

**DISTRIBUTIVE LATTICES AND REPRESENTATIONS
OF THE RANK TWO SIMPLE LIE ALGEBRAS**

A Thesis

Presented to

the Faculty of the Department of Mathematics and Statistics

Murray State University

Murray, Kentucky

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Leslie Wyatt Alverson II

July 2003

**DISTRIBUTIVE LATTICES AND REPRESENTATIONS
OF THE RANK TWO SIMPLE LIE ALGEBRAS**

DATE APPROVED: _____

Thesis Advisor

Member, Thesis Committee

Member, Thesis Committee

Collegiate Graduate Coordinator

Dean of the College

Graduate Studies Coordinator

Provost

ACKNOWLEDGMENTS

I take this opportunity to express my profound gratitude and deep regards to my advisor Dr. Rob Donnelly for his immense inspiration and guidance while preparing this thesis. Sincere appreciation is extended to Dr. Robert Pervine and Dr. Scott Lewis for taking the time to serve on my committee during this endeavor.

I am also indebted to my colleagues and friends Jon Ernstberger, Ken MacPherson, Richard Robinson, and Andy Perkins for their interest and encouragement throughout the writing of this thesis.

I would also like to express my gratitude to the faculty and administration of Murray State University for providing the opportunity and facilities to complete this work, and particularly to Dr. Don Bennett, the Chair of the Department of Mathematics and Statistics, for his support.

Finally, my deepest gratitude goes to my parents Leslie and the late Susan Alverson, siblings Melissa, Jessica, Elizabeth, and Joshton, girlfriend Elizabeth Doss, and son Ethan Wyatt Alverson, for their support, understanding, patience, and love.

To all of you, thank you.

ABSTRACT

Three families of distributive lattices (so-called “semistandard lattices for A_2 , B_2 , and G_2 ”) which arise naturally in a certain algebraic context are considered. Two of these families of semistandard lattices (A_2 and G_2) were studied previously by McClard [Mc]. From a combinatorial point of view, the lattices are quite striking and exhibit some pleasant symmetries and certain enumerative niceties. Evidence is presented suggesting that many of these lattices provide a suitable combinatorial environment for realizing linear representations of certain Lie algebras. This evidence includes results of McClard and others which led to a conjecture that all semistandard lattices could be used to realize Lie algebra representations. This conjecture has been confirmed for several special classes of semistandard lattices. A primary contribution of this thesis is a demonstration that the conjecture fails for many other classes of semistandard lattices.

TABLE OF CONTENTS

LIST OF FIGURES	vi
Chapter 1 INTRODUCION	1
Lie algebra actions	5
Rank two simple Lie algebras acting on posets	6
Identifying posets as supporting graphs	7
The Semistandard Lattice Conjecture and our main results	12
Chapter 2 DISTRIBUTIVE LATTICES AND REPRESENTATIONS OF A_2	17
Chapter 3 DISTRIBUTIVE LATTICES AND REPRESENTATIONS OF B_2	26
Chapter 4 DISTRIBUTIVE LATTICES AND REPRESENTATIONS OF G_2	38
Chapter 5 KEY ALGORITHMS FOR INVESTIGATING THE SEMISTANDARD LATTICE CONJECTURE	45
Chapter 6 APPLICATIONS OF ALGORITHMS	57
B_2 Counterexamples to the Semistandard Lattice Conjecture	57
Examples of Semistandard Lattices	62
APPENDICES	71
Appendix A Maple Implementation	71
Appendix B Data for the Rank Two Lie Algebras	94
BIBLIOGRAPHY	102

LIST OF FIGURES AND TABLES

Table 1.1	Does the SLC hold for B_2 -semistandard lattice L_λ when $\lambda = (a, b)$?	14
Table 1.2	Does the SLC hold for G_2 -semistandard lattice L_λ when $\lambda = (a, b)$?	15
Figure 2.1	The A_2 -semistandard lattice corresponding to the shape $\lambda = (1, 1)$	18
Figure 2.2	4-element poset of “irreducibles”	19
Figure 3.1	The B_2 -semistandard lattice corresponding to the shape $\lambda = (1, 1)$	27
Figure 3.2	7-element poset of “irreducibles”	28
Table 4.1	Filling restrictions for G_2 -tableaux	38
Figure 4.1	The G_2 -semistandard lattice corresponding to the shape $\lambda = (0, 1)$	39
Figure 3.2	10-element poset of “irreducibles”	40
Figure 6.1	Example lattice of the B_2 -semistandard lattice corresponding to $\lambda = (2, 1)$	58
Figure 6.2	Clarification of vertices 13 and 14 for Figure 6.1	58
Figure 6.3	Top 5 ranks (levels) in a generic B_2 -semistandard lattice	60
Table 6.1	Vertices in the top 5 ranks (levels) of a generic B_2 -semistandard lattice	61
Figure 6.4	The A_2 -semistandard lattice corresponding to shape $\lambda = (1, 1)$	63
Figure 6.5	The A_2 -semistandard lattice corresponding to shape $\lambda = (5, 5)$	64
Figure 6.6	The A_2 -semistandard lattice corresponding to shape $\lambda = (10, 10)$	65
Figure 6.7	The B_2 -semistandard lattice corresponding to shape $\lambda = (1, 1)$	66
Figure 6.8	The B_2 -semistandard lattice corresponding to shape $\lambda = (5, 0)$	67

Figure 6.9	The B_2 -semistandard lattice corresponding to shape $\lambda = (0, 5)$	68
Figure 6.10	The G_2 -semistandard lattice corresponding to shape $\lambda = (0, 1)$	69
Figure 6.11	The G_2 -semistandard lattice corresponding to shape $\lambda = (5, 0)$	70

CHAPTER 1

INTRODUCTION

The subject of this thesis lies in the intersection of combinatorics and Lie algebra representation theory. Although it will take just a little time to make the next statement precise, for now we can state the main goal of this thesis as follows: we aim to investigate a conjecture on what the actions of rank two simple Lie algebras on finite-dimensional vector spaces “look like” when presented as actions on finite partially ordered sets. This might seem like a strange problem at first: what is a Lie algebra action anyway, and why might such an action be of combinatorial interest? But indeed there is growing evidence (see for example [Don] and [Eve]) that actions of simple Lie algebras on finite-dimensional vector spaces are connected with some unexpectedly rich combinatorial structures.

Lie theory itself is a very deep and perhaps fundamental subject, connected to areas as diverse as topology (e.g. knot theory), special functions, enumeration, and physics. (See [How] for a readable survey and introduction to the subject. Two recent book reviews [Row], [Kna] make for illuminating reading as well: Rowe reviews a book on the early history of Lie theory, and Knapp reviews two new introductory Lie theory texts; these books might be of interest to the curious reader.) Many connections between combinatorics and Lie theory have been explored in the last half-century or so. Many of the applications go in one direction, however: Lie theoretic techniques are applied to prove results that are, in their statements, purely combinatorial. In [Pr], for example, one finds a lively and

interesting exposition of two combinatorial problems that can be resolved using Lie algebra representation theory. Broadly speaking, we are interested in an application that goes in the other direction: we want to know when certain combinatorial information is sufficient to “construct” or “recover” Lie algebra representations.

It follows from a beautiful classification theorem due to Cartan that there are precisely three “rank two simple Lie algebras,” denoted A_2 , B_2 , and G_2 respectively. Each of these Lie algebras is a finite-dimensional complex vector space equipped with a certain non-associative “multiplication.” This multiplication is denoted by a “bracket,” so for Lie algebra elements x and y , their product is written $[xy]$. The Lie algebra A_2 has dimension eight as a vector space over the complex numbers, the Lie algebra B_2 has dimension ten, and the Lie algebra G_2 has dimension fourteen. Each of these Lie algebras is “simple” in the sense that it does not have any nontrivial proper vector subspaces that are “inside-outside closed.” Or, put another way, let \mathfrak{g} be one of A_2 , B_2 , or G_2 , and let I be a vector subspace of \mathfrak{g} with the property that for any x in \mathfrak{g} and y in I , the product $[xy]$ remains in I . The simplicity of \mathfrak{g} means that I must be the trivial subspace $\{0\}$ or that I must be all of \mathfrak{g} . A simple Lie algebra is the analog of a simple group; a simple Lie algebra has no nontrivial proper “ideals.” Finally, the Lie algebras A_2 , B_2 , and G_2 are “rank two” in the sense that each has a nice set of generators consisting of two “ x ’s,” two “ y ’s,” and two “ h ’s.” That is, when \mathfrak{g} is one of A_2 , B_2 , or G_2 , we have

$$\mathfrak{g} = \langle x_1, y_1, h_1, x_2, y_2, h_2 \mid \text{certain relations} \rangle$$

The “certain relations” mentioned above are called “Serre relations” (these are recorded explicitly in [Mc] p. 19). These relations are somewhat tedious to produce, but for each

algebra A_2 , B_2 , or G_2 , the critical information needed to write down the Serre relations is encoded in the “Cartan matrix.” The Cartan matrix uniquely identifies a simple Lie algebra in the sense that one can use the Cartan matrix to recover the defining Serre relations for the Lie algebra (see for example Theorem 2.2.2 in [Eve]). For the rank two simple Lie algebras the Cartan matrices are:

Lie Algebra	Associated Cartan Matrix
A_2	$\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$
B_2	$\begin{pmatrix} 2 & -1 \\ -2 & 2 \end{pmatrix}$
G_2	$\begin{pmatrix} 2 & -1 \\ -3 & 2 \end{pmatrix}$

In [Mc] one can find a very readable introduction to the main combinatorial and Lie algebra representation theoretic notions needed to understand our main results. We will not attempt to duplicate that exposition here; the reader is referred to that thesis for any Lie algebra or combinatorics terminology not defined here. Much of the exposition in [Mc] is drawn from [Hum] (Lie algebra representation theory basics) and from [Sta] (combinatorics background). Some parts of this thesis will depend crucially on other results from [Mc] as well. In some sense, this thesis extends McClard’s discussion of the representations of the simple Lie algebras A_2 and G_2 to include the representations of the simple Lie algebra B_2 (see Chapter 3). This thesis will also provide a partial answer to a generalization of Conjecture 6.2.3 of [Mc].

This conjecture, formulated in generality by Donnelly as a result of collaborations with Scott Lewis, Marti McClard, Robert Pervine, and Norman Wildberger (henceforth referred to as the Semistandard Lattice Conjecture, or SLC), proposes that all representations of the rank two simple Lie algebras can be “constructed” in a certain sense from some very pretty families of distributive lattices. A solid body of evidence for the veracity of the SLC has been compiled in recent years. In Chapters 2, 3, and 4 we survey some of these results, which are interesting in their own right.

One of the main contributions of this study is many counterexamples to the Semistandard Lattice Conjecture (see Table 1.1, Table 1.2, and Theorem 6.1). These counterexamples were initially found by computer experimentation. By suitably modifying some fairly involved algorithms developed by Donnelly, Lewis, and Pervine [DLP2], we were able to use Maple to perform the many tedious computations needed to check the SLC. These algorithms (Chapter 5), along with the Maple code which implemented these algorithms (Appendix A) as well as their output and applications (Chapter 6) are among the other mathematical contributions of this thesis. The smallest counterexample to the SLC we found has dimension 35, which is small enough to be verified by hand. We have subsequently been able to verify with formal proofs many other counterexamples (see Theorem 6.1). Although these counterexamples show the SLC fails in full generality, it does hold for certain special infinite families (see Theorems 2.6, 3.6 and 4.6). These results have led us to a refinement of the conjecture. Finally, it should be noted that our work leaves open the question of whether other families of distributive lattices might work in place of those used to formulate the SLC.

In the “Lie algebra actions” and the “Rank two simple Lie algebras acting on posets” subsections below, we develop some of the technical language needed to more fully understand our results. In the “Identifying posets as supporting graphs” subsection below, we motivate the results which will be recorded in Chapters 2, 3, and 4. Finally, we close this section with a precise statement of the Semistandard Lattice Conjecture and give a summary of our investigations which in some cases verify and in some cases disprove the conjecture. The details of the algorithms we use, along with their applications, Maple implementation, and output, are contained in Chapters 5, 6, and Appendix A.

Lie algebra actions

Let \mathfrak{g} be a simple Lie algebra, and suppose \mathfrak{g} acts on a complex d -dimensional vector space V by way of the Lie algebra homomorphism

$$\phi : \mathfrak{g} \longrightarrow gl(V).$$

Here $gl(V) := \{T \mid T \text{ is a linear mapping from } V \text{ to } V\}$ is a d^2 -dimensional complex vector space with Lie bracket “multiplication” defined by $[S, T] := ST - TS$ for S, T in $gl(V)$. The homomorphism ϕ is a linear mapping satisfying $\phi([x, y]) = [\phi(x), \phi(y)]$ for x, y in \mathfrak{g} . We say that ϕ is a “representation” of \mathfrak{g} since it allows us to view elements of \mathfrak{g} as linear transformations; abusing notation, we often say that V is a “representation” of \mathfrak{g} when the mapping ϕ is implied. Since \mathfrak{g} is simple, the kernel $\ker(\phi)$ must be trivial or all of \mathfrak{g} . Assuming that ϕ is not a trivial mapping sending everything to 0, i.e. assuming $\ker(\phi) \neq \mathfrak{g}$, then it follows that $\ker(\phi) = \{0\}$, and so ϕ is injective. For $x \in \mathfrak{g}$ and $v \in V$, it is customary to write

$$x.v := \phi(x)(v).$$

In this case notice that (1) $x.(av + bw) = ax.v + bx.w$, (2) $(\alpha x + \beta y).v = \alpha(x.v) + \beta(y.v)$, and (3) $[x, y].v = x.y.v - y.x.v$. This notation gives another perspective on what it means to have a Lie algebra “acting on” a vector space. The action ϕ of \mathfrak{g} on V is “irreducible” if V has no nontrivial proper subspaces that “remain stable” under the action of \mathfrak{g} . That is, ϕ is irreducible if whenever W is a vector subspace of V for which $x.w$ remains in W for all x in \mathfrak{g} and w in W , then W must be either the trivial subspace $\{0\}$ or W must be all of V .

Rank two simple Lie algebras acting on posets

For now let us limit our consideration to the actions of simple Lie algebras of rank two. Suppose \mathfrak{g} is a rank two simple Lie algebra acting on a complex vector space V via the homomorphism $\phi : \mathfrak{g} \longrightarrow gl(V)$. Moreover, suppose that the representation ϕ is irreducible. Another wonderful classification result states that there is precisely one irreducible representation $\phi : \mathfrak{g} \longrightarrow gl(V)$ corresponding to each pair (a, b) of non-negative integers a and b . Let us briefly explain what this classification result means. First, when $\phi : \mathfrak{g} \longrightarrow gl(V)$ is irreducible, then in V there exists a unique nonzero vector v (up to scalar multiples) such that $x_1.v = 0 = x_2.v$. Moreover, it is the case that $h_1.v = av$ and $h_2.v = bv$ for some non-negative integers a and b . This vector v is called the “maximal vector” for the irreducible representation. The pair of integers (a, b) corresponding to the maximal vector v is called the “dominant weight,” and is usually denoted by λ . Second, the classification theorem tells us that given any pair of non-negative integers $\lambda = (a, b)$, there exists an irreducible representation $\phi : \mathfrak{g} \longrightarrow gl(V)$ with maximal vector v having dominant weight λ . And third, any two irreducible representations with the same dominant weight are “equivalent.” That is, suppose $\psi : \mathfrak{g} \longrightarrow gl(W)$ is another irreducible representation

with dominant weight $\lambda = (a, b)$. Then there exists an isomorphism $f : V \longrightarrow W$ of vector spaces such that $f \circ \phi(x) = \psi(x) \circ f$ for all x in \mathfrak{g} .

Suppose now that $\phi : \mathfrak{g} \longrightarrow gl(V)$ is irreducible. Assuming V has dimension d , let $\{v_s\}_{s \in P}$ be a basis for V , where the index set P has size d . We will create an edge-colored directed graph using the elements of P as vertices. Let \mathbf{s} and \mathbf{t} be elements of P and let i be either 1 or 2. We can expand $x_i.v_s$ and $y_i.v_t$ relative to the basis $\{v_s\}_{s \in P}$. That is, write $x_i.v_s = \sum_{\mathbf{x} \in P} c_{\mathbf{x},s} v_{\mathbf{x}}$ and $y_i.v_t = \sum_{\mathbf{x} \in P} d_{\mathbf{x},t} v_{\mathbf{x}}$. Place a directed edge of color i from \mathbf{s} to \mathbf{t} if $c_{\mathbf{t},s} \neq 0$ or $d_{\mathbf{s},t} \neq 0$. In this case we write $\mathbf{s} \xrightarrow{i} \mathbf{t}$. We call the set P together with this collection of colored, directed edges the *supporting graph* for the basis $\{v_s\}_{s \in P}$ of the representation $\phi : \mathfrak{g} \longrightarrow gl(V)$.

What will a supporting graph P look like? It could be unreasonably messy for all we know. But, as it turns out, any edge-colored directed graph arising in this way will be connected, and will be the Hasse diagram for a rank symmetric and rank unimodal partially ordered set. As it turns out, for some well chosen bases for certain representations, the posets corresponding to the supporting graphs are distributive lattices. This leads to the main question we are concerned with:

Given an irreducible representation of a rank two simple Lie algebra, can we find a distributive lattice and color the edges of its Hasse diagram in such a way that we will be looking at a supporting graph for the representation?

Identifying posets as supporting graphs

Given an edge-colored directed graph P , what does it take in order to know that P arises as the supporting graph obtained by fixing some basis for an irreducible representation of a

rank two simple Lie algebra? Let \mathfrak{g} be a simple Lie algebra of rank two, and let $\lambda = (a, b)$ be a dominant weight. Consult [Don] and [DLP1] for justification that the following requirements and conditions on P are necessary for P to be a supporting graph.

- 1. Combinatorial Requirement for $\mathfrak{g}(\lambda)$.** If an edge-colored directed graph P is a supporting graph for the irreducible representation of \mathfrak{g} with dominant weight λ , then (1) P must be a connected graph with just two edge colors (“1” or “2”), (2) P must be the Hasse diagram for a ranked poset, and (3) P must have a unique maximal and a unique minimal element. An edge-colored directed graph P meets the Combinatorial Requirement for $\mathfrak{g}(\lambda)$ if it satisfies properties (1), (2), and (3).

REMARK: In addition, a supporting graph P for an irreducible representation of \mathfrak{g} must be rank symmetric and rank unimodal. Rank symmetry is often easy to prove directly. On the other hand, rank unimodality is usually difficult to verify directly, and is something one normally concludes after demonstrating the Structure and Character Conditions for $\mathfrak{g}(\lambda)$ (discussed below).

- 2. Dimension Requirement for $\mathfrak{g}(\lambda)$.** In the 1920’s Hermann Weyl worked out formulas for the dimensions of the irreducible representations of \mathfrak{g} .

Algebra	Dimension of irreducible module corresponding to dominant weight $\lambda = (a, b)$
A_2	$\frac{1}{2}(a+1)(b+1)(a+b+2)$
B_2	$\frac{1}{3!}(a+1)(b+1)(a+b+2)(a+2b+3)$
G_2	$\frac{1}{5!}(a+1)(b+1)(a+b+2)(a+2b+3)(a+3b+4)(2a+3b+5)$

If P is to be a supporting graph for the irreducible representation of \mathfrak{g} corresponding to the dominant weight λ , then the size of the vertex set for P must agree with

the appropriate formula from the above table. So we say P meets the Dimension Requirement for $\mathfrak{g}(\lambda)$ if $|P|$ agrees with Weyl's dimension formula for the irreducible representation of \mathfrak{g} with dominant weight λ .

- 3. Structure Condition for \mathfrak{g} .** Fix an edge color $i = 1$ or 2 . For any \mathbf{s} in P , the “ i -component containing \mathbf{s} ” consists of all vertices of P which can be reached from \mathbf{s} by walking along edges of color i , together with all edges of color i connecting these vertices. Note that the i -component containing \mathbf{s} is automatically connected. If \mathbf{s} is maximal (resp. minimal) in some i -component, we say \mathbf{s} is “ i -maximal” (resp. “ i -minimal”) in P . Since it has already been checked that P is the Hasse diagram for a connected ranked poset, it follows that the i -component containing \mathbf{s} is also the Hasse diagram for some ranked poset. Define $\rho_i(\mathbf{s})$ to be the rank of the element \mathbf{s} in its i -component, and let $l_i(\mathbf{s})$ be the length of the i -component containing \mathbf{s} . Define

$$wt(\mathbf{s}) = (2\rho_1(\mathbf{s}) - l_1(\mathbf{s}), 2\rho_2(\mathbf{s}) - l_2(\mathbf{s}))$$

We say P satisfies the Structure Condition for \mathfrak{g} if for each edge $\mathbf{s} \xrightarrow{i} \mathbf{t}$ in P we have

$$wt(\mathbf{s}) + \alpha_i = wt(\mathbf{t}),$$

where α_i is the i th row vector of the Cartan matrix corresponding to \mathfrak{g} .

REMARK: If \mathbf{m} is the unique maximal element of P , then for P to be a supporting graph for the irreducible representation corresponding to λ , we need it to be the case (among other things) that $wt(\mathbf{m}) = \lambda$.

- 4. Character Condition for $\mathfrak{g}(\lambda)$.** Corresponding uniquely to the irreducible representation V of \mathfrak{g} with dominant weight λ is a Laurent polynomial in the variables X and

Y which has non-negative integral coefficients. We denote this polynomial, called the “character” of the representation V , by the notation $char_\lambda(X, Y)$, and write

$$char_\lambda(X, Y) = \sum_{\mu=(\mu_1, \mu_2) \in \mathbb{Z} \times \mathbb{Z}} d_\lambda(\mu) X^{\mu_1} Y^{\mu_2}.$$

In particular, it is the case that each $d_\lambda(\mu)$ is a non-negative integer, and that $char_\lambda(1, 1) = \dim(V)$. Thus, only finitely many of the $d_\lambda(\mu)$'s can be non-zero. In fact, each non-zero $d_\lambda(\mu)$ counts the dimension of a certain eigen-subspace of V . See [Hum] for a thorough discussion of the character polynomial.

We say P satisfies the Character Condition for $\mathfrak{g}(\lambda)$ if $d_\lambda(\mu) = |\{\mathbf{s} \in P \mid wt(\mathbf{s}) = \mu\}|$ for each μ in $\mathbb{Z} \times \mathbb{Z}$.

REMARK: In practice, this is quite hard to check directly, and one must usually appeal to some more sophisticated algebraic arguments in order to verify this condition. For the lattices we consider in Chapters 2, 3, and 4, we appeal to work of Littelmann [Lit] to settle this question. Note that the Character Condition implies the Dimension Requirement. If P meets the Combinatorial Requirement, the Structure Condition, and the Character Condition, then P is rank symmetric and rank unimodal.

- 5. Actions.** All of the preceding requirements and conditions are necessary, but not sufficient, to conclude that P is a supporting graph. However, if P is the Hasse diagram for a distributive lattice, if P meets the Combinatorial Requirement for $\mathfrak{g}(\lambda)$, the Dimension Requirement for $\mathfrak{g}(\lambda)$, and the Structure Condition for \mathfrak{g} , and if the weight of the maximal element of P is λ , then a description of “actions” as follows is sufficient to imply that P is indeed a supporting graph for the irreducible representation of \mathfrak{g} with highest weight λ . In particular, sorting out “actions” will automatically imply that

the Character Condition is satisfied! As it turns out, however, explicitly describing “actions” from scratch can be quite difficult.

To describe “actions,” one must attach an “ x -coefficient” $c_{\mathbf{t},\mathbf{s}}$ and a “ y -coefficient” $d_{\mathbf{s},\mathbf{t}}$ to each edge $\mathbf{s} \xrightarrow{i} \mathbf{t}$ in P . (The x -coefficient should be thought of as “acting” in the “up” direction, i.e. in the direction of the edge $\mathbf{s} \xrightarrow{i} \mathbf{t}$, while the y -coefficient “acts” in the “down” direction.) Let $\pi_{\mathbf{s},\mathbf{t}}$ denote the product $c_{\mathbf{t},\mathbf{s}}d_{\mathbf{s},\mathbf{t}}$ of the x -coefficient and y -coefficient on this edge. These coefficients must meet the following two conditions:

(1) The Diamond Condition: Whenever elements \mathbf{r} , \mathbf{s} , \mathbf{t} and \mathbf{u} form a diamond of

edges $\mathbf{t} \begin{array}{c} \nearrow \mathbf{u} \\ \searrow \mathbf{r} \end{array} \mathbf{s}$ in P , then $c_{\mathbf{u},\mathbf{s}}d_{\mathbf{t},\mathbf{u}} = d_{\mathbf{r},\mathbf{s}}c_{\mathbf{t},\mathbf{r}}$ and $c_{\mathbf{u},\mathbf{t}}d_{\mathbf{s},\mathbf{u}} = d_{\mathbf{r},\mathbf{t}}c_{\mathbf{s},\mathbf{r}}$.

(2) The Crossing Condition: For any \mathbf{s} in P and any color i ,

$$\sum_{\mathbf{r}:\mathbf{r}\xrightarrow{i}\mathbf{s}} \pi_{\mathbf{r},\mathbf{s}} - \sum_{\mathbf{t}:\mathbf{s}\xrightarrow{i}\mathbf{t}} \pi_{\mathbf{s},\mathbf{t}} = 2\rho_i(\mathbf{s}) - l_i(\mathbf{s}).$$

In what sense do these coefficients describe “actions”? Suppose P is a distributive lattice meeting all of the above requirements, and suppose we have assigned edge coefficients which meet the Diamond and the Crossing Conditions. Now let $V[P]$ denote the complex vector space freely generated by $\{v_{\mathbf{s}}\}_{\mathbf{s}\in P}$. Recall that \mathfrak{g} has the special set of generators $\{x_1, y_1, h_1, x_2, y_2, h_2\}$. Define linear transformations X_1, Y_1, H_1, X_2, Y_2 , and H_2 acting on $V[P]$ as follows:

$$X_i(v_{\mathbf{s}}) := \sum_{\mathbf{t}:\mathbf{s}\xrightarrow{i}\mathbf{t}} c_{\mathbf{t},\mathbf{s}}v_{\mathbf{t}} \quad Y_i(v_{\mathbf{t}}) := \sum_{\mathbf{s}:\mathbf{s}\xrightarrow{i}\mathbf{t}} d_{\mathbf{s},\mathbf{t}}v_{\mathbf{s}} \quad H_i(v_{\mathbf{s}}) := (2\rho_i(\mathbf{s}) - l_i(\mathbf{s}))v_{\mathbf{s}}$$

for $i = 1$ or $i = 2$. The Diamond, Crossing, and Structure Conditions are now sufficient to guarantee that $\{X_1, Y_1, H_1, X_2, Y_2, H_2\}$ satisfy the Serre relations for \mathfrak{g} . For example, one can use the Diamond Condition to check that X_i commutes with Y_j

when $i \neq j$. In [Don] one can find the details for an argument that the remaining Serre relations hold. We can now let $\phi : \mathfrak{g} \longrightarrow gl(V[P])$ be the Lie algebra homomorphism induced by mapping $x_i \xrightarrow{\phi} X_i$, $y_i \xrightarrow{\phi} Y_i$, and $h_i \xrightarrow{\phi} H_i$ for $i = 1, 2$. Then $\phi : \mathfrak{g} \longrightarrow gl(V[P])$ is an irreducible representation of \mathfrak{g} with dominant weight λ . Moreover, the graph P is the supporting graph for the basis $\{v_s\}_{s \in P}$ for this action of \mathfrak{g} in $V[P]$.

