm insistence on regular milestones are probably the major factors in this document being ready in 2007 rather than early 2008.

Dr. Cheryl Dudek and Dr. Lydia Sharman started what was to become the Generative Design project back in 2002, and took a chance on hiring a young, undergraduate me to take some tentative steps into computerized grammars. The project provides a context for the work in this thesis, and has been a space for me to learn, teach and develop a lot, in addition to being work that has allowed me to pursue my studies. Thanks especially to Cheryl for co-ordinating the undertaking and for many good hours of working together on technological things, writing and the arts and infrastructure aspects of the project. Thanks also Dr. Sudhir Mudur and Dr. Thomas Fevens for bringing their expertise to the project and thus exposing me to a variety of perspectives on the challenges of working with shape grammars and images. I'd like to extend thanks to Drs. Szabo, Dudek and Mudur, as well as to committee chair Dr. Hal Proppe, for taking the time to evaluate this thesis and sit on my committee, and to Ann-Marie Agnew, the Mathematics and Statistics department's graduate secretary, for kindness, encouragement and guidance with the various administrative processes of graduate life.

Ramgopal Rajagopalan, with GUI help from Yojana Joshi and further significant refinements from Nasim Sedag, wrote the grammar interpreter and design system that allowed me to explore grammar creation and modification much faster than manual methods would permit. Dania El-Khechen helped think through some important issues while we worked on the paper that sparked this thesis. Ramgopal and Dania were also excellent comrades in arms when arguing out what symmetry group to assign to real-world artefacts, contributing greatly to my understanding of and confidence with them. Dr. Ivo Rosenberg at Université de Montréal introduced me to the lattice and order theory with which much of the heavy lifting in this thesis is done.

The open source community, in particular the contributors to *Inskcape*, *GraphViz* and LaTeX (including the Mac front end *TeXShop*), have created powerful tools whose features have made writing, illustrating and organizing this thesis considerably easier and often more fun.

Of course, one of the most original parts of any single-author document is the collection of mistakes, omissions, hand-waving and other problems. These are my problem alone.

I would like to thank Ian Rennie, my principal for stretches of both elementary and high school, for sharing his enthusiasm for computing, geometry and mathematics with me and a gaggle of computer clubbers after hours at Ormstown Elementary School; Mary-Ellen O'Neil and Gregg Edwards for coaching the Intellectual Olympics team at Chateauguay Valley Regional and Rick Laverty and Hugh Maynard for guiding and trusting a bunch of geeky youth to undertake web design projects for SchoolNet, and giving us the opportunities to learn and solve problems for ourselves. All of these educators went out of their way to expose me, and many others, to exciting ideas and projects.

I would also like to thank Kevin Murphy, formerly of Statistics Canada, for patiently guiding me through writing my first thesis-sized document in 2002. When writing this thesis seemed impossible, the co-op term spent researching, hacking and documenting under his supervision was a happy counterexample.

My parents, John Hortop and Lynne Stockwell, have supported and encouraged me through my entire (somewhat circuitous) educational journey, encouraging me to become well-rounded and giving me a good start in life. My fiancée Elizabeth Bruce has been there, not only for the final copy edit, but also for almost all my graduate career, listening, encouraging, taking up the slack when I've been engrossed in abstract things and bringing happiness into my life just by being herself.

Contents

List of Figures

List of Tables

List of Notation

$\langle a, b, c, \ldots \rangle$	an ordered n-tuple
$\{a, b, c, \ldots\}$	a set
(a, b)	a point on the plane
$a \to b$	a shape grammar rule taking a (the right-hand side) and replacing it with b (the left-hand side)
$\varnothing$	the empty set (whether in text or in a grammar illustration)
$a \in A$	set membership: a is a member of A
$\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$	the sets of natural numbers (not including zero), integers, rational numbers and real numbers, respectively
$c_n, d_2 n$	cyclic and dihedral groups of order n and $2n$, respectively
$A \rtimes B$	the semidirect product of A and B
$a \iff b$	a if and only if b
$m \mid n$	m divides n
$\vec{a}$	the vector $\vec{a}$
$b \circ a$	b follows a (the composition of a and b)
$a \vee b$	the join of a and b
$a \wedge b$	the meet of a and b
A^{∂}	the dual of A
$\top$	the top element of a lattice
$\bot$	the bottom element of a lattice
$\forall x$	for all x
$\exists x$	there exists x such that
$\mathfrak{L}$	a lattice of sets
$\wp(X)$	power set, i.e. the set of all subsets of X
$a \preceq b$	a is less than or equal to b in a particular ordering
$a \preccurlyeq b$	a is less than or equal to b in a particular ordering
$a \prec b$	a is less than b in a particular ordering
$a \| b$	a is perpendicular to b
$\angle(\vec{a}, \vec{b})$	the angle between vectors $\vec{a}$ and $\vec{b}$, in radians
$\text{proj}\left(\vec{a}, \vec{b}\right)$	the projection of $\vec{a}$ onto $\vec{b}$
$\|\|\vec{a}\|\|$	the Euclidean length of $\vec{a}$

Chapter 1

Introduction

Art elicits questions and invites stories. In a representational piece, a viewer might want to know what the objects in the frame represent, how the treatment or choice of the subject reflects the artist's life and times, or how it can be described in any number of critical frameworks. In a non-representational piece, there are no objects except the piece: questions of feeling, context and history arise, but also one of particular interest from a mathematical point of view: how is the piece organized? What rules, if any, underlie it?

A viewer might approach that question by noticing how elements fit together: spacing, orientation and shape. Observations might be translated into relationships between pairs of elements, or sequences of steps for building the pattern, or both.

This might lead to a question of what those relationships allow, maybe in particular what symmetries are allowed by or required by the organization of the piece, maybe even what structure the piece would have if it extended infinitely beyond the frame it was given.

A desirable tool would take instructions for re-creating a pattern and return some information on that pattern. This thesis outlines a specific type of instructions, a subset of shape grammars, and a process for extracting a symmetry classification. The approach starts with the basic elements of the instructions, symbols and rules, and works up to strands, strips, tilings and then the whole pattern. This process requires ascribing symmetry information to elements of the design and, taking classifications and parameters into account, using operations on an infinite lattice to find the symmetry information of their combination. This tool could be useful in classifying designs, in

tracking the symmetry effects of changes to a grammar, or in aiding the creation of group-enriched shape grammars [9, p. 50]. In essence, this thesis adds a rendering-free semantics to shape grammars, which allows the reading of symmetry information directly from grammar rules and their interrelations.

In this thesis, the instructions for recreating a family of similar patterns are encoded in a *shape grammar*, a visual representation of how to build up designs using replacement rules, a concept borrowed from linguistics and computer science. The symmetry information sought is expressed using *symmetry groups*, which are groups of transformations which map designs to themselves, and organized using lattice and order theory. Finding the symmetry of a design is thus reduced to finding parts of a grammar which correspond to symmetries, situating them on lattices, and performing simple lattice operations iteratively until they have all been taken into account. This method skips the rendering of families of images and checking of their individual symmetries by moving the problem from grammar-reading to algebra.

The particular project from which the ideas and momentum for this thesis come is the Generative Design project, an interdisciplinary endeavour to apply mathematical and computer-science approaches to decorative repeating pattern.

1.1 The Generative Design Project

The Generative Design Project grew out of work in 2002 to assign shape grammars to traditional designs and classify the designs based on mathematical characteristics. Initiated by artist Dr. Cheryl Kolak Dudek and designer Dr. Lydia Sharman, the project was soon joined by Dr. Fred Szabo, who is also my supervisor, and from Computer Science, Dr. Sudhir Mudur, Dr. Ching Y. Suen and Dr. Thomas Fevens from Computer Science.

To date, the faculty on the project and numerous students have investigated and contributed work with neural networks, shape grammars of different types, image processing techniques, historical information and tool design, and have disseminated information about the project locally and internationally.

The goals of the project include a "metapattern database" of artefacts and grammars, and creating a means for lay people to interact with real-world and synthetic patterns via grammars and

symmetry [1, 3].

My work with the project has included initial literature searches and exploratory programming, writing grammars, investigating symmetry classification schemes and taking part in the dissemination efforts. This thesis continues and connects my work on grammars and symmetry classification, and builds upon explorations of the topic undertaken in large part with another student, Dania El-Khechen [5].

1.1.1 Shape Grammar Design System and Interpreter

Along with an annotated collection of images, the Generative Design project has produced design and interpreter software for context-free shape grammars (see Section 2.2 for details on the restrictions). The interpreter, whose capabilities are more fully outlined in *Inference and Design in Bakuba and Zillij Art with Shape Grammars* [15], takes an XML file describing a grammar and a particular sequence of rule applications from it, and returns an image of the result, as in Figure 1.1.

The design system, built around the interpreter, provides tools to visually author grammars, pick sequences of rules, and view the resulting designs.

Some of the restrictions imposed on grammars in this thesis, notably the context-free stipulation, stem from the capabilities of the interpreter.

1.1.2 Sources of traditional repeating pattern for Generative Design

As a consequence of the research interests of Drs. Dudek and Sharman, I have been involved in the study of Congolese Kuba textiles and Moorish zillij mosaics, written shape grammars which aim to express them well and ascribed symmetry classifications to them. Zillij mosaics, and the "Islamic star patterns" they display, have also been well-studied by many researchers, including Grünbaum [8], Kaplan [10] and Ostromoukhov [13]. This thesis examines a grammar-to-symmetry group bridge for a class of designs chosen to include most Islamic star patterns and many designs found in Kuba cloth.

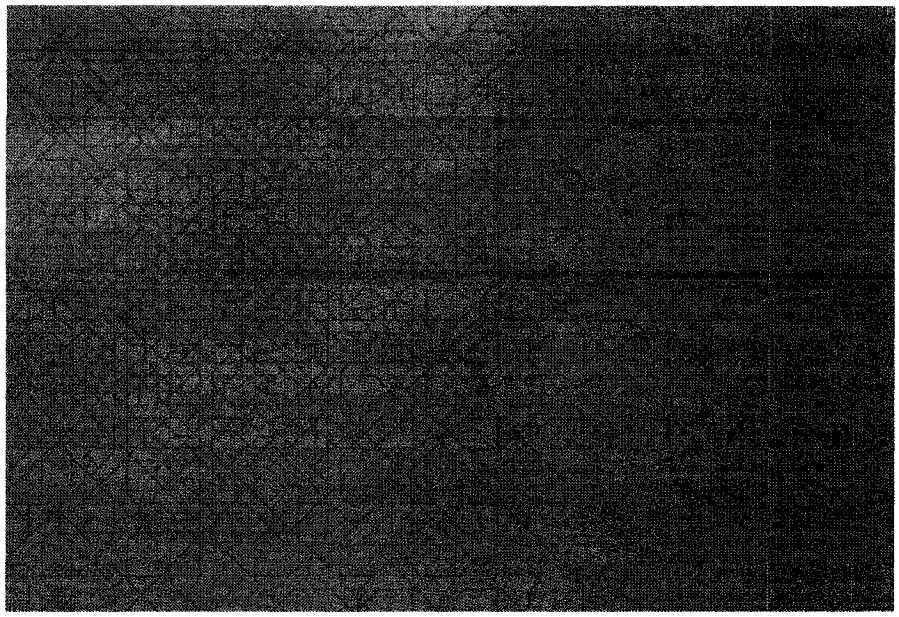

Figure 1.1: Output from the XML interpreter based on the zillij mosaic in Figure 1.3.

Kuba textiles

Kuba cloth is an embroidered and appliquéd textile on a single ply of woven raffia, from the Shoowa region of central Congo. The designs on the textile appear on one side only. Pieces are square or rectangular, typically 30–70 cm on a side. Individual pieces are used singly and in combination ceremonially, and are also sold to tourists and collectors. The abstract, rectilinear designs are typically monochromatic: natural raffia colour and dyed fibres in varying darknesses of one hue, which depends on the cloth's origin. [12]

In our collection of photographs and cloths, Kuba tends to have rough translational symmetry on one or two axes, sometimes over the whole cloth and sometimes in "sectors" with improvisations within the symmetric designs. Rotational and reflective symmetry are sometimes present as well.

Zillij mosaics

Zillij mosaics are glazed ceramic mosaics found across the areas touched by Moorish influence in the middle ages. The mosaics we study are composed of single-coloured glazed tiles organized in

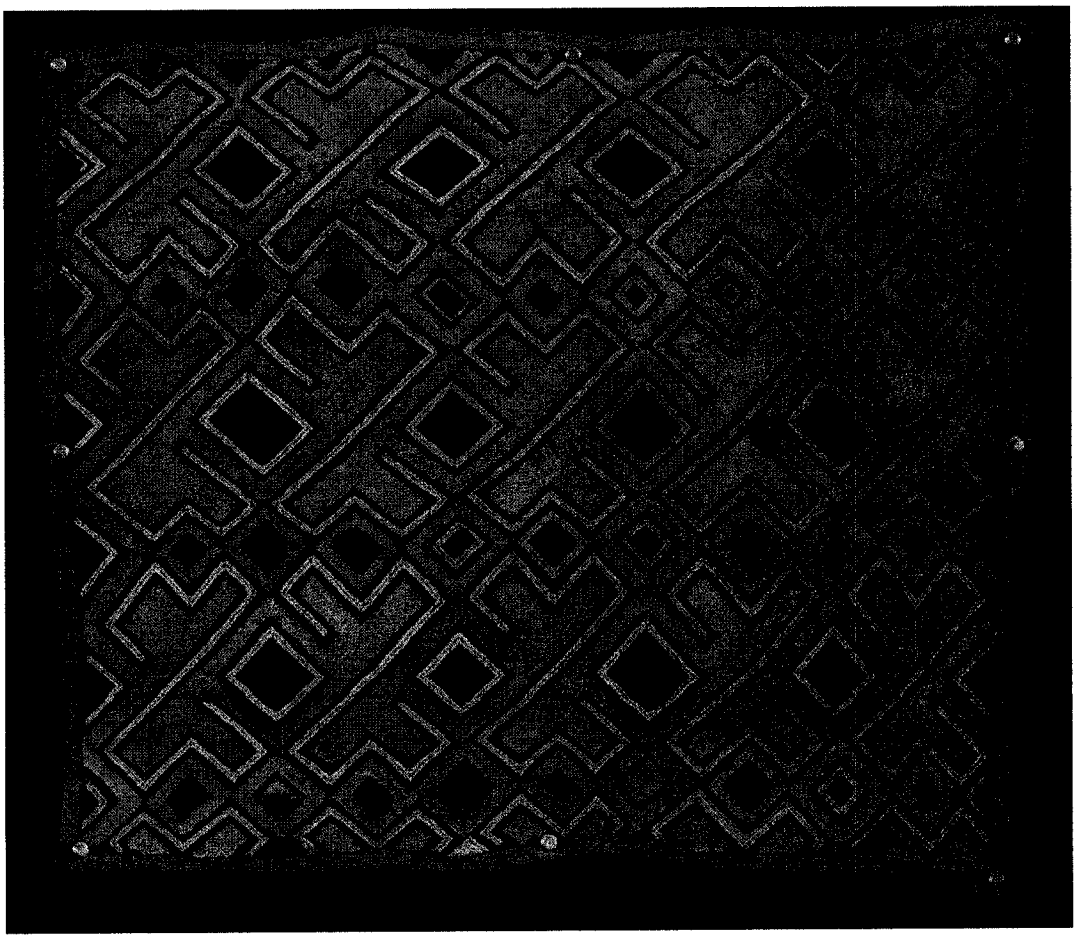

Figure 1.2: Kuba cloth from the collection of Cheryl Kolak Dudek

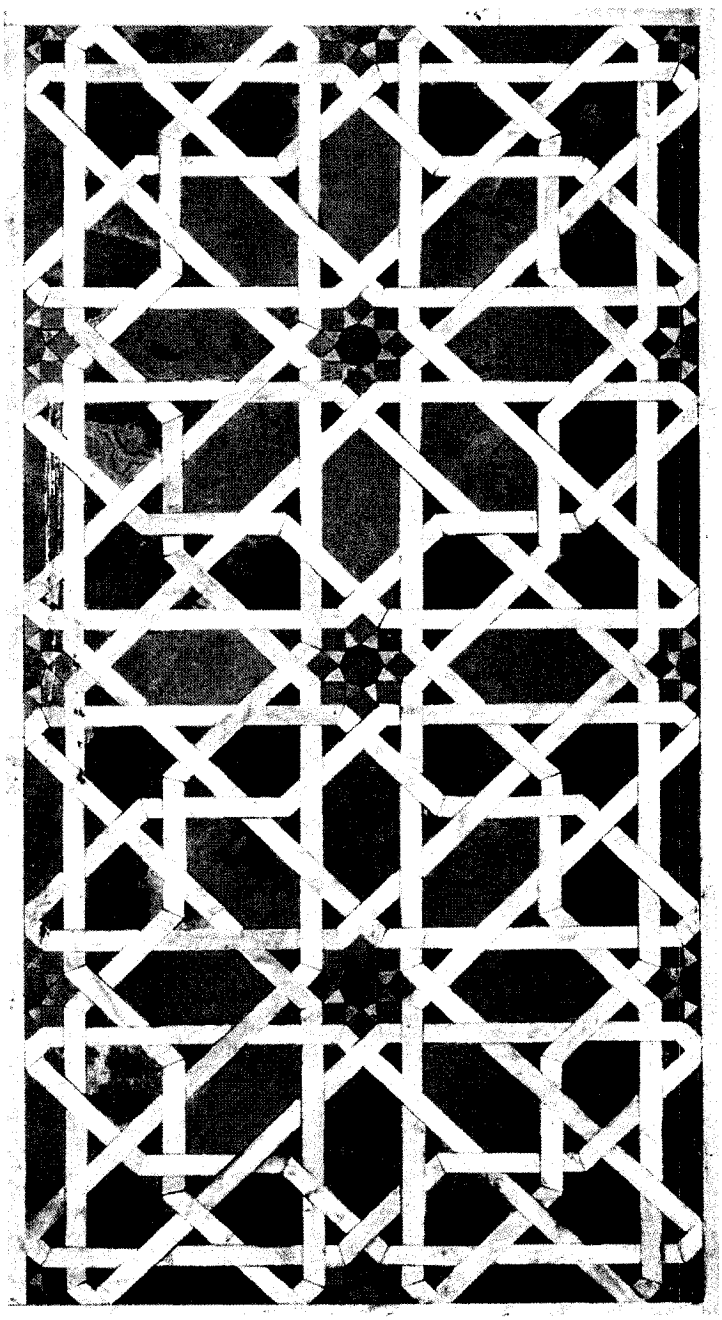

Figure 1.3: Zillij mosaic from Ummayad mosque, Damascus, Syria. Photo: Souleima Bacha.

regular patterns, typically with "strands" of narrow white tiles between non-white tiles. I follow Kaplan [10, p. 49–50] in restricting my attention to patterns with strand-like behaviour as shown by its vertices:

1. Vertices shall have degree two or four.

2. Vertices of degree four shall be "perfect crossings" in that pairs of line segments touching the vertices shall be parallel, so that the vertex could be read as two crossed line segments. Vertices of degree two can be read as bends in a strand, so the entire pattern can be read as interwoven strands.

A mosaic can have a repeat unit of tile shapes coloured different ways, so that the period of a pattern discounting colour can be shorter than the period of a pattern taking colour into account. In our work, we usually focus solely on shape, using the smaller repeat units and representing patterns as collections of strands [8].

1.2 Repeating patterns

For the purposes of this thesis, a repeating pattern is a pattern with translational symmetries. It doesn't necessarily tile the plane: its translational symmetries could all be parallel, thus resulting in a *frieze pattern*, but it can fill it in the sense that there is some finite constant R such that any point on the plane is at most R units away from the origin transformed by one of the symmetries of the pattern.

Real-world patterns on flat surfaces don't quite fit this definition: all translations move the edges of the pattern and thus don't map the pattern to itself, strictly speaking. We get around this by assuming that patterns continue off into the plane in "the obvious way," namely by assuming that at least one repeat unit is visible in the available portion of the pattern and that the rest of the pattern on the artefact is predicted by a repetition of that repeat unit under affine transformations (ideally just translations). With these fairly strict assumptions, we avoid some of the problems of how to extend an ambiguous pattern brought up in Kaplan [11]. Grünbaum and Shephard imply these assumptions in their work on interlace patterns [8, p. 147], by working with strands as

repeating, and by claiming a regularity of interweaving such that "a repeat is unlikely to have been visible in any actual ornament."

1.2.1 Strand-based patterns

A strand-based pattern is a pattern made up completely of strands, as defined in Section 1.1.2. A strand can be a repeating pattern unto itself, the definition allows for a polyline composed of a repeating pattern of segments, and so long as that polyline doesn't have any bad vertices (i.e. vertices of degree other than two or four, or non-perfect-crossing vertices of degree four), it can be both a strand and a repeating pattern. A strand-based pattern can have multiple strands, or in fact infinitely many (consider an infinite sheet of graph paper as a simple example: two countably-infinite sets of infinitely-long strands orthogonal to each other). It turns out that many Islamic star patterns of the sort found on zillij mosaics are strand-based patterns [13].

In this thesis, we will deal with grammars for strand-based patterns and for patterns that resemble strand-based patterns sufficiently to apply strand-classification and merging techniques.

1.3 From description to classification

The workflow which this thesis assumes takes an object or image, and yields a grammatical representation and symmetry information about it. Another process kept in mind is a user taking an already-written grammar, modifying it, and wanting to know what changes, if any, arise to its symmetries. The process is summarized in Figure 1.4.

The real-world image that starts the process has a design somewhere on it, and the first decision by a researcher is one which sorts out which features of the design are salient, and how to represent them. What variation of size and orientation is part of the design, and what is due to imperfections in the creation process or subsequent wear and tear? How are lines in the design going to be represented? Is a band of colour to be represented by a single line, or as its edges? Experience with the Generative Design project indicates that habits will form in the answers to these questions. In the case of the project and in this thesis, we favour representing bands as polylines rather than by their edges, and push to see connected strands as described in Section 1.2.1 wherever plausible. We

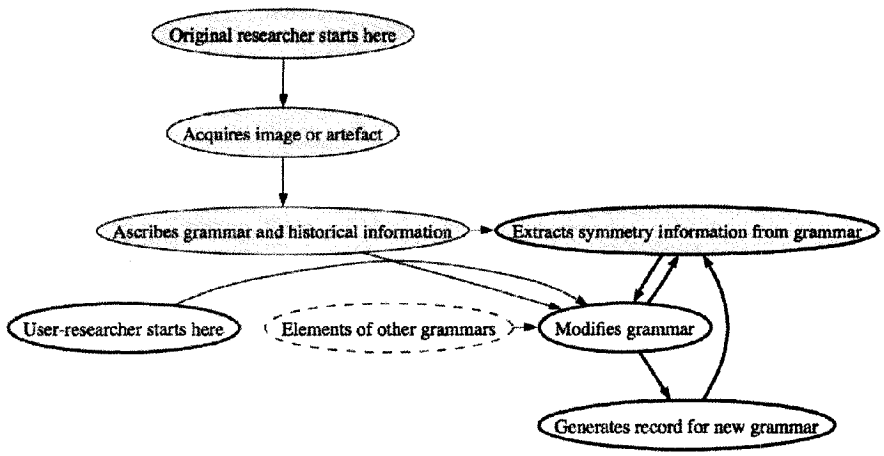

Figure 1.4: Workflow of an original researcher (grey) and a user-researcher (bold) on a given design.

assume symmetry where there are small deviations, and strive for a fairly simple extension of the design to the plane. Once the researcher has a sense of how to extend the design to the plane, the next step is writing a grammar.

From a strand-based or almost-strand-based pattern which stretches across the plane, the researcher needs to write a shape grammar. This grammar should follow the restrictions in Section 6.1, which are designed to be fairly accommodating to a strand-based design.

Next, the step where this thesis comes in: given a grammar, we follow a process for extracting symmetry information from the grammar. Searching for clues to symmetries in the grammar, we assign symmetry groups to elements of the design, locate those symmetries on a lattice using the tools in Chapter 5, and travel down the lattice, combining points with the *meet* operation, until we have a symmetry group for the whole design.

The researcher then gets back a symmetry classification for their design, adds in other information about the artefact (location, medium, artist, etc.), and adds a record to their collection.

Later on, someone finds that same record, opens up its associated grammar, and makes some changes: modifies a motif, adds or removes transformations, or changes the flow of the grammar by changing markers in various rules. Elements, such as shapes or transformations, from other grammars may be integrated into the grammar. They may generate an image of a design in the

language of the grammar, and create a record of their revised image. They then resubmit the grammar to the symmetry-assignment system, and get back a symmetry classification for the revised grammar.

Chapter 2

Shape grammars

This thesis presents a classification process, and the objects classified are not designs. They are instructions for creating designs: shape grammars. Shape grammars contain the information required to extract symmetry information about the designs they descibe (their languages).

A shape grammar [18] is a two-dimensional analogue of a phrase grammar. It is composed of some primitive symbols (shapes composed of line segments and curves) which are divided into markers and terminals, an initial state, and some replacement rules. The symbols play two roles: markers are symbols which are not present in the final design, which provide sites for the application of rules, organize the design process and represent choices in building a design. Terminals are symbols which are found in the final design. The intersection of the two sets should be empty, and furthermore, no combination of affine transformations of terminals should yield a marker: only by specific placement by a rule should a marker emerge in the design. The initial state is a set of markers and terminals from which all designs in the grammar "grow", and the replacement rules specify allowable changes in the design in terms of deleting a specific arrangement of symbols (anywhere in the design, at any scale or orientation) and replacing them with another specific arrangement (under the same transformation as the replaced arrangement).

A shape grammar can be written to include a given design, and if carefully written can express the design choices involved in creating a family of designs. Shape grammars as defined above can be arbitrarily complicated. Recognizing patterns to be replaced in an image can be very hard, computationally. Sets of rules to do the same thing can be defined many ways. As marker symbols

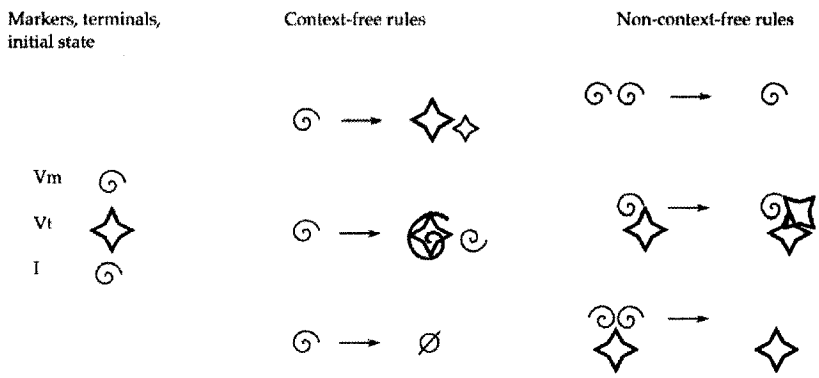

Figure 2.1: A sample grammar with context-free and non-context-free rules. Note the ways in which symbols in V_m and V_t can be transformed in grammar rules. The last rule in the context-free column simply deletes a marker (the right-hand side is the empty set, not a drawing of the empty set symbol).

will always disappear by the time the design is finished, there is often no single best place to put them in a rule, so long as rules are planned to work together. A simple sample grammar is shown in Figure 2.1.

2.1 Formal definition of shape grammars

Gips [6] and Stiny [17] define a shape grammar as a quadruple: $SG = \langle V_T, V_M, R, I \rangle$where:

V_T is a finite set of shapes. Sets of affine transformations of one or more elements of V_T, in which any element of V_T can be used multiple times, give the set V_T^* . Also we define $V_T^+ = V_T^* - \varnothing$.

V_M is a finite set of shapes defined so that the sets V_T^+ and V_M^+ (the analogous set of arrangements of a finite number of elements of V_M) are disjoint.

R is a finite set of shape rules of the form $u \rightarrow v$, where u and v are shapes formed by the shape union of shapes in V_T^* and V_M^*. The shape u must have at least one sub-shape that is a shape in V_M^+ . The shape v may be the empty shape.

I is the shape formed by the shape union of shapes in the power sets V_T^* and V_M^*. I must have at least one sub-shape that is a shape in V_M^+.

Shapes in the sets V_T^* or V_T^+ are called terminal shapes (or **terminals**). Shapes in the sets V_M^* or V_M^+ are called non-terminals (or **markers**). Terminals and markers are distinguishable, i.e., given a shape formed by the shape union of terminals and markers, the terminals occurring in the shape can be uniquely separated from the markers occurring in the shape. No shape in V_T^+ is a sub-shape of any shape in V_M^+ and vice versa. For the shape rule $u \rightarrow v$, u is called the **left side** and v the **right side** of the shape rule. The shape consisting of all the terminals in the left side of the shape rule is called the left terminal. The shape consisting of all the markers in the left side of a shape rule is called the left marker. (The left marker has at least one sub-shape that is a marker.) Right terminal and right marker are defined similarly. The shapes u and v are represented in identically-sized canvases to show the correspondence between them (in terms of their Euclidian transformations). I is called the **initial shape**. The shape consisting of all the markers in I has at least one sub-shape that is a marker. A shape is generated from a shape grammar by beginning with the initial shape and recursively applying the shape rules. The shape generation process is terminated when no shape rule in the shape grammar can be applied. The sentential set of a shape grammar, $SS(SG)$, is the set of shapes (sentential shapes) which contains the initial shape and all shapes which can be generated from the initial shape using the shape rules. The **language** of a shape grammar $L(SG)$, is the set of designs that contain only terminals. The language of a shape grammar may be finite or infinite set of shapes.

2.2 Context-free shape grammars

In the general case, a shape grammar can have any combination of markers and terminals on the left-hand side of a rule: in essence, the shape grammar can instruct the pattern builder to look for any combination of symbols in the design at each step. This general definition can require miserably difficult searching to follow a grammar. A subset of shape grammars, called **context-free**, have rules whose left sides contain only a single symbol from V_M (which is also a member of V_M^+, albeit a very simple one). Searching the design is then guaranteed to be a search for a single object. Rather than having to find each element of the left-hand side of the rule and then look for n-tuples where all elements are transformed correctly relative to each other, the search ends once all instances of the single marker are found, as there is no "wrong transformation" for a single

element which prevents it from being a point of application of the rule. This constraint greatly simplifies writing an interpreter, and was used by Rajagopalan in writing the interpreter used by the Generative Design team [15]. Context-free and non-context-free rules are shown in Figure 2.1.

2.3 Shape grammars for Kuba cloth

Writing grammars for complete Kuba cloths can be complicated by sectored designs and by improvisation. The writer must make decisions in the writing process as to whether a particular variation in size, coloration or design is salient, and visualize how the cloth could be extended or shrunk, in order to make a useful and representative shape grammar for the piece. A simpler alternative is to associate different regions with different grammars, each capable of tiling the plane or an infinite finite-width band. The example in Figures 1.2 and 2.2 is simple in that it does not need to be broken down into sectors.

2.4 Shape grammars for zillij mosaics

We write grammars for strand-based designs by placing, duplicating and extending strands. Two methods have worked well for us. One is "omnidirectional," where we start at the intersection of copies of a strand, place segments out to all adjacent intersections (connected to the starting point by strands), and repeat the process at all the intersections. The other is "basis-wise," where we place starting points for all required strands first, and then extend the strands, essentially deciding how big a design we want ahead of time. The "basis-wise" approach to writing a grammar makes open strands easier to recognize, and in an interpreter that doesn't remove copies of markers occupying exactly the same point, it avoids the problem of numbers of markers growing exponentially with the number of rule applications. Here, we assume the grammar is written "basis-wise."

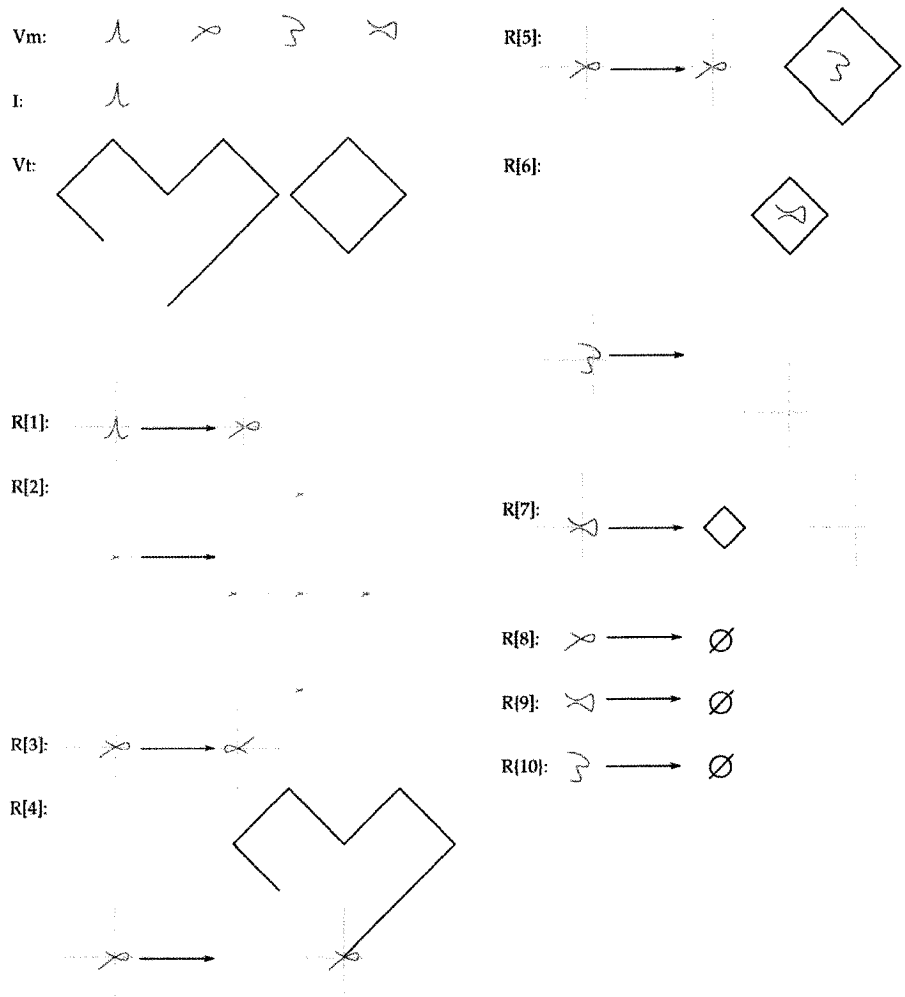

Figure 2.2: A grammar to recreate the major lines in Figure 1.2 with chains of line segments. The grammar obeys the restrictions to be introduced in Section 6.1. Crosshairs are used to indicate the origin in the left- and right-hand sides of rules where required. Notice that $R[2]$ is scaled down with respect to the other rules. The grammar doesn't break the scaling restrictions mentioned later because both sides of the rule are at the same scale. The different sizes of diamond obey scaling restrictions because the diamonds are terminals.

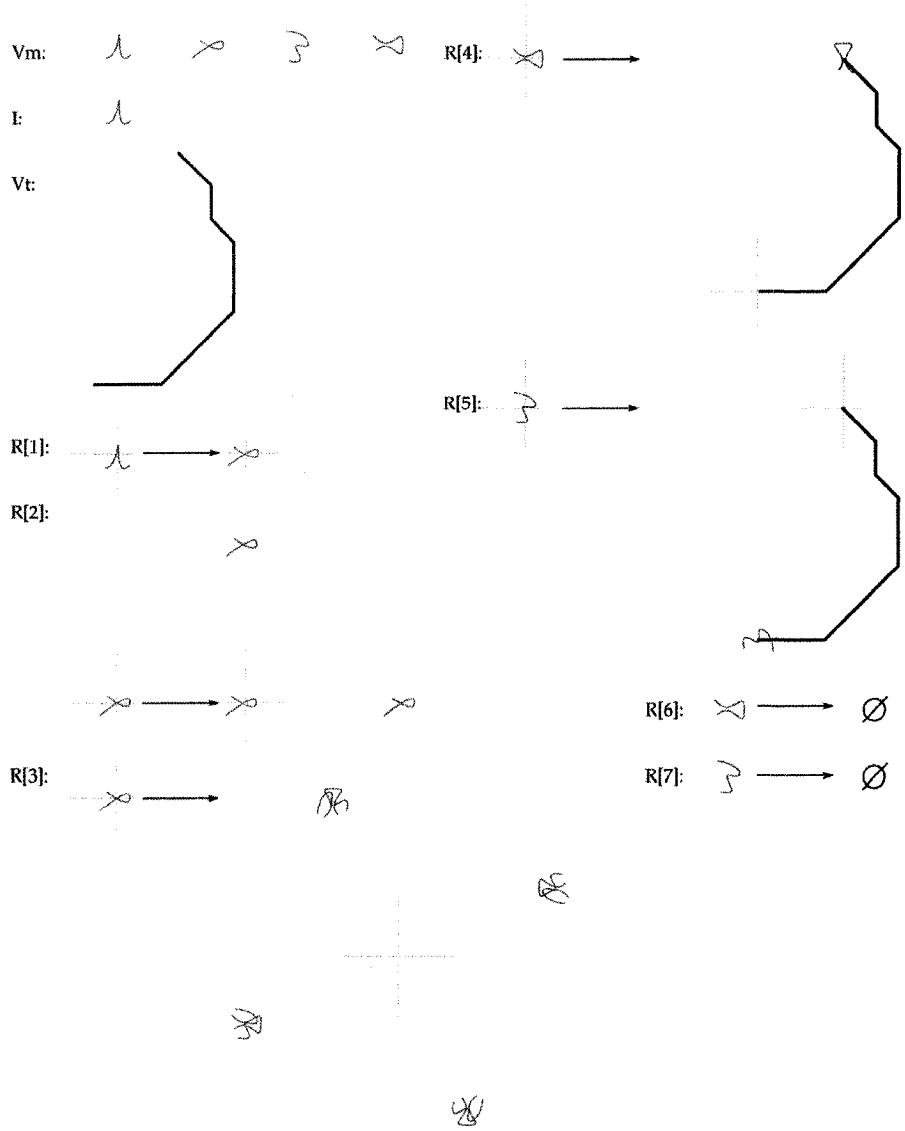

Figure 2.3: A grammar to recreate the white paths in the zillij mosaic in Figure 1.3 with chains of line segments. The grammar obeys the restrictions to be introduced in Section 6.1. Crosshairs are used to indicate the origin in the left- and right-hand sides of rules where required. Notice the overlapping markers in rule $R[3]$ and the reflected versions of terminals in $R[4]$ and $R[5]$

2.5 Summary

Shape grammars are the raw material this thesis shows how to classify. They have four major parts: markers, terminals, rules and an initial state. Rather than attempting to classify all shape grammars, this thesis uses a subset of grammars: context-free grammars, which have only one left marker per rule. The advantage of this is that such grammars are more feasible to implement (reducing search complexity and simplifying grammar design). The chapter ends with two example grammars, one each for the Kuba cloth and the zilllij moisaic presented in the introduction, which we will return to later as examples. The next few chapters deal mostly with the classifications and tools required to work with them. Chapters 3, 4 and 5 work more or less backward from our desired output to the machinery required to get it. We will return, well-prepared, to grammars and the tools required to assign symmetries to them in Chapter 6.