REMARK: Finding two coefficients $c_{\mathbf{t},\mathbf{s}}$ and $d_{\mathbf{s},\mathbf{t}}$ to attach to each edge $\mathbf{s} \xrightarrow{i} \mathbf{t}$ in P can be a chore. When P is a distributive lattice, it actually suffices to simply supply a single nonzero complex coefficient $\pi_{\mathbf{s},\mathbf{t}}$ to each edge. The Crossing Condition remains the same, but the Diamond Condition simplifies to checking that $\pi_{\mathbf{s},\mathbf{u}}\pi_{\mathbf{t},\mathbf{u}} = \pi_{\mathbf{r},\mathbf{s}}\pi_{\mathbf{r},\mathbf{t}}$ whenever elements \mathbf{r} , \mathbf{s} , \mathbf{t} and \mathbf{u} form a diamond of edges $\mathbf{t} \begin{array}{c} \nearrow \mathbf{u} \\ \searrow \mathbf{r} \end{array} \mathbf{s}$ in P . Then on any edge $\mathbf{s} \xrightarrow{i} \mathbf{t}$, use the principal square root of $\pi_{\mathbf{s},\mathbf{t}}$ for $c_{\mathbf{t},\mathbf{s}}$ and $d_{\mathbf{s},\mathbf{t}}$.

The Semistandard Lattice Conjecture and our main results

In Chapters 2, 3, and 4, we define certain families of edge-colored distributive lattices which we will call “semistandard lattices.” For a given rank two simple Lie algebra \mathfrak{g} and a given dominant weight $\lambda = (a, b)$, there is precisely one \mathfrak{g} -semistandard lattice L_λ . Semistandard lattices are built from certain combinatorial objects called “tableaux.” The main content of Chapters 2, 3, and 4 is to confirm the Combinatorial Requirement, the Dimension Requirement, the Structure Condition, and the Character Condition for each semistandard lattice. Moreover, one can also describe Actions for semistandard lattices in many special cases. Further, whenever such actions exist, they are “uniquely determined by the lattice” (see Theorem 5.4.4 and succeeding remarks). So, in a precise sense, whenever a semistandard lattice is a supporting graph, it combinatorially encodes exactly enough

information to be able to recover a unique basis for the corresponding representation as well as actions of generators on that basis. This evidence led to the following conjecture:

The Semistandard Lattice Conjecture: *Let \mathfrak{g} be a rank two simple Lie algebra, and let $\lambda = (a, b)$ for non-negative integers a and b . Let L_λ denote the corresponding \mathfrak{g} -semistandard lattice. Then L_λ is a supporting graph for the irreducible representation of \mathfrak{g} with dominant weight λ .*

The “tableaux” and the lattices described in Chapters 2, 3, and 4 may at first glance seem strange and perhaps somewhat arbitrary. However, the tableaux descriptions can be obtained (with some work) from a single result of [Lit]. In that paper, Littelmann offers a “uniform” description of tableau-like objects for almost all irreducible representations of the simple Lie algebras. Littelmann does not consider the lattice ordering of tableaux which we present here, however. On the other hand, Donnelly and Wildberger [DW] have found a combinatorial procedure which “uniformly” generates the semistandard lattices and which also yields sets of combinatorial objects which are easily converted to the tableaux of this thesis. (By “uniform” we mean independent of the type of the Lie algebra.)

By implementing the algorithms presented in Chapter 5 using the computer based mathematical manipulation language Maple V (see Appendix A), we are now able to easily construct examples of these beautiful edge-colored distributive lattices and test them against the SLC. In Chapter 6 we provide many examples of these distributive lattices.

One initial goal of this study was to confirm the SLC for all representations with “small” dimension, say, less than 10,000. In other words, given a rank two simple Lie algebra \mathfrak{g} , we wanted to demonstrate that the \mathfrak{g} -semistandard lattice L_λ is indeed a supporting

Table 1.2 Does the SLC hold for the G_2 -semistandard lattice L_λ when $\lambda = (a, b)$?

		b							
		0	1	2	3	4	5	6	7
a	0	y	y	n	n	n	n	n	n
	1	y	n	n	n	n	n	n	
	2	y	n	n	n	n			
	3	y	n	n	n	n			
	4	y	n	n	n				
	5	y	n	n					
	6	y	n	n					
	7	y	n						
	8	y	n						
	9	y							
	10	y							
	11	y							

Based in part on these findings, we offer the following refinement of the SLC:

A Refinement of the Semistandard Lattice Conjecture: *Let \mathfrak{g} be a rank two simple Lie algebra. Let a and b be non-negative integers. Then the \mathfrak{g} -semistandard lattice L_λ is a supporting graph for the irreducible representation of \mathfrak{g} with dominant weight λ if and only if $\lambda = (a, b)$ when $\mathfrak{g} = A_2$; $\lambda = (a, 0)$, $\lambda = (0, b)$ or $\lambda = (1, b)$ when $\mathfrak{g} = B_2$; and $\lambda = (a, 0)$ or $\lambda = (0, 1)$ when $\mathfrak{g} = G_2$.*

The A_2 part of the previous statement is the content of Theorem 2.6. In Theorem 4.6 we cite a result of [DLP1] which confirms the SLC for G_2 and all dominant weights of the form $\lambda = (a, 0)$. In Theorem 3.6 we use another result of [DLP1] to confirm the SLC for B_2 and all dominant weights of the form $\lambda = (0, b)$. In addition, we also confirm the SLC for B_2 and all dominant weights of the form $\lambda = (a, 0)$; while this result was known previously to Donnelly and others, it appears here for the first time as a formal statement with proof. In Theorem 6.1, we prove that for a dominant weight $\lambda = (a, b)$ with $a \geq 2$ and $b \geq 1$, the B_2 -semistandard lattice is not a supporting graph for a representation of B_2 .

In addition to these results, there are several other related questions worth pursuing. Confirming the SLC for B_2 and all dominant weights of the form $\lambda = (1, b)$ is an open question at this time. At the time of this writing we are also working toward a formal proof that the SLC fails for any G_2 -semistandard lattice L_λ if $\lambda = (a, b)$ with $b \geq 2$ or if λ has form $\lambda = (a, 1)$ with $a \geq 1$. Another possible project might be to re-formulate the definition of “semistandard lattice” in such a way that semistandard lattices can serve as supporting graphs for more classes of dominant weights. However, this already seems to be a challenging problem in the case of the algebra B_2 and the dominant weight $\lambda = (2, 1)$.

CHAPTER 2

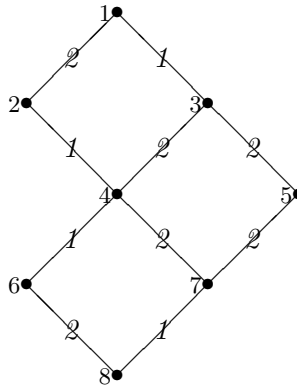
DISTRIBUTIVE LATTICES AND REPRESENTATIONS OF A_2

Semistandard lattices for A_2 . We begin by defining some combinatorial objects called “tableaux” which we will then partially order. The resulting poset will be a distributive lattice. We color the edges of the Hasse diagram for this lattice to obtain what we call a “semistandard lattice for A_2 .”

Fix non-negative integers a and b and set $\lambda = (a, b)$. We identify λ with the “Ferrers diagram” of the partition $(a + b, b)$, which we refer to as $sh(\lambda)$ and read “shape λ .” This is a grid with two rows of left-justified boxes; the top row has $a + b$ boxes and the bottom row has b boxes. Put another way, the shape λ has $a + b$ columns; the first b columns have two boxes each, and the final a columns have just one box each. For example, if $\lambda = (2, 1)$, then $sh(\lambda)$ is $\begin{array}{|c|c|c|} \hline \square & \square & \square \\ \hline \square & & \\ \hline \end{array}$. A *tableaux of shape λ* is a filling of the boxes in $sh(\lambda)$ with integers. The filling is said to be *semistandard* if the box-entries of the tableaux weakly increase from left to right across the rows and strictly increase from top to bottom down the columns. If \mathbf{t} is a tableau of shape λ , we often write $\mathbf{t} = (\mathbf{t}_1, \dots, \mathbf{t}_b, \mathbf{t}_{1+b}, \dots, \mathbf{t}_{a+b})$, where \mathbf{t}_q is the q th column of \mathbf{t} for $1 \leq q \leq a + b$; note that \mathbf{t}_q is a tableau of shape $\begin{array}{|c|} \hline \square \\ \hline \end{array}$ when $1 \leq q \leq b$ and is a tableau of shape $\begin{array}{|c|} \hline \square \\ \hline \square \\ \hline \end{array}$ when $1 + b \leq q \leq a + b$. We also index box-entries of \mathbf{t} as we would the entries of a matrix: $\mathbf{t}_{p,q}$ refers to the integer entry in the p th row, q th column of \mathbf{t} . An A_2 -*tableaux of shape λ* is any semistandard filling of $sh(\lambda)$ with box-entries taken from the set $\{1, 2, 3\}$. An example of an A_2 -tableau of shape $(2, 1)$ is $\mathbf{t} = \begin{array}{|c|c|c|} \hline 1 & 1 & 2 \\ \hline 3 & & \\ \hline \end{array}$.

Partially order the set of all A_2 -tableaux of shape λ by *reverse componentwise comparison*: that is, for A_2 -tableaux \mathbf{s} and \mathbf{t} of shape λ , we have $\mathbf{s} \leq \mathbf{t}$ if and only if each box-entry in \mathbf{t} is no bigger than the corresponding box-entry in \mathbf{s} . One can easily see that relative to this partial order \mathbf{s} is covered by \mathbf{t} , written $\mathbf{s} \longrightarrow \mathbf{t}$, if and only if there exist indices k and l such that $\mathbf{s}_{k,l} = \mathbf{t}_{k,l} + 1$ while $\mathbf{s}_{p,q} = \mathbf{t}_{p,q}$ for $p \neq k$ or $q \neq l$. In this case let $i := \mathbf{t}_{k,l}$; then we assign the “color” i to this edge, and write $\mathbf{s} \xrightarrow{i} \mathbf{t}$. The A_2 -semistandard lattice L_λ is the set of A_2 -tableaux of shape λ together with the reverse componentwise partial ordering and the above assignment of colors to the covering relations of this partial order.

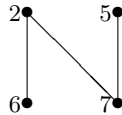
Figure 2.1 The A_2 -semistandard lattice corresponding to the shape $\lambda = (1, 1)$



As an example, consider the A_2 -semistandard lattice L_λ when $\lambda = (1, 1)$ (see Figure 2.1). We will sometimes refer to vertex k in this picture by the symbol v_k . The tableaux corresponding to these vertices are: $v_1 = \begin{array}{|c|c|} \hline 1 & 1 \\ \hline 2 & \\ \hline \end{array}$, $v_2 = \begin{array}{|c|c|} \hline 1 & 1 \\ \hline 3 & \\ \hline \end{array}$, $v_3 = \begin{array}{|c|c|} \hline 1 & 2 \\ \hline 2 & \\ \hline \end{array}$, $v_4 = \begin{array}{|c|c|} \hline 1 & 2 \\ \hline 3 & \\ \hline \end{array}$, $v_5 = \begin{array}{|c|c|} \hline 1 & 3 \\ \hline 2 & \\ \hline \end{array}$, $v_6 = \begin{array}{|c|c|} \hline 2 & 2 \\ \hline 3 & \\ \hline \end{array}$, $v_7 = \begin{array}{|c|c|} \hline 1 & 3 \\ \hline 3 & \\ \hline \end{array}$, and $v_8 = \begin{array}{|c|c|} \hline 2 & 3 \\ \hline 3 & \\ \hline \end{array}$. It is clear by inspection that this is a connected graph with just two edge colors (“1” and “2”) and is the Hasse diagram for a ranked partially ordered set. Additionally, one can see that this graph has a unique maximal and a unique minimal element (v_1 and v_8 , respectively). Together, these three conditions show that the

Combinatorial Requirement is met. With just a little work, one can see that this is a distributive lattice. The best way to see this is to apply the Fundamental Theorem of Finite Distributive Lattices (see [Sta]), and observe that the poset of Figure 2.1 is isomorphic to the poset of order ideals taken from the 4-element poset of “irreducibles” in Figure 2.2. To verify the Dimension Requirement, note that the number of vertices in Figure 2.1 is 8, which agrees with the Weyl dimension formula for A_2 with $a = 1$ and $b = 1$.

Figure 2.2 4-element poset of “irreducibles”



There are 10 edges to check in order to verify the Structure Condition. Take, for example, edge $v_4 \xrightarrow{2} v_3$. Recall that

$$wt(\mathbf{s}) = (2\rho_1(\mathbf{s}) - l_1(\mathbf{s}), 2\rho_2(\mathbf{s}) - l_2(\mathbf{s}))$$

where $\rho_i(\mathbf{s})$ is the rank of the element \mathbf{s} in its i -component and $l_i(\mathbf{s})$ is the length of the i -component containing \mathbf{s} . Since $\rho_1(v_4) = 1$, $\rho_2(v_4) = 1$, $l_1(v_4) = 2$, and $l_2(v_4) = 2$, one can verify that $wt(v_4) = (0, 0)$. Likewise, since $\rho_1(v_3) = 0$, $\rho_2(v_3) = 2$, $l_1(v_3) = 1$, and $l_2(v_3) = 2$, one can verify that $wt(v_3) = (-1, 2)$. It follows that

$$wt(v_4) + \alpha_2 = (0, 0) + (-1, 2) = (-1, 2) = wt(v_3).$$

One can do similar computations on the other 9 edges to verify the Structure Condition.

Using Diamond and Crossing relations, one can easily determine that the coefficients $\pi_{p,q}$ for this example are: $\pi_{2,1} = 1$, $\pi_{3,1} = 1$, $\pi_{4,2} = 2$, $\pi_{4,3} = \frac{1}{2}$, $\pi_{5,3} = \frac{3}{2}$, $\pi_{6,4} = 2$, $\pi_{7,4} = \frac{1}{2}$,

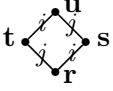
$\pi_{7,5} = \frac{3}{2}$, $\pi_{8,6} = 1$, and $\pi_{8,7} = 1$. Recall that the Crossing Condition for any vertex \mathbf{s} and any color i is:

$$\sum_{\mathbf{r}:\mathbf{r}\xrightarrow{i}\mathbf{s}} \pi_{\mathbf{r},\mathbf{s}} - \sum_{\mathbf{t}:\mathbf{s}\xrightarrow{i}\mathbf{t}} \pi_{\mathbf{s},\mathbf{t}} = 2\rho_i(\mathbf{s}) - l_i(\mathbf{s}).$$

Choosing color 2 at vertex v_5 for example, this identity holds since

$$\sum_{\mathbf{r}:\mathbf{r}\xrightarrow{2}v_5} \pi_{\mathbf{r},v_5} - \sum_{\mathbf{t}:v_5\xrightarrow{2}\mathbf{t}} \pi_{v_5,\mathbf{t}} = \pi_{7,5} - \pi_{5,3} = 0 = 2\rho_2(v_5) - l_2(v_5),$$

where $\rho_2(v_5) = 1$ and $l_2(v_5) = 2$. Recall that the Diamond Condition is checked by

$\pi_{\mathbf{s},\mathbf{u}}\pi_{\mathbf{t},\mathbf{u}} = \pi_{\mathbf{r},\mathbf{s}}\pi_{\mathbf{r},\mathbf{t}}$ whenever elements \mathbf{r} , \mathbf{s} , \mathbf{t} and \mathbf{u} form a diamond of edges  for any color i and j . Taking, for example, diamond 7-4-6-8, we see that

$$\pi_{7,4}\pi_{6,4} = 1 = \pi_{8,7}\pi_{8,6}.$$

The other diamond and crossing relations can be similarly verified.

Proposition 2.1 *Let $\lambda = (a, b)$ for non-negative integers a and b . Then the semistandard lattice L_λ is a distributive lattice and meets the Combinatorial Requirement for $A_2(\lambda)$.*

Proof. Distributivity of L_λ is just Corollary 5.1.2 of [Mc]. By definition edges in L_λ take only one of two colors. Finally, since L_λ is a distributive lattice, it is a ranked poset with a unique maximal element and a unique minimal element. \square

Proposition 2.2 *Let $\lambda = (a, b)$ for non-negative integers a and b . Then the semistandard lattice L_λ meets the Dimension Requirement for $A_2(\lambda)$. That is, the number of A_2 -tableaux of shape λ agrees with Weyl's dimension formula for A_2 , so*

$$|L_\lambda| = \frac{1}{2}(a+1)(b+1)(a+b+2)$$

Proof. For this proof we let $\binom{m}{n}$ (read “ m multichoose n ”) denote the number of ways to select n objects from among m objects with repetition allowed. It can be seen

that $\binom{m}{n} = \binom{m+n-1}{n}$. As an example, $\binom{m}{n}$ counts the number of sequences of integers (a_1, \dots, a_n) satisfying $1 \leq a_1 \leq \dots \leq a_n \leq m$.

Let $\mathbf{t} \in L_\lambda$. Then \mathbf{t} can be one of three types: (1) $\mathbf{t}_b = \boxed{\frac{1}{2}}$, (2) $\mathbf{t}_b = \boxed{\frac{1}{3}}$, or (3) $\mathbf{t}_b = \boxed{\frac{2}{3}}$. In case (1), it follows that $\mathbf{t}_q = \boxed{\frac{1}{2}}$ for $1 \leq q \leq b-1$. There are $\binom{3}{a}$ ways to choose the remaining a columns $\mathbf{t}_{1+b}, \dots, \mathbf{t}_{a+b}$ of \mathbf{t} . In case (2), the columns $\mathbf{t}_1, \dots, \mathbf{t}_{b-1}$ can be chosen from $\left\{ \boxed{\frac{1}{2}}, \boxed{\frac{1}{3}} \right\}$. There are $\binom{2}{b-1}$ ways to choose these columns. There are $\binom{3}{a}$ ways to choose the remaining a columns $\mathbf{t}_{1+b}, \dots, \mathbf{t}_{a+b}$ of \mathbf{t} . In case (3), the columns $\mathbf{t}_1, \dots, \mathbf{t}_{b-1}$ can be chosen from $\left\{ \boxed{\frac{1}{2}}, \boxed{\frac{1}{3}}, \boxed{\frac{2}{3}} \right\}$. There are $\binom{3}{b-1}$ ways to choose these columns. There are $\binom{2}{a}$ ways to choose the remaining a columns $\mathbf{t}_{1+b}, \dots, \mathbf{t}_{a+b}$ of \mathbf{t} , since $\mathbf{t}_{1+b}, \dots, \mathbf{t}_{a+b}$ must be selected from the set $\left\{ \boxed{2}, \boxed{3} \right\}$.

Now add up the totals from each the three types:

$$\binom{3}{a} + \binom{2}{b-1} \binom{3}{a} + \binom{3}{b-1} \binom{2}{a}$$

This simplifies to $\frac{1}{2}(a+1)(b+1)(a+b+2)$. □

With some work, the above description of A_2 -tableau can be derived from [Lit]. Littelmann supplied his own rule for computing the weight of an A_2 -tableaux. With a little more work, this rule can be seen to be

$$wt_{Lit}(\mathbf{s}) := \left(n_1(\mathbf{s}) - n_2(\mathbf{s}), n_2(\mathbf{s}) - n_3(\mathbf{s}) \right),$$

where $n_i(\mathbf{s})$ is the number of times the integer i appears as a box-entry in the tableau \mathbf{s} . Littelmann's weight rule is known to meet the Character Condition for $A_2(\lambda)$. That is, Littelmann shows that $d_\lambda(\mu) = |\{\mathbf{s} \in P \mid wt_{Lit}(\mathbf{s}) = \mu\}|$ for each μ in $\mathbb{Z} \times \mathbb{Z}$.

Lemma 2.3 *Let $\lambda = (a, b)$ for non-negative integers a and b . Let \mathbf{s} be in L_λ . Let i be in $\{1, 2\}$. (1.) Let \mathbf{s}_q ($1 \leq q \leq a+b$) be the leftmost (respectively, rightmost) column of \mathbf{s} that*

is not i -maximal (respectively, i -minimal) in L_\square or L_\square . Let \mathbf{x}_q denote an element of L_\square or L_\square that covers (respectively, is covered by) \mathbf{s}_q along an edge of color i . Form the tableau

$$\mathbf{x} := (\mathbf{s}_1, \dots, \mathbf{s}_{q-1}, \mathbf{x}_q, \mathbf{s}_{q+1}, \dots, \mathbf{s}_{a+b}).$$

Then \mathbf{x} is in L_λ , and $\mathbf{s} \xrightarrow{i} \mathbf{x}$ (respectively, $\mathbf{x} \xrightarrow{i} \mathbf{s}$). (2.) The tableau \mathbf{s} is i -maximal (respectively, i -minimal) in L_λ if and only if \mathbf{s}_q is i -maximal (respectively i -minimal) in L_\square or L_\square for all $1 \leq q \leq a+b$. (3.) The functions l_i and ρ_i are additive in the columns of \mathbf{s} , i.e. $l_i(\mathbf{s}) = l_i(\mathbf{s}_1) + l_i(\mathbf{s}_2) + \dots + l_i(\mathbf{s}_{a+b})$ and $\rho_i(\mathbf{s}) = \rho_i(\mathbf{s}_1) + \rho_i(\mathbf{s}_2) + \dots + \rho_i(\mathbf{s}_{a+b})$. The weight rule wt is additive in the columns of \mathbf{s} , so $wt(\mathbf{s}) = wt(\mathbf{s}_1) + wt(\mathbf{s}_2) + \dots + wt(\mathbf{s}_{a+b})$. (4.) If $\mathbf{s} \xrightarrow{i} \mathbf{t}$, then $wt(\mathbf{s}) + \alpha_i = wt(\mathbf{t})$. (5.) Littelmann's weight rule is additive in the columns of \mathbf{s} , so $wt_{Lit}(\mathbf{s}) = wt_{Lit}(\mathbf{s}_1) + wt_{Lit}(\mathbf{s}_2) + \dots + wt_{Lit}(\mathbf{s}_{a+b})$. (6.) $wt_{Lit}(\mathbf{s}) = wt(\mathbf{s})$.

Proof. Note that the 1-maximal columns in L_\square are $\boxed{1}$ and $\boxed{3}$, and the 2-maximal columns in L_\square are $\boxed{1}$ and $\boxed{2}$. The 1-minimal columns in L_\square are $\boxed{2}$ and $\boxed{3}$, and the 2-minimal columns in L_\square are $\boxed{1}$ and $\boxed{3}$. The 1-maximal columns in L_\square are $\boxed{\frac{1}{2}}$ and $\boxed{\frac{1}{3}}$, and the 2-maximal columns in L_\square are $\boxed{\frac{1}{2}}$ and $\boxed{\frac{2}{3}}$. The 1-minimal columns in L_\square are $\boxed{\frac{1}{2}}$ and $\boxed{\frac{2}{3}}$, and the 2-minimal columns in L_\square are $\boxed{\frac{1}{3}}$ and $\boxed{\frac{2}{3}}$.

For (1), we consider cases. First take $i = 1$ and assume \mathbf{s}_q is the leftmost column of \mathbf{s} that is not 1-maximal. It follows that \mathbf{s}_q must be either $\boxed{2}$ or $\boxed{\frac{2}{3}}$. If $\mathbf{s}_q = \boxed{2}$, then the semistandard condition implies that the box in tableau \mathbf{s} immediately to the left of \mathbf{s}_q (entry $\mathbf{s}_{1,q-1}$) must be either a 1 or a 2. But if $\mathbf{s}_{1,q-1} = 2$, then \mathbf{s}_{q-1} is either $\boxed{2}$ or $\boxed{\frac{2}{3}}$, which contradicts that fact that \mathbf{s}_q is the leftmost column of \mathbf{s} that is not 1-maximal. Therefore $\mathbf{s}_{1,q-1}$ is 1. To the right of the column \mathbf{s}_q we must have either $\boxed{2}$ or $\boxed{3}$. Thus, if we set $\mathbf{x}_q := \boxed{1}$, then the tableau $\mathbf{x} := (\mathbf{s}_1, \dots, \mathbf{s}_{q-1}, \mathbf{x}_q, \mathbf{s}_{q+1}, \dots, \mathbf{s}_{a+b})$ will still meet the

semistandard requirement. Then \mathbf{x} is in L_λ . Moreover, it is clear that \mathbf{s} and \mathbf{x} only differ in the entry $(1, q)$, and that $\mathbf{s} \xrightarrow{1} \mathbf{x}$. Similarly, if $\mathbf{s}_q = \boxed{\frac{2}{3}}$, then the semistandard condition implies that the box in tableau \mathbf{s} immediately to the left of \mathbf{s}_q (entry $\mathbf{s}_{1,q-1}$) must be either a 1 or a 2. But if $\mathbf{s}_{1,q-1} = 2$, then \mathbf{s}_{q-1} is $\boxed{\frac{2}{3}}$, which contradicts that fact that \mathbf{s}_q is the leftmost column of \mathbf{s} that is not 1-maximal. Therefore $\mathbf{s}_{1,q-1}$ is 1, and \mathbf{s}_{q-1} is either $\boxed{\frac{1}{2}}$ or $\boxed{\frac{1}{3}}$. To the right of the column \mathbf{s}_q we have one of $\boxed{2}$, $\boxed{3}$, or $\boxed{\frac{2}{3}}$. Thus, if we set $\mathbf{x}_q := \boxed{\frac{1}{3}}$, then the tableau $\mathbf{x} := (\mathbf{s}_1, \dots, \mathbf{s}_{q-1}, \mathbf{x}_q, \mathbf{s}_{q+1}, \dots, \mathbf{s}_{a+b})$ will still meet the semistandard requirement. Then \mathbf{x} is in L_λ . Moreover, it is clear that \mathbf{s} and \mathbf{x} only differ in the entry $(1, q)$, and that $\mathbf{s} \xrightarrow{1} \mathbf{x}$. The remaining $i = 2$ cases and the i -minimal arguments are entirely similar.