Chapter 3

Symmetry Groups

One powerful way of capturing the overall structure of a design is through symmetry classification. The Generative Design Project has used symmetry information as one objective, mathematical descriptor of artefacts in its database, and other researchers such as Washburn and Crowe [19] have done work showing that the mix of symmetry groups in a body of work correlates strongly with its culture of origin. This thesis defines a method for ascribing symmetry groups to grammars, skipping the steps of rendering designs in the language and examining them for symmetries.

This chapter provides detail on the symmetry features of interest in designs. All are possible symmetries of the plane, but some are also important in intermediate steps of the classification process. Describing many of these features involves tools and standard groups from group theory. For group theory terms, notation and definitions, I follow Dummit and Foote [4] unless otherwise noted.

3.1 Symmetries

Generally, a symmetry is an isometry of a design that leaves it unchanged. This section will provide an overview of symmetries of two-dimensional designs which are representable as rigid motions, including reflections. One concept that comes up regularly is that of a **generator**: generators of a group are a set of n elements $g_i \in G, i \in 1, 2, \ldots, n$ such that, for any element $h \in G$, there exists a sequence of generators and their inverses that, when multiplied out, yields h. In other words,

every element of a group should be expressible in terms of a set of generators for that group.

In all designs, a trivial identity function like $f(x_1, x_2) = (x_1, x_2)$ leaves the design unchanged.

In finite two-dimensional designs, possible symmetries can include rotations of $\frac{2\pi}{n}, n \in \mathbb{Z}$ for some values of n particular to the design about a point, and reflections across one or more lines which intersect at a point (the same point as for rotations, if applicable). The groups which model these symmetries, **cyclic groups** (rotations only), and **dihedral groups** (reflections with or without rotations) will be collectively called **rosette groups**.

If we accept the possibility of infinite designs, the possibilities become somewhat richer. Infinite designs can still have rosette symmetries : visually these designs have a centre with identical wedges (intersections of two half-planes) radiating out from them. An infinite design can also have translational symmetry: in more intuitive terms, the design repeats itself at regular intervals, and the translational symmetries move any given region of the design to a copy of itself. If there is one translational symmetry, there are an infinite number of translational symmetries which are multiples of that translation. This is clear, as if $f(D)$ maps a design D to a copy of itself, then $f(f(D))$ and $f^n(D)$, for $n \in \mathbb{N}$ must as well. If $f(D)$ translates the design by (a, b) and is a symmetry, then functions translating the design by (na, nb), for $n \in \mathbb{N}$, must as well.

There are two broad classes of plane symmetry groups which arise from translational symmetry: if the design has translational symmetries which are all parallel to each other, i.e. the generator of all the translational symmetries is a single vector, then the design has a **frieze group** of symmetries. If a two-dimensional design's translational symmetries aren't all parallel, then there must be two generating vectors, and the design's symmetries form a **wallpaper group**. Frieze groups' name arises from the image of a long strip of copies of a design: a frieze is a decorative band at the top of a wall in architecture (not necessarily possessing translational symmetry: many have long, narrative images in real-world friezes). Despite the name, a design having a frieze group needn't be a finite-width strip: it can be infinitely wide but simply not accept translational symmetries beyond a set with parallel vectors. The wallpaper groups' name suggests the instructions they contain for covering the whole plane with a design which repeats in two directions: "along the roll" and "strip to strip."

I follow terminology from Huang [9] and the approach from Grünbaum and Shephard [7] for

Section 3.1. Illustration conventions, detailed in Remark 3.1, are adapted from Grünbaum and Shephard.

3.1.1 Rosette groups

Rosette groups are symmetry groups isomorphic to cyclic and dihedral groups.

The cyclic groups c_n are generated by a single rotation about a single point and include all the transformations achievable by repeating that rotation. The generator can be $\frac{1}{n}$ turn, with $n \in \mathbb{N}$, or $\frac{m}{n}$ turn, with m and n co-prime. In the second case, the group can be generated by a $\frac{1}{n}$ turn, as well. In this thesis, we will use the smallest generator. The symmetries of a cyclic group of order n are called **n-fold rotational symmetry**.

The dihedral groups d_{2n} are the semidirect product of two cyclic groups: $c_n \rtimes c_2$, where c_n is the turn component, and c_2 contains the identity and a flip about some axis which travels through the centre of rotation.

A cyclic group c_n is a subgroup of the corresponding dihedral group d_{2n}. Cyclic groups are also subgroups of other cyclic groups: $c_m \leq c_n \iff m|n$, and likewise for dihedral groups. Rosette groups do not usually represent the symmetries of designs examined by the Generative Design Project, but they do play an important role in describing the local symmetries of closed strands (see Section 4.1).

3.1.2 Frieze groups

Frieze groups, or "strip groups" in Grunbaüm and Shephard [7], are groups of symmetries with a single translation among their minimal set of generators. There are seven types of frieze groups, classified by the types of symmetries involved. The types of symmetries which appear in frieze groups are:

translation As noted above, a frieze group's translational symmetries are all integer multiples of a single vector (i.e. all parallel and evenly spaced). All frieze groups have translational symmetries, although some frieze groups can be generated without them (as other symmetries, when repeated, yield translations). Translation symmetries are **free vectors**: they represent a

Figure 3.1: A $p111$ pattern with translation vector indicated.

Figure 3.2: A $p112$ pattern with translation vector and two classes of rotation (diamonds of different orientation) indicated.

movement of the whole design, with no specified point or line of application, unlike the other symmetries below.

rotation A frieze group may have rotational symmetries. The only rotation possible is a half-turn, π radians (which can also be considered a "reflection in a point"). Like translations, a frieze group with rotational symmetries always has a countably infinite number of them, evenly spaced. However, we need but one centre of rotation to generate a frieze group with infinitely many rotation centres: other rotations can be had by composing translations generated by $\vec{t}$ and a single generating rotation at point $\vec{o}$: the resulting rotations are around points $\vec{o} + \frac{n\vec{t}}{2}$.

In the tables, I use the term "flippable" to indicate a rotation which coincides with a reflection. This indicates that the design's symmetries have a subgroup which is dihedral.

"horizontal" reflection Going back to our model of a frieze running along the top of a wall, we let "horizontal" reflection be reflection across a line parallel to all the translations. We need to locate this line with an intercept or some other point, but its slope is a consequence of the slope of the translational symmetries. There may be at most one horizontal reflection in a frieze group, two or more parallel axes would create a translational symmetry not parallel to the ones we assume are the only ones in the design.

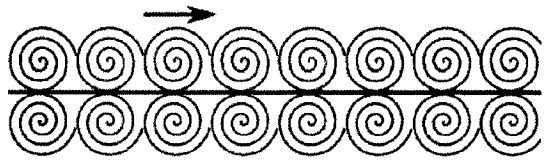

Figure 3.3: A $p1m1$ pattern with translation vector and axis of horizontal reflection (bold line) indicated.

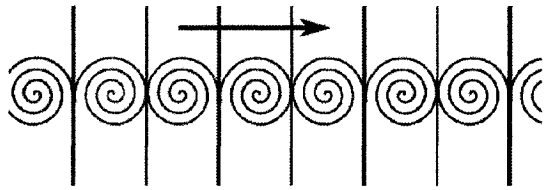

Figure 3.4: A $pm11$ pattern with translation vector and two classes of axes of vertical reflection (bold and thinner lines) indicated.

"vertical" reflection "Vertical" reflections are reflections across lines orthogonal to the translational symmetries in a design. Like rotations, vertical reflections come either as a countable infinity of reflections generated by a reflection across one axis, or do not exist in a design with a frieze group of symmetries. Like horizontal reflection, the translation vector already determines these reflections' slope, so an intercept is all that is required to specify the generating axis of reflection. Like rotations, combining translations and a single vertical reflection generates the rest of the vertical reflections, spaced by half the generating translation. Other than horizontal and vertical reflections, no other reflections can occur in a frieze group, as mirrors with other orientations would introduce translations which break the parallel assumption. To specify a vertical reflection, as with a horizontal one, we need only an intercept as the slope of the reflection axes and their spacing are determined by the translational symmetries of the frieze group.

glide reflection Glide reflections are a combination of a reflection and a translation parallel to the axis of reflection. For classification purposes, we only ascribe a glide reflection to patterns without horizontal reflection. The translation portion of a glide reflection will always be parallel to and half the length of the translational symmetry of its frieze pattern, and the

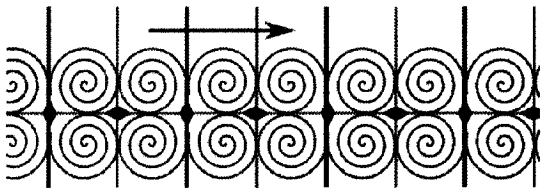

Figure 3.5: A $pmm2$ pattern with translation vector, horizontal and two classes of vertical reflections and two classes of rotations.

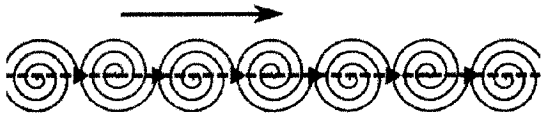

Figure 3.6: A $p1a1$ pattern with translation vector and a glide reflection axis (with arrowheads indicating translation distance for the glide) indicated.

presence of a glide reflection g removes the need for that translation t among its generators, as $g \circ g = t$. To specify a glide reflection, we minimally need an intercept and a free vector, i.e. as much information as a translation and a horizontal reflection.

Putting these elements together, we can enumerate the possible frieze groups. There is a fair amount of choice in notation. I will use the crystallographic notation used by Grünbaum and Shephard [7]. Table 3.1 lists the name, generators, symmetries present (other than generators and generators under a translational symmetry), and parameters required to specify each type of frieze group. When symmetries are labelled A and B, they are equivalence classes of symmetries which are mapped onto each other by the translational symmetries of the design, i.e. they sit on the same

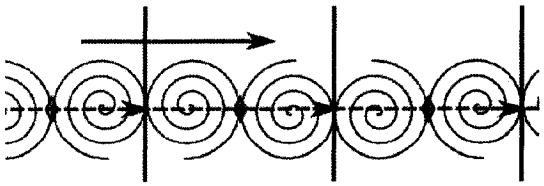

Figure 3.7: A $pma2$ pattern with translation vector, vertical reflections, rotations and a glide reflection axis (with arrowheads indicating translation distance for the glide) indicated.

name	generators	other symmetries	parameters required
$p111$	translation	—	free vector
$p112$	translation, rotation A	rotation B	free vector, fixed vector
$p1m1$	translation, horizontal reflection	—	free vector, intercept
$pm11$	translation,vertical reflection A	vertical reflection B	free vector, intercept
$pmm2$	translation, horizontal reflection, vertical reflection A[1]	vertical reflection B, flippable rotations A, B	free vector, two intercepts
$p1a1$	glide reflection	translation	free vector, intercept
$pma2$	glide reflection, vertical reflection[2]	translation, rotation	free vector, two intercepts

[1] can also be generated by a translation, either reflection and a rotation, in this case, think of the intercepts as a fixed vector
[2] can also be generated by glide reflection and rotation, in this case, think of the intercepts as a fixed vector

Table 3.1: List of frieze groups

"part" of the pattern.

Like rosette groups, frieze groups are not commonly associated with patterns under examination by the Generative Design project, but they do play an important role in describing symmetries of open strands (see Section 4.1) at intermediate steps in the classification process.

Remark 3.1 (symmetry illustration conventions) *To simplify distinguishing symmetry markers from actual designs in this chapter, all non-marker elements in each design are spirals or distorted spirals. Hence, any straight edge belongs to a reflection axis or rotation polygon.*

Rotation polygons are regular polygons with as many sides as the rotation they mark, except for two-fold rotations which are marked by diamonds. Solid polygons mark rotations on a reflection axis (which are automatically "flippable" or dihedral) and outlined polygons mark rotations with no coinciding reflection (a glide reflection can run through a non-flippable rotation).

Solid lines represent reflection axes, glide reflections by dashed lines, either with barbs indicating their period or with their periods noted in the caption.

Arrows represent representative minimal translation symmetries. Shaded areas represent fundamental regions for wallpaper groups.

3.1.3 Wallpaper groups

Wallpaper groups, or "crystallographic groups" in Grünbaum and Shephard [7], are groups of symmetries with two non-parallel translational symmetries. There are seventeen wallpaper groups.

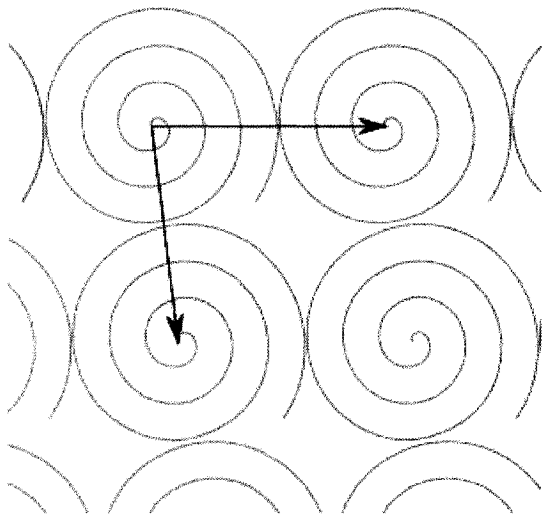

Figure 3.8: A $p1$ pattern with translation vectors indicated.

For one complete proof of why there are exactly seventeen, see Huang [9, pp. 69–90]. Many of the symmetries from previous subsections also occur in wallpaper groups, but how they interact is sometimes different from the frieze groups, as the restriction that all translational symmetries be parallel is gone.

The familiar symmetries of frieze and rosette groups undergo the following changes:

translation Replacing the singly-generated group of translational symmetries from frieze groups is a pair of generators for all translational symmetries. Again, these are free vectors: translations move everything in the design the same distance, so they are not "about" or "along" any particular part of the design.

rotation Rotations in a wallpaper group are freer than in a frieze group, but still very restricted by the need to tile them. A presentation of the restriction theorem is in Huang [9, pp. 62–68]: the result is that centres of rotation of the whole design can be only 2-, 3-, 4- or 6-fold. Local features, however, can have any degree of rotational symmetry: there may be patches of pattern with n-fold symmetry, but disallowed rotational symmetries cannot coexist with translational symmetries. One helpful way to visualize this is to think of the limited number of tilings of the plane with regular polygons (and parallellograms, in the case of 2-fold symmetry). A

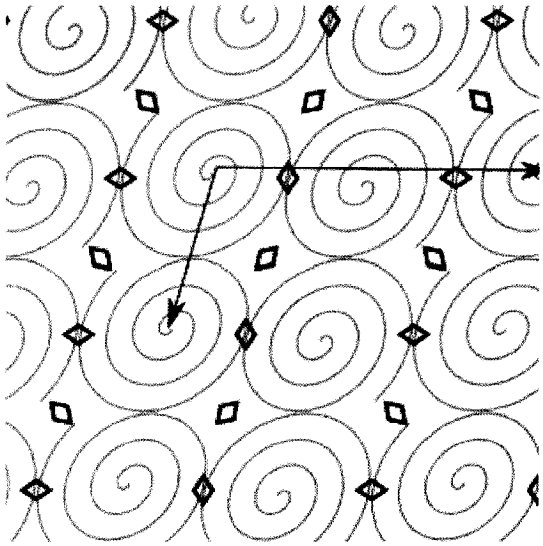

Figure 3.9: A $p2$ pattern with translation vectors and two classes of rotations indicated with diamonds of different orientations. A translation unit is the bounding box two spirals side-by-side, a fundamental region is the bounding box of one spiral.

repeating rotational symmetry corresponds to one of these tilings.

Different equivalence classes of rotations can exist in a design: like rotations and vertical reflections in frieze groups, two centres are in the same class if a translational symmetry maps from one to the other.

"Flippable" (dihedral) n-fold rotations in wallpaper groups have a minimum number of symmetry axes: $\frac{n}{2}$ axes if n is even, and n axes if n is odd.

reflection Gone are the two neat categories of "horizontal" and "vertical" reflections. Axes will obey the translational symmetries of the design. Non-parallel axes of reflection imply centres of rotation if there exists an axis of reflection and a centre of rotation, there will be a non-parallel axis of reflection. Like centres of rotation, different equivalence classes, also defined by the translations of the design, may exist. Reflections and glide reflections come in pairs of parallel equivalence classes: a pair may include two reflections, a glide reflection and a reflection, or two glide reflections.

Unlike frieze groups, rotations can be on or off an axis of reflection.

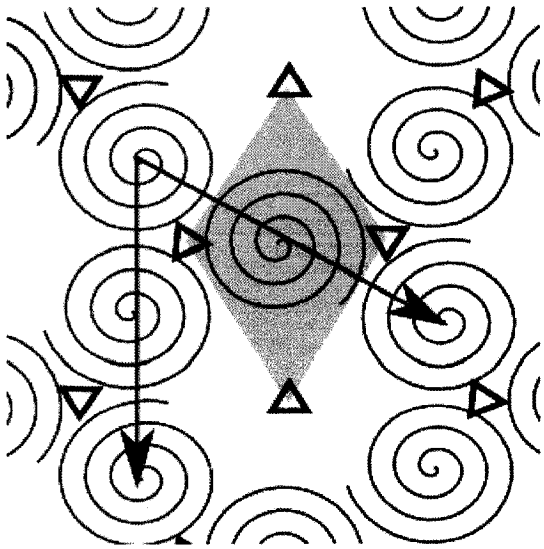

Figure 3.10: A $p3$ pattern with translation vectors and three classes of rotations indicated with triangles of different orientations. A translation unit is a rhomb with sides equal to the translational symmetries, a fundamental region is indicated by the shaded area.