Part (2) follows immediately from (1). For (3), let $i \in \{1, 2\}$. For $1 \leq q \leq a+b$, let \mathbf{u}_q be the i -maximal element in the i -component of \mathbf{s}_q in L_\square or L_\square , and let \mathbf{r}_q be the corresponding i -minimal element. Observe that the number of steps from \mathbf{r}_q to \mathbf{s}_q in L_\square or L_\square is just $\rho_i(\mathbf{s}_q)$, and the number of steps from \mathbf{r}_q to \mathbf{u}_q is $l_i(\mathbf{s}_q)$. For $1 \leq j \leq a+b$, form tableaux $\mathbf{u}^{(j)}$ and $\mathbf{r}^{(j)}$ by the rules

$$\mathbf{u}^{(j)} := (\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_j, \mathbf{s}_{j+1}, \dots, \mathbf{s}_{a+b})$$

$$\mathbf{r}^{(j)} := (\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_{j-1}, \mathbf{r}_j, \dots, \mathbf{r}_{a+b})$$

Parts (1) and (2) of this lemma imply that $\mathbf{u}^{(j)}$ and $\mathbf{r}^{(j)}$ are in the i -component of \mathbf{s} in L_λ , and that $\mathbf{u}^{(a+b)}$ and $\mathbf{r}^{(1)}$ are respectively the maximal and minimal elements in this i -component. Moreover, from part (1) we see that the number of steps from $\mathbf{r}^{(a+b+1-j)}$ to \mathbf{s} in L_λ is $\rho_i(\mathbf{s}_{a+b}) + \rho_i(\mathbf{s}_{a+b-1}) + \dots + \rho_i(\mathbf{s}_{a+b+1-j})$ for $1 \leq j \leq a+b$. Of course, $\rho_i(\mathbf{s})$ is just the number of steps from $\mathbf{r}^{(1)}$ to \mathbf{s} in L_λ . So by setting $j = a+b$, we get $\rho_i(\mathbf{s}) = \rho_i(\mathbf{s}_{a+b}) + \rho_i(\mathbf{s}_{a+b-1}) + \dots + \rho_i(\mathbf{s}_1)$. Similarly we see that the number of steps from

$\mathbf{r}^{(1)}$ to $\mathbf{u}^{(j)}$ is $l_i(\mathbf{s}_1) + l_i(\mathbf{s}_2) + \cdots + l_i(\mathbf{s}_j) + \rho_i(\mathbf{s}_{j+1}) + \cdots + \rho_i(\mathbf{s}_{a+b})$ for $1 \leq j \leq a + b$.

But $l_i(\mathbf{s})$ is the number of steps from $\mathbf{r}^{(1)}$ to $\mathbf{u}^{(a+b)}$. So by setting $j = a + b$, we see that

$l_i(\mathbf{s}) = l_i(\mathbf{s}_1) + l_i(\mathbf{s}_2) + \cdots + l_i(\mathbf{s}_{a+b})$. In particular, notice now that

$$\begin{aligned} wt(\mathbf{s}) &= (2\rho_1(\mathbf{s}) - l_1(\mathbf{s}), 2\rho_2(\mathbf{s}) - l_2(\mathbf{s})) \\ &= (2\rho_1(\mathbf{s}_1) - l_1(\mathbf{s}_1), 2\rho_2(\mathbf{s}_1) - l_2(\mathbf{s}_1)) \\ &\quad + \cdots + (2\rho_1(\mathbf{s}_{a+b}) - l_1(\mathbf{s}_{a+b}), 2\rho_2(\mathbf{s}_{a+b}) - l_2(\mathbf{s}_{a+b})) \\ &= wt(\mathbf{s}_1) + \cdots + wt(\mathbf{s}_{a+b}) \end{aligned}$$

For (4), suppose $\mathbf{s} \xrightarrow{i} \mathbf{t}$ in L_λ . Now \mathbf{s} and \mathbf{t} only differ in one entry. In particular, there is an index k ($1 \leq k \leq a + b$), such that $\mathbf{s}_q = \mathbf{t}_q$ for $q \neq k$ while $\mathbf{s}_k \xrightarrow{i} \mathbf{t}_k$ in L_\square or L_\square . One can check by hand that for any such edge in L_\square or L_\square , we will have $wt(\mathbf{s}_k) + \alpha_i = wt(\mathbf{t}_k)$. (There is a total of only four edges in L_\square or L_\square .) Of course $wt(\mathbf{s}_q) = wt(\mathbf{t}_q)$ for $q \neq k$. We get:

$$\begin{aligned} wt(\mathbf{s}) + \alpha_i &= wt(\mathbf{s}_1) + \cdots + wt(\mathbf{s}_{k-1}) + wt(\mathbf{s}_k) + wt(\mathbf{s}_{k+1}) + \cdots + wt(\mathbf{s}_{a+b}) + \alpha_i \\ &= wt(\mathbf{t}_1) + \cdots + wt(\mathbf{t}_{k-1}) + wt(\mathbf{s}_k) + wt(\mathbf{t}_{k+1}) + \cdots + wt(\mathbf{t}_{a+b}) + \alpha_i \\ &= wt(\mathbf{t}_1) + \cdots + wt(\mathbf{t}_{k-1}) + wt(\mathbf{t}_k) + wt(\mathbf{t}_{k+1}) + \cdots + wt(\mathbf{t}_{a+b}) \\ &= wt(\mathbf{t}) \end{aligned}$$

For (5), observe that the function n_i is additive in the columns of \mathbf{s} , that is, $n_i(\mathbf{s}) = n_i(\mathbf{s}_1) + n_i(\mathbf{s}_2) + \cdots + n_i(\mathbf{s}_{a+b})$. It follows that wt_{Lit} will be additive as well. Finally, for (6) one can easily check by hand that for any column \mathbf{s}_q in L_\square or L_\square , we have $wt_{Lit}(\mathbf{s}_q) = wt(\mathbf{s}_q)$.

(There are only six different elements in L_{\square} and L_{\square} to be checked.) Then

$$\begin{aligned} wt_{Lit}(\mathbf{s}) &= wt_{Lit}(\mathbf{s}_1) + wt_{Lit}(\mathbf{s}_2) + \cdots + wt_{Lit}(\mathbf{s}_{a+b}) \\ &= wt(\mathbf{s}_1) + wt(\mathbf{s}_2) + \cdots + wt(\mathbf{s}_{a+b}) \\ &= wt(\mathbf{s}) \end{aligned} \quad \square$$

Proposition 2.4 *Let $\lambda = (a, b)$ for non-negative integers a and b . Then the semistandard lattice L_{λ} satisfies the Structure Condition for A_2 .*

Proof. This is just Lemma 2.3.4. □

Proposition 2.5 *Let $\lambda = (a, b)$ for non-negative integers a and b . Then the semistandard lattice L_{λ} satisfies the Character Condition for $A_2(\lambda)$.*

Proof. In the paragraph preceding Lemma 2.3, we noted that Littelmann's weight rule is known to meet the Character Condition for $A_2(\lambda)$. The proposition now follows from Lemma 2.3.6. □

The next theorem confirms the Semistandard Lattice Conjecture for A_2 for any dominant weight λ . An explicit description of Actions (as discussed in Chapter 1) on L_{λ} can be found in [DLP1].

Theorem 2.6 *Let $\lambda = (a, b)$ for non-negative integers a and b . Then the semistandard lattice L_{λ} is a supporting graph for the irreducible representation of A_2 with dominant weight λ .*

Proof. See Proposition 4.1 of [Don]. There, a construction originally due to Gelfand and Tsetlin [GT] is used to obtain a basis with supporting graph L_{λ} for the irreducible representation of A_2 with dominant weight λ . This result is re-derived in [DLP1]. □

CHAPTER 3

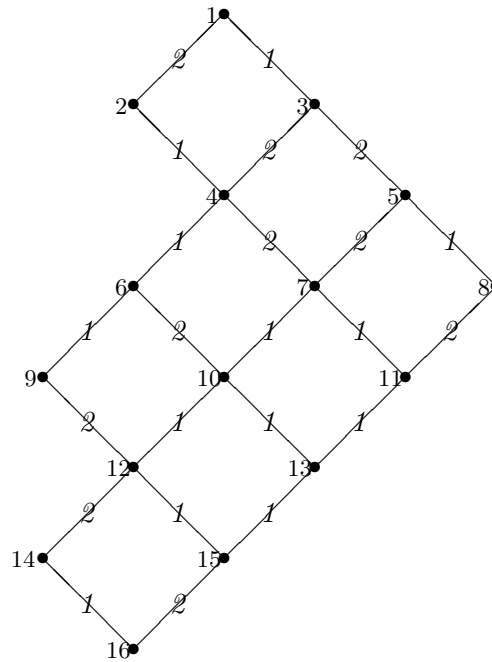
DISTRIBUTIVE LATTICES AND REPRESENTATIONS OF B_2

Semistandard lattices for B_2 . The results of this chapter extend to the algebra B_2 all of the A_2 results of Chapter 2, with the exception of Theorem 2.6. One can also view this chapter as a B_2 version of McClard’s thesis [Mc] in the sense that it records for the algebra B_2 all of the major results obtained in [Mc] for G_2 . While these B_2 results were known previously to Donnelly and others, most of them do not appear anywhere in the literature, and so they are appearing here for the first time as formal statements with proofs.

Following Chapter 2, a B_2 -tableau \mathbf{t} of shape λ is any semistandard filling of $sh(\lambda)$ with box-entries taken from the set $\{1, 2, 3, 4\}$ such that \mathbf{t} has no columns of the form $\begin{bmatrix} 1 \\ 4 \end{bmatrix}$ and at most one column of the form $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$. An example of a B_2 -tableau of shape $(2,1)$ is $\mathbf{t} = \begin{bmatrix} 1 & 1 & 2 \\ 3 & & \end{bmatrix}$.

As in Chapter 2, partially order the set of all B_2 -tableaux of shape λ by reverse componentwise comparison. One can easily see that relative to this partial order \mathbf{s} is covered by \mathbf{t} , written $\mathbf{s} \longrightarrow \mathbf{t}$, if and only if there exist indices k and l such that $\mathbf{s}_{k,l} = \mathbf{t}_{k,l} + 1$ while $\mathbf{s}_{p,q} = \mathbf{t}_{p,q}$ for $p \neq k$ or $q \neq l$. Let $i := 1$ if $\mathbf{t}_{k,l} \in \{1, 3\}$, and let $i := 2$ otherwise; then we assign the “color” i to this edge, and write $\mathbf{s} \xrightarrow{i} \mathbf{t}$. The B_2 -semistandard lattice L_λ is the set of B_2 -tableaux of shape λ together with the reverse componentwise partial ordering and the above assignment of colors to the covering relations of this partial order.

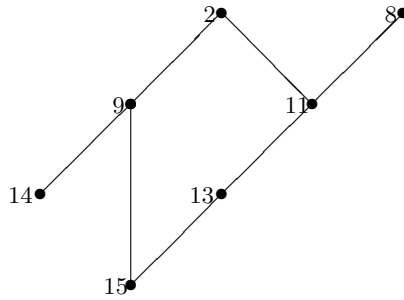
As an example, consider the B_2 -semistandard lattice L_λ when $\lambda = (1, 1)$ (see Figure 3.1). As in Chapter 2, we will sometimes refer to vertex k in this picture by the symbol

Figure 3.1 The B_2 -semistandard lattice corresponding to shape $\lambda = (1, 1)$ 

v_k . The tableaux corresponding to these vertices are: $v_1 = \begin{bmatrix} 1 & 1 \\ 2 \end{bmatrix}$, $v_2 = \begin{bmatrix} 1 & 1 \\ 3 \end{bmatrix}$, $v_3 = \begin{bmatrix} 1 & 2 \\ 2 \end{bmatrix}$, $v_4 = \begin{bmatrix} 1 & 2 \\ 3 \end{bmatrix}$, $v_5 = \begin{bmatrix} 1 & 3 \\ 2 \end{bmatrix}$, $v_6 = \begin{bmatrix} 2 & 2 \\ 3 \end{bmatrix}$, $v_7 = \begin{bmatrix} 1 & 3 \\ 3 \end{bmatrix}$, $v_8 = \begin{bmatrix} 1 & 4 \\ 2 \end{bmatrix}$, $v_9 = \begin{bmatrix} 2 & 2 \\ 4 \end{bmatrix}$, $v_{10} = \begin{bmatrix} 2 & 3 \\ 3 \end{bmatrix}$, $v_{11} = \begin{bmatrix} 1 & 4 \\ 3 \end{bmatrix}$, $v_{12} = \begin{bmatrix} 2 & 3 \\ 4 \end{bmatrix}$, $v_{13} = \begin{bmatrix} 2 & 4 \\ 3 \end{bmatrix}$, $v_{14} = \begin{bmatrix} 3 & 3 \\ 4 \end{bmatrix}$, $v_{15} = \begin{bmatrix} 2 & 4 \\ 4 \end{bmatrix}$, and $v_{16} = \begin{bmatrix} 3 & 4 \\ 4 \end{bmatrix}$. It is clear by inspection that this is a connected graph with just two edge colors (“1” and “2”) and is the Hasse diagram for a ranked partially ordered set. Additionally, one can see that this graph has a unique maximal and a unique minimal element (v_1 and v_{16} , respectively). Together, these three conditions show that the Combinatorial Requirement is met. With just a little work, one can see that this is a distributive lattice. The best way to see this is to apply the Fundamental Theorem of Finite Distributive Lattices (see [Sta]), and observe that the poset of Figure 3.1 is isomorphic to the poset of order ideals taken from the 7-element poset of “irreducibles” in Figure 3.2. To verify the Dimension Requirement, note that the number

of vertices in Figure 3.1 is 16, which agrees with the Weyl dimension formula for B_2 with $a = 1$ and $b = 1$.

Figure 3.2 7-element poset of “irreducibles”



There are 23 edges to check in order to verify the Structure Condition. Take, for example, edge $v_7 \xrightarrow{2} v_5$. Recall that

$$wt(\mathbf{s}) = (2\rho_1(\mathbf{s}) - l_1(\mathbf{s}), 2\rho_2(\mathbf{s}) - l_2(\mathbf{s}))$$

where $\rho_i(\mathbf{s})$ is the rank of the element \mathbf{s} in its i -component and $l_i(\mathbf{s})$ is the length of the i -component containing \mathbf{s} . Since $\rho_1(v_7) = 3$, $\rho_2(v_7) = 0$, $l_1(v_7) = 3$, and $l_2(v_7) = 2$, one can verify that $wt(v_7) = (3, -2)$. Likewise, since $\rho_1(v_5) = 1$, $\rho_2(v_5) = 1$, $l_1(v_5) = 1$, and $l_2(v_5) = 2$, one can verify that $wt(v_5) = (1, 0)$. It follows that

$$wt(v_7) + \alpha_2 = (3, -2) + (-2, 2) = (1, 0) = wt(v_5).$$

One can do similar computations on the other 22 edges to verify the Structure Condition.

Using Diamond and Crossing relations, one can easily determine that the coefficients $\pi_{p,q}$ for this example are: $\pi_{2,1} = 1$, $\pi_{3,1} = 1$, $\pi_{4,2} = 3$, $\pi_{4,3} = \frac{1}{3}$, $\pi_{5,3} = \frac{5}{3}$, $\pi_{6,4} = 4$, $\pi_{7,4} = \frac{1}{3}$, $\pi_{7,5} = \frac{5}{3}$, $\pi_{8,5} = 1$, $\pi_{9,6} = 3$, $\pi_{10,6} = 1$, $\pi_{10,7} = \frac{4}{3}$, $\pi_{11,7} = \frac{5}{3}$, $\pi_{11,8} = 1$, $\pi_{12,9} = 2$, $\pi_{12,10} = \frac{3}{2}$, $\pi_{13,10} = \frac{5}{6}$, $\pi_{13,11} = \frac{8}{3}$, $\pi_{14,12} = 2$, $\pi_{15,12} = \frac{1}{2}$, $\pi_{15,13} = \frac{5}{2}$, $\pi_{16,14} = 1$, and $\pi_{16,15} = 1$. Recall

that the Crossing Condition for any vertex \mathbf{s} and any color i is:

$$\sum_{\mathbf{r}: \mathbf{r} \xrightarrow{i} \mathbf{s}} \pi_{\mathbf{r}, \mathbf{s}} - \sum_{\mathbf{t}: \mathbf{s} \xrightarrow{i} \mathbf{t}} \pi_{\mathbf{s}, \mathbf{t}} = 2\rho_i(\mathbf{s}) - l_i(\mathbf{s}).$$

Choosing color 1 at vertex v_{10} for example, this identity holds since

$$\sum_{\mathbf{r}: \mathbf{r} \xrightarrow{1} v_{10}} \pi_{\mathbf{r}, v_{10}} - \sum_{\mathbf{t}: v_{10} \xrightarrow{1} \mathbf{t}} \pi_{v_{10}, \mathbf{t}} = (\pi_{12,10} + \pi_{13,10}) - \pi_{10,7} = 1 = 2\rho_1(v_{10}) - l_1(v_{10}),$$

where $\rho_1(v_{10}) = 2$ and $l_1(v_{10}) = 3$. Recall that the Diamond Condition is checked by

$\pi_{\mathbf{s}, \mathbf{u}} \pi_{\mathbf{t}, \mathbf{u}} = \pi_{\mathbf{r}, \mathbf{s}} \pi_{\mathbf{r}, \mathbf{t}}$ whenever elements \mathbf{r} , \mathbf{s} , \mathbf{t} and \mathbf{u} form a diamond of edges $\mathbf{t} \begin{array}{c} \nearrow \mathbf{u} \\ \searrow \end{array} \mathbf{s}$ for any colors i and j . Taking, for example, diamond 8-5-7-11, we see that

$$\pi_{8,5} \pi_{7,5} = \frac{5}{3} = \pi_{11,8} \pi_{11,7}.$$

The other diamond and crossing relations can be similarly verified.

Proposition 3.1 *Let $\lambda = (a, b)$ for non-negative integers a and b . Then the semistandard lattice L_λ is a distributive lattice and meets the Combinatorial Requirement for $B_2(\lambda)$.*

Proof. Let $\mathbf{s}, \mathbf{t} \in L_\lambda$. Define $\mathbf{min}(\mathbf{s}, \mathbf{t})$ to be the componentwise max of entries from \mathbf{s}, \mathbf{t} (since L_λ is ordered by reverse componentwise comparison). In particular, the (i, j) -entry of $\mathbf{min}(\mathbf{s}, \mathbf{t})$ is the $\max\{\mathbf{s}_{i,j}, \mathbf{t}_{i,j}\}$. (Note that $i \in \{1, 2\}$ and $1 \leq j \leq a + b$.) Let $\mathbf{r} = \mathbf{min}(\mathbf{s}, \mathbf{t})$. (1) Suppose $\mathbf{r}_{i,j} > \mathbf{r}_{i,j+1}$. Without loss of generality, assume $\mathbf{s}_{i,j} \geq \mathbf{t}_{i,j}$. Therefore, by definition $\mathbf{r}_{i,j} = \mathbf{s}_{i,j}$. Also, $\mathbf{r}_{i,j+1} \geq \mathbf{s}_{i,j+1}$. So, $\mathbf{s}_{i,j} = \mathbf{r}_{i,j} > \mathbf{r}_{i,j+1} \geq \mathbf{s}_{i,j+1}$, hence $\mathbf{s}_{i,j} > \mathbf{s}_{i,j+1}$. But, $\mathbf{s}_{i,j} \leq \mathbf{s}_{i,j+1}$ for every i, j since $\mathbf{s} \in L_\lambda$. Therefore we must have $\mathbf{r}_{i,j} \leq \mathbf{r}_{i,j+1}$ for every j , so \mathbf{r} meets the criteria that *box-entries of a B_2 -tableau weakly increase from left to right across the rows*. (2) Similarly suppose entry $\mathbf{r}_{1,j} \geq \mathbf{r}_{2,j}$. Without loss of generality, assume $\mathbf{s}_{1,j} \geq \mathbf{t}_{1,j}$. Therefore, by definition $\mathbf{r}_{1,j} = \mathbf{s}_{1,j}$. Also, $\mathbf{r}_{2,j} \geq \mathbf{s}_{2,j}$. So, $\mathbf{s}_{1,j} = \mathbf{r}_{1,j} \geq \mathbf{r}_{2,j} \geq \mathbf{s}_{2,j}$, hence $\mathbf{s}_{1,j} \geq \mathbf{s}_{2,j}$. But, $\mathbf{s}_{1,j} < \mathbf{s}_{2,j}$ for every j since $\mathbf{s} \in L_\lambda$.

Therefore $r_{1,j} < r_{2,j}$ for each j , so \mathbf{r} meets the criteria that *box-entries of a B_2 -tableau strictly increase from top to bottom down the columns*. (3) Now, assume $\begin{bmatrix} 1 \\ 4 \end{bmatrix}$ is the j th column in \mathbf{r} . Since $r_{1,j}$ is a 1, then $s_{1,j}$ and $t_{1,j}$ are both 1. Therefore, either $s_{2,j}$ or $t_{2,j}$ must be a 4, which means that $\begin{bmatrix} 1 \\ 4 \end{bmatrix}$ is the j th column in either \mathbf{s} or \mathbf{t} . But $\mathbf{s}, \mathbf{t} \in L_\lambda$, hence $\begin{bmatrix} 1 \\ 4 \end{bmatrix}$ does not appear as a column in \mathbf{s} or \mathbf{t} . Therefore, $\begin{bmatrix} 1 \\ 4 \end{bmatrix}$ is not in \mathbf{r} , so \mathbf{r} meets the criteria that *a B_2 -tableau has no columns of the form $\begin{bmatrix} 1 \\ 4 \end{bmatrix}$* . (4) Finally, assume the j th and $(j+1)$ st columns of \mathbf{r} are $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$. Since $r_{2,j}$ is a 3, then without loss of generality $s_{2,j}$ is a 3. Since $s_{2,j} \leq s_{2,j+1}$, then $s_{2,j+1} \in \{3, 4\}$. If $s_{2,j+1}$ was a 4, then $r_{2,j+1}$ would have to be a 4. But $r_{2,j+1} = 3$ by assumption. Then $s_{2,j+1} = 3$. At this point, the j th and $(j+1)$ st columns of \mathbf{s} must be $\begin{bmatrix} 1 & 1 \\ 3 & 3 \end{bmatrix}$, $\begin{bmatrix} 1 & 2 \\ 3 & 3 \end{bmatrix}$, or $\begin{bmatrix} 2 & 2 \\ 3 & 3 \end{bmatrix}$. If $\begin{bmatrix} 1 & 1 \\ 3 & 3 \end{bmatrix} \in \mathbf{s}$, then the j th and $(j+1)$ st columns of \mathbf{t} must be $\begin{bmatrix} 2 & 2 \\ 3 & 3 \end{bmatrix}$ to obtain \mathbf{r} . But $\mathbf{t} \in L_\lambda$, so $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ appears at most once in \mathbf{t} . Similarly, if $\begin{bmatrix} 1 & 2 \\ 3 & 3 \end{bmatrix} \in \mathbf{s}$, then (again) \mathbf{t} must contain $\begin{bmatrix} 2 & 2 \\ 3 & 3 \end{bmatrix}$ to obtain \mathbf{r} . Since $\mathbf{s} \in L_\lambda$, \mathbf{s} can not contain $\begin{bmatrix} 2 & 2 \\ 3 & 3 \end{bmatrix}$. Therefore, the j th and $(j+1)$ st columns of \mathbf{r} are not $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$, so \mathbf{r} meets the criteria that in a B_2 -tableau, $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ appears at most once.

By combining (1) through (4), we see that $\mathbf{r} \in L_\lambda$, hence $\mathbf{min}(\mathbf{s}, \mathbf{t}) \in L_\lambda$ for every $\mathbf{s}, \mathbf{t} \in L_\lambda$. A similar argument can be used to show that $\mathbf{max}(\mathbf{s}, \mathbf{t}) \in L_\lambda$, where $\mathbf{max}(\mathbf{s}, \mathbf{t})$ is the componentwise min of entries from \mathbf{s} and \mathbf{t} . Note that $L_\lambda \subseteq L_{\square} \times \cdots \times L_{\square} \times L_{\square} \times \cdots \times L_{\square}$. Note also that L_{\square} for B_2 is a chain with four elements, while L_{\square} is a chain with five elements. Chains are known to be distributive lattices. Therefore by Lemma 2.2.3 of [Mc], L_λ is a distributive lattice, with $\mathbf{s} \wedge \mathbf{t} := \mathbf{min}(\mathbf{s}, \mathbf{t})$ and $\mathbf{s} \vee \mathbf{t} := \mathbf{max}(\mathbf{s}, \mathbf{t})$. By definition, the edges in L_λ can only take one of two colors. Finally, since L_λ is a distributive lattice, it is a ranked poset with a unique maximal element and a unique minimal element. \square

Proposition 3.2 *Let $\lambda = (a, b)$ for non-negative integers a and b . Then the semistandard lattice L_λ meets the Dimension Requirement for $B_2(\lambda)$. That is, the number of B_2 -tableaux of shape λ agrees with Weyl's dimension formula for B_2 , so*

$$|L_\lambda| = \frac{1}{3!}(a+1)(b+1)(a+b+2)(a+2b+3)$$

Proof. Let $\mathbf{t} \in L_\lambda$. Then \mathbf{t} can be one of five types: (1) $\mathbf{t}_b = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, (2) $\mathbf{t}_b = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$, (3) $\mathbf{t}_b = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$, (4) $\mathbf{t}_b = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$, or (5) $\mathbf{t}_b = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$. In case (1), it follows that $\mathbf{t}_q = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ for $1 \leq q \leq b-1$. There are $\binom{4}{a}$ ways to choose the remaining a columns $\mathbf{t}_{1+b}, \dots, \mathbf{t}_{a+b}$ of \mathbf{t} . In case (2), the columns $\mathbf{t}_1, \dots, \mathbf{t}_{b-1}$ can be chosen from $\left\{ \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 3 \end{bmatrix} \right\}$. There are $\binom{2}{b-1}$ ways to choose these columns. There are $\binom{4}{a}$ ways to choose the remaining a columns $\mathbf{t}_{1+b}, \dots, \mathbf{t}_{a+b}$ of \mathbf{t} . In case (3), the columns $\mathbf{t}_1, \dots, \mathbf{t}_{b-1}$ can be chosen from $\left\{ \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 3 \end{bmatrix} \right\}$, but not $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ since it can appear at most once in \mathbf{t} . There are $\binom{2}{b-1}$ ways to choose these columns. There are $\binom{3}{a}$ ways to choose the remaining a columns $\mathbf{t}_{1+b}, \dots, \mathbf{t}_{a+b}$ of \mathbf{t} , since $\mathbf{t}_{1+b}, \dots, \mathbf{t}_{a+b}$ must be selected from the set $\left\{ \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \begin{bmatrix} 3 \\ 4 \end{bmatrix} \right\}$. Since there can only be one $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ in \mathbf{t} , cases (4) and (5) can be broken up into two subcases: (a) $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ does not appear, and (b) $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ appears only once. In case (4), the columns $\mathbf{t}_1, \dots, \mathbf{t}_{b-1}$ can be chosen from $\left\{ \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \begin{bmatrix} 2 \\ 4 \end{bmatrix} \right\}$. So, for case (4a), there are $\binom{3}{b-1}$ ways to choose these columns, while in case (4b), there are $\binom{3}{b-2}$ ways to choose the columns. There are $\binom{3}{a}$ ways to choose the remaining a columns $\mathbf{t}_{1+b}, \dots, \mathbf{t}_{a+b}$ of \mathbf{t} for both case (a) and case (b). In case (5), the remaining columns of $\mathbf{t}_1, \dots, \mathbf{t}_{b-1}$ can be chosen from $\left\{ \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \begin{bmatrix} 2 \\ 4 \end{bmatrix}, \begin{bmatrix} 3 \\ 4 \end{bmatrix} \right\}$. There are $\binom{4}{b-1}$ ways to choose these columns in case (5a), and $\binom{4}{b-2}$ ways to choose these columns in case (5b). There are $\binom{2}{a}$ ways to choose the remaining a columns $\mathbf{t}_{1+b}, \dots, \mathbf{t}_{a+b}$ of \mathbf{t} for both case (a) and case (b), since \mathbf{t} must be selected from the set $\left\{ \begin{bmatrix} 3 \\ 4 \end{bmatrix}, \begin{bmatrix} 4 \\ 4 \end{bmatrix} \right\}$.

Now add up the totals from each of the five types:

$$\begin{aligned} & \left(\binom{4}{a} \right) + \left(\binom{2}{b-1} \right) \left(\binom{4}{a} \right) + \left(\binom{2}{b-1} \right) \left(\binom{3}{a} \right) + \\ & \left\{ \left(\binom{3}{b-1} \right) + \left(\binom{3}{b-2} \right) \right\} \left(\binom{3}{a} \right) + \left\{ \left(\binom{4}{b-1} \right) + \left(\binom{4}{b-2} \right) \right\} \left(\binom{2}{a} \right) \end{aligned}$$

This simplifies to $\frac{1}{3!}(a+1)(b+1)(a+b+2)(a+2b+3)$. \square

With some work, the above description of B_2 -tableaux can be derived from [Lit]. Littelmann supplied his own rule for computing the weight of an B_2 -tableaux. With a little more work, this rule can be seen to be

$$wt_{Lit}(\mathbf{s}) := \left(n_1(\mathbf{s}) - n_2(\mathbf{s}) + n_3(\mathbf{s}) - n_4(\mathbf{s}), n_2(\mathbf{s}) - n_3(\mathbf{s}) \right),$$

where $n_i(\mathbf{s})$ is the number of times the integer i appears as a box-entry in the tableau \mathbf{s} . Littelmann's weight rule is known to meet the Character Condition for $B_2(\lambda)$. That is, Littelmann shows that $d_\lambda(\mu) = |\{\mathbf{s} \in P \mid wt_{Lit}(\mathbf{s}) = \mu\}|$ for each μ in $\mathbb{Z} \times \mathbb{Z}$.

Lemma 3.3 *Let $\lambda = (a, b)$ for non-negative integers a and b . Let \mathbf{s} be in L_λ . Let i be in $\{1, 2\}$. (1.) Let \mathbf{s}_q ($1 \leq q \leq a+b$) be the leftmost (respectively, rightmost) column of \mathbf{s} that is not i -maximal (respectively, i -minimal) in L_\square or L_\square . Let \mathbf{x}_q denote an element of L_\square or L_\square that covers (respectively, is covered by) \mathbf{s}_q along an edge of color i . Form the tableau*

$$\mathbf{x} := (\mathbf{s}_1, \dots, \mathbf{s}_{q-1}, \mathbf{x}_q, \mathbf{s}_{q+1}, \dots, \mathbf{s}_{a+b}).$$

Then \mathbf{x} is in L_λ , and $\mathbf{s} \xrightarrow{i} \mathbf{x}$ (respectively, $\mathbf{x} \xrightarrow{i} \mathbf{s}$). (2.) The tableau \mathbf{s} is i -maximal (respectively, i -minimal) in L_λ if and only if \mathbf{s}_q is i -maximal (respectively i -minimal) in L_\square or L_\square for all $1 \leq q \leq a+b$. (3.) The functions l_i and ρ_i are additive in the columns of \mathbf{s} , i.e. $l_i(\mathbf{s}) = l_i(\mathbf{s}_1) + l_i(\mathbf{s}_2) + \dots + l_i(\mathbf{s}_{a+b})$ and $\rho_i(\mathbf{s}) = \rho_i(\mathbf{s}_1) + \rho_i(\mathbf{s}_2) + \dots + \rho_i(\mathbf{s}_{a+b})$. The weight

rule wt is additive in the columns of \mathbf{s} , so $wt(\mathbf{s}) = wt(\mathbf{s}_1) + wt(\mathbf{s}_2) + \cdots + wt(\mathbf{s}_{a+b})$. (4.) If $\mathbf{s} \xrightarrow{i} \mathbf{t}$, then $wt(\mathbf{s}) + \alpha_i = wt(\mathbf{t})$. (5.) Littelmann's weight rule is additive in the columns of \mathbf{s} , so $wt_{Lit}(\mathbf{s}) = wt_{Lit}(\mathbf{s}_1) + wt_{Lit}(\mathbf{s}_2) + \cdots + wt_{Lit}(\mathbf{s}_{a+b})$. (6.) $wt_{Lit}(\mathbf{s}) = wt(\mathbf{s})$.

Proof. Note that the 1-maximal columns in L_{\square} are $\boxed{1}$ and $\boxed{3}$, and the 2-maximal columns in L_{\square} are $\boxed{1}$, $\boxed{2}$, and $\boxed{4}$. The 1-minimal columns in L_{\square} are $\boxed{2}$ and $\boxed{4}$, and the 2-minimal columns in L_{\square} are $\boxed{1}$, $\boxed{3}$, and $\boxed{4}$. The 1-maximal columns in L_{\square} are $\boxed{\frac{1}{2}}$, $\boxed{\frac{1}{3}}$, and $\boxed{\frac{3}{4}}$, and the 2-maximal columns in L_{\square} are $\boxed{\frac{1}{2}}$, $\boxed{\frac{2}{3}}$, and $\boxed{\frac{2}{4}}$. The 1-minimal columns in L_{\square} are $\boxed{\frac{1}{2}}$, $\boxed{\frac{2}{4}}$, and $\boxed{\frac{3}{4}}$, and the 2-minimal columns in L_{\square} are $\boxed{\frac{1}{3}}$, $\boxed{\frac{2}{3}}$, and $\boxed{\frac{3}{4}}$.

For (1), we consider cases. First take $i = 1$ and assume \mathbf{s}_q is the leftmost column of \mathbf{s} that is not 1-maximal. It follows that \mathbf{s}_q must be either $\boxed{2}$, $\boxed{4}$, $\boxed{\frac{2}{3}}$ or $\boxed{\frac{2}{4}}$. If $\mathbf{s}_q = \boxed{2}$, then the semistandard condition implies that the box in tableau \mathbf{s} immediately to the left of \mathbf{s}_q (entry $\mathbf{s}_{1,q-1}$) must be either a 1 or a 2. But if $\mathbf{s}_{1,q-1} = 2$, then \mathbf{s}_{q-1} is either $\boxed{2}$, $\boxed{\frac{2}{3}}$ or $\boxed{\frac{2}{4}}$, which contradicts that fact that \mathbf{s}_q is the leftmost column of \mathbf{s} that is not 1-maximal. Therefore $\mathbf{s}_{1,q-1}$ is 1. To the right of the column \mathbf{s}_q we must have either $\boxed{2}$, $\boxed{3}$, or $\boxed{4}$. Thus, if we set $\mathbf{x}_q := \boxed{1}$, then the tableau $\mathbf{x} := (\mathbf{s}_1, \dots, \mathbf{s}_{q-1}, \mathbf{x}_q, \mathbf{s}_{q+1}, \dots, \mathbf{s}_{a+b})$ will still meet the semistandard requirement. Then \mathbf{x} is in L_{λ} . Moreover, it is clear that \mathbf{s} and \mathbf{x} only differ in the entry $(1, q)$, and that $\mathbf{s} \xrightarrow{1} \mathbf{x}$. If $\mathbf{s}_q = \boxed{4}$, then the semistandard condition implies that $\mathbf{s}_{1,q-1}$ must be a 1, 2, 3, or 4. But if $\mathbf{s}_{1,q-1} = 4$, then \mathbf{s}_{q-1} is $\boxed{4}$, which contradicts the fact that \mathbf{s}_q is the leftmost column of \mathbf{s} that is not 1-maximal. Also, if $\mathbf{s}_{1,q-1} = 2$, then \mathbf{s}_{q-1} is either $\boxed{2}$, $\boxed{\frac{2}{3}}$, or $\boxed{\frac{2}{4}}$, which again contradicts the fact that \mathbf{s}_q is the leftmost column of \mathbf{s} that is not 1-maximal. Therefore $\mathbf{s}_{1,q-1}$ is either a 1 or a 3. To the right of the column \mathbf{s}_q we must have $\boxed{4}$. Thus, if we set $\mathbf{x}_q := \boxed{3}$, then the tableau $\mathbf{x} := (\mathbf{s}_1, \dots, \mathbf{s}_{q-1}, \mathbf{x}_q, \mathbf{s}_{q+1}, \dots, \mathbf{s}_{a+b})$ will still meet the semistandard requirement. Then \mathbf{x}

is in L_λ . Moreover, it is clear that \mathbf{s} and \mathbf{x} only differ in the entry $(1, q)$, and that $\mathbf{s} \xrightarrow{1} \mathbf{x}$. Now, if $\mathbf{s}_q = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$, then the semistandard condition implies that $\mathbf{s}_{1, q-1}$ must be a 1 or a 2. But if $\mathbf{s}_{1, q-1} = 2$, then \mathbf{s}_{q-1} is $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$, which contradicts the fact that \mathbf{s}_q is the leftmost column of \mathbf{s} that is not 1-maximal (it also contradicts the fact that $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ appears at most once in \mathbf{s}). Therefore $\mathbf{s}_{1, q-1}$ is 1, and \mathbf{s}_{q-1} is either $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ or $\begin{bmatrix} 1 \\ 3 \end{bmatrix}$. To the right of the column \mathbf{s}_q we must have one of $\begin{bmatrix} 2 \\ 2 \end{bmatrix}$, $\begin{bmatrix} 3 \\ 3 \end{bmatrix}$, $\begin{bmatrix} 4 \\ 4 \end{bmatrix}$, $\begin{bmatrix} 2 \\ 4 \end{bmatrix}$, or $\begin{bmatrix} 3 \\ 4 \end{bmatrix}$. Thus, if we set $\mathbf{x}_q := \begin{bmatrix} 2 \\ 4 \end{bmatrix}$, then the tableau $\mathbf{x} := (\mathbf{s}_1, \dots, \mathbf{s}_{q-1}, \mathbf{x}_q, \mathbf{s}_{q+1}, \dots, \mathbf{s}_{a+b})$ will still meet the semistandard requirement. Then \mathbf{x} is in L_λ . Moreover, it is clear that \mathbf{s} and \mathbf{x} only differ in the entry $(1, q)$, and that $\mathbf{s} \xrightarrow{1} \mathbf{x}$. Similarly, if $\mathbf{s}_q = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$, then the semistandard condition implies that $\mathbf{s}_{1, q-1}$ must be either a 1 or a 2. But if $\mathbf{s}_{1, q-1} = 2$, then \mathbf{s}_{q-1} is either $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ or $\begin{bmatrix} 2 \\ 4 \end{bmatrix}$, both of which contradict the fact that \mathbf{s}_q is the leftmost column of \mathbf{s} that is not 1-maximal. Therefore $\mathbf{s}_{1, q-1}$ is 1, and \mathbf{s}_{q-1} is either $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ or $\begin{bmatrix} 1 \\ 3 \end{bmatrix}$. To the right of the column \mathbf{s}_q we have one of $\begin{bmatrix} 2 \\ 2 \end{bmatrix}$, $\begin{bmatrix} 3 \\ 3 \end{bmatrix}$, $\begin{bmatrix} 4 \\ 4 \end{bmatrix}$, $\begin{bmatrix} 2 \\ 4 \end{bmatrix}$, or $\begin{bmatrix} 3 \\ 4 \end{bmatrix}$. Thus, if we set $\mathbf{x}_q := \begin{bmatrix} 2 \\ 3 \end{bmatrix}$, then the tableau $\mathbf{x} := (\mathbf{s}_1, \dots, \mathbf{s}_{q-1}, \mathbf{x}_q, \mathbf{s}_{q+1}, \dots, \mathbf{s}_{a+b})$ will still meet the semistandard requirement. Additionally, since \mathbf{s}_{q-1} could not have been $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$, then \mathbf{x} meets the restriction requirement that $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ appears at most once. Then \mathbf{x} is in L_λ . Moreover, it is clear that \mathbf{s} and \mathbf{x} only differ in the entry $(1, q)$, and that $\mathbf{s} \xrightarrow{1} \mathbf{x}$. The remaining $i = 2$ cases and the i -minimal arguments are entirely similar.

Note that L_\square for B_2 is a chain with four vertices and three edges, while L_\square is a chain with five vertices and four edges. Otherwise, the proofs of parts (2) through (6) of Lemma 3.3 are identical to the proofs of parts (2) through (6) given in Lemma 2.3. \square

Theorem 3.4 *Let $\lambda = (a, b)$ for non-negative integers a and b . Then the semistandard lattice L_λ satisfies the Structure Condition for B_2 .*

Proof. This is just Lemma 3.3.4. □

Theorem 3.5 *Let $\lambda = (a, b)$ for non-negative integers a and b . Then the semistandard lattice L_λ satisfies the Character Condition for $B_2(\lambda)$.*

Proof. In the paragraph preceding Lemma 3.3, we noted that Littelmann's weight rule is known to meet the Character Condition for $B_2(\lambda)$. The proposition now follows from Lemma 3.3.6. □

The next theorem confirms the Semistandard Lattice Conjecture for B_2 for two special classes of dominant weights. For one of these families of dominant weights, the result follows as an application of the main result of [DLP1].

Theorem 3.6 *Let $\lambda = (a, b)$ for some non-negative integers a and b such that $a = 0$ or $b = 0$. Then the semistandard lattice L_λ is a supporting graph for the irreducible representation of B_2 with dominant weight λ .*

Proof. For $\lambda = (0, b)$, the semistandard lattice L_λ is just the Molev lattice $L_B^{Mol}(b, 4)$ from [DLP1]. To see this, associate to a tableau $\mathbf{s} = (\mathbf{s}_1, \dots, \mathbf{s}_b)$ in L_λ the 5-tuple $mol(\mathbf{s}) := (s_1, s_2, s_3, s_4, s_5)$ in $L_B^{Mol}(b, 4)$, where s_1 counts the number of columns of the form $\boxed{\frac{1}{2}}$ in \mathbf{s} , s_2 counts the number of columns of the form $\boxed{\frac{1}{3}}$ in \mathbf{s} , s_3 counts the number of columns of the form $\boxed{\frac{2}{3}}$ in \mathbf{s} , s_4 counts the number of columns of the form $\boxed{\frac{2}{4}}$ in \mathbf{s} , and s_5 counts the number of columns of the form $\boxed{\frac{3}{4}}$ in \mathbf{s} . One can check that $\mathbf{s} \longrightarrow \mathbf{t}$ in L_λ if and only if $mol(\mathbf{s}) \longrightarrow mol(\mathbf{t})$ in $L_B^{Mol}(b, 4)$. The Cartan matrix for B_2 used in [DLP1] is $\begin{pmatrix} 2 & -2 \\ -1 & 2 \end{pmatrix}$, so one must reverse the colors on the edges of L_λ to get the same arrangement of edge colors in $L_B^{Mol}(b, 4)$. Theorem 2.1.Molev of [DLP1] shows that the edge-colored lattice $L_B^{Mol}(b, 4)$,

and hence L_λ , is a supporting graph for the irreducible representation of B_2 with dominant weight λ .

Disregarding edge colors, the B_2 lattice L_λ for $\lambda = (a, 0)$ coincides with the A_3 lattice denoted $L_A^{GT-left}(3, (a, 0, 0))$ in [Don]. That is, L_λ and $L_A^{GT-left}(3, (a, 0, 0))$ are isomorphic as lattices, but with slightly different edge coloring schemes. For the remainder of this proof, set $L_A^{GT-left} := L_A^{GT-left}(3, (a, 0, 0))$. In [Don], edges of the A_3 lattice $L_A^{GT-left}$ are colored 1, 2, or 3: the edge $\mathbf{s} \rightarrow \mathbf{t}$ is given color i if an entry \boxed{i} in \mathbf{t} changes to $\boxed{i+1}$ to form the tableau \mathbf{s} . It is easy to see that all i -components in $L_A^{GT-left}$ are chains. Pick an i -component \mathcal{C} in $L_A^{GT-left}$ and supply the edges of this chain with coefficients as follows: If $\mathbf{s} \xrightarrow{i} \mathbf{t}$ is an edge in \mathcal{C} , set $c_{\mathbf{s}, \mathbf{t}} = \rho_i(\mathbf{t})$ and $d_{\mathbf{s}, \mathbf{t}} = l_i(\mathbf{t}) - \rho_i(\mathbf{t}) + 1$. (For an example of these coefficients, see Example 4.1.1 of [Mc].) It now follows from Theorem 6.4 of [Don] that with this assignment of edge coefficients, $L_A^{GT-left}$ meets the Diamond and Crossing Conditions. (In the language of that paper, $L_A^{GT-left}$ is now a representation diagram for a “one-dimensional weight space representation” of A_3 .) Keep this assignment of edge coefficients, but now change all edges of color 3 in $L_A^{GT-left}$ to color 1 to obtain the B_2 lattice L_λ . Clearly this assignment of coefficients to the edges of L_λ will satisfy the Diamond condition at any diamond, and will satisfy the Crossing Condition for color 2 because the 2-components of L_λ are the same as the 2-components of $L_A^{GT-left}$. We now only need to check the Crossing Condition for color 1 in the B_2 lattice L_λ .

The color 1 component of a tableau \mathbf{t} in L_λ coincides with the $\{1, 3\}$ -component of \mathbf{t} in $L_A^{GT-left}$. Let $k := l_1(\mathbf{t})$ and $p := \rho_1(\mathbf{t})$ (the length of the 1-component of \mathbf{t} in $L_A^{GT-left}$ and the rank of \mathbf{t} in the 1-component, respectively). Let $l := l_3(\mathbf{t})$ and $q := \rho_3(\mathbf{t})$, where we think of \mathbf{t} as an element of $L_A^{GT-left}$. One can check that the color 1 edge below \mathbf{t} (if

it exists) has product $(k - p + 1)(p)$, the color 3 edge below \mathbf{t} (if it exists) has product $(l - q + 1)(q)$, the color 1 edge above \mathbf{t} (if it exists) has product $(k - p)(p + 1)$, and the color 3 edge above \mathbf{t} (if it exists) has product $(l - q)(q + 1)$. Now change color 3 to color 1. It is clear that the length of the 1-component containing \mathbf{t} in L_λ is $k + l$, and that the rank of \mathbf{t} in this 1-component is $p + q$. To check the Crossing Condition for color 1 at \mathbf{t} in L_λ now requires us to simply verify the identity

$$(l - q + 1)(q) + (k - p + 1)(p) - (k - p)(p + 1) - (l - q)(q + 1) = 2p + 2q - l - k.$$

One can check by hand that this identity holds. □

CHAPTER 4

DISTRIBUTIVE LATTICES AND REPRESENTATIONS OF G_2

Semistandard lattices for G_2 . Fix non-negative integers a and b and set $\lambda = (a, b)$.

Following Chapter 6 of [Mc], a G_2 -tableaux of shape λ is any semistandard filling of $sh(\lambda)$ with box-entries taken from the set $\{1, 2, 3, 4, 5, 6, 7\}$, adhering to the restrictions in Table

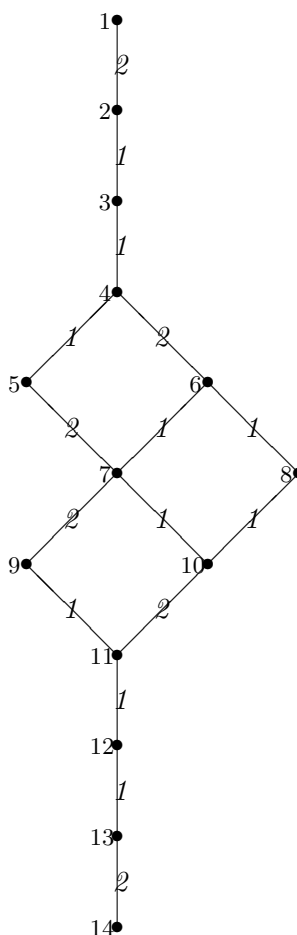
4.1. An example of a G_2 -tableau of shape $(2,1)$ is $\mathbf{t} = \begin{array}{|c|c|c|} \hline 1 & 1 & 2 \\ \hline 3 & & \\ \hline \end{array}$.

Table 4.1 Filling restrictions for G_2 -tableaux

This column in $\mathbf{t} \in \mathcal{T}_\lambda \dots$... cannot be succeeded by a column containing these entries:
$\begin{array}{ c } \hline 4 \\ \hline \end{array}$	$\begin{array}{ c } \hline 4 \\ \hline \end{array}$
$\begin{array}{ c } \hline 1 \\ 4 \\ \hline \end{array}$	$\begin{array}{ c } \hline 1 \\ \hline \end{array}$
$\begin{array}{ c } \hline 1 \\ 5 \\ \hline \end{array}$	$\begin{array}{ c } \hline 1 \\ \hline \end{array}$
$\begin{array}{ c } \hline 1 \\ 6 \\ \hline \end{array}$	$\begin{array}{ c } \hline 1 \text{ or } 2 \\ \hline \end{array}$
$\begin{array}{ c } \hline 1 \\ 7 \\ \hline \end{array}$	$\begin{array}{ c } \hline 1, 2, 3, \text{ or } 4 \\ \hline \end{array}$
$\begin{array}{ c } \hline 2 \\ 6 \\ \hline \end{array}$	$\begin{array}{ c } \hline 2 \\ \hline \end{array}$
$\begin{array}{ c } \hline 2 \\ 7 \\ \hline \end{array}$	$\begin{array}{ c } \hline 2, 3, \text{ or } 4 \\ \hline \end{array}$
$\begin{array}{ c } \hline 3 \\ 7 \\ \hline \end{array}$	$\begin{array}{ c } \hline 3 \text{ or } 4 \\ \hline \end{array}$
$\begin{array}{ c } \hline 4 \\ 7 \\ \hline \end{array}$	$\begin{array}{ c } \hline 4 \\ \hline \end{array}$

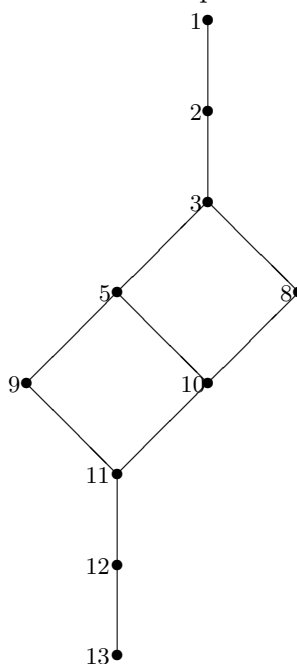
As in Chapter 2, partially order the set of all G_2 -tableaux of shape λ by reverse componentwise comparison. It follows from Lemma 6.1.4 of [Mc] that relative to this partial order \mathbf{s} is covered by \mathbf{t} , written $\mathbf{s} \longrightarrow \mathbf{t}$, if and only if there exist indices k and l such that $\mathbf{s}_{k,l} = \mathbf{t}_{k,l} + 1$ while $\mathbf{s}_{p,q} = \mathbf{t}_{p,q}$ for $p \neq k$ or $q \neq l$. Let $i := 1$ if $\mathbf{t}_{k,l} \in \{1, 3, 4, 6\}$, and let $i := 2$ otherwise; then we assign the “color” i to this edge, and write $\mathbf{s} \xrightarrow{i} \mathbf{t}$. The G_2 -semistandard lattice L_λ is the set of G_2 -tableaux of shape λ together with the reverse componentwise partial ordering and the above assignment of colors to the covering relations of this partial order.

Figure 4.1 The G_2 -semistandard lattice corresponding to the shape $\lambda = (0, 1)$



As an example, consider the G_2 -semistandard lattice L_λ when $\lambda = (0, 1)$ (see Figure 4.1). We will sometimes refer to vertex k in this picture by the symbol v_k . The tableaux associated to these vertices can be found in Appendix B. It is clear by inspection that this is a connected graph with just two edge colors (“1” and “2”) and is the Hasse diagram for a ranked partially ordered set. Additionally, one can see that this graph has a unique maximal and a unique minimal element (v_1 and v_{14} , respectively). Together, these three conditions show that the Combinatorial Requirement is met. With just a little work, one can see that this is a distributive lattice. The best way to see this is to apply the Fundamental Theorem of Finite Distributive Lattices (see [Sta]), and observe that the poset in Figure 4.1 is isomorphic to the poset of order ideals taken from the 10-element poset of “irreducibles” in Figure 4.2. To verify the Dimension Requirement, note that the number of vertices in Figure 4.1 is 14, which agrees with the Weyl dimension formula for B_2 with $a = 0$ and $b = 1$.

Figure 4.2 10-element poset of “irreducibles”



There are 16 edges to check in order to verify the Structure Condition. Take, for example, edge $v_6 \xrightarrow{2} v_4$. Recall that

$$wt(\mathbf{s}) = (2\rho_1(\mathbf{s}) - l_1(\mathbf{s}), 2\rho_2(\mathbf{s}) - l_2(\mathbf{s}))$$

where $\rho_i(\mathbf{s})$ is the rank of the element \mathbf{s} in its i -component and $l_i(\mathbf{s})$ is the length of the i -component containing \mathbf{s} . Since $\rho_1(v_6) = 2$, $\rho_2(v_6) = 0$, $l_1(v_6) = 2$, and $l_2(v_6) = 1$, one can verify that $wt(v_6) = (2, -1)$. Likewise, since $\rho_1(v_4) = 1$, $\rho_2(v_4) = 1$, $l_1(v_4) = 3$, and $l_2(v_4) = 1$, one can verify that $wt(v_4) = (-1, 1)$. It follows that

$$wt(v_6) + \alpha_2 = (2, -1) + (-3, 2) = (-1, 1) = wt(v_4).$$

One can do similar computations on the other 15 edges to verify the Structure Condition.