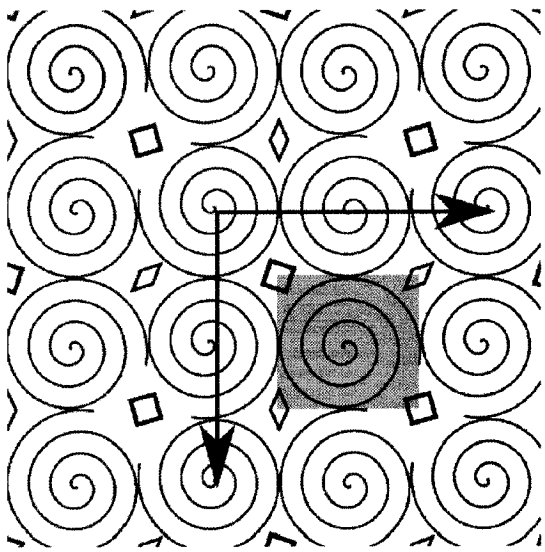

Figure 3.11: A $p4$ pattern with translation vectors and two classes each of two-fold and four-fold indicated with squares and diamonds of different orientations. A translation unit is a square with sides equal to the translational symmetries, a fundamental region is indicated by the shaded area.

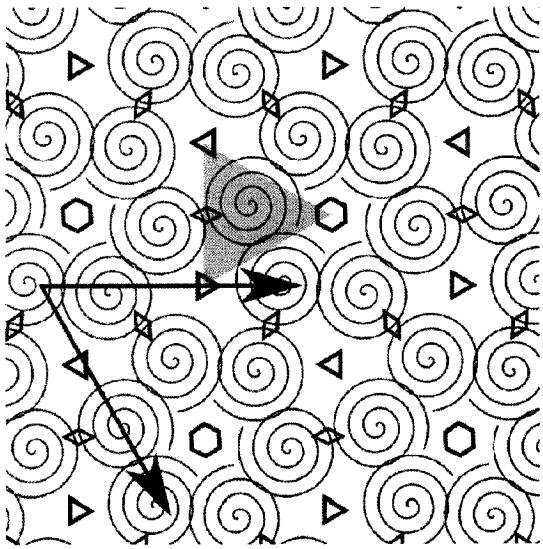

Figure 3.12: A $p6$ pattern with translation vectors and rotation centres of orders 2,3 and 6 (hexagons). A translation unit is a square with sides equal to the translational symmetries, a fundamental region is indicated by the shaded area.

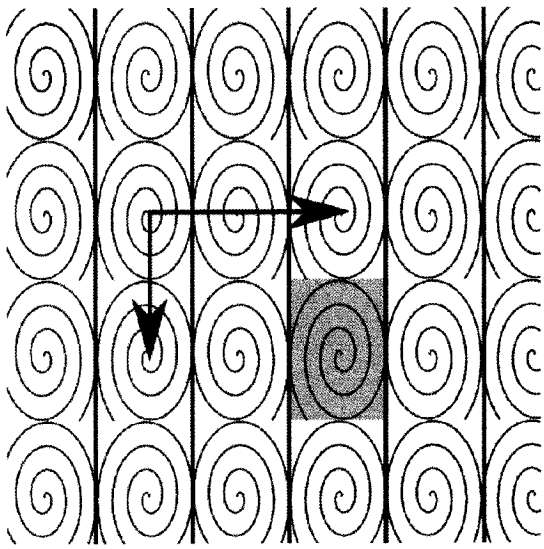

Figure 3.13: A pm pattern with translation vectors and reflection axes (two classes alternate: one between "tails" of spirals and one between "backs" of spirals). A translation unit is a square with sides equal to the translational symmetries, a fundamental region is indicated by the shaded area.

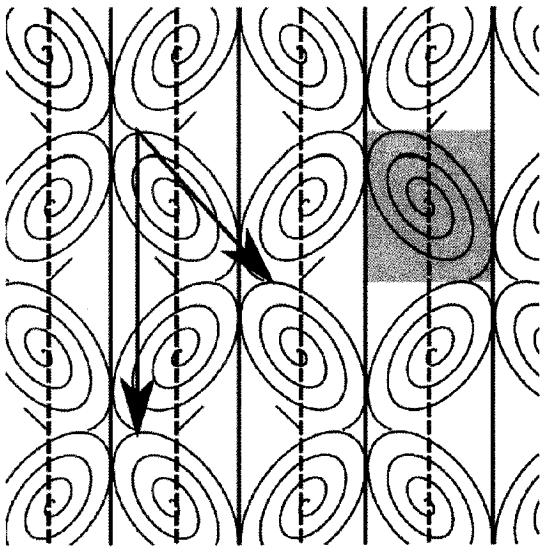

Figure 3.14: A *cm* pattern with translation vectors and reflection and glide axes (one class of each). A translation unit is a square with sides equal to the translational symmetries, a fundamental region is indicated by the shaded area. Notice that unlike p-celled patterns, the reflections are not orthogonal to either minimal translation vector.

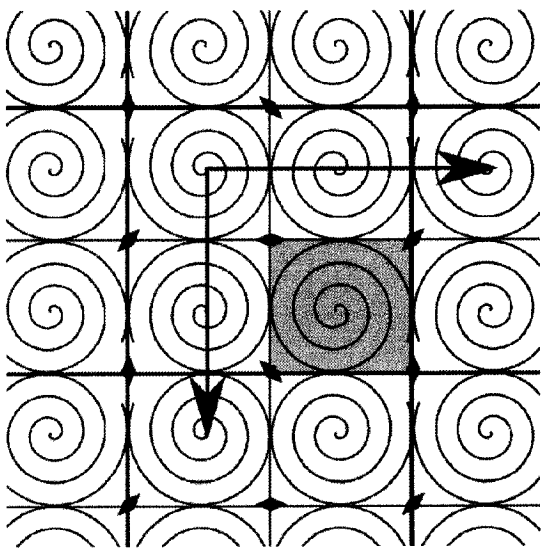

Figure 3.15: A *pmm* pattern with translation vectors, two classes of reflection and four classes of twofold rotation indicated: solid rotation markers indicate that they lie exactly on a reflection axis. A fundamental region is shaded.

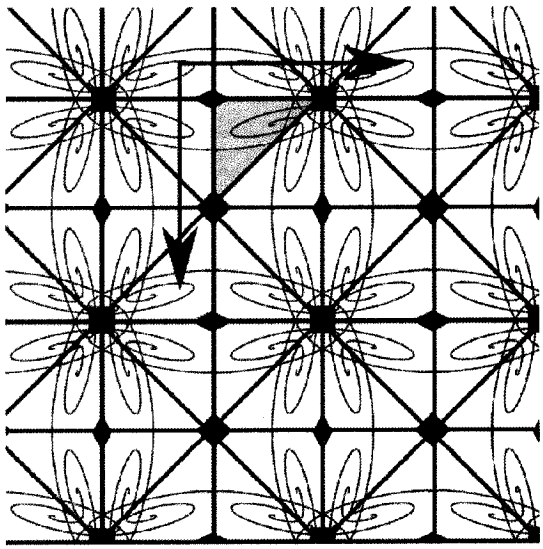

Figure 3.16: A $p4m$ pattern with translation vectors, four classes of reflection and two classes each of twofold and fourfold flippable rotations indicated. A fundamental region is shaded.

glide reflection Glide reflections are affected by the change to wallpaper groups much as reflections are. Glide reflections emerge in patterns where there are dihedral rotations (i.e. rotations around points on reflections).

With these elements, we can enumerate a further seventeen classes of symmetry group, presented in Table 3.2. Again, crystallographic group names are used, and the parameters involved are listed in addition to generating and other symmetries. More than frieze groups, wallpaper groups' generators listed in the table are not necessarily the only generators for the groups. For example, Grünbaum and Shephard [8, pp. 148–149] work with $p4m$ and $p6m$ as being generated by reflections. The lists for each group are as short as possible, but other generators can often be swapped in.

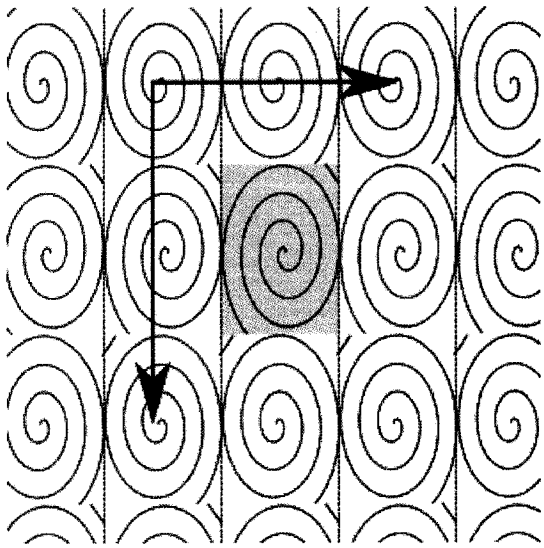

Figure 3.17: A *pg* pattern with translation vectors and glide reflections (dashed lines with barbs) indicated. A fundamental unit is shaded.

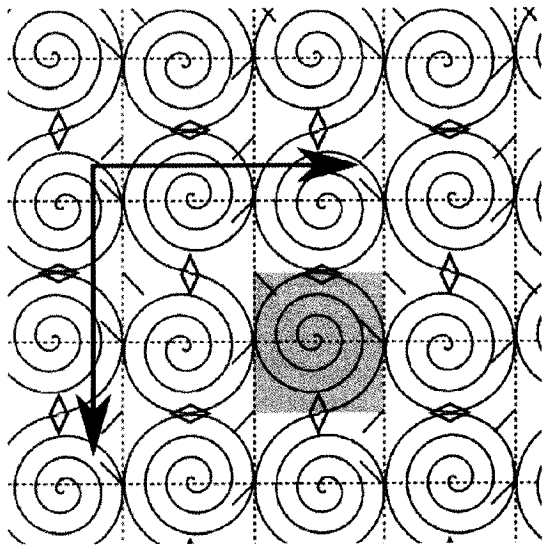

Figure 3.18: A *pgg* pattern with translation vectors, two classes of rotations and two classes of glide reflections (dashed lines with barbs) indicated. A fundamental region is shaded.

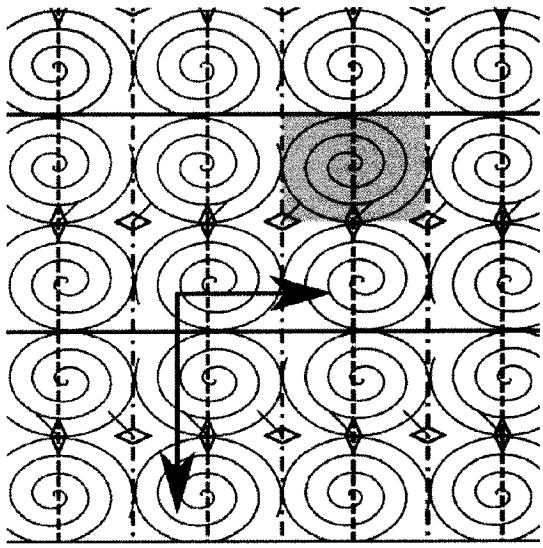

Figure 3.19: A *pmg* pattern with translation vectors, two classes of rotations, one class of reflection and two classes of glide reflections (dashed lines with barbs) indicated. A fundamental region is shaded.

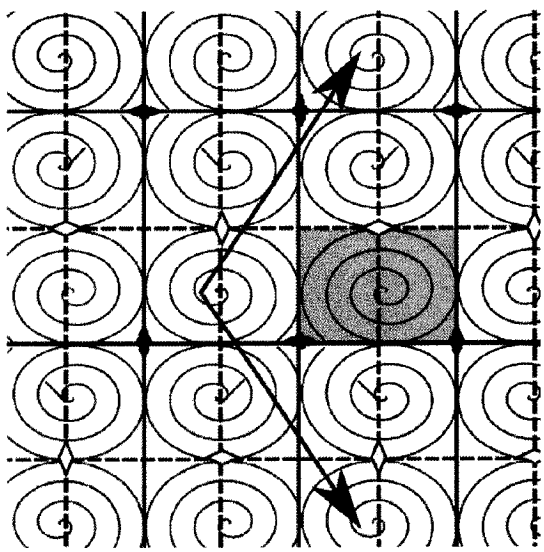

Figure 3.20: A *cmm* pattern with translation vectors, four classes of rotations, two classes of reflection and two classes of glide reflections (dashed lines with barbs) indicated. A fundamental region is shaded.

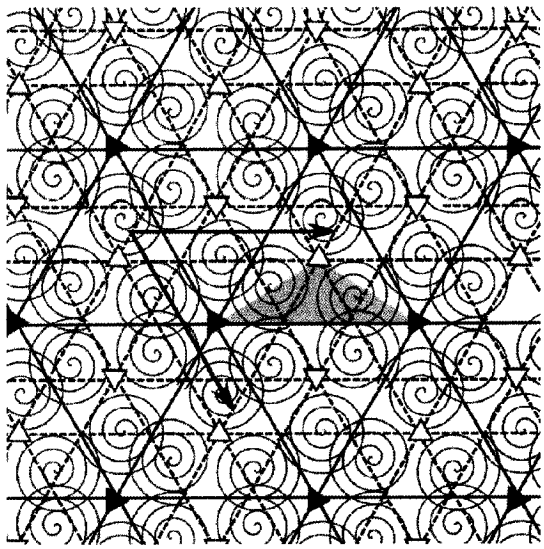

Figure 3.21: A $p31m$ pattern with translation vectors, three classes of rotations, three classes each of reflection and glide reflection (the period of the glide reflection is the minimum segment between reflections) indicated. A fundamental region is shaded.

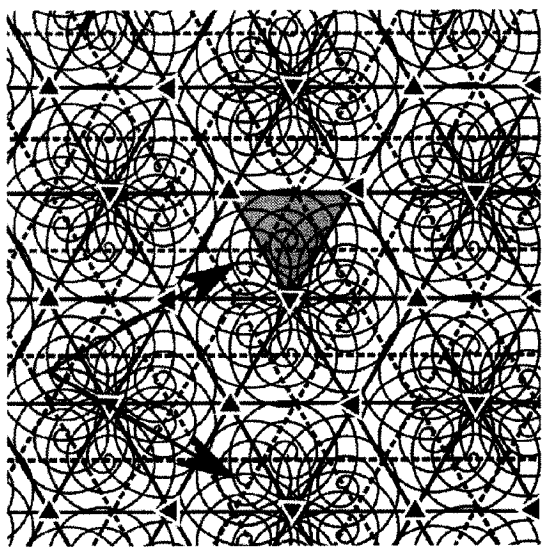

Figure 3.22: A $p3m1$ pattern with translation vectors, three classes of rotations, and three classes each of reflection and glide reflection (the period of the glide reflection is the minimum segment between reflections) indicated. A fundamental region is shaded.

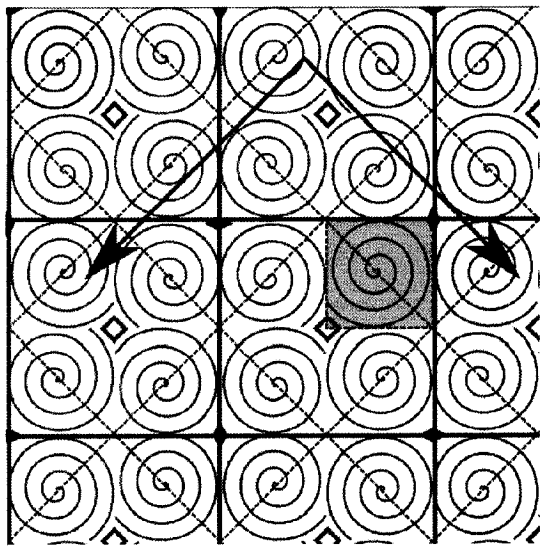

Figure 3.23: A $p4g$ pattern with translation vectors, three classes of rotations, and two classes each of reflection and glide reflection (the period of the glide reflection is the minimum segment between reflections) indicated. A fundamental region is shaded.

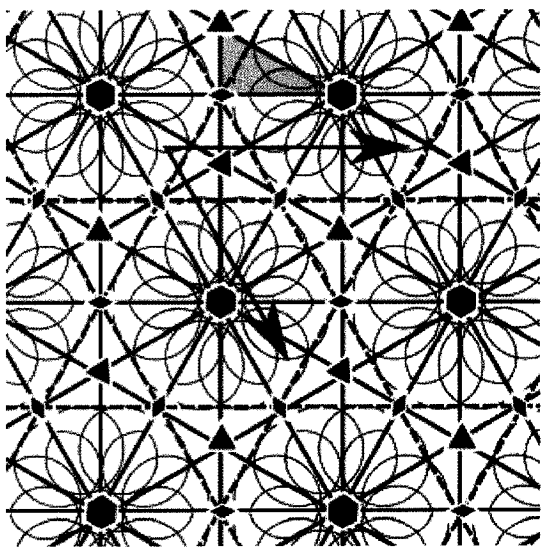

Figure 3.24: A $p6m$ pattern with translation vectors, six classes of rotations, six classes of reflection and three classes of glide reflection (the period of the glide reflection is the minimum segment between reflections) indicated. A fundamental region is shaded.

name	generators	other symmetries	parameters required
$p1$	translation A, B	—	two free vectors
$p2$	2-fold rotations A, B	translation A, B	fixed vector, two free vectors
$p3$	3-fold rotation A, 3-fold rotation B	3-fold rotation C	two fixed vectors
$p4$	4-fold rotations A, B	2-fold rotations A, B, translations A, B	two fixed vectors
$p6$	6-fold rotation, 3-fold rotation A	3-fold rotation B, 2-fold rotations A,B,C, translations A,B	two fixed vectors
pm	reflections A, B (axes in parallel), translation A (direction parallel to reflections)	translation B	two free vectors
cm	glide reflection, reflection (axes in parallel)	translation A, B	free vector, two intercepts
pmm	reflections A, B (axes orthogonal)	2-fold flippable rotations A, B, translations A, B	fixed vector, free vector (for one axis), magnitude (for other)
$p4m$	4-fold flippable rotations A,B	reflections A,B,C,D, glide reflections A,B, translations A,B	two fixed vectors
pg	glide reflections A, B (axes in parallel)	translations A, B	two free vectors
pgg	2-fold rotations A, B, glide reflections A, B (axes orthogonal)	—	fixed vector (rotation centre), free vector (first glide), intercept (orthogonal glide)
pmg	glide reflection, reflection (axes orthogonal)	2-fold rotations A, B, translations A, B	fixed vector, free vector (for one axis), magnitude (for other)
cmm	reflections A, B (axes orthogonal), glide reflection A	rotation, flippable rotation, glide reflection B, translations A, B	two fixed vectors (both on one reflection, define intercepts and slope for glide and other reflection), magnitude (for glide reflection)
$p31m$	3-fold flippable rotation (one reflection axis orthogonal to vector to non-flippable rotation), 3-fold rotation	reflection A, B, C, glide reflection A, B, C, translations A, B	two fixed vectors
$p3m1$	3-fold flippable rotations A, B (reflection axis joins centres)	3-fold flippable rotation C, reflections A, B, C, glide reflections A, B, C, translations A, B	two fixed vectors
$p4g$	4-fold rotation, glide reflection A	2-fold reflections A,B, glide reflections B, C, D, reflections A, B, translations A, B	fixed vector (for 4-fold centre and to provide intercept), free vector (for glide reflection)
$p6m$	6-fold flippable rotation, 3-fold flippable rotation A	3-fold flippable rotations B, C, 2-fold flippable rotations A,B,C, reflections A–F, glide reflections A–F	two fixed vectors

Table 3.2: List of wallpaper groups

3.1.4 Translational units and fundamental regions

Up to this point, symmetries have been grouped into equivalence classes by whether the translations of a design map them onto each other. This is unambiguous and easy to set up, but can result in a profusion of classes. A coarser division of symmetry is the **fundamental region**: a minimal region of the design from which we can, in combination with symmetry information, reconstitute the whole.