Using Diamond and Crossing relations, one can easily determine that the coefficients $\pi_{p,q}$ for this example are: $\pi_{2,1} = 1$, $\pi_{3,2} = 3$, $\pi_{4,3} = 4$, $\pi_{5,4} = 3$, $\pi_{6,4} = 1$, $\pi_{7,5} = 2$, $\pi_{7,6} = \frac{3}{2}$, $\pi_{8,6} = \frac{1}{2}$, $\pi_{9,7} = 2$, $\pi_{10,7} = \frac{3}{2}$, $\pi_{10,8} = \frac{1}{2}$, $\pi_{11,9} = 3$, $\pi_{11,10} = 1$, $\pi_{12,11} = 4$, $\pi_{13,12} = 3$, and $\pi_{14,13} = 1$. Recall that the Crossing Condition for any vertex \mathbf{s} and any color i is:

$$\sum_{\mathbf{r}: \mathbf{r} \xrightarrow{i} \mathbf{s}} \pi_{\mathbf{r}, \mathbf{s}} - \sum_{\mathbf{t}: \mathbf{s} \xrightarrow{i} \mathbf{t}} \pi_{\mathbf{s}, \mathbf{t}} = 2\rho_i(\mathbf{s}) - l_i(\mathbf{s}).$$

Choosing color 1 at vertex v_6 for example, this identity holds since

$$\sum_{\mathbf{r}: \mathbf{r} \xrightarrow{1} v_6} \pi_{\mathbf{r}, v_6} - \sum_{\mathbf{t}: v_6 \xrightarrow{1} \mathbf{t}} \pi_{v_6, \mathbf{t}} = (\pi_{7,6} + \pi_{8,6}) - 0 = 2 = 2\rho_1(v_6) - l_1(v_6),$$

where $\rho_1(v_6) = 2$ and $l_1(v_6) = 2$. Recall that the Diamond Condition is checked by

$\pi_{\mathbf{s}, \mathbf{u}} \pi_{\mathbf{t}, \mathbf{u}} = \pi_{\mathbf{r}, \mathbf{s}} \pi_{\mathbf{r}, \mathbf{t}}$ whenever elements \mathbf{r} , \mathbf{s} , \mathbf{t} and \mathbf{u} form a diamond of edges $\mathbf{t} \begin{array}{c} \nearrow^i \\ \searrow^j \end{array} \mathbf{u} \begin{array}{c} \nwarrow^j \\ \nearrow^i \end{array} \mathbf{s}$

for any colors i and j . Taking, for example, diamond 10-7-9-11, we see that

$$\pi_{10,7} \pi_{9,7} = 3 = \pi_{11,10} \pi_{11,9}.$$

The other diamond and crossing relations can be similarly verified.

Proposition 4.1 *Let $\lambda = (a, b)$ for non-negative integers a and b . Then the semistandard lattice L_λ is a distributive lattice and meets the Combinatorial Requirement for $G_2(\lambda)$.*

Proof. Distributivity of L_λ is just Theorem 6.2.2 of [Mc]. By definition edges in L_λ take only one of two colors. Finally, since L_λ is a distributive lattice, it is a ranked poset with a unique maximal element and a unique minimal element. \square

Proposition 4.2 *Let $\lambda = (a, b)$ for non-negative integers a and b . Then the semistandard lattice L_λ meets the Dimension Requirement for $G_2(\lambda)$. That is, the number of G_2 -tableaux of shape λ agrees with Weyl's dimension formula for G_2 , so*

$$|L_\lambda| = \frac{1}{5!}(a+1)(b+1)(a+b+2)(a+2b+3)(a+3b+4)(2a+3b+5)$$

Remarks on proof. While a combinatorial proof similar to the proofs of Proposition 2.2 and 3.2 would be nice, the number of subcases involved would make for a lengthy argument. Recall that the Character Condition for $G_2(\lambda)$ (Theorem 4.5 below) implies the Dimension Requirement. \square

McClard shows in Chapter 6 of [Mc] how certain tableaux originating in [Lit] can be translated into the G_2 -tableaux defined above. Note that McClard uses $L_G^{Lit}(2, \lambda)$ to denote L_λ , and writes $wt_P(\mathbf{t})$ instead of $wt(\mathbf{t})$. In [Lit], Littelmann prescribes his own “weight rule” for G_2 -tableaux. From Lemma 6.1.2 and other remarks on p. 39 of [Mc], one can see that Littelmann's weight rule is simply

$$wt_{Lit}(\mathbf{s}) = \left(n_1(\mathbf{s}) - n_2(\mathbf{s}) + 2n_3(\mathbf{s}) - 2n_5(\mathbf{s}) + n_6(\mathbf{s}) - n_7(\mathbf{s}), n_2(\mathbf{s}) - n_3(\mathbf{s}) + n_5(\mathbf{s}) - n_6(\mathbf{s}) \right),$$

where $n_i(\mathbf{s})$ is the number of times the integer i appears as a box-entry in the tableau \mathbf{s} . Littelmann's weight rule is known to meet the Character Condition for $G_2(\lambda)$. That is,

Littelmann shows that $d_\lambda(\mu) = |\{\mathbf{s} \in P \mid wt_{Lit}(\mathbf{s}) = \mu\}|$ for each μ in $\mathbb{Z} \times \mathbb{Z}$. The next lemma collects into one statement many results and observations from [Mc].

Lemma 4.3 *Let $\lambda = (a, b)$ for non-negative integers a and b . Let \mathbf{s} be in L_λ . Let i be in $\{1, 2\}$. (1.) Let \mathbf{s}_q ($1 \leq q \leq a+b$) be the leftmost (respectively, rightmost) column of \mathbf{s} that is not i -maximal (respectively, i -minimal) in L_\square or L_\square . Let \mathbf{x}_q denote an element of L_\square or L_\square that covers (respectively, is covered by) \mathbf{s}_q along an edge of color i . Form the tableau*

$$\mathbf{x} := (\mathbf{s}_1, \dots, \mathbf{s}_{q-1}, \mathbf{x}_q, \mathbf{s}_{q+1}, \dots, \mathbf{s}_{a+b}).$$

Then \mathbf{x} is in L_λ , and $\mathbf{s} \xrightarrow{i} \mathbf{x}$ (respectively, $\mathbf{x} \xrightarrow{i} \mathbf{s}$). (2.) The tableau \mathbf{s} is i -maximal (respectively, i -minimal) in L_λ if and only if \mathbf{s}_q is i -maximal (respectively i -minimal) in L_\square or L_\square for all $1 \leq q \leq a+b$. (3.) The functions l_i and ρ_i are additive in the columns of \mathbf{s} , i.e. $l_i(\mathbf{s}) = l_i(\mathbf{s}_1) + l_i(\mathbf{s}_2) + \dots + l_i(\mathbf{s}_{a+b})$ and $\rho_i(\mathbf{s}) = \rho_i(\mathbf{s}_1) + \rho_i(\mathbf{s}_2) + \dots + \rho_i(\mathbf{s}_{a+b})$. The weight rule wt is additive in the columns of \mathbf{s} , so $wt(\mathbf{s}) = wt(\mathbf{s}_1) + wt(\mathbf{s}_2) + \dots + wt(\mathbf{s}_{a+b})$. (4.) If $\mathbf{s} \xrightarrow{i} \mathbf{t}$, then $wt(\mathbf{s}) + \alpha_i = wt(\mathbf{t})$. (5.) Littelmann's weight rule is additive in the columns of \mathbf{s} , so $wt_{Lit}(\mathbf{s}) = wt_{Lit}(\mathbf{s}_1) + wt_{Lit}(\mathbf{s}_2) + \dots + wt_{Lit}(\mathbf{s}_{a+b})$. (6.) $wt_{Lit}(\mathbf{s}) = wt(\mathbf{s})$.

Proof. The proof of (1) can be found in the proof of Proposition 6.2.5 of [Mc]. Part (2) of the lemma is just a restatement of Proposition 6.2.5 in [Mc]. For part (3) of the lemma, additivity of ρ_i and l_i follow from Corollary 6.2.7 of [Mc]. Additivity of wt now follows:

$$\begin{aligned} wt(\mathbf{s}) + \alpha_i &= wt(\mathbf{s}_1) + \dots + wt(\mathbf{s}_{k-1}) + wt(\mathbf{s}_k) + wt(\mathbf{s}_{k+1}) + \dots + wt(\mathbf{s}_{a+b}) + \alpha_i \\ &= wt(\mathbf{t}_1) + \dots + wt(\mathbf{t}_{k-1}) + wt(\mathbf{s}_k) + wt(\mathbf{t}_{k+1}) + \dots + wt(\mathbf{t}_{a+b}) + \alpha_i \\ &= wt(\mathbf{t}_1) + \dots + wt(\mathbf{t}_{k-1}) + wt(\mathbf{t}_k) + wt(\mathbf{t}_{k+1}) + \dots + wt(\mathbf{t}_{a+b}) \\ &= wt(\mathbf{t}) \end{aligned}$$

For part (4) of the lemma, consult the proof of Theorem 6.2.8 of [Mc], p. 48. For part (5) of the lemma, additivity of wt_{Lit} follows from Lemma 6.1.2 of [Mc]. Part (6) of the lemma is just Theorem 6.2.8 of [Mc]. Part (6) can also be concluded from the additivity of wt and wt_{Lit} , together with the observation that wt and wt_{Lit} agree in L_{\square} and L_{\square} . \square

The next two theorems were the main results of [Mc]. We reformulate these results in the language and notation of this thesis.

Theorem 4.4 *Let $\lambda = (a, b)$ for non-negative integers a and b . Then the semistandard lattice L_{λ} satisfies the Structure Condition for G_2 .*

Proof. This is just Lemma 4.3.4. \square

Theorem 4.5 *Let $\lambda = (a, b)$ for non-negative integers a and b . Then the semistandard lattice L_{λ} satisfies the Character Condition for $G_2(\lambda)$.*

Proof. In the paragraph preceding Lemma 4.3, we noted that Littelmann's weight rule is known to meet the Character Condition for $G_2(\lambda)$. The theorem now follows from Lemma 4.3.6. \square

The next theorem confirms the Semistandard Lattice Conjecture for G_2 for a special class of dominant weights. This theorem restates a result from [DLP1].

Theorem 4.6 *Let $\lambda = (a, 0)$ for some non-negative integer a . Then the semistandard lattice L_{λ} is a supporting graph for the irreducible representation of G_2 with dominant weight λ .*

Proof. See Corollary 3.3 of [DLP1]. The lattice $L_G^{Lit}(2, a\omega_1)$ there is just the lattice L_{λ} when $\lambda = (a, 0)$. \square

CHAPTER 5

KEY ALGORITHMS FOR INVESTIGATING THE SEMISTANDARD LATTICE CONJECTURE

In order to investigate the Semistandard Lattice Conjecture, we developed three main algorithms. The first algorithm, **tableaux**, creates the set of all tableaux of shape λ . The second algorithm, **total_order**, defines a total ordering for the tableaux in L_λ . The final algorithm, **edges**, determines (if possible) the coefficient associated with any edge in a semistandard lattice with a view toward describing the “actions” of the Lie Algebra \mathfrak{g} .

Throughout this section, we let \mathfrak{g} denote a simple Lie Algebra of rank two, and we let $\lambda = (a, b)$, where a and b are non-negative integers. Define **lattice_dimension** (\mathfrak{g}, λ) to be the size of the \mathfrak{g} -semistandard lattice L_λ . This coincides with Weyl’s formula for the dimension of the irreducible representation of \mathfrak{g} with dominant weight λ (see Propositions 2.2, 3.2, and 4.2). Define a function **lattice_length** (\mathfrak{g}, λ) which returns the length of the \mathfrak{g} -semistandard lattice L_λ . That is, **lattice_length** (\mathfrak{g}, λ) counts the number of steps from the maximal tableau to the minimal tableau in L_λ . One can see that

$$\mathbf{lattice_length}(\mathfrak{g}, \lambda) = \begin{cases} 2a + 2b & \text{if } \mathfrak{g} = A_2 \\ 3a + 4b & \text{if } \mathfrak{g} = B_2 \\ 6a + 10b & \text{if } \mathfrak{g} = G_2 \end{cases}$$

by noting that the maximal tableau in all three cases is $\begin{array}{|c|c|c|c|c|} \hline 1 & \dots & 1 & 1 & \dots & 1 \\ \hline 2 & \dots & 2 & & & \\ \hline \end{array}$, while for A_2 the minimal tableau in L_λ is $\begin{array}{|c|c|c|c|c|} \hline 2 & \dots & 2 & 3 & \dots & 3 \\ \hline 3 & \dots & 3 & & & \\ \hline \end{array}$, for B_2 the minimal tableau is $\begin{array}{|c|c|c|c|c|c|} \hline 3 & \dots & 3 & 4 & \dots & 4 \\ \hline 4 & \dots & 4 & & & \\ \hline \end{array}$,

and for G_2 the minimal tableau is $\begin{array}{|c|c|c|c|c|} \hline 6 & \dots & 6 & 7 & \dots & 7 \\ \hline 7 & \dots & 7 & & & \\ \hline \end{array}$. For a tableau \mathbf{t} in the \mathfrak{g} -semistandard lattice L_λ , we let **descendant_set**($\mathfrak{g}, \lambda, \mathbf{t}$) denote the set $\{\mathbf{s} \in L_\lambda \mid \mathbf{s} \rightarrow \mathbf{t}\}$. That is, **descendant_set**($\mathfrak{g}, \lambda, \mathbf{t}$) is the set of all tableaux in L_λ covered by \mathbf{t} (regardless of the color of the edges). In the semistandard lattice L_λ for a rank two simple Lie Algebra \mathfrak{g} , tableaux can be organized according to “rank” or “level.” Let $L_\lambda^{(0)}$ denote the top level, so $L_\lambda^{(0)} := \left\{ \begin{array}{|c|c|c|c|c|} \hline 1 & \dots & 1 & 1 & \dots & 1 \\ \hline 2 & \dots & 2 & & & \\ \hline \end{array} \right\}$, and more generally, $L_\lambda^{(i)}$ will denote the set of tableaux which are i steps below the maximum tableau.

Algorithm 5.1 TABLEAUX

INPUT: A rank two simple Lie Algebra \mathfrak{g} and two non-negative integers a and b that indicate the size of the tableaux λ .

OUTPUT: A sequence (K_0, \dots, K_l) of sets of tableaux, where $l = \mathbf{lattice_length}(\mathfrak{g}, \lambda)$.

Step 1: Set $l := \mathbf{lattice_length}(\mathfrak{g}, \lambda)$. Let (K_0, K_1, \dots, K_l) denote a sequence

of sets. Initialize $K_0 := \left\{ \begin{array}{|c|c|c|c|c|} \hline 1 & \dots & 1 & 1 & \dots & 1 \\ \hline 2 & \dots & 2 & & & \\ \hline \end{array} \right\}$, and set $K_i := \emptyset$ for $1 \leq i \leq l$.

Step 2: FOR i from 1 to l LOOP

FOR \mathbf{t} in K_{i-1} LOOP

$K_i := K_i$ union **descendant_set**($\mathfrak{g}, \lambda, \mathbf{t}$)

END LOOP

END LOOP

Step 3: RETURN(K_0, \dots, K_l)

Proposition 5.2 Let $l := \mathbf{lattice_length}(\mathfrak{g}, \lambda)$. Let $(K_0, \dots, K_l) := \mathbf{tableaux}(\mathfrak{g}, \lambda)$.

Then $K_i = L_\lambda^{(i)}$ for $0 \leq i \leq l$.

Proof. Clearly, $K_0 = L_\lambda^{(0)} = \left\{ \begin{array}{|c|c|c|c|} \hline 1 & \dots & 1 & 1 & \dots & 1 \\ \hline 2 & \dots & 2 & & & \\ \hline \end{array} \right\}$. Assume $K_{i-1} = L_\lambda^{(i-1)}$. Choose an element \mathbf{s} in $L_\lambda^{(i)}$. By definition, \mathbf{s} is a descendant of an element of $L_\lambda^{(i-1)}$, say \mathbf{t} . Since $K_{i-1} = L_\lambda^{(i-1)}$, then \mathbf{t} is an element of K_{i-1} . By step 4 of algorithm **tableaux**, each descendant of \mathbf{t} will be an element of K_i . Therefore $\mathbf{s} \in K_i$, hence $L_\lambda^{(i)} \subset K_i$. Now choose an element \mathbf{s} in K_i . By algorithm **tableaux**, \mathbf{s} can only be added to K_i if it is the descendant of some element \mathbf{t} in K_{i-1} . Since $K_{i-1} = L_\lambda^{(i-1)}$, then \mathbf{t} is an element of $L_\lambda^{(i-1)}$. By definition, \mathbf{s} will be an element of $L_\lambda^{(i)}$ since it is the descendant of \mathbf{t} , an element of $L_\lambda^{(i-1)}$. Therefore, $\mathbf{s} \in L_\lambda^{(i)}$, hence $K_i \subset L_\lambda^{(i)}$. Then $K_i = L_\lambda^{(i)}$. The proposition statement now follows by induction. \square

In particular, it follows from Proposition 5.2 that the output of the **tableaux** algorithm is the set of all tableaux in L_λ . Our next algorithm totally orders the set of tableaux in L_λ . This will aid crucially in the iterative search for the edges and associated coefficients in the **edges** algorithm, which will follow shortly.

One tool that is very useful for assigning edge coefficients in a \mathfrak{g} -semistandard lattice is its “boundary.” The boundary is not easy to characterize abstractly, but roughly speaking it is a long chain of vertices and edges going from the minimal tableau up to the maximal tableau. We use edges in this chain as the starting point for edge coefficient computations between any two levels of vertices. In Appendix B, we explicitly define the sequences of boundary vertices for the semistandard lattices of each type (A_2 , B_2 , or G_2). The function **boundary**(\mathfrak{g}, λ) returns the sequence $(\mathbf{b}_0, \dots, \mathbf{b}_l)$ of boundary vertices in the \mathfrak{g} -semistandard lattice L_λ , where $\mathbf{b}_0 =$ maximal tableau, $\mathbf{b}_l =$ minimal tableau ($l =$ **lattice_length**(\mathfrak{g}, λ)), and \mathbf{b}_k is the unique boundary vertex in the level set $L_\lambda^{(k)}$.

For two tableaux $\mathbf{s}, \mathbf{t} \in L_\lambda$, we define a function $\mathbf{distance}(\mathfrak{g}, \lambda, \mathbf{s}, \mathbf{t})$ which returns the shortest number of steps between \mathbf{s} and \mathbf{t} , i.e. the length of any shortest path from \mathbf{s} to \mathbf{t} in the Hasse diagram for L_λ . In particular, if \mathbf{s} is the maximal tableau and \mathbf{t} is the minimal tableau in L_λ , then $\mathbf{distance}(\mathfrak{g}, \lambda, \mathbf{s}, \mathbf{t}) = \mathbf{lattice_length}(\mathfrak{g}, \lambda)$. Before we introduce a total ordering of tableaux in the \mathfrak{g} -semistandard lattice L_λ for an arbitrary shape λ (Algorithm 5.3), let us first describe a total ordering on columns in the \mathfrak{g} -semistandard lattices L_\square and L_\square . See Appendix B for explicit descriptions of these lattices. If \mathbf{s} and \mathbf{t} are two distinct \mathfrak{g} -semistandard columns of the same shape, then we say \mathbf{s} precedes \mathbf{t} in the total order on columns if (a) the rank of \mathbf{s} in L_\square or L_\square is greater than the rank of \mathbf{t} or (b) \mathbf{s} and \mathbf{t} have the same rank but \mathbf{s} is a boundary vertex. Notice that condition (b) is only relevant for the G_2 -semistandard lattice L_\square . For example, in B_2 -semistandard lattices, we say that $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ precedes $\begin{bmatrix} 3 \\ 4 \end{bmatrix}$, and we say $\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ precedes $\begin{bmatrix} 3 \\ 4 \end{bmatrix}$. In L_\square for G_2 , consider $\begin{bmatrix} 2 \\ 6 \end{bmatrix}$ and $\begin{bmatrix} 1 \\ 7 \end{bmatrix}$. While both sit on the same level of L_\square , the column $\begin{bmatrix} 2 \\ 6 \end{bmatrix}$ is a boundary vertex, so we say that $\begin{bmatrix} 2 \\ 6 \end{bmatrix}$ precedes $\begin{bmatrix} 1 \\ 7 \end{bmatrix}$. In the ‘‘Convention Adopted for Code’’ tables located in Appendix B, each column for the \mathfrak{g} -semistandard lattices L_\square and L_\square is assigned a unique positive integer. Using this assignment of integers one can easily compare two columns of the same shape for precedence in the total ordering on columns.

Now let λ be an arbitrary dominant weight. Let us describe how we will compare two tableaux with the same rank and the same distance from the boundary. Suppose \mathbf{s} and \mathbf{t} are at the same level $L_\lambda^{(k)}$ in the \mathfrak{g} -semistandard lattice L_λ , and suppose they have the same distance from the boundary vertex \mathbf{b}_k . Let \mathbf{s}_i (respectively \mathbf{t}_i) denote the i th column of \mathbf{s} (respectively \mathbf{t}). Say \mathbf{s} precedes \mathbf{t} in the righthand lexicographic ordering if there exists a j

such that (a) $\mathbf{s}_i = \mathbf{t}_i$ for $i > j$, and (b) \mathbf{s}_j precedes \mathbf{t}_j in the total order on the columns. We refer to this as the *righthand lexicographic rule*.

In the algorithm that follows, the concatenation of two finite sequences (a_1, \dots, a_m) and (b_1, \dots, b_n) , written $(a_1, \dots, a_m) \text{ concat } (b_1, \dots, b_n)$, is the new sequence $(a_1, \dots, a_m, b_1, \dots, b_n)$. For a sequence $S = (a_1, \dots, a_m)$, we let $\{S\}$ denote the set $\{a_1, \dots, a_m\}$. Note that $(a_1, \dots, a_m) \text{ concat } (\text{empty sequence}) = (a_1, \dots, a_m)$.

Algorithm 5.3 TOTAL_ORDER

INPUT: A rank two simple Lie Algebra \mathfrak{g} and two non-negative integers a and b that indicate the size of the tableaux λ .

OUTPUT: A ordered sequence T of tableaux of shape λ .

Step 1: Set $l := \mathbf{lattice_length}(\mathfrak{g}, \lambda)$, $dim := \mathbf{lattice_dimension}(\mathfrak{g}, \lambda)$,

$(K_0, \dots, K_l) := \mathbf{tableaux}(\mathfrak{g}, \lambda)$, and $(\mathbf{b}_0, \dots, \mathbf{b}_l) := \mathbf{boundary}(\mathfrak{g}, \lambda)$.

Step 2: For $0 \leq j \leq l$, let m_j be the maximum distance of a tableau at level

K_j from the boundary \mathbf{b}_j . Let $0 \leq k \leq m_j$. Let $S_j^k := \{\mathbf{s} \in K_j \mid \mathbf{distance}(\mathfrak{g}, \mathbf{s}, \mathbf{b}_j) = k\}$. Let $d(j, k) := |S_j^k|$.

Step 3: Now sort S_j^k using the righthand lexicographic rule to get the sequence

$T_j^k = (\mathbf{s}_1, \dots, \mathbf{s}_{d(j,k)})$. Here, T_j^k and S_j^k coincide as sets, and \mathbf{s}_p precedes \mathbf{s}_q in the right hand lexicographic ordering whenever $p < q$.

Step 4: Now, form a sequence T by concatenating the T_j^k 's by level and then

by distance within each level. That is:

Let $T := \text{empty sequence}$.

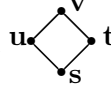
FOR j from 0 to l LOOP

FOR k from 0 to m_j LOOP
 $T := T$ concat T_j^k
 END LOOP
 END LOOP

Step 5: RETURN(T).

Proposition 5.4 *Let $T = \mathbf{total_order}(\mathbf{g}, \lambda)$. Then each tableau in L_λ appears once and only once in the sequence T , and the set $\{T\}$ coincides with L_λ .*

Proof. Keep the notation of Algorithm 5.3 and the preceding paragraphs. Let $\mathbf{t} \in \{T\}$. Since $\{T\} = \bigcup \{T_j^k\}$, then $\mathbf{t} \in \{T_j^k\}$ for some $0 \leq j \leq l$ and $0 \leq k \leq m_j$. Since $S_j^k = \{T_j^k\}$ as sets, then $\mathbf{t} \in S_j^k$. By definition, $S_j^k \subseteq K_j \subseteq L_\lambda$, therefore $\mathbf{t} \in L_\lambda$. We see then that $\{T\} \subseteq L_\lambda$. Now, let $\mathbf{t} \in L_\lambda$. By definition, $\mathbf{t} \in K_j$ for some $0 \leq j \leq l$. Also, by definition, $\mathbf{t} \in S_j^k$ for some $0 \leq k \leq m_j$. Since $S_j^k = \{T_j^k\}$ as sets, then \mathbf{t} appears in the sequence T_j^k . Since T_j^k appears as a subsequence of T , then \mathbf{t} appears in the sequence T . Therefore each tableau in L_λ appears in T , and hence $L_\lambda \subseteq \{T\}$. Moreover, since $\mathbf{t} \in L_\lambda$ can appear in at most one subsequence T_j^k of T , \mathbf{t} can appear at most once in T . \square

Define a function $\mathbf{lub}(\mathbf{r}, \mathbf{s})$ which returns the tableau $\mathbf{r} \vee \mathbf{s}$. Recall that the (i, j) -entry of $\mathbf{r} \vee \mathbf{s}$ is the $\min \{s_{i,j}, t_{i,j}\}$. For an edge $\mathbf{s} \xrightarrow{i} \mathbf{t}$ in L_λ , let $\mathbf{diamond_set}(\mathbf{g}, \lambda, \mathbf{s}, \mathbf{t})$ denote the set $\{\mathbf{u} \in L_\lambda \mid \mathbf{u} \text{ precedes } \mathbf{t} \text{ in } \mathbf{total_order}(\mathbf{g}, \lambda), \mathbf{u} \text{ is at the same level as } \mathbf{t}, \text{ and } \mathbf{u} \text{ covers } \mathbf{s}\}$. We visualize $\mathbf{s} \xrightarrow{i} \mathbf{t}$ as the “southeast” edge of the diamond  whenever $u \in \mathbf{diamond_set}(\mathbf{g}, \lambda, \mathbf{s}, \mathbf{t})$ and $v = \mathbf{lub}(\mathbf{u}, \mathbf{t})$. Additionally, define a function $m_i(\mathbf{g}, \mathbf{t})$ which returns the i th coordinate of the weight vector $wt(\mathbf{t})$. This number can be calculated using

Littelmann's definition of the weight rule:

$$m_i(\mathfrak{g}, \mathbf{t}) = \begin{cases} n_1(\mathbf{t}) - n_2(\mathbf{t}) & \text{for } i = 1 \text{ and } \mathfrak{g} = A_2 \\ n_2(\mathbf{t}) - n_3(\mathbf{t}) & \text{for } i = 2 \text{ and } \mathfrak{g} = A_2 \\ n_1(\mathbf{t}) - n_2(\mathbf{t}) + n_3(\mathbf{t}) - n_4(\mathbf{t}) & \text{for } i = 1 \text{ and } \mathfrak{g} = B_2 \\ n_2(\mathbf{t}) - n_3(\mathbf{t}) & \text{for } i = 2 \text{ and } \mathfrak{g} = B_2 \\ n_1(\mathbf{t}) - n_2(\mathbf{t}) + 2n_3(\mathbf{t}) - 2n_5(\mathbf{t}) + n_6(\mathbf{t}) - n_7(\mathbf{t}) & \text{for } i = 1 \text{ and } \mathfrak{g} = G_2 \\ n_2(\mathbf{t}) - n_3(\mathbf{t}) + n_5(\mathbf{t}) - n_6(\mathbf{t}) & \text{for } i = 2 \text{ and } \mathfrak{g} = G_2 \end{cases}$$

where $n_j(\mathbf{t})$ is the number of times the integer j appears as a box-entry in the tableau \mathbf{t} .

Algorithm 5.5 EDGES

INPUT: A rank two simple Lie Algebra \mathfrak{g} and two non-negative integers a and b that indicate the size of the tableaux λ .

OUTPUT: A set of edge coefficients or a failure error.

Step 1: Set $T := \mathbf{total_order}(\mathfrak{g}, \lambda)$, so $T = (\mathbf{t}_1, \dots, \mathbf{t}_d)$, where $d = \mathbf{lattice_}$

$\mathbf{dimension}(\mathfrak{g}, a, b)$.

Step 2: FOR j from 1 to d LOOP

 Set $\mathbf{t} := \mathbf{t}_j$.

 Sort $\mathbf{descendant_set}(\mathfrak{g}, \lambda, \mathbf{t})$ by the righthand lexicographic rule

 to get a sequence $(\mathbf{s}_1, \dots, \mathbf{s}_r)$. (Here, $\mathbf{descendant_set}(\mathfrak{g}, \lambda, \mathbf{t}) =$

$\{\mathbf{s}_1, \dots, \mathbf{s}_r\}$, where $r = |\mathbf{descendant_set}(\mathfrak{g}, \lambda, \mathbf{t})|$.)

 FOR k from 1 to r LOOP

 Set $\mathbf{s} := \mathbf{s}_k$.

 Determine color i of edge $\mathbf{s} \longrightarrow \mathbf{t}$.

Set **diamond** := **diamond_set**(**g**, **s**, **t**).

IF **diamond** \neq $\{\emptyset\}$ THEN

Let **u** \in **diamond**

IF $\pi_{\mathbf{s},\mathbf{u}} = 0$ THEN

RETURN(“Division by zero error”)

ELSE

Let **v** := **lub**(**u**, **t**).

Set $\pi_{\mathbf{s},\mathbf{t}} := \frac{\pi_{\mathbf{u},\mathbf{v}}\pi_{\mathbf{t},\mathbf{v}}}{\pi_{\mathbf{s},\mathbf{u}}}$

END IF

diamond := **diamond** \setminus $\{\mathbf{u}\}$.

WHILE **diamond** \neq $\{\emptyset\}$ LOOP

Let **u'** \in **diamond**

Let **v'** := **lub**(**u'**, **t**)

IF $\pi_{\mathbf{s},\mathbf{t}} \neq \frac{\pi_{\mathbf{u}',\mathbf{v}'}\pi_{\mathbf{t},\mathbf{v}'}}{\pi_{\mathbf{s},\mathbf{u}'}}$ THEN

RETURN(“Inconsistent Diamond Relation”)

END IF

diamond := **diamond** \setminus $\{\mathbf{u}'\}$.

END LOOP

ELSE

$\pi_{\mathbf{s},\mathbf{t}} := m_i(\mathbf{g}, \mathbf{t}) + \sum_{\mathbf{v}} \pi_{\mathbf{t},\mathbf{v}} - \sum_{\mathbf{r}} \pi_{\mathbf{r},\mathbf{t}}$. The first summation is taken over the set of all **v** such that $\mathbf{t} \xrightarrow{i} \mathbf{v}$. The second summation is taken over the set of all **r** such that $\mathbf{r} \neq \mathbf{s}$ and

```

         $\mathbf{r} \xrightarrow{i} \mathbf{t}.$ 

    END IF

END LOOP

FOR  $i$  from 1 to 2 LOOP

    IF  $m_i(\mathbf{g}, \mathbf{t}) \neq \sum_{\mathbf{s}:\mathbf{s} \xrightarrow{i} \mathbf{t}} \pi_{\mathbf{s},\mathbf{t}} - \sum_{\mathbf{v}:\mathbf{t} \xrightarrow{i} \mathbf{v}} \pi_{\mathbf{t},\mathbf{v}}$  THEN

        RETURN("Inconsistent Crossing Relation")

    END IF

END LOOP

END LOOP

```

Step 3: RETURN(Π) where Π is the sequence of all edge coefficients $\pi_{\mathbf{s},\mathbf{t}}$ indexed by the edges $\mathbf{s} \xrightarrow{i} \mathbf{t}$ in L_λ .

The total ordering of the **total_order** algorithm is critical in order to ensure that the **edges** algorithm is “well-defined,” i.e. at any stage in the algorithm, we have enough information to compute a new edge coefficient. This is the content of part (1) of the following theorem.

Theorem 5.6 *Let a and b be non-negative integers. Let $\lambda = (a, b)$. Let \mathfrak{g} be a rank two simple Lie Algebra.*

(1) *Preserving the notation of the preceding algorithm, suppose $\mathbf{diamond_set}(\mathfrak{g}, \mathbf{s}, \mathbf{t}) = \emptyset$ for edge $\mathbf{s} \xrightarrow{i} \mathbf{t}$. (That is, there does not exist a tableau \mathbf{u} such that (a) \mathbf{u} precedes \mathbf{t} in the total order, (b) \mathbf{u} is at the same level as \mathbf{t} , and (c) \mathbf{u} covers \mathbf{s} .) Then for any \mathbf{s}' such that $\mathbf{s}' \xrightarrow{i} \mathbf{t}$, it is the case that the coefficient $\pi_{\mathbf{s}',\mathbf{t}}$ has already been calculated. In*

particular, if there is no diamond with $\mathbf{s} \xrightarrow{i} \mathbf{t}$ as its “southeast” edge, then $\pi_{\mathbf{s},\mathbf{t}}$ can be computed with the crossing relation for color i at vertex \mathbf{t} .

- (2) If $\mathbf{edges}(\mathfrak{g}, \lambda)$ returns “Inconsistent” (i.e. the algorithm returns a consistency error), then the \mathfrak{g} -semistandard lattice L_λ is NOT a supporting graph for a representation of \mathfrak{g} .
- (3) If $\mathbf{edges}(\mathfrak{g}, \lambda)$ returns non-zero coefficients, then the \mathfrak{g} -semistandard lattice L_λ is a supporting graph for the representation of \mathfrak{g} with dominant weight λ .
- (4) Suppose L_λ is the supporting graph for a basis $\{v_{\mathbf{s}}\}$ for the irreducible representation of \mathfrak{g} with dominant weight λ , and suppose that relative to this basis the product of coefficients $\pi_{\mathbf{s},\mathbf{t}}^*$ attached to any edge $\mathbf{s} \xrightarrow{i} \mathbf{t}$ in L_λ is nonzero. Then $\mathbf{edges}(\mathfrak{g}, \lambda)$ returns a sequence of nonzero coefficients, and this sequence is precisely $(\pi_{\mathbf{s},\mathbf{t}}^*)$.

Part (4) of Theorem 5.4 is something of a converse to part (3). It says that whenever L_λ is known to be a supporting graph, then the actions along the edges of L_λ (or, more precisely, the products of the coefficients on any edge) are *uniquely* determined by the \mathbf{edges} algorithm. From Theorems 2.6, 3.6, and 4.6 we see that Theorem 5.4.4 applies to L_λ when λ is any dominant weight for A_2 ; when λ has the form $(a, 0)$ or $(0, b)$ for B_2 ; and when λ has the form $(a, 0)$ or $(0, 1)$ for G_2 . (The references cited in the proofs of Theorems 2.6, 3.6, and 4.6 actually demonstrate that these lattices are supporting graphs for bases whose actions yield *nonzero* edge coefficients.) It can be shown ([DLP2]) that this uniqueness of edge coefficients implies “uniqueness” of the basis in the representing space. That is, keeping the notation of the theorem statement, suppose $\{w_{\mathbf{s}}\}$ is another basis for the representation of \mathfrak{g} with dominant weight λ , and suppose L_λ is the supporting graph for this basis as well.

Then for each $\mathbf{s} \in L_\lambda$, there exists a nonzero scalar $a_{\mathbf{s}}$ such that $w_{\mathbf{s}} = a_{\mathbf{s}}v_{\mathbf{s}}$. It is in this sense that the combinatorics of a semistandard lattice supporting graph uniquely determines a basis for the corresponding representation as well as actions of the generators on that basis.

Proof of Theorem 5.6. The proof of part (1) has been demonstrated by Donnelly, Lewis, and Pervine [DLP2] by analyzing cases. Crucial to this case analysis is the total ordering of tableaux in the `total_order` algorithm. For part (2), we proceed by contradiction. Suppose that Algorithm 5.5 terminates at vertex \mathbf{t} with an inconsistency, but suppose L_λ is a supporting graph for a representation of \mathfrak{g} . Since L_λ is a supporting graph, it follows from the discussion of Chapter 1 that we can assign coefficients $c_{\mathbf{q},\mathbf{p}}^*$ and $d_{\mathbf{p},\mathbf{q}}^*$ to each edge $\mathbf{p} \xrightarrow{i} \mathbf{q}$ in L_λ in such a way that these coefficients meet the Diamond and Crossing Conditions. For any such edge $\mathbf{p} \xrightarrow{i} \mathbf{q}$, set $\pi_{\mathbf{p},\mathbf{q}}^* := c_{\mathbf{q},\mathbf{p}}^* d_{\mathbf{p},\mathbf{q}}^*$. It is easy to check that $\pi_{\mathbf{p},\mathbf{m}}^* = \pi_{\mathbf{p},\mathbf{m}}$ for any edge $\mathbf{p} \xrightarrow{i} \mathbf{m}$, where $\mathbf{m} = \max$ tableau of L_λ . (This follows from the Crossing Condition, since the i -component of any such edge $\mathbf{p} \xrightarrow{i} \mathbf{m}$ is a chain.) Let us suppose that Algorithm 5.5 terminates with a Diamond inconsistency at edge $\mathbf{s} \xrightarrow{i} \mathbf{t}$. Let $\mathbf{p} \xrightarrow{i} \mathbf{q}$ be any edge in L_λ with coefficient $\pi_{\mathbf{p},\mathbf{q}}$ successfully computed prior to termination. Since $\pi_{\mathbf{p},\mathbf{q}}$ must have been obtained from a Diamond or Crossing Relation, it follows easily from an inductive argument that $\pi_{\mathbf{p},\mathbf{q}}^* = \pi_{\mathbf{p},\mathbf{q}}$. Now following the Algorithm we see that

$$\pi_{\mathbf{s},\mathbf{t}} = \frac{\pi_{\mathbf{u},\mathbf{v}}\pi_{\mathbf{t},\mathbf{v}}}{\pi_{\mathbf{s},\mathbf{u}}} = \frac{\pi_{\mathbf{u},\mathbf{v}}^*\pi_{\mathbf{t},\mathbf{v}}^*}{\pi_{\mathbf{s},\mathbf{u}}^*} = \pi_{\mathbf{s},\mathbf{t}}^*.$$

Moreover, since L_λ is a supporting graph, the Diamond Condition implies that

$$\pi_{\mathbf{s},\mathbf{t}}^* = \frac{\pi_{\mathbf{u}',\mathbf{v}'}^*\pi_{\mathbf{t},\mathbf{v}'}^*}{\pi_{\mathbf{s},\mathbf{u}'}^*} = \frac{\pi_{\mathbf{u}',\mathbf{v}'}\pi_{\mathbf{t},\mathbf{v}'}}{\pi_{\mathbf{s},\mathbf{u}'}}.$$

But the algorithm terminates with

$$\pi_{s,t} \neq \frac{\pi_{u',v'} \pi_{t,v'}}{\pi_{s,u'}},$$

a contradiction. So suppose the algorithm terminates with a Crossing inconsistency. As before, for any edge $\mathbf{p} \xrightarrow{i} \mathbf{q}$ encountered prior to termination, we have $\pi_{\mathbf{p},\mathbf{q}}^* = \pi_{\mathbf{p},\mathbf{q}}$. But now just prior to termination we have

$$m_i(\mathbf{g}, \mathbf{t}) \neq \sum_{s:\mathbf{s} \xrightarrow{i} \mathbf{t}} \pi_{s,t} - \sum_{\mathbf{v}:\mathbf{t} \xrightarrow{i} \mathbf{v}} \pi_{t,v}.$$

However, the Crossing condition for L_λ implies that

$$m_i(\mathbf{g}, \mathbf{t}) = \sum_{s:\mathbf{s} \xrightarrow{i} \mathbf{t}} \pi_{s,t}^* - \sum_{\mathbf{v}:\mathbf{t} \xrightarrow{i} \mathbf{v}} \pi_{t,v}^* = \sum_{s:\mathbf{s} \xrightarrow{i} \mathbf{t}} \pi_{s,t} - \sum_{\mathbf{v}:\mathbf{t} \xrightarrow{i} \mathbf{v}} \pi_{t,v},$$

a contradiction. Then L_λ cannot be a supporting graph for a representation of \mathbf{g} .

For part (3), if $\mathbf{edges}(\mathbf{g}, \lambda)$ returns non-zero coefficients, then we have a unique assignment of non-zero coefficients to the edges of L_λ . Now one needs only to observe that every Diamond and every Crossing relation has been verified by Algorithm 5.5. Returning to the discussion of Chapter 1, we see that with this assignment of edge coefficients, L_λ meets all requirements sufficient to imply that the lattice is a supporting graph for the irreducible representation of \mathbf{g} with dominant weight λ .

Similar to the proof of part (2), an inductive argument for part (4) will show that $\pi_{\mathbf{p},\mathbf{q}} = \pi_{\mathbf{p},\mathbf{q}}^*$ for any edge $\mathbf{p} \xrightarrow{i} \mathbf{q}$ in L_λ . The assumption that each $\pi_{\mathbf{p},\mathbf{q}}^*$ is nonzero is necessary in order to avoid division by zero whenever a diamond relation is encountered. \square

CHAPTER 6

APPLICATIONS OF ALGORITHMS

Our first application of the algorithms of Chapter 5 and Theorem 5.4 will be an analysis of certain B_2 -semistandard lattices. In Theorem 6.1 we will show that if $\lambda = (a, b)$ with $a \geq 2$ and $b \geq 1$, then the B_2 -semistandard lattice L_λ will NOT be a supporting graph for a representation of B_2 . That is, we show that the SLC fails for these B_2 lattices. This result verifies, at least in part, data obtained experimentally and summarized in Table 1.1. Before the statement and proof of Theorem 6.1, we illustrate the **edges** algorithm for the concrete 35-dimensional B_2 -semistandard lattice L_λ with $\lambda = (2, 1)$.

In this chapter we also present a sampling of the \mathfrak{g} -semistandard lattices investigated during the course of this project. The pictures we present were obtained using the computer algebra system Maple to implement the algorithms of Chapter 5. (We used Maple V Release 5.1.) See Appendix A for the actual Maple code. Our presentations of the many lattice pictures below was significantly aided by our use of the “**posets**” package for Maple developed by John Stembridge of the University of Michigan [Stem]. In particular, this package contains several Maple procedures that were useful for allowing us to visualize the lattices once the edges were created.

B_2 Counterexamples to the Semistandard Lattice Conjecture

We begin by considering the 35-element B_2 -semistandard lattice L_λ corresponding to weight $\lambda = (2, 1)$ (see Figure 6.1). We will show the output of **edges**(B_2, λ) is an “Inconsis-

tent diamond relation” error. It follows from Theorem 5.6.2 that the B_2 lattice L_λ cannot be a supporting graph for a representation of B_2 . In the analysis of this lattice that follows, we will sometimes refer to vertex number k in Figure 6.1 by the symbol v_k .

Figure 6.1 Example lattice of the B_2 -semistandard lattice corresponding to $\lambda = (2, 1)$

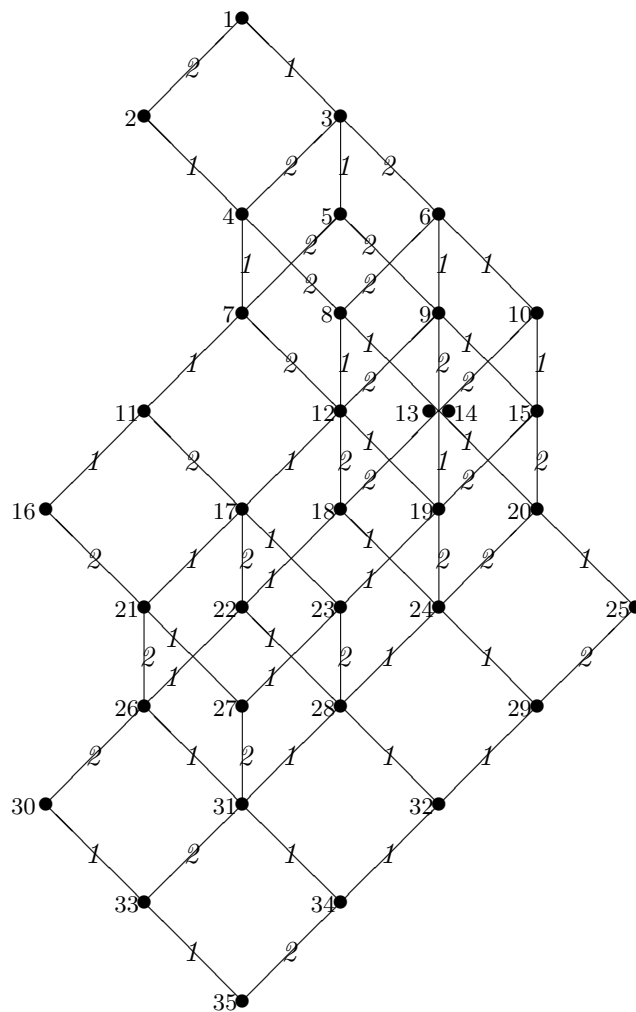
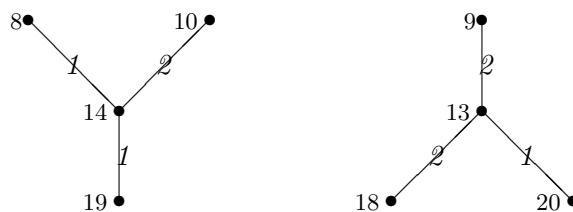
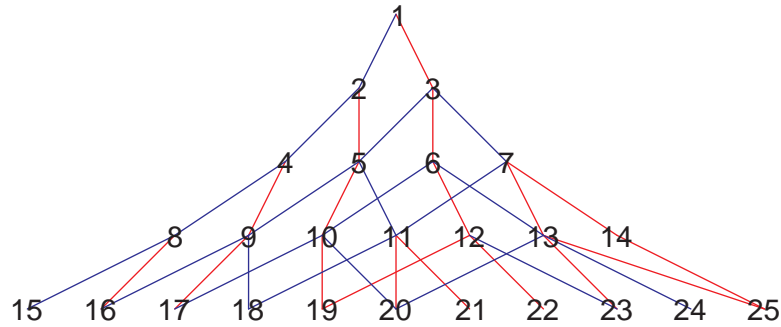


Figure 6.2 Clarification of vertices 13 and 14 for Figure 6.1



Start with vertex v_1 . Consider the edge $v_2 \xrightarrow{2} v_1$. Since this is not the “southeast” edge for any diamond in L_λ , we cannot use a diamond relation to compute $\pi_{2,1}$. Instead, utilize the crossing condition for color 2 to get $\pi_{2,1} = 1$. Next consider $v_3 \xrightarrow{1} v_1$. Again there is no appropriate diamond, but utilizing the crossing condition for color 1 we get $\pi_{3,1} = 2$. The **edges** algorithm now moves to vertex v_2 . Consider $v_4 \xrightarrow{1} v_2$. Again, a diamond appropriate for computing $\pi_{4,2}$ does not exist, so utilize the crossing condition for color 1 to get $\pi_{4,2} = 4$. Moving on to the next vertex v_3 , we see that the next edge $v_4 \xrightarrow{2} v_3$ is part of the 4-3-1-2 diamond. This gives $\pi_{4,3} = \frac{1}{2}$. The next edge $v_5 \xrightarrow{1} v_3$ cannot be computed using a diamond relation, but the crossing condition for color 1 gives $\pi_{5,3} = 2$. Edge $v_6 \xrightarrow{2} v_3$ does not involve an appropriate diamond either, but using the crossing condition for color 2 (note that $\pi_{4,3}$ has already been found) we get $\pi_{6,3} = \frac{3}{2}$. Continuing as such, one obtains: $\pi_{2,1} = 1$, $\pi_{3,1} = 2$, $\pi_{4,2} = 4$, $\pi_{4,3} = \frac{1}{2}$, $\pi_{5,3} = 2$, $\pi_{6,3} = \frac{3}{2}$, $\pi_{7,4} = 6$, $\pi_{8,4} = \frac{1}{2}$, $\pi_{7,5} = \frac{1}{6}$, $\pi_{9,5} = \frac{17}{6}$, $\pi_{8,6} = \frac{3}{2}$, $\pi_{9,6} = \frac{18}{17}$, $\pi_{10,6} = \frac{16}{17}$, $\pi_{11,7} = 6$, $\pi_{12,7} = \frac{7}{6}$, $\pi_{12,8} = \frac{18}{7}$, and $\pi_{14,8} = \frac{10}{7}$.

Now, moving to vertex v_9 and choosing the edge $v_{12} \xrightarrow{2} v_9$, there exist two diamonds: 9-5-7-12 and 9-6-8-12. Coefficients on all of the edges in these diamonds have been computed already, with the exception of $\pi_{12,9}$. Diamond 9-5-7-12 yields $\pi_{12,9} = \frac{17}{42}$, but diamond 9-6-8-12 yields $\pi_{12,9} = \frac{21}{34}$. Thus the Diamond Condition will fail on at least one of these diamonds. Therefore, we have an inconsistency in the diamond relations. By Theorem 5.6.2, the B_2 -semistandard lattice L_λ with $\lambda = (2, 1)$ is not a supporting graph for a representation of B_2 . The next theorem provides a more general argument demonstrating that the SLC fails for a large class of B_2 -semistandard lattices.

Figure 6.3 Top 5 ranks (levels) in a generic B_2 -semistandard lattice L_λ 

Theorem 6.1 Let $\lambda = (a, b)$ for integers $a \geq 2$ and $b \geq 1$. Then the B_2 -semistandard lattice L_λ is NOT a supporting graph for a representation of B_2 .

Proof. We show that for any such λ , $\mathbf{edges}(B_2, \lambda)$ returns a diamond inconsistency. It will follow then from Theorem 5.4.2 that the B_2 -semistandard lattice L_λ cannot be a supporting graph for a representation of B_2 . We will assume throughout that $a \geq 4$ and $b \geq 4$. The remaining cases ($a \in \{2, 3\}$ and $b \geq 1$; $b \in \{2, 3\}$ and $a \geq 2$) can, with a little work, be viewed as special cases of the argument that follows.

Let P_λ be the subposet of all vertices of L_λ within 4 steps (inclusive) of the max tableau of L_λ , together with the ordering on these tableaux induced by L_λ . As a set, $P_\lambda = L_\lambda^0 \vee L_\lambda^1 \vee L_\lambda^2 \vee L_\lambda^3 \vee L_\lambda^4$. Let $\mu := (4, 4)$. Observe that P_λ and P_μ are identical (isomorphic as posets). To see this, note that if a tableau \mathbf{t} is within 4 steps of the max tableau, then it differs from the max tableau in at most 4 different columns. Thus the first $b - 4$ columns of \mathbf{t} will be $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$'s, while $\mathbf{t}_{b+1} = \mathbf{t}_{b+2} = \dots = \mathbf{t}_{a+b-4} = \begin{bmatrix} 1 \end{bmatrix}$. So we can identify the tableau \mathbf{t} in P_λ with the tableau $(\mathbf{t}_{b-3}, \mathbf{t}_{b-2}, \mathbf{t}_{b-1}, \mathbf{t}_b, \mathbf{t}_{a+b-3}, \mathbf{t}_{a+b-2}, \mathbf{t}_{a+b-1}, \mathbf{t}_{a+b} \in P_\mu)$.