Both the translational unit and the fundamental region (and particularly the length and orientation of the associated vectors) are important characteristics of local or feature symmetries. Combinations of strands and other features may retain or lose symmetries based on the relationship between the fundamental regions and translational units of the combined elements.

We start with the idea of a **translational unit**. Given two minimal, non-parallel translation vectors $\vec{u}, \vec{v}$, we can form a parallelogram. We can then take a portion of the design bounded by any translation of that parallelogram, and using the design's translational symmetries, we can re-create the whole (periodic) design faithfully. Perhaps less usefully, we can take an infinite rectangle or parallelogram, such that a line parallel to one translational symmetry intersected with the region yields a segment which is a translation of a minimal translation vector in that direction (this approach is forced in the case of an unbounded frieze pattern): this is a translational unit. Rosette designs, having no translational symmetries, can be considered to be their own translational units or to lack them entirely.

Examining the translational unit, we may find some symmetries. Because $\vec{u}$ and $\vec{v}$ were defined to be minimal (and because the unit is finite), we should not find any translational symmetries. We may, however, find symmetries of the parallelogram: a rotation of degree 2 or 4 at the centre $(\frac{\vec{u}}{2}, \frac{\vec{v}}{2})$, reflections cutting either corner-to-corner (in some rhombic fundamental regions) or from the middle of a face to the middle of the opposite face (in some rectangular regions), or both (in some square regions). Infinite translation units may have symmetries as well, even if the location of axes and centres isn't quite as prescribed. These symmetries, if they exist, divide the translaional unit into congruent pieces, either parallelograms or triangles. These pieces are the **fundamental region**. One of these regions, plus symmetry information, is sufficient to re-create the entire design. Naturally, a fundamental region is a subset of a translational unit, but not a proper subset: if the

translaiona unit has no symmetries (as is the case in a design with $p1$ symmetry), the two are equal. As the previous sentence implies, two designs with the same frieze group or wallpaper group have the same fundamental region, up to isomorphism by some affine transformation.

Grünbaum and Shephard, in their interlace paper [8], work from knowing the design's symmetry and fundamental region to getting information about the strands in it. This thesis proceeds in a different direction: we are given information about the strands and their behaviour, and combine it to find out about the symmetries of the design, including information about its translation units and its fundamental regions.

3.1.5 Nomenclature

The notation in Tables 3.1 and 3.2, according to Schattschneider [16, pp. 441–446] is adapted from the International Tables for X-Ray Crystallography. To construct the symbol for a pattern, we start with a four-character symbol. The first is p or c: p means a "primitive cell," where the corners of the translational unit are made to correspond with the highest-order centres of rotation and that translational unit is taken as the cell for further symbols, and c means "centred cell," where the cell is made up of two translational units with a rotational centre of highest order at the centre. The second symbol is a number, the maximum order of a centre of rotation in the group. The third symbol the type of reflection across some axis orthogonal to one side of the cell: 1 for no reflections, g for glide reflections, and m for reflections. Centred cells are used when low-order rotations meet reflection axes that are not orthogonal to a translational symmetry, but are orthogonal to the sum of minimal translation symmetries. The last symbol is the type of reflection parallel to a second axis: parallel to the side for designs with maximum rotational order 2, at an angle of $\frac{\pi}{4}$ for designs with maximum rotational order 4, and at an angle of $\frac{\pi}{3}$ for designs with maximum rotational order 6. In many cases, symbols can be dropped from the end of the name without ambiguity, and some combinations of reflections force a certain maximum rotational symmetry, allowing the dropping of the second symbol. Shortened names and their reconstituted long names, are provided in Table 3.3.

Frieze notation is more ambiguous according to Grünbaum and Shephard [7], but readable in a similar way: all friezes use primitive cells, making the first character p, the second character

short	$p2$	pm	pg	cm	pmm	pmg	pgg	cmm	$p4m$	$p4g$	$p6m$
long	$p211$	$p1m1$	$p1g1$	$c1m1$	$p2mm$	$p2mg$	$p2gg$	$c2mm$	$p4mm$	$p4gm$	$p6mm$

Table 3.3: Conversions from short names to long names of wallpaper groups

indicates reflection normal to the translation axis, the third indicates reflection parallel to the translation axis (with a replacing g as the glide reflection notation), and the last indicating the maximum rotational symmetry (either 1 for none or 2 for two-fold).

3.2 Symmetry Groups

Tables 3.1 and 3.2 describe classes of symmetries. Members of a class have the same types of symmetry and numbers of equivalence classes of each. Maybe surprisingly at first glance, an affine transformation of a design can move it from one class to another: centres of rotation can appear or disappear under non-uniform scaling.

A symmetry group is the group of specific transformations taking a design to itself. Any of the classes of groups in Sections 3.1.1, 3.1.2 and 3.1.3 can be made into a symmetry groups by "filling in the blanks:" rotations need centres, reflections (including those in dihedral symmetries) need axes, translations need their free vectors, and so on.

Sometimes only general information about the class of symmetry group is required about a design, but even when that is the case, more detailed information about constituent parts is required to assemble a global symmetry group using the methods in Chapter 6.

3.2.1 Parameters and degrees of freedom

The tables above list a collection of parameters required: free and fixed vectors and the occasional intercept. With each vector counting as two degrees of freedom, and each intercept counting as one, we can determine a total number of degrees of freedom. No table is provided, but purely rotational (cyclic) symmetry has a single centre to specify (two degrees), and rotational-with-flips (dihedral) symmetry has a single centre as well as an intercept or angle, totalling three degrees of freedom.

A problem with these specifications is that they are not entirely standard: each group is specified by a different mix of vectors and intercepts with no obvious correspondence. A more standardized

approach to specifying the parameters for a symmetry group is treated in Section 5.2.1.

3.3 Summary

This chapter catalogues the features we are looking for in a grammar and establishes the concepts and vocabulary for working with them. A few necessary terms from group theory are introduced, and then an illustrated catalogue is presented. It may be helpful to return to the illustrations of various groups when confronted with general statements about symmetries and parameters in later chapters. The catalogue is ordered, broadly, by the number of independent translations and then by which other symmetries are required to describe them. Rosette groups have zero translations and are important mostly for describing local symmetry. Frieze groups' linear translations are all multiples of a single generating translation, and are particularly useful for describing individual strands in a pattern. Wallpaper groups tend to be the end product of the classification process, and have the maximum number of linearly independent translational symmetries in a plane pattern: two. After the survey of patterns, the chapter outlines the structure of group names and highlights the variability within classes by presenting parameters and degrees of freedom for symmetry groups. Chapter 5 will establish tools for working with various characteristics of symmetry groups, and Chapter 6 sets out a process for assigning symmetry groups to grammars, which is the problem addressed in this thesis.

Chapter 4

Strand-based patterns

The idea of a strand is introduced in Sections 1.1.2 and 1.2.1. Grünbaum and Shephard [8] take the idea of a strand, particularly in Islamic and Moorish art, and combine it with knowledge of fundamental regions to describe strand behaviour and extend strands indefinitely (if infinite) or complete them (if finite).

Strands happen to be a very common feature in the patterns of interest to the Generative Design Project: the zillij mosaics almost universally have strand patterns in the sense described in this chapter and in Section 1.2.1. They are also a powerful and proven tool for exploring symmetry, as in Grünbaum and Shephard [8].

4.1 Open and closed strands

In the context of a shape grammar written as specified in this thesis, we can divide strands in a design into two classes. The simpler type, **open strands**, can repeat forever: they are made up of repeating segments with frieze group symmetries between them: either half-turns around their ends, glide reflections which join their ends, reflections across parallel axes crossing their endpoints, or translations which connect endpoints of strands. These open strands have at least the symmetries implied by their duplication pattern, and possibly some others determined by the structure of the strand: minimally those of the strand itself, but it is possible for new symmetries to emerge from "offset" versions of the strand [5]. At the top of Figure 4.1, we see a strand unit

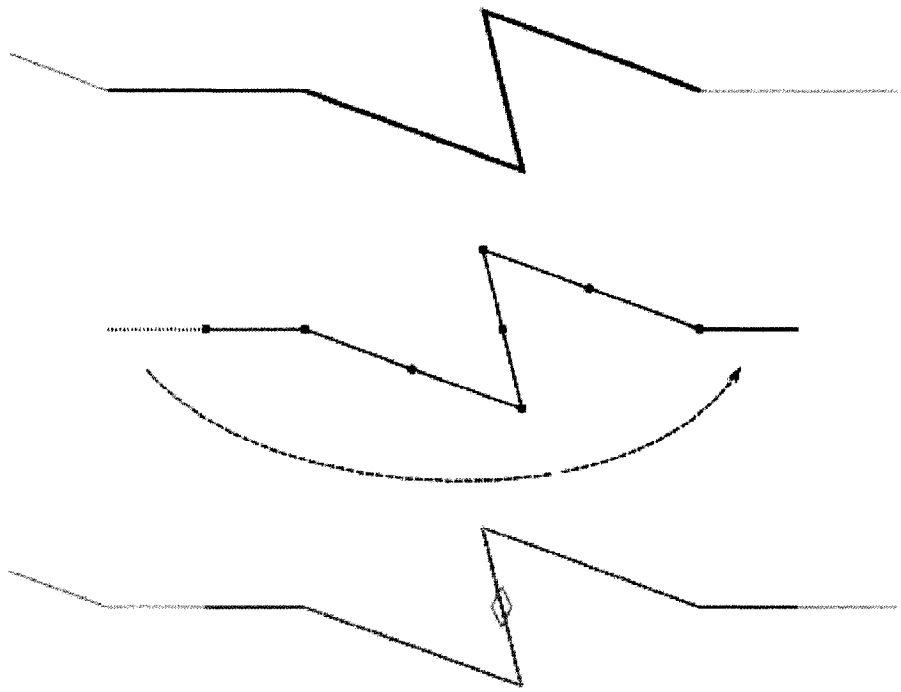

Figure 4.1: Offset symmetry of a strand unit

with no symmetries. In the middle row, the strand is broken into half-segments, and one is moved from left to right, making the unit 2-fold rotationally symmetric, with one of the centres, shown as an open diamond, the infinite strand's frieze group (shown in the bottom part). Other centres are the images of the centre under the translational symmetries of the design, and halfway between neighbouring centres generated that way.

Strands which are forced to be finite are **closed strands**: segments are duplicated by rotations commensurable with a full turn or are complete without duplication, and can only involve a finite number of copies of the strand before redundant copying sets in and the design stops changing. Closed strands thus only have the symmetries possible for finite designs (rosette groups, see Section 3.1.1).

4.2 Strandlike behaviour

We have somewhat over-defined strands, sometimes we want to have objects that are not strands, but treat them as such. The main properties of strands that we use are predictable repetition and a limited number of possibilities for symmetry at the one-terminal level.

Kuba cloth designs in particular are less likely to be purely strand-based than zillij mosaics. We can treat finite polylines as closed strands of order 1, even if they do not form closed polygons, or string rows of them together to form long open strands with gaps.

4.3 Interactions of strands and wallpaper groups

In their work on interlace patterns, Grünbaum and Shephard use the idea of an **induced symmetry**: a portion of the design (in their case, a strand) has its own symmetries in isolation, but when placed in the concept of a design, some are cancelled out by neighbouring elements which don't follow those symmetries. Those symmetries of the element which are left are the induced symmetries of that element. The design may have more symmetries than those induced symmetries, but those symmetries must take the element into account, hence those symmetries of the design not among the induced symmetries of the element must indicate transformations under which copies of the element must be found.

From this, we can say something about the symmetries of the design U, $\mathcal{S}(U)$, the symmetries of an element $\mathcal{S}(E)$ and the symmetries of its duplication $\mathcal{S}(D)$, with multiplication of elements of a symmetry group being the usual composition:

$$\mathcal{S}(U) \leq \mathcal{S}(E)\mathcal{S}(D) \cup \mathcal{S}(D)\mathcal{S}(E)$$

This means that we can establish an upper bound on the symmetries present in a design as soon as we have a lattice of translationally symmetric points and symmetry information for strands in the design. Furthermore, we can tighten this bound by successively combining strands and finding the symmetry and distribution of the combined sub-designs.

In practice, this only works in the case that we have congruent strands all represented by a

single symbol in a grammar, so that $S(D)$ can be read off the grammar. We demand this condition in Section 6.1 for this reason: a somewhat more organized grammar reduces the complexity of assigning symmetries considerably.

4.4 Summary

Strands are an easily recognizable feature in design, and strand-like behaviour aids in organizing visual elements. They are also included or suggested in the artwork studied by the Generative Design Project. They come in open and closed varieties: open strands continue indefinitely across the plane, whereas closed strands are not extendible or, when extended, loop back upon themselves. Sometimes it is beneficial to relax the concept of strand somewhat, to include strandlike behaviour, allowing us to apply strand techniques to repeated discrete objects. In closing this chapter, the concept of induced symmetry allows us to establish a crude upper bound on the symmetries of a combination of strands. As we will see in Chapter 6, strands are also recognizable in grammar form, and will be building blocks for assigning symmetries to grammars.

Chapter 5

Lattice methods

This thesis approaches combining the symmetries of simpler patterns by organizing symmetries of two-dimensional designs into a lattice. This lattice has the trivial symmetry of a blank, featureless plane at its top, and that of a completely asymmetric design at its bottom. In between are all of the symmetry groups from Chapter 3. When combining two patterns, we map their lattice points to a resulting lattice point using the *meet* operator. This can be iterated over finite numbers of lattice points, so we can take features we find locally in a shape grammar and glean information for the entire design. A major advantage of using lattices is that they can represent many of the features of interest in determining the symmetries of shape grammars, so that in Chapter 6, large parts of the symmetry assignment process can be described with a small vocabulary, once the details are established in this chapter.

5.1 Lattice basics

A lattice is an ordered set $L = \langle E, \leq \rangle$ with the additional properties that for any two elements, we can find a *join* and a *meet*, defined as follows:

Definition 5.1 (join) *The join of x and y, written $x \vee y$, is the least upper bound of x and y, i.e. the least element greater than x and greater than y.*

Definition 5.2 (meet) *The meet of x and y, written $x \wedge y$, is the greatest lower bound of x and y, and is the*

join of x and y in the dual, L^∂, of L.

The dual is more or less what we would expect, and makes proving things about ordered sets easier: for any statement about ordered sets in general, the dual statement is also true.

Definition 5.3 (dual) *The **dual of an ordered set** S, written S^∂, is an order on the same set where $\leq_\partial$, the new order, is defined in terms of $\leq$, the old order, as follows: $x \leq y \iff y \leq_\partial x$. In essence, the dual of an ordered set is that set "flipped upside-down." The **dual of a statement** is the statement created by swapping the meanings of $\leq$ and $\geq$ in the original statement.*

A useful property of lattices is that they can be defined either as special ordered sets or directly with $\vee$ and $\wedge$. The relation $\leq$ can be defined in terms of $\vee$ and $\wedge$, as in the following lemma, from Davey and Priestley [2, p. 39]:

Lemma 5.4 (Connecting Lemma) *Let L be a lattice, and let $a, b \in L$. Then the following are equivalent:*

1. $a \leq b$;
2. $a \vee b = b$;
3. $a \wedge b = a$.

When $a \leq b$, then the least element greater than or equal to b is b. The set of elements greater than both a and b must be a subset of all elements greater than b, and b is in the set of elements greater than a, so $a \leq b \Rightarrow a \vee b = b$. By duality, $a \leq b \Rightarrow a \wedge b = a$. In the opposite direction, $a \vee b = b$ and $a \wedge b = a$ both imply by definition that $a \leq b$. □

Lattices can have a top and a bottom. If they do exist, they are defined as follows:

Definition 5.5 (top and bottom) *The top of a lattice (or, more generally, an order) L is an element $\top \in L$, such that $\forall x \in L, x \leq \top$. Dually, the bottom, $\bot$, is an element such that $\forall x \in L, \bot \leq x$. A lattice or order may have a top, bottom, both or neither, but if they exist, they are unique.*

In building lattices, we can define a product of lattices. One visual way of thinking of the product of two lattices is to take the elements of the first lattice, place copies of the second lattice "inside" each, and connect corresponding points of each copy. Algebraically, the product of two lattices is a set of ordered pairs, with join and meet defined element-wise.

Definition 5.6 (product of lattices) *The product of lattices L and K, $L \times K$, is a set of ordered pairs (l, k) with $l \in L$ and $k \in K$. Join and meet are defined as follows:*

$$\begin{aligned}(l_1, k_1) \vee (l_2, k_2) &= (l_1 \vee l_2, k_1 \vee k_2) \\ (l_1, k_1) \wedge (l_2, k_2) &= (l_1 \wedge l_2, k_1 \wedge k_2)\end{aligned}$$

5.2 Lattices of sets

Defining a meet of plane symmetry groups to find the result of combining two patterns with known symmetry groups will be the central tool in Section 6.3. A first step is to build a lattice of the basic categories of symmetries. Using this lattice of plane symmetry groups alone will not be sufficient, as plane symmetry groups leave 4–6 degrees of freedom (translation vectors and positions of reflection axes and rotation centres) undefined, but this lattice will form a slice of the larger lattice of symmetry groups. The parameters of the symmetries involved may push the actual design symmetry into the down-set of the meet of the symmetry groups, requiring a more complicated lattice than one based solely on categories of symmetries.

A plane symmetry group (including the simpler symmetry groups: frieze groups, dihedral groups and cyclic groups) is a set of symmetries. Different plane symmetry groups can have symmetries in common, and the intersection of the symmetries of two plane symmetry groups always yields a plane symmetry group or no symmetry at all (we will treat no symmetry at all as a degenerate plane symmetry group.) We can then treat plane symmetry groups as a *lattice of sets.*

Definition 5.7 (lattice of sets) *A lattice of sets is a subset of the power set $\varnothing \leq \mathfrak{L} \leq \wp(X)$ which is closed under finite unions and intersections. If it is closed under arbitrary unions and intersections, then it is a* ***complete lattice of sets.***

5.2.1 Kali-style generators

A distillation of symmetry groups down to standardized sets of generators is required to take advantage of this structure. One such distillation, implemented in Java, comes from Amenti, Weeks and Phillips, in the symmetry group drawing software Kali [14]. They have a set of four axes,

flagged as either reflection or glide reflection axes, defined relative to the cell consisting of a translational unit of the design with one corner on a centre of rotation (this cell is the primitive cell of all p groups, and half of the primitive cell of the c groups, as discussed in Section 3.1.5).

When building a lattice of sets of symmetries, we can consider a reflection to include a "long glide" under certain circumstances, which we will detail in Section 6.3. We can always consider a reflection of order n to include reflections of orders of the factors of n.

The four **Kali axes** involved are either "x" (parallel to an arbitrary first translation) or "y" (orthogonal to the first translation), and either go through a rotation of highest order or do not: axes going through high-order centres will have a "c" appended to their label, uncentred aces will have single-character labels (Kali uses 0 and 4 for implementation reasons), and will include a point either half (for reflections) or one-quarter (for glide reflections) of the appropriate (non-parallel) translation symmetry from the corresponding c axis.

This lattice of sets of symmetries, working by meets, gives the resulting symmetries of two "harmonized" translation units. The tools for harmonizing translation units will be established in Section 5.3.2, and the process will be outlined in Section 6.3

5.3 A divisibility lattice and its generalizations

The natural numbers can be ordered the usual way and form a very simple lattice, but this is not the only possibility. The "divides" relation [2, pp. 4, 37] organizes the real numbers in a way that lends itself to use with rosette groups (see Section 3.1.1).

These lattices will enable us to find symmetry parameters for designs resulting from the combination of designs with known translational and rotational symmetries.

Definition 5.8 (lattice by divisibility) *Given* $x, y \in \mathbb{N}$, *define* $\preceq$ *as* $x \preceq y \iff x|y$.

In the lattice $\langle \mathbb{N}, \preceq \rangle$, $x \vee y = \operatorname{lcm}\{x, y\}$ *and* $x \wedge y = \gcd\{x, y\}$

The lattice $\langle \mathbb{N}, \preceq \rangle$ has a bottom of 1 (all natural numbers are multiples of 1), and a top of 0 (zero is a multiple of everything).

name	reflection axis	glide axis	max. rotation	notes
c_n	—	—	n	$n \in \mathbb{N}$
d_{2n}	xc	—	n	$n \in \mathbb{N}$
$p111$	—	—	1	
$p1m1$	yc	—	1	
$pm11$	xc	—	1	
$p112$	—	—	2	
$pmm2$	yc	—	2	
$pma2$	x	—	2	
$p1a1$	—	yc	1	
$p\,n$	—	—	n	$n \in \{1, 2, 3, 4, 6\}$
pmm	xc	—	2	
$p3m1$	xc	—	3	
$p4m$	xc	—	4	
$p6m$	xc	—	6	
$p4g$	—	x	4	changed from [14]
$p31m$	yc	—	3	changed from [14]
cmm	x	xc	2	
pmg	x	—	2	
pm	xc	—	1	
cm	xc	x	1	
pg	—	xc	1	
pgg	—	x	2	

Table 5.1: Kali-style generators for symmetry groups

5.3.1 A lattice from an order by commensurability

While this thesis uses lattices of sets to represent general types of symmetry, lattices of sets don't lend themselves to dealing with the real-vauled parameters that crop up in symmetries: a $p4m$ symmetry still has distances between its lattice points and lines of reflection which vary across that group of isomorphic symmetry groups. The divisibility lattice in Section 5.3 holds some promise, but only deals in the natural numbers. We can construct a lattice containing the nonnegative real numbers, however, that has some nice properties for representing length parameters of translational and glide symmetry, using an order relating pairs of commensurable numbers.

Definition 5.9 (Order by commensurability) *Let $P = \mathbb{R}_{\geq 0} \cup \{\infty\}$ Let addition, multiplication and division be defined as usual on the elements of P.*

Given $x, y \in P_{\geq 0}$, define $\preccurlyeq$ as follows:

$$x \preccurlyeq y \iff x = y = 0 \text{ or } \frac{y}{x} \in \mathbb{Z} \text{ or } y = \infty$$

The definitions are such that $\langle P, \preccurlyeq \rangle$ is a lattice, has a top $\top = \infty$, and has a bottom $\bot = 0$. Elements a, b have a join of their "least common multiple", i.e.:

$$a \vee b = \min \{\{x \mid \exists m, n \in \mathbb{Z}_{\geq 0} \cup \{\infty\} : ma = nb = x\}\}$$

Meets are "greatest common divisors" in the same sense:

$$a \wedge b = \max \left\{ \left\{ x \mid \exists m, n \in \mathbb{N} : \frac{a}{m} = \frac{b}{n} = x \right\} \cup \{0\} \right\}$$

If a set S contains two incommensurable numbers a and b, then $\bigvee S = \infty$, which should be interpreted as the least common multiple not existing in the reals (and that $\infty \times a = \infty \times b = \infty$).

The sublattice of this lattice consisting only of the natural numbers is exactly the divisibility lattice from Section 5.3.

5.3.2 A vector lattice

The previous two lattices are lattices of numbers. For two-dimensional patterns, we also need a vector lattice: one whose meet represents a pair of minimal vectors constructible from two different pairs of vectors and their inverses. As with the commensurability lattice (Section 5.9), incommensurable quantities should yield defined but symmetry-wrecking results.

The points on this lattice are the ordered quadruples $\langle x_1, y_1, x_2, y_2 \rangle$ with all four elements coming from $\mathbb{R} \cup \infty$. Additionally, all points satisfy the following properties:

$$\begin{aligned}
\angle\left(\langle 1,0\rangle, \langle x_1, y_1\rangle\right) \leq \angle\left(\langle 1,0\rangle, \langle x_2, y_2\rangle\right) & \\
\angle\left(\langle 1,0\rangle, \langle x_1, y_1\rangle\right) &< \pi \\
\angle\left(\langle x_1, y_1\rangle, \langle x_2, y_2\rangle\right) - \angle\left(\langle 1,0\rangle, \langle x_1, y_1\rangle\right) &< \pi \\
\angle\left(\langle x_1, y_1\rangle, \langle x_2, y_2\rangle\right) = 0 &\Rightarrow x_1 = x_2 \text{ and } y_1 = y_2
\end{aligned}$$

In other words, the vector closest to the x-axis comes first, the first vector must lie in the first two quadrants, the second vector can't have an inverse which lies at a positive angle from the first, and parallel vectors must be equal. The idea is to have either two standard minimal vectors which generate a two-dimensional grid of points, or two copies of a vector which generates a one-dimensional row. These sets of points can represent the translational symmetries of designs.

Let $\vec{a} = \langle x_1, y_1, x_2, y_2 \rangle$ and $\vec{b} = \langle x_1', y_1', x_2', y_2' \rangle$ be points on this lattice. We define $\preccurlyeq$ as follows:

$$\vec{a} \preccurlyeq \vec{b} \iff \exists m_1, n_1, m_2, n_2 \in \mathbb{Z} \cup \infty : \begin{array}{l} m_1 \langle x_1, y_1\rangle + n_1 \langle x_2, y_2\rangle = \langle x_1', y_1'\rangle \\ \text{and } m_2 \langle x_1, y_1\rangle + n_2 \langle x_2, y_2\rangle = \langle x_2', y_2'\rangle \end{array}$$

The join of two points on this lattice is a lattice point consisting of a pair of (possibly identical) vectors which are "constructible" from integer multiples of the vectors in both operand points. The meet, defined by the relation above, is a lattice point with a pair of vectors that can "pack" both pairs of vectors. Finding these in general is an integer programming problem not treated in this thesis, but the smooth operation of joins and meets given the order is established by Lemma 5.4.

5.3.3 Harmonized joins in vector lattices

One property of a given join that we will want later on is whether it is **harmonized** with respect to one or both of their Kali axes (see Section 5.2.1. Similar to the meaning of the word when referring to sound, this means that features of the two shorter-period elements line up with those of the longer-period one: those features are the axes of reflective or glide symmetry.

A join

$$\vec{a} \vee \vec{b} = \langle x_3, y_3, x_4, y_4 \rangle = \text{concatenate}(\vec{j_1}, \vec{j_2})$$

(j_1 and j_2 are two-element vectors) is harmonized with respect to an axis h (whose type, glide or reflection, is known) if and only if there exist:

$$\vec{a^*} \in \{(x_1, y_1), (x_2, y_2)\}, \vec{b^*} \in \{(x_1', y_1'), (x_2', y_2')\}$$

with $\{a_*\} = \{(x_1, y_1), (x_2, y_2)\} - \{a^*\}$ and $[\{b_*\} = \{(x_1', y_1'), (x_2', y_2')\} - \{b^*\}$, such that the following are true:

- $\vec{a^*} \| \vec{b^*}$
- $\left(\left(\vec{a^*} \| h \text{ and } \vec{b^*} \| h\right) \text{ or } \left(\angle\left(\vec{a^*}, h\right) = \pm\frac{\pi}{2} \text{ and } \angle\left(\vec{b^*}, h\right) = \pm\frac{\pi}{2}\right)\right)$
- if $\vec{a^*} \| h$ then either $\vec{h}$ is named xc or yc (i.e. h passes through a centre of highest-order rotation), or otherwise (if $\vec{h}$ is named x or $\vec{h}$ is named y) then

$$\frac{||\vec{a_*} - \text{proj}\left(\vec{a_*}, \vec{a^*}\right)|| \vee ||\vec{b_*} - \text{proj}\left(\vec{b_*}, \vec{b^*}\right)||}{||\vec{a_*} - \text{proj}\left(\vec{a_*}, \vec{a^*}\right)||}$$

and

$$\frac{||\vec{a_*} - \text{proj}\left(\vec{a_*}, \vec{a^*}\right)|| \vee ||\vec{b_*} - \text{proj}\left(\vec{b_*}, \vec{b^*}\right)||}{||\vec{b_*} - \text{proj}\left(\vec{b_*}, \vec{b^*}\right)||}$$

must both be odd, in order to ensure the coincidence of a pair of integer multiples of $\vec{a^*}$ and $\vec{b^*}$ or $\vec{a_*}$ and $\vec{b_*}$.

- if $\angle\left(\vec{a^*}, \vec{h}\right) = \pm\frac{\pi}{2}$, then the join is harmonized with respect to $\vec{h}$ if $\vec{h} \| \vec{j_n}$ for $n = 1$ or 2 and either $\vec{h}$ runs through a highest-order centre (and has a c in its name) or, similarly to the case

above, if

$$\frac{||\vec{a^*}|| \vee ||\vec{b^*}||}{||\vec{a^*}||} \text{ and } \frac{||\vec{a^*}|| \vee ||\vec{b^*}||}{||\vec{b^*}||}$$

are odd.

The conditions are fairly unwieldy to express, but the visual idea is that, starting with an axis on the edge of a primitive cell, or halfway across and parallel to an edge, or a quarter of the way across and parallel to an edge in each cell in the join, when the joined cell is formed, a copy of that each axis will coincide in the right areas of the joined cell: in short, if a join is harmonized with respect to that axis, we do a regular meet on a lattice of sets of symmetries, if not, then we exclude that axis and then perform the meet.

5.4 Summary

With a set of lattices describing relationships of interest between sets of symmetries, periods, frequencies and vectors, we now have the tools to succinctly combine the known symmetries of pairs of designs inscribed on a common plane. Some of these lattices, such as the lattice of sets, wrap familiar operations in lattice structure, whereas in the case of the vector lattice, previous lattices and other concepts are assembled to yield the machinery that symmetry classification requires. This process will be applied in the next chapter. In this chapter, we also introduced a uniform scheme of generators borrowed from the symmetry group rendering code of *Java Kali*, and then included some awareness of this scheme in the vector lattice.

Chapter 6

Assembling semantics

This chapter narrows down the class of grammars usable as inputs, then explores some properties of graphs of grammar rules that yield natural sub-designs which serve as starting points for moving the problem into the semantics of lattices of symmetries. That process concludes the rest of the chapter.

6.1 Grammar restrictions

We use only context-free grammars (see Section 2.2). A rule in a context-free grammar looks for a single, atomic marker and replaces it with some elements. More complex matching is not allowed. This means an application of a rule at a marker adds a predictable number of matches to the design: the number of markers in the replacing set. This number is exact, as a grammar with no matches for an introduced marker has some sequences of rule applications that lead to unavoidably invalid designs.

All rules in a grammar should be **reachable**, i.e. it should actually contribute to the language of the grammar.

Definition 6.1 (reachable) *For any reachable rule r_{a_0}, there exists some chain of rules back to the initial rule I in which r_{a_i} has in its right-hand-side some marker m and either $r_{a_i} = I$ or m is in the left-hand-side of $r_{a_{i+1}}$.*

We establish a grid (synonymous with "lattice" in most of the sources for this thesis, but the term "grid" is used to avoid confusion with lattices from ordered sets) for closed strands, or starting points for a lattice of open strands, before filling it in.

We use rules with only markers in their right-hand sides to place the starting points for open strands or for rows of closed strands or other closed motifs, then apply rules to grow those strands. If extra rules are required to fill in detail, these must be applied after the grid is built.

Rules in a grammar may transform markers solely by translation, rotation and reflection: scaling is not allowed on markers, only on terminals. This makes it possible to assume that repeated non-parallel copying of a marker will lead to a bounded design and (if the angle is commensurable with 2π) a closed strand or similar object. This property will be referred to as **scale-preserving**.

We avoid duplication of terminals. A strand terminal should contain one repeat unit of a strand, and similar strands (including, for our purposes, strands which are reflections or glide reflections of each other) should use the same terminals. This property will be referred to as **terminal-disjoint**. One reason for avoiding terminal duplication is to make finding the symmetry group of a set of strands easier. If two strands have are made up of different terminals, then we can assume that they are in fact different (even after being extended to frieze patterns).

For every marker in a conforming grammar, there must also exist a tree of rules, connected as above, which has no markers in any of its leaf nodes: this corresponds to the possibility of arriving at a design in the language of the grammar after introducing that marker. This property will be called **realizable**.

6.2 Extracting local symmetries from grammars

The sequence of rules in a grammar is constrained by the markers in each rule. It can be useful to represent these constraints with a directed graph.

Algorithm 6.2 (directed graph construction from grammars) *Taking the grammars from Figures 2.3 and 2.2, we can create the graphs of rules in Figures 6.1 and 6.2. Each graph has two types of node: elliptical rule nodes and diamond choice nodes. Every rule has an elliptical node, and for each marker in the right-hand side of the rule, that node connects to all nodes in the with that marker in their left-hand side. If only one rule*

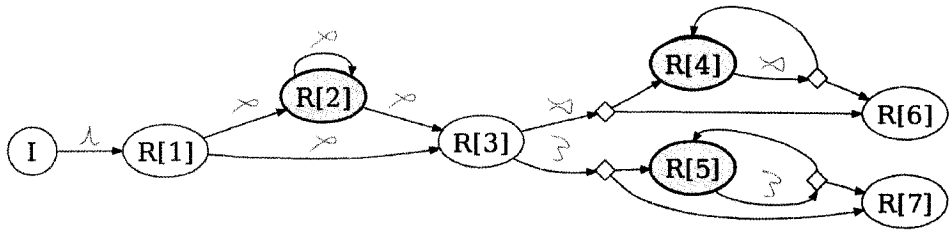

Figure 6.1: Directed graph of rules in the Ummayad zillij grammar from Figure 2.3. Interpretation details are in the text of Section 6.2.

has that marker in its left-hand side, then connect it directly; if more than one rule (including the originating rule) has a given marker in its left-hand side, the departing edge should point to a choice node, which then has outgoing edges to all rules with that marker in their left-hand side. In the graphs illustrated, each edge departing a rule is labelled with its associated marker, although those labels aren't strictly necessary once the graph is built. The initial state is treated as a rule with no left-hand side, and the initial state's symbols in its right-hand side.

Algorithm 6.3 (rule application sequences on a directed graph) *The progress of applying rules can be represented with markers on the nodes of the graph. This representation is not sufficient to keep track of the state of the design, but it can show that a grammar is realizable. If a list of rules touched is kept, it will be a sequence of rule applications which leads to a design in the language of the grammar. A grammar starts with the I node marked. At any point, if all nodes marked have no edges leading out, the grammar has terminated. Otherwise, follow* all *edges out from one marked node. For any edges that lead to a rule node, mark the rule node they point to. If an edge leads to a diamond, choose one edge leading out from the diamond and mark that rule node. After following edges, un-mark the original node.*

One rule b **follows** another rule a in a grammar G if a would be reachable (Definition 6.1) in a grammar identical to G but with its initial shape set to the right-hand side of a. If $b \prec a$ but $a \not\prec b$, then we will say that b **strictly follows** a. If $a \prec b$ and $b \prec a$, then we say that a and b can **recurse**. A pair of rules which *must* recurse, however, cannot lead to a design, as neither rule in that pair is realizable. An example is a pair of rules a, b where a is the only rule with a marker m in its left-hand side and has n in its right-hand side, and the only rule with n in its left-hand side has m

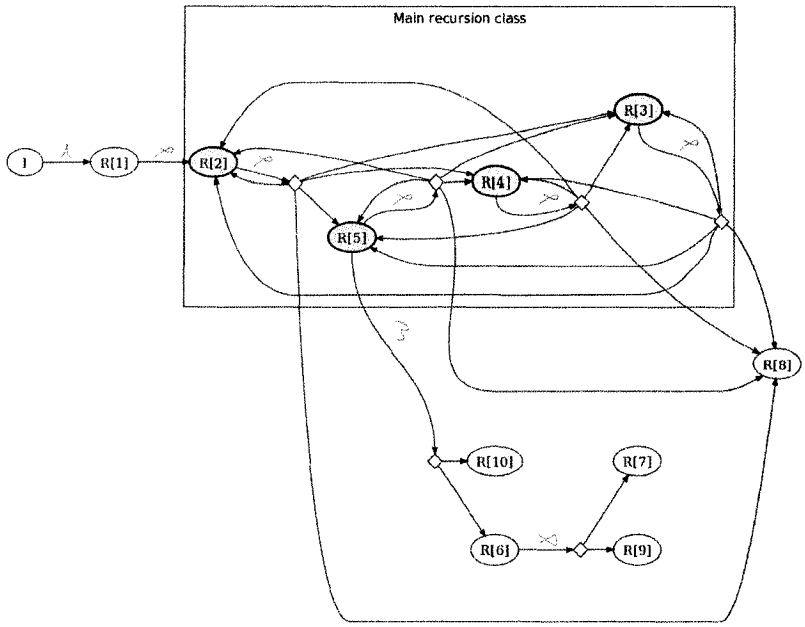

Figure 6.2: Directed graph of rules in the Kuba cloth grammar from Figure 2.2. Interpretation details are in the text of Section 6.2.

in its right-hand side. Larger loops and looplike structures can also break terminability. As these conditions break the grammar restrictions, we can safely assume that all rules that can recurse are realizable, but can be chained together into arbitrarily long sequences. This fits nicely with the idea of a strand from Section 1.2.1 and Chapter 4.

Having defined pairs of rules that recurse, we can use it as a relation to define equivalence classes of rules. Inside an equivalence class, we can look for symmetries.

$$a \equiv b(\text{recursion}) \iff a \text{ and } b \text{ can recurse}$$

This is almost an equivalence relation: $a \equiv b \iff b \equiv a$ by how "can recurse" is defined, and if $a \equiv c$ and $c \equiv b$, then there exist chains of rules from a to b and back, both via c. We can then define an antisymmetric, transitive relation $\prec$ between classes $\overline{r_i} \in \overline{R}$:

$$\overline{r_i} \prec \overline{r_j} \iff \exists(r_{ia} \in \overline{r_i}, r_{jb} \in \overline{r_j}) : r_{ia} \prec r_{jb}$$

Intuitively, the relationship exists between two classes if it exists between any members of the classes. The relationship is antisymmetric because if it went both ways, then chains would exist in both directions and the two classes would be equal. The one property that keeps it from being an equivalence relation is that some rules recurse with no rules, and some recurse with themselves. Rules in classes with more than one element always recurse with themselves, but there are rules which don't belong to any class. The relation is an equivalence relation among rules that can recurse, though, with classes of one or more rules, all of which have the property that they can repeat indefinitely. In Figure 6.1, each bold, grey rule node is a recursion class on its own. In Figure 6.2, there is one large recursion class in the box, other rules don't have recursion classes.