We depict a generic P_λ in Figure 6.3. In this and all remaining figures, the red edges in these lattices correspond to color 1 while the blue edges correspond to color 2. In Table

Table 6.1 Vertices in the top 5 ranks (levels) of a generic B_2 -semistandard lattice

Level	Vertex \mathbf{t}	Distance from Boundary	$m_1(\mathbf{t}) = 2\rho_1(\mathbf{t}) - l_1(\mathbf{t})$	$m_2(\mathbf{t}) = 2\rho_2(\mathbf{t}) - l_2(\mathbf{t})$
0	$v_1 = \begin{smallmatrix} 1 & \dots & 1 & 1 & \dots & 1 \\ 2 & \dots & 2 & & & \end{smallmatrix}$	0	a	b
1	$v_2 = \begin{smallmatrix} 1 & \dots & 1 & 1 & 1 & \dots & 1 \\ 2 & \dots & 2 & 3 & & & \end{smallmatrix}$	0	$a + 2$	$b - 2$
	$v_3 = \begin{smallmatrix} 1 & \dots & 1 & 1 & \dots & 1 & 2 \\ 2 & \dots & 2 & & & & \end{smallmatrix}$	2	$a - 2$	$b + 1$
2	$v_4 = \begin{smallmatrix} 1 & \dots & 1 & 1 & 1 & 1 & \dots & 1 \\ 2 & \dots & 2 & 3 & 3 & & & \end{smallmatrix}$	0	$a + 4$	$b - 4$
	$v_5 = \begin{smallmatrix} 1 & \dots & 1 & 1 & 1 & \dots & 1 & 2 \\ 2 & \dots & 2 & 3 & & & & \end{smallmatrix}$	2	a	$b - 1$
	$v_6 = \begin{smallmatrix} 1 & \dots & 1 & 1 & \dots & 1 & 2 & 2 \\ 2 & \dots & 2 & & & & & \end{smallmatrix}$	4	$a - 4$	$b + 2$
	$v_7 = \begin{smallmatrix} 1 & \dots & 1 & 1 & \dots & 1 & 3 \\ 2 & \dots & 2 & & & & \end{smallmatrix}$	4	a	$b - 1$
3	$v_8 = \begin{smallmatrix} 1 & \dots & 1 & 1 & 1 & 1 & 1 & \dots & 1 \\ 2 & \dots & 2 & 3 & 3 & 3 & & & \end{smallmatrix}$	0	$a + 6$	$b - 6$
	$v_9 = \begin{smallmatrix} 1 & \dots & 1 & 1 & 1 & 1 & \dots & 1 & 2 \\ 2 & \dots & 2 & 3 & 3 & & & & \end{smallmatrix}$	2	$a + 2$	$b - 3$
	$v_{10} = \begin{smallmatrix} 1 & \dots & 1 & 1 & 1 & \dots & 1 & 2 & 2 \\ 2 & \dots & 2 & 3 & & & & & \end{smallmatrix}$	4	$a - 2$	b
	$v_{11} = \begin{smallmatrix} 1 & \dots & 1 & 1 & 1 & \dots & 1 & 3 \\ 2 & \dots & 2 & 3 & & & & \end{smallmatrix}$	4	$a + 2$	$b - 3$
	$v_{12} = \begin{smallmatrix} 1 & \dots & 1 & 1 & \dots & 1 & 2 & 2 & 2 \\ 2 & \dots & 2 & & & & & & \end{smallmatrix}$	6	$a - 6$	$b + 3$
	$v_{13} = \begin{smallmatrix} 1 & \dots & 1 & 1 & \dots & 1 & 2 & 3 \\ 2 & \dots & 2 & & & & & \end{smallmatrix}$	6	$a - 2$	b
	$v_{14} = \begin{smallmatrix} 1 & \dots & 1 & 1 & \dots & 1 & 4 \\ 2 & \dots & 2 & & & & \end{smallmatrix}$	6	$a - 2$	b
4	$v_{15} = \begin{smallmatrix} 1 & \dots & 1 & 1 & 1 & 1 & 1 & 1 & \dots & 1 \\ 2 & \dots & 2 & 3 & 3 & 3 & 3 & & & \end{smallmatrix}$	0	$a + 8$	$b - 8$
	$v_{16} = \begin{smallmatrix} 1 & \dots & 1 & 1 & 1 & 1 & 1 & \dots & 1 & 2 \\ 2 & \dots & 2 & 3 & 3 & 3 & & & & \end{smallmatrix}$	2	$a + 4$	$b - 5$
	$v_{17} = \begin{smallmatrix} 1 & \dots & 1 & 1 & 1 & 1 & \dots & 1 & 2 & 2 \\ 2 & \dots & 2 & 3 & 3 & & & & & \end{smallmatrix}$	4	a	$b - 2$
	$v_{18} = \begin{smallmatrix} 1 & \dots & 1 & 1 & 1 & 1 & \dots & 1 & 3 \\ 2 & \dots & 2 & 3 & 3 & & & & \end{smallmatrix}$	4	$a + 4$	$b - 5$
	$v_{19} = \begin{smallmatrix} 1 & \dots & 1 & 1 & 1 & \dots & 1 & 2 & 2 & 2 \\ 2 & \dots & 2 & 3 & & & & & & \end{smallmatrix}$	6	$a - 4$	$b + 1$
	$v_{20} = \begin{smallmatrix} 1 & \dots & 1 & 1 & 1 & \dots & 1 & 2 & 3 \\ 2 & \dots & 2 & 3 & & & & & \end{smallmatrix}$	6	a	$b - 2$
	$v_{21} = \begin{smallmatrix} 1 & \dots & 1 & 1 & 1 & \dots & 1 & 4 \\ 2 & \dots & 2 & 3 & & & & \end{smallmatrix}$	6	a	$b - 2$
	$v_{22} = \begin{smallmatrix} 1 & \dots & 1 & 1 & \dots & 1 & 2 & 2 & 2 & 2 \\ 2 & \dots & 2 & & & & & & & \end{smallmatrix}$	8	$a - 8$	$b + 4$
	$v_{23} = \begin{smallmatrix} 1 & \dots & 1 & 1 & \dots & 1 & 2 & 2 & 3 \\ 2 & \dots & 2 & & & & & & \end{smallmatrix}$	8	$a - 4$	$b + 1$
	$v_{24} = \begin{smallmatrix} 1 & \dots & 1 & 1 & \dots & 1 & 3 & 3 \\ 2 & \dots & 2 & & & & & \end{smallmatrix}$	8	a	$b - 2$
	$v_{25} = \begin{smallmatrix} 1 & \dots & 1 & 1 & \dots & 1 & 2 & 4 \\ 2 & \dots & 2 & & & & & \end{smallmatrix}$	8	$a - 4$	$b + 1$

6.1, we list the 25 vertices of P_λ , record the distance of each vertex from the corresponding boundary element, and compute the weight of each vertex (thought of now as a element of L_λ). Using this data, we proceed through the **edges** algorithm. The coefficients for the edges below vertices v_1 through v_{12} are $\pi_{2,1} = b$, $\pi_{3,1} = a$, $\pi_{4,2} = 2(b-1)$, $\pi_{5,2} = a+2$, $\pi_{5,3} = \frac{ab}{a+2}$, $\pi_{6,3} = 2(a-1)$, $\pi_{7,3} = \frac{a+2b+2}{a+2}$, $\pi_{8,4} = 3(b-2)$, $\pi_{9,4} = a+4$, $\pi_{9,5} = \frac{2(a+2)(b-1)}{a+4}$, $\pi_{10,5} = 2(a+1)$, $\pi_{11,5} = \frac{a(a+2b+2)}{(a+2)(a+4)}$, $\pi_{10,6} = \frac{ab(a-1)}{(a+1)(a+2)}$, $\pi_{12,6} = 3(a-2)$, $\pi_{13,6} = \frac{2(2ab+b+a^2+3a+2)}{(a+1)(a+2)}$, $\pi_{11,7} = \frac{b(a+4)}{a+2}$, $\pi_{13,7} = \frac{(a-1)(a+1)(a+2b+2)}{2ab+b+a^2+3a+2}$, $\pi_{14,7} = \frac{(a+2)(a+b+1)}{2ab+b+a^2+3a+2}$, $\pi_{15,8} = 4(b-3)$, $\pi_{16,8} = a+6$, $\pi_{16,9} = \frac{3(a+4)(b-2)}{a+6}$, $\pi_{17,9} = 2(a+3)$, $\pi_{18,9} = \frac{a(a+2b+2)}{(a+6)(a+4)}$, $\pi_{17,10} = \frac{2(a+1)(a+2)(b-1)}{(a+3)(a+4)}$, $\pi_{19,10} = 3a$, $\pi_{20,10} = \frac{2(2a^3b+7a^2b+7ab+8b+a^4+6a^3+13a^2+12a+4)}{(a+1)(a+2)(a+3)(a+4)}$, $\pi_{18,11} = \frac{2(a+6)(b-1)}{a+4}$, $\pi_{20,11} = \frac{a(a+1)^2(a+3)(a+2b+2)}{2a^3b+7a^2b+7ab+8b+a^4+6a^3+13a^2+12a+4}$, $\pi_{21,11} = \frac{(a+4)(a^3+a^2b+4a^2+3ab+5a+4b+2)}{2a^3b+7a^2b+7ab+8b+a^4+6a^3+13a^2+12a+4}$, $\pi_{19,12} = \frac{b(a-1)(a-2)}{(a+1)(a+2)}$, $\pi_{22,12} = 4(a-3)$, and $\pi_{23,12} = \frac{3(2ab+a^2+3a+2)}{(a+1)(a+2)}$.

One can now check that at vertex v_{13} , diamond 20-13-6-10 returns

$$\pi_{20,13} = \frac{ab(a-1)(2ab+b+a^2+3a+2)(a+3)(a+4)}{(2a^3b+7a^2b+7ab+8b+a^4+6a^3+13a^2+12a+4)(a+1)(a+2)}$$

while diamond 20-13-7-11 returns

$$\pi_{20,13} = \frac{b(a+4)(a-1)(2a^3b+7a^2b+7ab+8b+a^4+6a^3+13a^2+12a+4)}{a(a+1)(a+2)(a+3)(2ab+b+a^2+3a+2)}.$$

Thus, **edges**(B_2, λ) terminates with an inconsistent diamond relation. \square

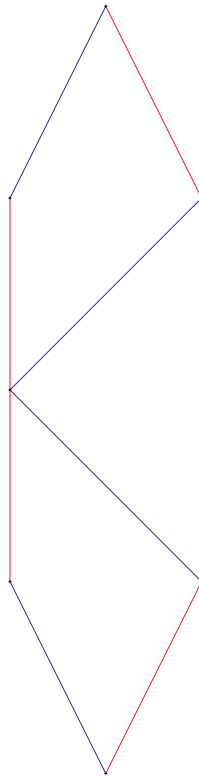
Examples of Semistandard Lattices

In what follows, we present examples of A_2 , B_2 , and G_2 semistandard lattices, each of which is in fact a supporting graph for a representation of the appropriate rank two simple Lie algebra. These examples were generated using Maple procedures which can be found in Appendix A. Three of these examples (Figures 6.4, 6.7 and 6.10) have appeared previously in Chapters 2, 3, and 4 of this thesis. Our programs are capable of attaching to each

vertex the index of that vertex relative to the total ordering supplied by Algorithm 5.3. In displaying each of the following examples, we have omitted these vertex numbers; however, the reader should note that the vertices are depicted in order if one reads the lattice from top to bottom, and left to right across each level.

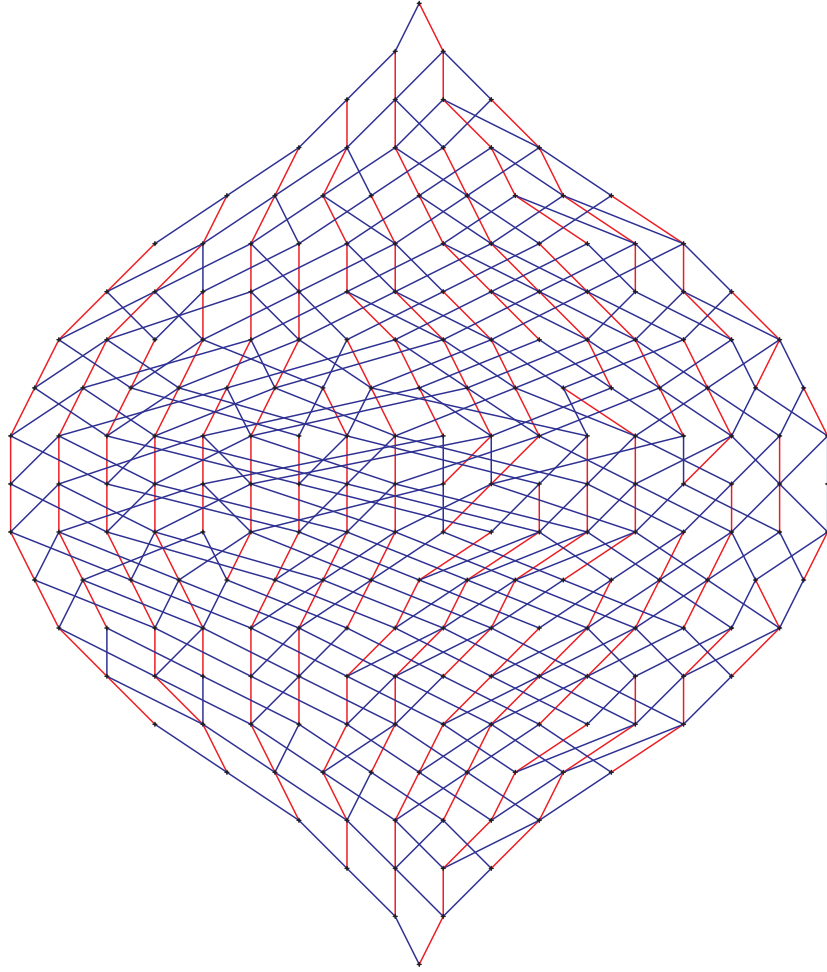
Figure 6.4 is the computer generated output for the example detailed in Example 2.1. As a reminder, it is an 8-dimensional A_2 semistandard lattice. Additionally, it has 5 levels, 4 red edges and 6 blue edges.

Figure 6.4 The A_2 -semistandard lattice corresponding to shape $\lambda = (1, 1)$



The next example, Figure 6.5, is of a 216-dimensional A_2 semistandard lattice with 20 levels, 180 red edges and 330 blue edges.

Figure 6.5 The A_2 -semistandard lattice corresponding to shape $\lambda = (5, 5)$



The next example, Figure 6.6, is a 1331-dimensional A_2 semistandard lattice with 40 levels, 1210 red edges and 2310 blue edges.

Figure 6.6 The A_2 -semistandard lattice corresponding to shape $\lambda = (10, 10)$

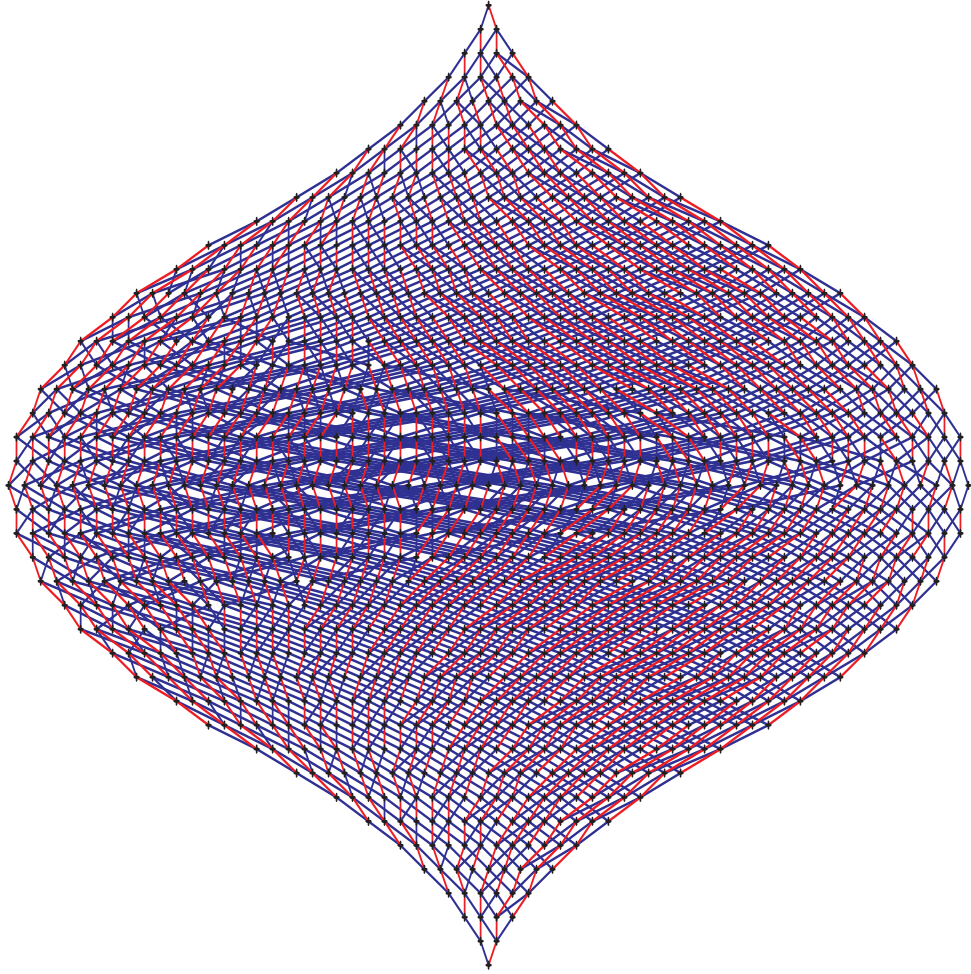
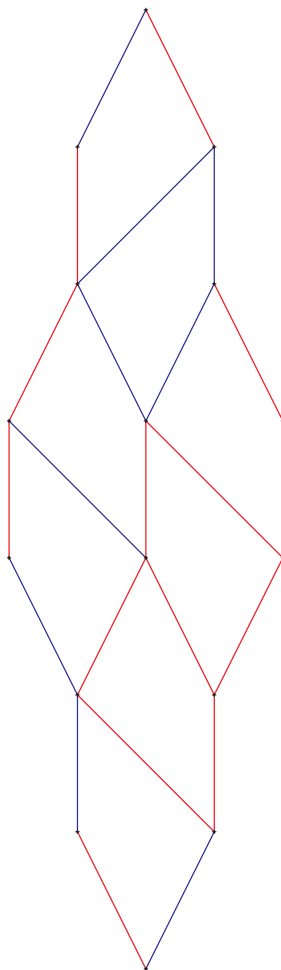


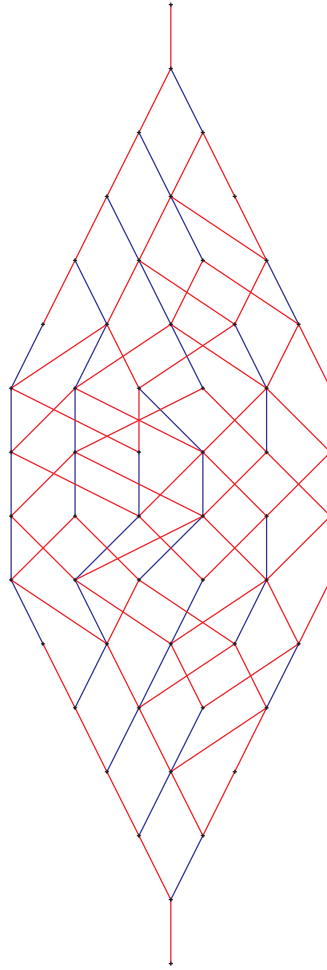
Figure 6.7 is the computer generated output for the example detailed in Example 3.1. As a reminder, it is a 16-dimensional B_2 semistandard lattice. Additionally, it has 8 levels, 13 red edges and 10 blue edges.

Figure 6.7 The B_2 -semistandard lattice corresponding to shape $\lambda = (1, 1)$



The next example, Figure 6.8, is of a 56-dimensional B_2 semistandard lattice with 16 levels, 70 red edges and 35 blue edges.

Figure 6.8 The B_2 -semistandard lattice corresponding to shape $\lambda = (5, 0)$



The next example, Figure 6.9, is of a 91-dimensional B_2 semistandard lattice with 21 levels, 70 red edges and 110 blue edges.

Figure 6.9 The B_2 -semistandard lattice corresponding to shape $\lambda = (0, 5)$

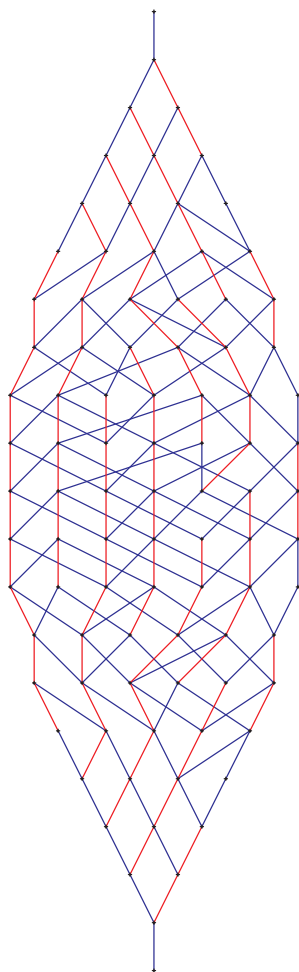
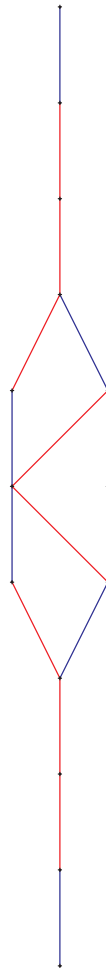


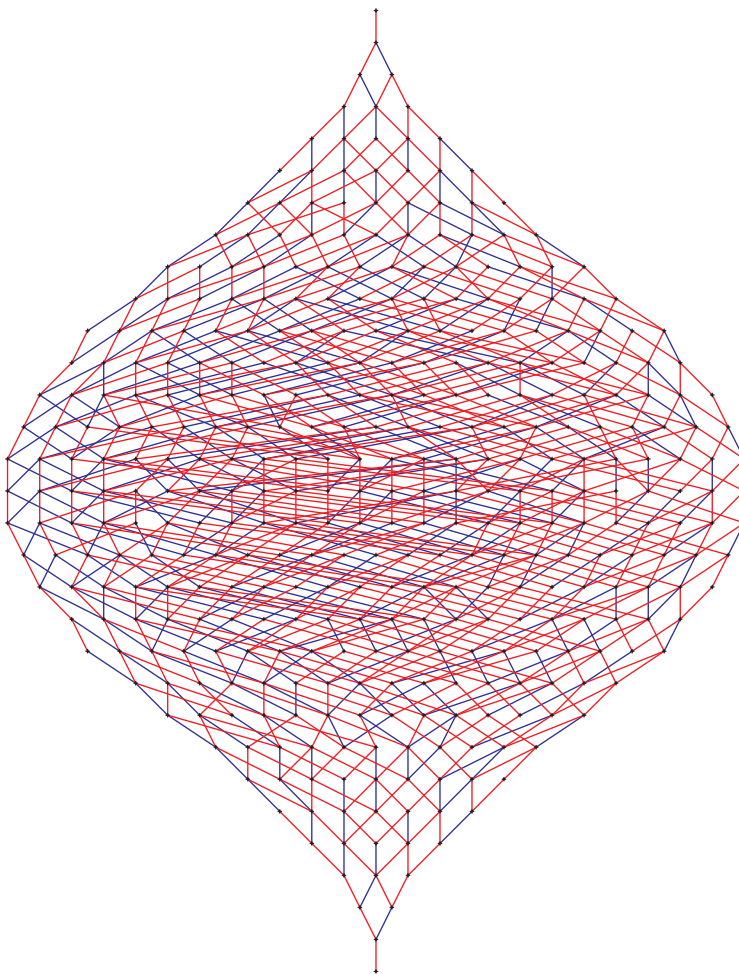
Figure 6.10 is the computer generated output for the example detailed in Example 4.1. As a reminder, it is a 14-dimensional G_2 semistandard lattice. Additionally, it has 11 levels, 10 red edges and 6 blue edges.

Figure 6.10 The G_2 -semistandard lattice corresponding to shape $\lambda = (0, 1)$



The next example, Figure 6.11, is of a 378-dimensional G_2 semistandard lattice with 31 levels, 616 red edges and 364 blue edges.

Figure 6.11 The G_2 -semistandard lattice corresponding to shape $\lambda = (5, 0)$



APPENDIX A

MAPLE IMPLEMENTATION

This appendix contains the actual Maple implementation of the three algorithms and numerous functions discussed in Chapter 5. Prior to each Maple procedure is a paragraph describing output, any special requirements of procedure input, and any other noteworthy particulars.