Each of these classes of rule is the primitive object in which we look for symmetry. Their relationships to each other and to the initial state determine their roles and how they combine.

The unclassed rules still play important roles in the design. One important type of rule which will never be in a recursion class is the type of rule which removes markers. For a design to be in the language of a grammar, all markers must be gone, so all grammars must include one or more marker-removing rules. Such rules may introduce one or more terminals into the design, or simply delete unneeded markers. As a marker-removing rule has no marker in its right-hand side, it will never be in a recursion class. Another type of non-classed rule is a "transition" rule which bridges (or forms part of a bridge) between two classes. Note that a transition rule is not necessary to get from one class to another: so long as there is no path back, any rule in a class can produce one or more markers which lead directly into other classes. Rules bridging between the initial state and minimal classes under $\prec$ are also possible.

6.2.1 Symmetries from recursion classes

Any loop in the directed graph of rules in a recursion class indicates the potential for a strand, with its fundamental unit being at most the result of applying all the rules in the loop in order. Examining the transformation resulting from the composition of all the rules in the loop can yield further information about the unit.

If the transformation is "parallel", i.e. a pure translation or a translation combined with a reflection on an axis parallel to the direction of translation, then the design from that loop is an open

strand (see Section 4.1). In this case, if there is no reflection then the translation unit divides the loop's translation. If there is reflection, then the design has glide symmetry with a vector equal to the loop's translation and an axis parallel to it running between rows of unflipped and flipped markers. As with all glide reflections, the translation unit is twice the glide one.

If the translation is not parallel, then again the symmetry depends on whether there is a reflection. If there is a reflection, then the loop generates a frieze symmetry, with a glide translation vector of the one-loop translation projected onto the translation resulting from two runs through the loop of rules. If there is no reflection, then the loop will result in a closed strand if the rotation of one application of the loop is commensurable with 2π, otherwise it will result in an infinite but bounded strand which possesses rotational symmetry (in this case, the symmetries of a circle) only as it approaches infinite repetitions.

6.2.2 Minimal and limiting symmetries

No element in the subset of the language $L(SG)$ of a shape grammar SG generated by a finite number of rule applications will ever have translational symmetry, let alone a wallpaper group: a wallpaper group predicts translational symmetries of arbitrarily large distance. Any scale-preserving grammar with a finite number of rules has a finite maximum translation distance k, hence for a finite sequence of m rule applications, no translational symmetry can be of a distance more than km units.

Nonetheless, with some grammars, we can fill arbitrarily large discs with a pattern that, if extended "naturally" (i.e. with perfectly overlapping translated copies mapping grid points on one copy to grid points on the other), would have true translational symmetry. If this is the case, then the grammar has a **limiting symmetry** for every symmetry posessed by the natural extension. If the grammar is composed entirely of closed strands not arranged on a lattice, then the limiting symmetry may just be the symmetries of a finite-application member of L(SG). This is in fact a degenerate case of the above definition: symmetries of finite designs have one point on their grid of translation symmetries: the origin. If points on the grid must map to each other, then all copies of the design must lie on top of each other.

Some element or elements of $L(SG)$ will also be the least symmetric. In some cases, this will be

the trivial symmetry (the identity function maps the design to itself and no other function does), but in other cases either the symmetries of the terminals or how they must be arranged will force a minimal symmetry on the design.

6.2.3 Symmetries from unclassed rules

Unclassed rules play different roles depending on whether they precede or follow a class. The definitions of precede and follow for these mixed types are as one would expect, treating the unclassed rule more or less as a class with one member:

$$x \prec \overline{r} \iff \exists r \in \overline{r} : x \prec r$$

and

$$\overline{r} \prec x \iff \exists r \in \overline{r} : r \prec x$$

Rules after a class, if they generate terminals, can add features to the designs created by that class, and should be absorbed into the symmetries of that class before that class is combined with copies of itself or other classes.

Rules without terminals before a class determine the number and transformations of copies of a class. Rules which generate terminals before any class will be applied a finite number of times, and can narrow the possibilities for design symetries considerably if they must be applied (see Section 6.2.4).

6.2.4 Mandatory and optional rules

In order to produce a design in a grammar's language, rules must be applied in such a way as to remove all markers created. In some cases, there is only one rule which removes a marker which necessarily occurs in a grammar. This rule is a **mandatory rule**. Such a rule, including any effects it has on design symmetry, is guaranteed to apply in every element of the language of the grammar.

One way to make a marker necessarily appear is to have it in the initial state. Another is to have it occur in a mandatory rule. A third is more complicated: if a marker is **locally mandatory** with respect to each of the rules which **immediately follows** a mandatory rule, then that marker is

necessary and it can in turn be used to determine mandatory rules.

Definition 6.4 (immediately follows) *A rule y immediately follows another rule x if the intersection of the right-hand side of x and the left-hand side of y is non-empty.*

Definition 6.5 (locally mandatory) *A rule y is locally mandatory with respect to a rule x if it is mandatory in a local grammar composed of all rules which follow x, with the right-hand side of x being the initial state for the local grammar.*

A mandatory rule can be applied more than once in a grammar if and only if it is in a recursion class with another rule. It may not, however, be in a singleton recursion class if the grammar is realizable, as its right-hand marker would then never disappear from the design state.

6.2.5 Strand copying

So far, we have defined how to break a grammar into manageable portions and extract symmetry information about the portions. What remains is to recursively combine the portions and their symmetry information to yield information about the entire grammar.

We have two types of grammar portions: **grid-building** portions which distribute other portions, and strandlike ones, which print designs spaced evenly by multiples of a translation vector with or without an accompanying rotation (which, in this case, could be a vector of zero length: hence, we include single-use rules and rosette-group generating rules). We can tell which group a rule or class belongs to simply: rules between the initial state and a recursion class are grid-builders, and any recursion class or rule following a recursion class is strandlike. Strandlike rules may have other strands branching off of them, but no rule in the first class can exhibit strandlike behaviour.

In order to use the lattice techniques of section 6.3, we need to apply the effects of grid-building rules to strandlike rules which follow them, then take the resulting symmetries and work with them on the symmetry lattices.

6.2.6 Examples

The grammars in Figures 2.2 and 2.3 yield the directed graphs of rules in Figures 6.1 and 6.2, respectively. The exact measurements and implementation details can be read off the XML source in Appendix A. Those grammars also include descriptive names and comments to aid in comprehension of how they work.

Ummayad zillij grammar

In the case of the zillij grammar's graph in Figure 6.1, its mandatory rules are $R[1]$, $R[3]$, $R[6]$ and $R[7]$. The recursion classes are three singleton sets: $R[2]$, $R[4]$ and $R[5]$. The fact that the set of mandatory rules and recursion classes are all pairwise disjoint is coincidental, but the fact that all the recursion classes are pairwise disjoint from each other stems from the properties of equivalence classes.

The initial state is established by $R[1]$: this is in fact a concession to the XML format, and not strictly necessary. In any case, it takes a single marker to another single marker with no transformation, and is mandatory, so does not affect the end symmetry.

Looking at $R[2]$, we find that the second marker symbol is translated 240 units in two orthogonal directions and that it has no symmetries or terminals laid down. At its limit, it establishes a $p1$ wallpaper group with a square grid. Minimally, it does nothing to the symmetry: the initial rule has a $c1$ (trivial) symmetry, and a finite number of translated copies of an asymmetric design does, too.

The first mandatory rule of much significance is $R[3]$: it takes all copies of what should be the only marker in the design, and creates four each of two types of marker, rotated evenly around a common centre with a radius of 255 units. All rules following $R[3]$ involving its markers will produce designs with centres of fourfold symmetry. There will be as many centres as were created by $R[2]$: they will tile the plane in the limiting case, and there will be only one if $R[2]$ was skipped.

The similarities between $R[4]$ and $R[5]$ are apparent: they extend strands of repeatedly-flipped copies of the same element forward and back. They both recurse independently. Looking at the terminal they copy, even shifting it to check for offset symmetry as described in Section 4.1, we find no symmetries aside from the glide symmetry forced by the repeated flipping and translation in

the rule. We can then assign this portion of the pattern a $p1a1$ frieze symmetry. The translation of one application of the rule is $(130, 350)$, but we cannot double it as it is not parallel to the axis of reflection: rather we add it to its reflection, $(350, 130)$, and get a translation vector of $(480, 480)$. To specify the group exactly, we need a glide axis intercept: the glide axis goes through the lines of slope 1 going through a point and its glide-reflected image, hence it goes through $\frac{(0,130)+(0,144)}{2} = (0, 137)$.

The role of $R[6]$ and $R[7]$ is to clean up markers: once both have been applied, the design consists only of terminals and a design in the language remains. There is a degenerate case where all markers can be removed without placing a single terminal. That empty canvas's symmetries include every conceivable affine transformation.

Four starting points for this frieze pattern are mandatory according to $R[3]$, in two parallel pairs (with parallel strands travelling in opposite directions). The translation period of these frieze patterns is twice the distance between centres, so in the limiting case of lots of rule applications, the members of the will be pairs will be reflections of each other. Combined with the known rotation centre, this means that if copies of the strands extend forever (or seem to because they fill the viewing port), then the whole design can have a symmetry group of $p4m$. Because there is only one patch of design left, we do not need to apply any lattice operations to this grammar. We have, therefore, a minimal symmetry of c_4 for small numbers of rule applications, a maximal non-degenerate symmetry of $p4m$, and the possibility of degenerate symmetries when this grammar yields a blank canvas.

Kuba textile grammar

The Kuba textile grammar in Figure 6.2 has mandatory rules $R[1]$ and $R[8]$, and one big recursion class consisting of $R[2]$, $R[3]$, $R[4]$ and $R[5]$.

Like the zillij example, $R[1]$ plays the role of the initial state for the XML interpreter, and doesn't change the symmetries of the design: it takes one marker and replaces it with another.

The meat of the grammar is in the main recursion class: $R[2]$ copies a marker with orthogonal translations of different lengths, $R[3]$ rotates it one half-turn, $R[4]$ places an asymmetric terminal and $R[5]$ places a symmetric terminal and a marker. These can be applied repeatedly in any order,

and the application of $R[8]$ breaks out of the class, leaving any remaining copies of the third and fourth markers to be cleaned up by $R[6]$, $R[7]$, $R[9]$ and $R[10]$.

$R[2]$, like its analogue in the zillij grammar, creates a grid of markers, and a similar (isomorphic but differently proportioned) $p1$ limiting symmetry. A centre of twofold rotation (or a grid of centres under $R[2]$ can be created by $R[3]$, which may affect the first terminal or the chain of copies of the square terminal created by $R[5]$, $R[6]$ and $R[7]$.

We are left with a choice of zero, one or two starting parts of design, possibly repeated using two orthogonal generating tranalations:

- The "hook" terminal can occur either alone (asymmetric) or with a rotated copy, establishing a c_2 rosette centre.
- The square terminal (posessing a d_8 rosette group when alone), possibly with a smaller copy (reducing the symmetry in question to d_2, or d_4 when repeated vertically see Section 4.1 for an illustration of offset symmetry for strands: this pattern is strandlike) and possibly a third copy that retains that symmetry. These chains of squares, if extended simultaneously or at different points in the rule application sequence, may or may not have centres of order 2.

  From these square terminals, then, we have either pure translations, d_4 centres coinciding with the "hook" centres if they exist, interspersed with reflections, or d_8 centres spaced horizontally between the d_2 centres of the "hooks", if applicable.

Note that this profusion of options is due to the grammar being flexible and supporting a varied language: in Figure 1.2, it is clear that the hook design is rotationally symmetric, and all the squares are in the pattern.

6.3 Lattice operations with post-processing

Once the grammar has yielded a set of designs with known symmetries, the lattice techniques from Chapter 5 allow combined symmeties to be assembled from the available pieces.

In combining two designs, first the translational symmetries of the output are required. Given the 4-tuples $\vec{a}, \vec{b}$ made up of the symmetries of the inputs, the translational symmetries of the output are $\vec{c} = \vec{a} \vee \vec{b}$ (in the vector lattice: joins and meets in this thesis are unambiguous, as each one from

Chapter 5 is associated with a different type of input). This establishes the translational symmetries of the result, but we still need to know whether rotation centres are common. If the symmetries turn out to be equal, then we have a frieze group. If the symmetries turn out to be infinite, then we have at most a rosette group (if centres coincide), or a design with only trivial symmetry.

Next, we take the meet of the sets of generating symmetries of both designs, remembering to include all subgroups of rotational centres: a 6-fold centre includes a 2-fold one and a 3-fold one, and a 4-fold centre includes a 2-fold one (duplicates can be ignored). We also count reflections as "long glides" in one case: a reflection axis r in a symmetry with translations $\vec{a}$ also counts as a glide in a join $\vec{c} = \vec{a} \vee \vec{b}$ if $\vec{a}$ and $\vec{c}$ have translations a,c parallel to r and $2|(\frac{||c||}{||a||})$. This check is only really necessary if the second symmetry has a glide reflection and no reflections, though. This represents a best-case set of symmetries.

To establish whether rotation centres coincide, let $\vec{d}$ be the difference of positions between known highest-order centres $\vec{r_a}, \vec{r_b}$. If $\vec{a} \wedge \vec{b} \preccurlyeq \vec{d}$, then the centres coincide and if $\vec{c}$ is formed from two independent vectors we have a grid of centres (of order yet to be determined) across the pattern. If the representative higher-order centres do not coincide, then additional centres should be considered depending on the order of the high-order ones. Let $\vec{c_0}$ be the higher-order centre and $\vec{t_1}, \vec{t_2}$ be the minimal independent translations for the design. For patterns with maximally rotationally-symmetric centres with a twofold subgroup, add twofold-centres at $\vec{c_0} + \frac{\vec{t_1}+\vec{t_2}}{2}$, $\vec{c_0} + \frac{\vec{t_1}}{2}$ and $\vec{c_0} + \frac{\vec{t_2}}{2}$. For patterns with centres with a threefold subgroup, add a threefold-centre at $\vec{c_0} + \frac{\vec{t_1}+\vec{t_2}}{2}$. For patterns with four-centres maximal, add an additional four-centre at $\vec{c_0} + \frac{\vec{t_1}+\vec{t_2}}{2}$. Check these centres pairwise (with a pair having an element from each design), from highest to lowest order, and take the first coinciding pair of centres.

If one of the patterns has no rotation centres but both have reflections or glide reflections, the same coincidence test can be done between parallel axes, taking points where the axes intersect a line parallel to a translation vector orthogonal to both. Additional reflection axes can be generated similarly to twofold centres, using only the orhogonal translation $\frac{\vec{t}}{2}$.

In both cases, features which do not coincide should be discarded. If centres exist in both patterns and do not coincide, then the resulting pattern is a simple $p1$ or $p111$, depending on whether the translation vectors of in $\vec{c}$ are independent.

Next, we check to see if the join that produced $\vec{c}$ is harmonized with respect to any axes present in both designs, as detailed in Section 5.3.3. This is essentially another coincidence test. If the join is not harmonized with respect to an axis, it should be dropped.

Whatever symmetries are left after dropping ones that don't coincide or for which the join isn't harmonized, generate the symmetries of the merged design. If any centres remain, then one can be found by searching points on a grid of points reachable via integer multiples of the vectors in $\vec{a} \wedge \vec{b}$. That information can then be put back into the process if more designs are to be combined to form the whole. The potential mass dropping of symmetries should make it abundantly clear why strict equality cannot be the only option in Section 4.3.

Once all elements have been combined, we have a design symmetry. If the graph of the grammar includes optional rules, the process should be run on different allowable combinations of designs.

6.3.1 Example

The previous section finished off the classification of the zillij mosaic, but the Kuba textile remains.

All of the parts mentioned can occur independently, so the grammar can definitely generate patterns of symmetry groups d_8, d_4, c_2 and c_1 and (with limiting symmetry) their two-translation cousins $p4m$, pmm, $p2$ and $p1$.

The grammars' centres coincide as follows:

$(0, 0)$ **and translations if applicable:** 2-centre in p_2

$(100, 0)$ **and translations if applicable:** flippable 4-centre in $p4m$; reflection axis $x = 100$ in pmm

$(100, 150)$ **and translations if applicable:** flippable 2-centre in pmm; reflection axis $x = 100$ in pmm; reflection axis $y = 150$ in pmm

Obviously, the $p1$ pattern contributes no symmetries.

Due to the organization of the grammar, all translational symmetries are the same despite differing centres in the various designs: translational symmetries are generated by $(200, 0)$ and $(0, 300)$.

The combinations allowed are essentially (asymmetric hooks **or** c_2 hooks **or** nothing) + a diamond pattern with (d_8 **or** d_4 **or** d_2) centres.

In the case of asymmetric hooks with anything, the set intersection yields only the translational symmetries (in the limiting case), so $p1$ and c_1 are possible final symmetries.

In the case of nothing from the first choice combined with anything, the diamond pattern's symmetries remain, and so the symmetries cited above for the diamond portions are possible final symmetries for the grammar as well.

This leaves the combination of the c_2 hooks with the various diamond patterns. The meet of sets of generating symmetries includes no reflections or glide reflections, only centres of two-fold rotational symmetry. The representative symmetries do not coincide, so we create two-fold centres at half-translations from the original c_2 pattern. These centres match, so we have another way of obtaining the p_2 wallpaper pattern (and one of these combinations yields the design pictured in Figure 1.2).

Having run through the symmetries available, we have a list whose maximal element (excluding the degenerate symmetry of the empty canvas) is $p4m$, and includes the symmetries mentioned above. The grammar's minimal symmetry is c_1, or $p1$ if the design must have a pair of translations.

6.4 Summary

This chapter covered a lot of ground, due in part to the tools developed in previous chapters.

Grammar restrictions gave us a method of writing and, indirectly, a method of interpreting grammars that remains flexible but brings some order to the classification process. The middle of the chapter addressed graphing and classing rules, using markers to establish edges to classify rules as mandatory or optional, recursive or not, by examining their place in the graph. Once these properties were established, we demonstrated how to build up symmetries portions of the design using "grid-building" and "strandlike" subgraphs. The idea of limiting symmetry reconciles a desire to classify designs as having wallpaper groups of symmetries while dealing with the finite nature of shape grammars.

The ideas above sufficed to completely classify the zillij pattern that has served as one of our examples.

We then detailed lattice operations and checks for coincidence for combining portions of design with known symmetries, bringing the lattice methods in Chapter 5 to bear and adding measures to

catch less-apparent symmetries. These tools allowed the classification of the example Kuba design to finish.

Chapter 7

Conclusion

This thesis has laid a framework for systematically examining a broad class of shape grammars and assessing the symmetries to be found in their languages, particularly in the case of maximally symmetric members. The purpose of this is to use the information in grammars to draw conclusions about their languages of designs, rather than "flattening" them into rendered patterns and searching for symmetries in the whole design.

The problems with a render-and-search strategy are several: first, many grammars have infinitely large languages, and picking arbitrary elements to examine for symmetries may or may not be representative of the grammar's range. If we use rules in the grammar to suggest transformations, then we may as well use the grammar rules for more than that. Also, combinations of rule transformations may be required, so a more systematic approach is preferable.

In examining the grammar more systematically, we need to overcome a few problems. First is the profusion of symmetries in some groups, and the specific ways they can occur together: not every mishmash of symmetries works in the plane. Second is the problem of finding these allowable symmetries in a shape grammar.

This thesis has described the key features and inputs in mathematical and visual terms, and designated the area of interest to coincide with grammar users' needs and expectations in the Generative Design Project. Part of that definition includes a standard list of generators for wallpaper groups, and ways of combining them using lattices, with the additional tool of checking for harmonized joins, to correct for skewing of translation units when designs with differently scaled and

oriented translation vectors are combined.

This thesis defines a precise class of grammars, defined in terms of how compliant grammars should be written from the user standpoint, which strengthens the meaning of various grammar features, particularly groups of rules that can recurse, and features which indicate order in grammars. The class defined still allows a grammar writer sufficient freedom to describe patterns of interest to Generative Design: two very different test cases are detailed throughout the thesis, working within the restrictions given.

The fundamental contribution is that of a system of ascribing symmetry information to these grammars which bypasses having to render elements in the language of a grammar. Much of the material in Chapters 5 and 6 was crafted especially for this problem, and is original work which applies standard and well-understood tools in a new ways to this problem.

Shape grammars, with their description of design by how elements fit together and by what decisions are made in their creation, are a natural candidate for examinations of symmetry. Much of the work on symmetry breaks patterns down into fundamental regions or strands, which can be represented quite well by shape grammars, but an explicit bridge from grammars has eluded my search if it exists.

Moving early from enumerated classes of ornamental groups to actual groups and using the concept of induced symmetry keeps the amount of information at various steps in the process manageable. Lattices highlight many of the important decisions in the process, compartmentalize it somewhat, and by careful definition ensure defined results from each step. Directed graphs and equivalence classes help carve a grammar into units that function together and influence each other.

This thesis increases the breadth of tools available to researchers examining repeating patterns. It introduces a package of techniques that grow out of established mathematics and are practical to understand and apply for mathematicians, computer scientists and interested researchers from less technical disciplines.

Appendix A

Sample grammar XML code

A.1 Ummayad zillij grammar

Below is the XML code, in the format outlined in Rajagopalan et al. [15], for the Ummayad zillij grammar in Figure 2.3. It provides exact lengths and transformations for all the rules illustrated. The markers are coded by colour rather than shape, due to the limitations of the input format.

```
<?xml version="1.0" encoding="utf-8"?>
<!--single-terminal "frieze" grammar for ummayadmsq_damas_r03_16, a p4m zillij mosaic with saft @Eric H.-->
<!DOCTYPE ShapeGrammar SYSTEM "sg_ver_0.06.dtd">
<ShapeGrammar>
    <ShapeList>

<!--
Marker Shapes
-->
        <Shape type="marker" shapeID="initial">
            <Circle cx="0" cy="0" r="5" color="255,0,255" opacity="1.0" stroke="1.0" />
        </Shape>
        <Shape type="marker" shapeID="centre">
            <Circle cx="0" cy="0" r="5" color="255,255,0" opacity="1.0" stroke="1.0" />
        </Shape>
        <Shape type="marker" shapeID="strandFwd">
            <Circle cx="0" cy="0" r="4" color="0,0,255" opacity="1.0" stroke="1.0" />
        </Shape>
        <Shape type="marker" shapeID="strandBack">
            <Circle cx="0" cy="0" r="3" color="255,0,0" opacity="1.0" stroke="1.0" />
        </Shape>

<!--
Terminal Shapes
-->

        <Shape type="terminal" shapeID="Strand">
            <Polyline color="64,64,64" opacity="0.0" stroke="1.0">
                <Point x="0" y="0" />
                <Point x="104" y="0" />
                <Point x="215" y="111" />
                <Point x="215" y="215" />
                <Point x="180" y="250" />
                <Point x="180" y="300" />
                <Point x="130" y="350" />
            </Polyline>
        </Shape>
        <Shape type="terminal" shapeID="null">
            <Circle cx="0" cy="0" r="0" color="255,255,255" opacity="0.0" stroke="1.0" />
        </Shape>

    </ShapeList>
    <RuleList><Rule ruleID="start">
            <Left>
                <CompositeShape>
                    <RuleShape shapeRef="initial" />
                </CompositeShape>
            </Left>
            <Right>
                <CompositeShape transformation="scale(0.35,0.35);">
                    <RuleShape shapeRef="centre" />
                </CompositeShape>
            </Right>
```

```
        </Rule>
        <Rule ruleID="dupCentre">
            <Left>
                <CompositeShape>
                    <RuleShape shapeRef="centre" />
                </CompositeShape>
            </Left>
            <Right>
                <CompositeShape>
                    <RuleShape shapeRef="centre" />
                    <RuleShape shapeRef="centre" transformation="translate(240,0);" />
                    <RuleShape shapeRef="centre" transformation="translate(0,240);" />
                </CompositeShape>
            </Right>
        </Rule>
        <Rule ruleID="spinStrands">
            <Left>
                <CompositeShape>
                    <RuleShape shapeRef="centre" />
                </CompositeShape>
            </Left>
            <Right>
                <CompositeShape>
                    <CompositeShape transformation="translate(-233,-103);">
                        <RuleShape shapeRef="strandBack" />
                        <RuleShape shapeRef="strandFwd" />
                    </CompositeShape>
                    <CompositeShape transformation="rotate(90);translate(-233,-103);">
                        <RuleShape shapeRef="strandBack" />
                        <RuleShape shapeRef="strandFwd" />
                    </CompositeShape>
                    <CompositeShape transformation="rotate(180);translate(-233,-103);">
                        <RuleShape shapeRef="strandBack" />
                        <RuleShape shapeRef="strandFwd" />
                    </CompositeShape>
                    <CompositeShape transformation="rotate(270);translate(-233,-103);">
                        <RuleShape shapeRef="strandBack" />
                        <RuleShape shapeRef="strandFwd" />
                    </CompositeShape>
                </CompositeShape>
            </Right>
        </Rule>
        <Rule ruleID="extStrandsBack">
            <Left>
                <CompositeShape>
                    <RuleShape shapeRef="strandBack" />
                </CompositeShape>
            </Left>
            <Right>
                <CompositeShape transformation="rotate(45);scale(1,-1);rotate(-45);translate(-130,-350);">
                    <RuleShape shapeRef="Strand" />
                    <RuleShape shapeRef="strandBack" />
                </CompositeShape>
            </Right>
        </Rule>
        <Rule ruleID="extStrandsFwd">
            <Left>
                <CompositeShape>
                    <RuleShape shapeRef="strandFwd" />
                </CompositeShape>
            </Left>
            <Right>
                <CompositeShape><RuleShape shapeRef="Strand" />
                    <RuleShape shapeRef="strandFwd" transformation="translate(130,350);rotate(45);scale(1,-1);rotate(-45);" />
                </CompositeShape>
            </Right>
        </Rule>
        <Rule ruleID="endBack">
            <Left>
                <CompositeShape>
                    <RuleShape shapeRef="strandBack" />
                </CompositeShape>
            </Left>
            <Right>
                <CompositeShape>
                    <RuleShape shapeRef="null" />
                </CompositeShape>
            </Right>
        </Rule>
        <Rule ruleID="endFwd">
            <Left>
                <CompositeShape>
                    <RuleShape shapeRef="strandFwd" />
                </CompositeShape>
            </Left>
            <Right>
                <CompositeShape>
                    <RuleShape shapeRef="null" />
                </CompositeShape>
            </Right>
        </Rule>
    </RuleList>
    <SequenceList>
        <Sequence ruleRef="start" />
        <Sequence ruleRef="dupCentre" />
        <Sequence ruleRef="dupCentre" />
        <Sequence ruleRef="dupCentre" />
        <Sequence ruleRef="dupCentre" />
        <Sequence ruleRef="dupCentre" />
        <Sequence ruleRef="spinStrands" />
        <Sequence ruleRef="extStrandsBack" />
        <Sequence ruleRef="extStrandsBack" />
        <Sequence ruleRef="extStrandsBack" />
        <Sequence ruleRef="extStrandsFwd" />
        <Sequence ruleRef="extStrandsFwd" />
        <Sequence ruleRef="extStrandsFwd" />
        <Sequence ruleRef="endBack" />
        <Sequence ruleRef="endFwd" />
    </SequenceList>
</ShapeGrammar>
```

A.2 Kuba textile grammar

Below is the XML code, in the format outlined in Rajagopalan et al. [15], for the Kuba textile grammar in Figure 2.2. It provides exact lengths and transformations for all the rules illustrated. The markers are coded by colour rather than shape, due to the limitations of the input format.

```
<?xml version="1.0" encoding="utf-8"?>
<!--two-terminal closed-strand grammar for Kuba 9706-007, a p2 Kuba textile from the collection of Cheryl Kolak Dudek @Eric H.-->
<!DOCTYPE ShapeGrammar SYSTEM "sg_ver_0.06.dtd">
<ShapeGrammar>
    <ShapeList>

<!--
Marker Shapes
-->
        <Shape type="marker" shapeID="initial">
            <Circle cx="0" cy="0" r="5" color="255,255,0" opacity="1.0" stroke="3.0" />
        </Shape>
        <Shape type="marker" shapeID="centre">
            <Circle cx="0" cy="0" r="5" color="0,255,0" opacity="1.0" stroke="0.75" />
        </Shape>
        <Shape type="marker" shapeID="bigDiamond">
            <Circle cx="0" cy="0" r="5" color="0,223,0" opacity="1.0" stroke="1.5" />
        </Shape>
        <Shape type="marker" shapeID="medDiamond">
            <Circle cx="0" cy="0" r="5" color="0,191,0" opacity="1.0" stroke="2.0" />
        </Shape>

<!--
Terminal Shapes
-->
        <Shape type="terminal" shapeID="null">
            <!--Kludge to simulate an empty RHS-->
            <Circle cx="0" cy="0" r="0" color="255,255,255" opacity="0.0" stroke="1.0" />
        </Shape>
        <Shape type="terminal" shapeID="Strand">
            <Polyline color="0,0,0" opacity="0.0" stroke="6.0">
                <Point x="0" y="0" />
                <Point x="84" y="84" />
                <Point x="42" y="126" />
                <Point x="0" y="84" />
                <Point x="-42" y="126" />
                <Point x="-84" y="84" />
                <Point x="-49" y="49" />
            </Polyline>
        </Shape>
        <Shape type="terminal" shapeID="Diamond">
            <Polygon color="0,0,0" opacity="1.0" stroke="1.0">
                <Point x="100" y="0" />
                <Point x="0" y="100" />
                <Point x="-100" y="0" />
                <Point x="0" y="-100" />
            </Polygon>
        </Shape>
    </ShapeList>
<!--
Rules
-->
    <RuleList>
        <Rule ruleID="start">
            <!--Kludge to symbolize an initial state-->
            <Left>
                <CompositeShape>
                    <RuleShape shapeRef="initial" />
                </CompositeShape>
            </Left>
            <Right>
                <CompositeShape>
                    <RuleShape shapeRef="centre" />
                </CompositeShape>
            </Right>
        </Rule>
        <Rule ruleID="dupCentre">
            <Left>
                <CompositeShape>
                    <RuleShape shapeRef="centre" />
                </CompositeShape>
            </Left>
            <Right>
                <CompositeShape>
                    <RuleShape shapeRef="centre" />
                    <RuleShape shapeRef="centre" transformation="translate(200,0);" />
                    <RuleShape shapeRef="centre" transformation="translate(0,300);" />
                    <RuleShape shapeRef="centre" transformation="translate(-200,0);" />
                    <RuleShape shapeRef="centre" transformation="translate(0,-300);" />
                </CompositeShape>
            </Right>
        </Rule>
        <Rule ruleID="rotCentre">
            <Left>
                <CompositeShape>
                    <RuleShape shapeRef="centre" />
                </CompositeShape>
            </Left>
            <Right>
                <CompositeShape transformation="rotate(180);">
                    <RuleShape shapeRef="centre" />
                </CompositeShape>
            </Right>
        </Rule>
        <Rule ruleID="printStrand">
            <Left>
                <CompositeShape>
                    <RuleShape shapeRef="centre" />
                </CompositeShape>
            </Left>
```

```
            <Right>
                <CompositeShape>
                    <RuleShape shapeRef="Strand" />
                    <RuleShape shapeRef="centre" />
                </CompositeShape>
            </Right>
        </Rule>
        <Rule ruleID="throwBigDiamond">
            <Left>
                <CompositeShape>
                    <RuleShape shapeRef="centre" />
                </CompositeShape>
            </Left>
            <Right>
                <CompositeShape>
                        <RuleShape shapeRef="bigDiamond" />
                    <CompositeShape transformation="translate(100,0);">
                        <RuleShape shapeRef="Diamond" transformation="scale(0.45,0.45);"/>
                    </CompositeShape>
                    <RuleShape shapeRef="centre" />
                </CompositeShape>
            </Right>
        </Rule>
        <Rule ruleID="throwMedDiamond">
            <Left>
                <CompositeShape>
                    <RuleShape shapeRef="bigDiamond" />
                </CompositeShape>
            </Left>
            <Right>
                <CompositeShape transformation="translate(0,150);">
                    <RuleShape shapeRef="medDiamond" />
                    <RuleShape shapeRef="Diamond" transformation="scale(0.3,0.3);"/>
                </CompositeShape>
            </Right>
        </Rule>
        <Rule ruleID="throwSmallDiamond">
            <Left>
                <CompositeShape>
                    <RuleShape shapeRef="medDiamond" />
                </CompositeShape>
            </Left>
            <Right>
                <CompositeShape transformation="translate(-100,0);">
                    <RuleShape shapeRef="Diamond" transformation="scale(0.15,0.15);"/>
                </CompositeShape>
            </Right>
        </Rule>
        <Rule ruleID="killCentre">
            <Left>
                <CompositeShape>
                    <RuleShape shapeRef="centre" />
                </CompositeShape>
            </Left>
            <Right>
                <CompositeShape>
                    <RuleShape shapeRef="null"/>
                </CompositeShape>
            </Right>
        </Rule>
        <Rule ruleID="killBigDiamond">
            <Left>
                <CompositeShape>
                    <RuleShape shapeRef="bigDiamond" />
                </CompositeShape>
            </Left>
            <Right>
                <CompositeShape>
                    <RuleShape shapeRef="null"/>
                </CompositeShape>
            </Right>
        </Rule>
        <Rule ruleID="killMedDiamond">
            <Left>
                <CompositeShape>
                    <RuleShape shapeRef="medDiamond" />
                </CompositeShape>
            </Left>
            <Right>
                <CompositeShape>
                    <RuleShape shapeRef="null"/>
                </CompositeShape>
            </Right>
        </Rule>
    </RuleList>
    <SequenceList>
        <Sequence ruleRef="start" />
        <Sequence ruleRef="dupCentre" />
        <Sequence ruleRef="dupCentre" />
        <Sequence ruleRef="dupCentre" />
        <Sequence ruleRef="printStrand" />
        <Sequence ruleRef="throwBigDiamond" />
        <Sequence ruleRef="rotCentre" />
        <Sequence ruleRef="printStrand" />
        <Sequence ruleRef="throwBigDiamond" />
        <Sequence ruleRef="throwMedDiamond" />
        <Sequence ruleRef="throwSmallDiamond" />
        <Sequence ruleRef="killCentre" />
    </SequenceList>
</ShapeGrammar>
```

Bibliography

[1] S. BHAKAR, C. K. DUDEK, S. MUISE, L. SHARMAN, E. HORTOP, AND F. SZABO, *Textiles, patterns and technology: Digital tools for the geometric analysis of cloth and culture*, Textile: The Journal of Cloth and Culture, 2 (2003), pp. 308–324.

[2] B. DAVEY AND H. PRIESTLEY, *Introduction to lattices and order*, Cambridge University Press, 2002.

[3] C. K. DUDEK, L. SHARMAN, F. E. SZABO, S. BHAKAR, E. HORTOP, , AND W. PAN, *From ethnomathematics to generative design: Metapatterns and interactive methods for the creation of decorative art*, in 8th International Conference on Information Visualisation, 2004.

[4] D. S. DUMMIT AND R. M. FOOTE, *Abstract Algebra*, Wiley, Hoboken, New Jersey, 3rd ed., 2004.

[5] D. EL-KHECHEN AND E. HORTOP, *Shape grammars as a representation for automated classification in patterned artefacts.* in preparation, 2007.

[6] J. GIPS, *Shape Grammars and their Uses: Artificial Perception, Shape Generation and Computer Aesthetics*, Birkhaüser Verlag, Basel and Stuttgart, 1975.

[7] B. GRÜNBAUM AND G. C. SHEPHARD, *Tilings and Patterns: An Introduction*, W H Freeman & Co, New York, 1989.

[8] ——, *Interlace patterns in islamic and moorish art*, Leonardo, 25 (1992), pp. 331–339.

[9] X. W. HUANG, *Groups, grammars and designs: A mathematical analysis of artifactual patterns*, Master's thesis, Concordia University, 2007.

[10] C. S. KAPLAN, *Computer graphics and geometric ornamental design*, PhD thesis, University of Washington, 2002.

[11] ——, *A mediation on Kepler's Aa*, in Bridges London, Proceedings of the 2006 Conference on Mathematical Connections in Art, Music and Science, R. Sarhangi and J. Sharp, eds., 2006, pp. 465–472.

[12] G. MEURANT, *Shoowa Design*, Thames and Hudson, London, UK, 1986.

[13] V. OSTROMOUKHOV, *Mathematical Tools for Computer-Generated Ornamental Patterns*, vol. 1375 of Lecture Notes in Computer Science, Springer-Verlag, Heidelberg, 1998, pp. 193–223.

[14] M. PHILLIPS, J. WEEKS, AND N. AMENTI, *Java Kali source code and documentation.* Website, archived at http://www.geom.uiuc.edu/java/Kali/source.html, September 1996.

[15] R. RAJAGOPALAN, E. HORTOP, D. EL-KHECHEN, C. K. DUDEK, L. SHARMAN, F. SZABO, T. FEVENS, AND S. MUDUR, *Inference and design in Kuba and zillij art with shape grammars,* in Bridges London, Proceedings of the 2006 Conference on Mathematical Connections in Art, Music and Science, London, UK, 2006.

[16] D. SCHATTSCHNEIDER, *The plane symmetry groups: Their recognition and notation,* American Mathematical Monthly, 85 (1978), pp. 439–250.

[17] G. STINY, *Pictoral and Formal Aspects of Shape and Shape Grammars: On Computer Generation of Aesthetic Objects,* Birkhauser Verlag, Basel, 1975.

[18] G. STINY AND J. GIPS, *Algorithmic Aesthetics: Computer Models for Criticism and Design in the Arts,* University of California Press, Los Angeles, 1978.

[19] D. K. WASHBURN AND D. W. CROWE, *Cultural insights from symmetry studies,* in Bridges London, Proceedings of the 2006 Conference on Mathematical Connections in Art, Music and Science, R. Sarhangi and J. Sharp, eds., 2006, pp. 19–24.

Index