The procedure **lattice_length** takes as input a rank two simple Lie Algebra \mathfrak{g} and two non-negative integers a and b . These numbers indicate the shape for the tableaux that are to be generated; in algebraic terms, the numbers a and b also specify the irreducible representation of the Lie algebra \mathfrak{g} corresponding to the dominant weight $\lambda = (a, b)$. The output of the procedure is the total number of steps from the maximal tableaux to the minimal tableaux in the \mathfrak{g} -semistandard lattice L_λ .

```
lattice_length := proc(g,a,b)
  if g = 'A2' then
    RETURN(2*a+2*b):
  elif g = 'B2' then
    RETURN(3*a+4*b):
  elif g = 'G2' then
    RETURN(6*a+10*b):
  fi:
end:
```

The procedure **lattice_dimension** takes as input a Lie Algebra \mathfrak{g} and two non-negative integers a and b as described above. The output of the procedure is the total number of tableaux in L_λ .

```
lattice_dimension := proc(g,a,b)
```

```

if g = 'A2' then
  RETURN((a+1)*(b+1)*(a+b+2)/2):
elif g = 'B2' then
  RETURN((a+1)*(b+1)*(a+b+2)*(a+2*b+3)/3!):
elif g = 'G2' then
  RETURN((a+1)*(b+1)*(a+b+2)*(a+2*b+3)*(a+3*b+4)*(2*a+3*b+5)/5!):
fi:
end:

```

The procedure **tableaux** takes as input a Lie Algebra \mathfrak{g} and two non-negative integers a and b as described above. The output of the procedure is an array which has $\mathbf{lattice_length}(\mathfrak{g}, a, b) + 1$ entries. The i^{th} entry ($0 \leq i \leq \mathbf{lattice_length}(\mathfrak{g}, a, b)$) of this array is a set containing all tableaux that are i levels below the top level (so the maximal tableaux is at the 0^{th} level). To display the output as the procedure generates it, include in the procedure call the optional argument **display** as a third input parameter. To simplify the data storage, we adopted a slightly different convention for denoting the tableaux of shape λ (see Appendix B).

```

tableaux := proc(g,a,b)
  local L, length, t, i, j, display_flag:
  if nargs >= 4 and args[4]='display' then
    display_flag := true:
  else
    display_flag := false:
  fi:
  length:=lattice_length(g,a,b):
  L:=array(0..length):
  if g = 'A2' then
    L[0]:=[seq(1,i=1..b),seq(4,i=(b+1)..(a+b))]:
    if display_flag then print(L[0]) fi:
    for i from 1 to length do
      L[i]:={}:
      for t in L[i-1] do
        for j from 1 to (a+b) do
          if t[j] = 1 and (j = a+b or member(t[j+1],{2,3,4,5,6})) then
            t[j]:=2:
            L[i] := L[i] union {t}:
            t[j]:=1:
          elif t[j] = 2 and (j = a+b or member(t[j+1],{3,5,6})) then
            t[j]:=3:
            L[i] := L[i] union {t}:

```

```

        t[j]:=2:
    elif t[j] = 4 and (j = a+b or member(t[j+1],{5,6})) then
        t[j]:=5:
        L[i] := L[i] union {t}:
        t[j]:=4:
    elif t[j] = 5 and (j = a+b or member(t[j+1],{6})) then
        t[j] := 6:
        L[i] := L[i] union {t}:
        t[j] := 5:
    fi:
od:
od:
if display_flag then print(L[i]) fi:
od:
elif g = 'B2' then
    L[0]:={seq(1,i=1..b),seq(6,i=(b+1)..(a+b))}:
    if display_flag then print(L[0]) fi:
    for i from 1 to length do
        L[i]:={}:
        for t in L[i-1] do
            for j from 1 to (a+b) do
                if t[j] = 1 and (j = a+b or
                    member(t[j+1],{2,3,4,5,6,7,8,9})) then
                    t[j]:=2:
                    L[i] := L[i] union {t}:
                    t[j]:=1:
                elif t[j] = 2 and (j = a+b or member(t[j+1],{4,5,7,8,9})) then
                    t[j]:=3:
                    L[i] := L[i] union {t}:
                    t[j]:=2:
                elif t[j] = 3 and (j = a+b or member(t[j+1],{4,5,7,8,9})) then
                    t[j]:=4:
                    L[i] := L[i] union {t}:
                    t[j]:=3:
                elif t[j] = 4 and (j = a+b or member(t[j+1],{5,8,9})) then
                    t[j]:=5:
                    L[i] := L[i] union {t}:
                    t[j]:=4:
                elif t[j] = 6 and (j = a+b or member(t[j+1],{7,8,9})) then
                    t[j] := 7:
                    L[i] := L[i] union {t}:
                    t[j] := 6:
                elif t[j] = 7 and (j = a+b or member(t[j+1],{8,9})) then
                    t[j] := 8:
                    L[i] := L[i] union {t}:
                    t[j] := 7:
                elif t[j] = 8 and (j = a+b or member(t[j+1],{9})) then
                    t[j] := 9:
                    L[i] := L[i] union {t}:
                    t[j] := 8:
            fi:
        end
    end
end

```



```

    od:
  od:
  if display_flag then print(L[i]) fi:
od:
elif g = 'G2' then
  L[0]:={seq(1,i=1..b),seq(15,i=(b+1)..(a+b))}:
  if display_flag then print(L[0]) fi:
  for i from 1 to length do
    L[i]:={}:
    for t in L[i-1] do
      for j from 1 to (a+b) do
        if t[j] = 1 and (j = a+b or
          member(t[j+1],{2,3,4,5,6,7,8,9,10,11,12,13,
            14,15,16,17,18,19,20,21})) then
          t[j]:=2:
          L[i] := L[i] union {t}:
          t[j]:=1:
        elif t[j] = 2 and (j = a+b or
          member(t[j+1],{5,7,9,10,11,13,14,16,17,19,20,21})) then
          t[j]:=3:
          L[i] := L[i] union {t}:
          t[j]:=2:
        elif t[j] = 3 and (j = a+b or
          member(t[j+1],{5,7,9,10,11,13,14,16,17,19,20,21})) then
          t[j]:=4:
          L[i] := L[i] union {t}:
          t[j]:=3:
        elif t[j] = 4 and (j = a+b or
          member(t[j+1],{5,7,9,10,11,12,13,14,16,17,18,19,20,21})) then
          if (j = a+b or
            member(t[j+1],{9,11,12,13,14,17,18,19,20,21})) then
            t[j]:=6:
            L[i] := L[i] union {t}:
          fi:
          t[j]:=5:
          L[i] := L[i] union {t}:
          t[j]:=4:
        elif t[j] = 5 and (j = a+b or
          member(t[j+1],{9,11,12,13,14,17,18,19,20,21})) then
          t[j] := 7:
          L[i] := L[i] union {t}:
          t[j] := 5:
        elif t[j] = 6 and (j = a+b or
          member(t[j+1],{9,11,12,13,14,17,18,19,20,21})) then
          if (j = a+b or member(t[j+1],{13,14,19,20,21})) then
            t[j]:=8:
            L[i] := L[i] union {t}:
          fi:
          t[j]:=7:
          L[i] := L[i] union {t}:
          t[j]:=6:

```

```

elif t[j] = 7 and (j = a+b or
member(t[j+1],{9,11,12,13,14,17,18,19,20,21})) then
  if (j = a+b or member(t[j+1],{13,14,19,20,21})) then
    t[j]:=10:
    L[i] := L[i] union {t}:
  fi:
  t[j]:=9:
  L[i] := L[i] union {t}:
  t[j]:=7:
elif t[j] = 9 and (j = a+b or member(t[j+1],{13,14,19,20,21})) then
  t[j] := 11:
  L[i] := L[i] union {t}:
  t[j] := 9:
elif t[j] = 10 and (j = a+b or member(t[j+1],{13,14,19,20,21})) then
  t[j] := 11:
  L[i] := L[i] union {t}:
  t[j] := 10:
elif t[j] = 11 and (j = a+b or member(t[j+1],{13,14,19,20,21})) then
  t[j] := 12:
  L[i] := L[i] union {t}:
  t[j] := 11:
elif t[j] = 12 and (j = a+b or member(t[j+1],{13,14,19,20,21})) then
  t[j] := 13:
  L[i] := L[i] union {t}:
  t[j] := 12:
elif t[j] = 13 and (j = a+b or member(t[j+1],{14,20,21})) then
  t[j] := 14:
  L[i] := L[i] union {t}:
  t[j] := 13:
elif t[j] = 15 and (j = a+b or member(t[j+1],{16,17,18,19,20,21})) then
  t[j] := 16:
  L[i] := L[i] union {t}:
  t[j] := 15:
elif t[j] = 16 and (j = a+b or member(t[j+1],{17,18,19,20,21})) then
  t[j] := 17:
  L[i] := L[i] union {t}:
  t[j] := 16:
elif t[j] = 17 and (j = a+b or member(t[j+1],{19,20,21})) then
  t[j] := 18:
  L[i] := L[i] union {t}:
  t[j] := 17:
elif t[j] = 18 and (j = a+b or member(t[j+1],{19,20,21})) then
  t[j] := 19:
  L[i] := L[i] union {t}:
  t[j] := 18:
elif t[j] = 19 and (j = a+b or member(t[j+1],{20,21})) then
  t[j] := 20:
  L[i] := L[i] union {t}:
  t[j] := 19:
elif t[j] = 20 and (j = a+b or member(t[j+1],{21})) then
  t[j] := 21:

```

```

        L[i] := L[i] union {t}:
        t[j] := 20:
    fi:
od:
od:
if display_flag then print(L[i]) fi:
od:
fi:
RETURN(L):
end:

```

The procedure **boundary** takes as input a Lie Algebra \mathfrak{g} and two non-negative integers a and b as described above. The output of the procedure is an array which has $\mathbf{lattice_length}(\mathfrak{g}, a, b) + 1$ entries. The i^{th} entry ($0 \leq i \leq \mathbf{lattice_length}(\mathfrak{g}, a, b)$) of this array is the tableau that is i levels below the top level and is the “leftmost” among the tableaux of the i^{th} level. See Appendix B for a detailed definition of the boundary vertices for the semistandard lattices of each type (A_2 , B_2 , or G_2).

```

boundary := proc(g,a,b)
local B,i,k,length:
length:=lattice_length(g,a,b):
B := array(0..length):
if g = 'A2' then
for k from 0 to b do
B[k]:= [seq(1,i=1..b-k),seq(2,i=b-k+1..b),seq(4,i=(b+1)..(a+b))]:
od:
for k from 1 to a do
B[b+k]:= [seq(2,i=1..b),seq(4,i=(b+1)..(a+b-k)),seq(5,i=a+b-k+1..a+b)]:
od:
for k from 1 to b do
B[a+b+k]:= [seq(2,i=1..b-k),seq(3,i=b-k+1..b),seq(5,i=b+1..a+b)]:
od:
for k from 1 to a do
B[a+2*b+k]:= [seq(3,i=1..b),seq(5,i=b+1..a+b-k),seq(6,i=a+b-k+1..a+b)]:
od:
elif g = 'B2' then
for k from 0 to b do
B[k]:= [seq(1,i=1..b-k),seq(2,i=b-k+1..b),seq(6,i=(b+1)..(a+b))]:
od:
for k from 1 to a do
B[b+k]:= [seq(2,i=1..b),seq(6,i=(b+1)..(a+b-k)),seq(7,i=a+b-k+1..a+b)]:
od:
for k from 1 to b do

```

```

    B[a+b+2*k-1]:=[seq(2,i=1..b-k),3,seq(4,i=b-k+2..b),seq(7,i=b+1..a+b)]:
    B[a+b+2*k]:=[seq(2,i=1..b-k),seq(4,i=b-k+1..b),seq(7,i=b+1..a+b)]:
  od:
  for k from 1 to a do
    B[a+3*b+k]:=[seq(4,i=1..b),seq(7,i=b+1..a+b-k),seq(8,i=a+b-k+1..a+b)]:
  od:
  for k from 1 to b do
    B[2*a+3*b+k]:=[seq(4,i=1..b-k),seq(5,i=b-k+1..b),seq(8,i=(b+1)..(a+b))]:
  od:
  for k from 1 to a do
    B[2*a+4*b+k]:=[seq(5,i=1..b),seq(8,i=b+1..a+b-k),seq(9,i=a+b-k+1..a+b)]:
  od:
elif g = 'G2' then
  for k from 0 to b do
    B[k]:=[seq(1,i=1..b-k),seq(2,i=b-k+1..b),seq(15,i=(b+1)..(a+b))]:
  od:
  for k from 1 to a do
    B[b+k]:=[seq(2,i=1..b),seq(15,i=(b+1)..(a+b-k)),seq(16,i=a+b-k+1..a+b)]:
  od:
  for k from 1 to b do
    B[a+b+3*k-2]:=[seq(2,i=1..b-k),3,seq(5,i=b-k+2..b),seq(16,i=b+1..a+b)]:
    B[a+b+3*k-1]:=[seq(2,i=1..b-k),4,seq(5,i=b-k+2..b),seq(16,i=b+1..a+b)]:
    B[a+b+3*k]:=[seq(2,i=1..b-k),seq(5,i=b-k+1..b),seq(16,i=b+1..a+b)]:
  od:
  for k from 1 to a do
    B[a+4*b+k]:=[seq(5,i=1..b),seq(16,i=b+1..a+b-k),seq(17,i=a+b-k+1..a+b)]:
  od:
  for k from 1 to b do
    B[2*a+4*b+2*k-1]:=[seq(5,i=1..b-k),7,seq(9,i=b-k+2..b),seq(17,i=(b+1)..(a+b))]:
    B[2*a+4*b+2*k]:=[seq(5,i=1..b-k),seq(9,i=b-k+1..b),seq(17,i=(b+1)..(a+b))]:
  od:
  for k from 1 to a do
    B[2*a+6*b+2*k-1]:=[seq(9,i=1..b),seq(17,i=b+1..a+b-k),18,seq(19,i=a+b-k+2..a+b)]:
    B[2*a+6*b+2*k]:=[seq(9,i=1..b),seq(17,i=b+1..a+b-k),seq(19,i=a+b-k+1..a+b)]:
  od:
  for k from 1 to b do
    B[4*a+6*b+3*k-2]:=[seq(9,i=1..b-k),11,seq(13,i=b-k+2..b),seq(19,i=b+1..a+b)]:
    B[4*a+6*b+3*k-1]:=[seq(9,i=1..b-k),12,seq(13,i=b-k+2..b),seq(19,i=b+1..a+b)]:
    B[4*a+6*b+3*k]:=[seq(9,i=1..b-k),seq(13,i=b-k+1..b),seq(19,i=b+1..a+b)]:
  od:
  for k from 1 to a do
    B[4*a+9*b+k]:=[seq(13,i=1..b),seq(19,i=(b+1)..(a+b-k)),seq(20,i=a+b-k+1..a+b)]:
  od:
  for k from 1 to b do
    B[5*a+9*b+k]:=[seq(13,i=1..b-k),seq(14,i=b-k+1..b),seq(20,i=b+1..a+b)]:
  od:
  for k from 1 to a do
    B[5*a+10*b+k]:=[seq(14,i=1..b),seq(20,i=(b+1)..(a+b-k)),seq(21,i=a+b-k+1..a+b)]:
  od:
fi:
RETURN(B):

```

end:

The procedure `tableaux_distance` takes as input a Lie Algebra \mathfrak{g} and two tableaux \mathbf{s} and \mathbf{t} . In practice, this procedure is only called when \mathbf{s} and \mathbf{t} are tableaux from the same \mathfrak{g} -semistandard lattice L_λ . The output of the procedure is the shortest number of steps between \mathbf{s} and \mathbf{t} , i.e. the length of any shortest path from \mathbf{s} to \mathbf{t} in the Hasse diagram for L_λ . Recall that if \mathbf{s} is the maximal tableau in L_λ and \mathbf{t} is the minimal tableau, then $\text{tableaux_distance}(\mathfrak{g}, \mathbf{s}, \mathbf{t}) = \text{lattice_length}(\mathfrak{g}, a, b)$.

```

tableaux_distance := proc(g,s,t)
  local i,td:
  if g = 'A2' or g = 'B2' then
    RETURN(convert([seq(abs(s[i]-t[i]),i=1..nops(t))], '+')):
  elif g = 'G2' then
    td:=0:
    for i from 1 to nops(t) do
      if t[i]=1 then
        if member(s[i],{1,2,3,4,5}) then
          td:=td+abs(s[i]-t[i]):
        elif member(s[i],{6}) then
          td:=td+4:
        elif member(s[i],{7,8}) then
          td:=td+5:
        elif member(s[i],{9,10}) then
          td:=td+6:
        else
          td:=td+abs(abs(s[i]-t[i])-3):
        fi:
      elif t[i]=2 then
        if member(s[i],{1,2,3,4,5}) then
          td:=td+abs(s[i]-t[i]):
        elif member(s[i],{6}) then
          td:=td+3:
        elif member(s[i],{7,8}) then
          td:=td+4:
        elif member(s[i],{9,10}) then
          td:=td+5:
        else
          td:=td+abs(abs(s[i]-t[i])-3):
        fi:
      elif t[i]=3 then
        if member(s[i],{1,2,3,4,5}) then
          td:=td+abs(s[i]-t[i]):

```

```

elif member(s[i],{6}) then
    td:=td+2:
elif member(s[i],{7,8}) then
    td:=td+3:
elif member(s[i],{9,10}) then
    td:=td+4:
else
    td:=td+abs(abs(s[i]-t[i])-3):
fi:
elif t[i]=4 then
    if member(s[i],{1,2,3,4,5}) then
        td:=td+abs(s[i]-t[i]):
    elif member(s[i],{6}) then
        td:=td+1:
    elif member(s[i],{7,8}) then
        td:=td+2:
    elif member(s[i],{9,10}) then
        td:=td+3:
    else
        td:=td+abs(abs(s[i]-t[i])-3):
    fi:
elif t[i]=5 then
    if member(s[i],{1,2,3,4,5}) then
        td:=td+abs(s[i]-t[i]):
    elif member(s[i],{7}) then
        td:=td+1:
    elif member(s[i],{8}) then
        td:=td+3:
    elif member(s[i],{6,9,10}) then
        td:=td+2:
    else
        td:=td+abs(abs(s[i]-t[i])-3):
    fi:
elif t[i]=6 then
    if member(s[i],{1,2,3,4}) then
        td:=td+abs(abs(s[i]-t[i])-1):
    elif member(s[i],{5,9,10}) then
        td:=td+2:
    elif member(s[i],{7,8}) then
        td:=td+1:
    elif member(s[i],{6}) then
        td:=td:
    else
        td:=td+abs(abs(s[i]-t[i])-2):
    fi:
elif t[i]=7 then
    if member(s[i],{1,2,3,4}) then
        td:=td+abs(abs(s[i]-t[i])-1):
    elif member(s[i],{5,6,9,10}) then
        td:=td+1:
    elif member(s[i],{8}) then

```

```

        td:=td+2:
    elif member(s[i],{7}) then
        td:=td:
    else
        td:=td+abs(abs(s[i]-t[i])-2):
    fi:
elif t[i]=8 then
    if member(s[i],{1,2,3,4}) then
        td:=td+abs(abs(s[i]-t[i])-2):
    elif member(s[i],{5,9}) then
        td:=td+3:
    elif member(s[i],{6,7,10}) then
        td:=td+1:
    elif member(s[i],{8}) then
        td:=td:
    else
        td:=td+abs(abs(s[i]-t[i])-1):
    fi:
elif t[i]=9 then
    if member(s[i],{1,2,3,4,5}) then
        td:=td+abs(abs(s[i]-t[i])-2):
    elif member(s[i],{6,10}) then
        td:=td+2:
    elif member(s[i],{7}) then
        td:=td+1:
    elif member(s[i],{8}) then
        td:=td+3:
    elif member(s[i],{9}) then
        td:=td:
    else
        td:=td+abs(abs(s[i]-t[i])-1):
    fi:
elif t[i]=10 then
    if member(s[i],{1,2,3,4}) then
        td:=td+abs(abs(s[i]-t[i])-3):
    elif member(s[i],{5,6,9}) then
        td:=td+2:
    elif member(s[i],{7,8}) then
        td:=td+1:
    else
        td:=td+abs(s[i]-t[i]):
    fi:
elif t[i]=11 then
    if member(s[i],{1,2,3,4}) then
        td:=td+abs(abs(s[i]-t[i])-3):
    elif member(s[i],{5,6}) then
        td:=td+3:
    elif member(s[i],{7,8}) then
        td:=td+2:
    elif member(s[i],{9}) then
        td:=td+1:

```

```

else
    td:=td+abs(s[i]-t[i]):
fi:
elif t[i]=12 then
    if member(s[i],{1,2,3,4}) then
        td:=td+abs(abs(s[i]-t[i])-3):
    elif member(s[i],{5,6}) then
        td:=td+4:
    elif member(s[i],{7,8}) then
        td:=td+3:
    elif member(s[i],{9,10}) then
        td:=td+2:
    else
        td:=td+abs(s[i]-t[i]):
    fi:
elif t[i]=13 then
    if member(s[i],{1,2,3,4}) then
        td:=td+abs(abs(s[i]-t[i])-3):
    elif member(s[i],{5,6}) then
        td:=td+5:
    elif member(s[i],{7,8}) then
        td:=td+4:
    elif member(s[i],{9,10}) then
        td:=td+3:
    else
        td:=td+abs(s[i]-t[i]):
    fi:
elif t[i]=14 then
    if member(s[i],{1,2,3,4}) then
        td:=td+abs(abs(s[i]-t[i])-3):
    elif member(s[i],{5,6}) then
        td:=td+6:
    elif member(s[i],{7,8}) then
        td:=td+5:
    elif member(s[i],{9,10}) then
        td:=td+4:
    else
        td:=td+abs(s[i]-t[i]):
    fi:
else
    td:=td+abs(s[i]-t[i]):
fi:
od:
RETURN(td):
fi:
end:

```


The procedure **rhl_precedes** takes as input two tableaux **s** and **t**. In practice, this procedure is only called when **s** and **t** are at the same level of L_λ and have the same distance from the boundary. The output of the procedure is the Boolean value *true* if the first tableau precedes the second tableau relative to the righthand lexicographic rule. Otherwise, the procedure returns the Boolean value *false*.

```
rhl_precedes := proc(s,t)
  local i:
  for i from nops(t) to 1 by -1 do
    if s[i] < t[i] then RETURN( evalb(1=1) )
    elif s[i] > t[i] then RETURN( evalb(1=0) )
    fi:
  od:
  ## If it makes it this far, then s=t, so we'll return true. ##
  RETURN( evalb(1=1) ):
end:
```

The procedure **total_order** takes as input a Lie Algebra **g** and two non-negative integers *a* and *b* as described above. The output of the procedure is an array which has **lattice_dimension(g, a, b)** entries. The i^{th} entry of this array is a pair consisting of the i^{th} ordered tableau as determined by Algorithm 5.3 along with the distance of the i^{th} tableau from the boundary.

```
total_order := proc(g,a,b)
  local T,B,length,dimension,S,level_sum_so_far,k,temptable,t,d,end_while_flag,j,i:
  T:=tableaux(g,a,b):
  B:=boundary(g,a,b):
  length:=lattice_length(g,a,b):
  dimension:=lattice_dimension(g,a,b):
  S:=array(1..dimension):
  S[1]:=B[0],0,0]:
  level_sum_so_far := 1:
  for k from 1 to length do
    temptable[1]:=B[k],0,k]:
    for t in T[k] do
      d:=tableaux_distance(g,t,B[k]):
      if d > 0 then
        end_while_flag:=evalb(1=0):
        for j from 1 while not(end_while_flag) do
```

```

    if d < temptable[j][2] then
      for i from nops(op(op(temptable))) to j by -1 do
        temptable[i+1]:=temptable[i]:
      od:
      temptable[j]:=[t,d,k]:
      end_while_flag:=evalb(1=1):
    elif d = temptable[j][2] and rhl_precedes(t,temptable[j][1]) then
      for i from nops(op(op(temptable))) to j by -1 do
        temptable[i+1]:=temptable[i]:
      od:
      temptable[j]:=[t,d,k]:
      end_while_flag:=evalb(1=1):
    elif j=nops(op(op(temptable))) then
      temptable[j+1]:=[t,d,k]:
      end_while_flag:=evalb(1=1):
    fi:
  od:
  fi:
od:
for j from 1 to nops(T[k]) do
  S[level_sum_so_far + j] := temptable[j]:
od:
level_sum_so_far := level_sum_so_far + nops(T[k]):
temptable:='temptable':
od:
RETURN(S):
end:

```

The procedure `convert_tableaux` takes as input a Lie Algebra \mathfrak{g} and one tableau \mathbf{t} . The output of the procedure is an array of 6, 9, or 21 entries depending on \mathfrak{g} . The i^{th} entry of the array is the number of occurrences of a column of type i (see “Convention Adopted for Code” in Appendix B) in the tableau \mathbf{t} . This procedure is used inside the procedure `edges` to compute the coordinates of the weight vector $wt(\mathbf{t})$.

```

convert_tableaux := proc(g,t)
  local m,i,j,tsize:
  if g = 'A2' then
    tsize := 6:
  elif g = 'B2' then
    tsize := 9:
  elif g = 'G2' then
    tsize := 21:
  fi:
  m:=array(1..tsize):
  for i from 1 to tsize do

```

```

    m[i]:=0:
  od:
  for i from 1 to nops(t) do
    for j from 1 to tsize do
      if t[i] = j then
        m[j]:=m[j]+1:
      fi:
    od:
  od:
  RETURN(m);
end:

```

The procedure **bdry_index** takes as input an ordered array S . In practice, the input set S is the output of the **total_order** procedure. The output of **bdry_index** is an array which has $\mathbf{lattice_length}(\mathfrak{g}, a, b) + 1$ entries. The i^{th} entry of the array is the index of the boundary tableau \mathbf{b}_i relative to the total ordering of the tableaux of L_λ . That is, the i^{th} entry of the output array is the position of the tableau \mathbf{b}_i in the sequence S .

```

bdry_index := proc(S)
  local dimension,B,level,j:
  dimension:=nops(convert(op(S),set)):
  B:=array(0..S[dimension][3]):
  level:=0:
  for j from 1 to dimension do
    if S[j][2] = 0 then
      B[level] := j:
      level:=level+1:
    fi:
  od:
  RETURN(B):
end:

```

The procedure **find_index** takes as input a Lie Algebra \mathfrak{g} , an ordered S , the array B of the boundary indices, and a tableau \mathbf{t} . In practice, the array S is the output of the **total_order** procedure, B is the output of the **bdry_index** procedure, and \mathbf{t} is a tableau with the same shape as the tableaux in S . The output of the procedure is the index of tableau \mathbf{t} relative to the total ordering of the tableaux in L_λ . That is, the procedure returns the position the tableau \mathbf{t} appears in the sequence S .

```

find_index := proc(g,S,B,t)
  local level,dist,j;
  level:=tableaux_distance(g,t,S[1][1]):
  dist:=tableaux_distance(g,t,S[B[level]][1]):
  for j from B[level] do
    if S[j][2] = dist and S[j][1] = t then
      RETURN(j):
    fi:
  od:
end:

```

The procedure **rightmost_decrease** takes as input a Lie Algebra \mathfrak{g} and one tableau \mathbf{t} . The output of the procedure is the tableau that is the same as \mathbf{t} in every column except for the right most “decreasable” column \mathbf{t}_p . This column is “decreased” once along the edge $\mathbf{t}_p \longrightarrow \mathbf{v}$ where \mathbf{v} is the leftmost column above \mathbf{t}_p in L_{\square} or L_{\square} .

```

rightmost_decrease := proc(g,t)
  local r,i,already_done,x;
  if nargs = 3 and args[3]<=nops(t) then x:=args[3] else x:=nops(t) fi:
  r:=array(1..nops(t)):
  already_done:=evalb(0=1):
  if x<nops(t) then
    for i from nops(t) to x by -1 do
      r[i]:=t[i];
    od:
  fi:
  if g = 'A2' then
    for i from x to 2 by -1 do
      if not(t[i-1]=t[i]) and not(t[i-1]=3) and not(t[i]=4) and not(already_done) then
        r[i]:=t[i]-1:
        already_done:=evalb(1=1):
      elif not(t[i-1]=t[i]) and t[i-1]=3 and not(t[i]=5) and not(already_done) then
        r[i]:=t[i]-1:
        already_done:=evalb(1=1):
      else
        r[i]:=t[i]:
      fi:
    od:
    if already_done then
      r[1]:=t[1]:
    else
      if t[1]-1 = 0 then
        r[1]:=t[1]:
      else
        r[1]:=t[1]-1:
      fi:
    fi:
  fi:
end:

```

```

fi:
elif g = 'B2' then
  for i from x to 2 by -1 do
    if not(t[i-1]=t[i]) and t[i]=9 and not(already_done) then
      r[i]:=t[i]-1:
      already_done:=evalb(1=1):
    elif not(t[i-1]=t[i]) and t[i]=8 and not(member(t[i-1],{5})) and
      not(already_done) then
      r[i]:=t[i]-1:
      already_done:=evalb(1=1):
    elif not(t[i-1]=t[i]) and t[i]=7 and not(member(t[i-1],{5,4,3})) and
      not(already_done) then
      r[i]:=t[i]-1:
      already_done:=evalb(1=1):
    elif not(t[i-1]=t[i]) and t[i]=6 and not(already_done) then
      r[i]:=t[i]:
      already_done:=evalb(0=1):
    elif not(t[i-1]=t[i]) and t[i]=5 and not(already_done) then
      r[i]:=t[i]-1:
      already_done:=evalb(1=1):
    elif not(t[i-1]=t[i]) and t[i]=4 and not(t[i-1]=3) and not(already_done) then
      r[i]:=t[i]-1:
      already_done:=evalb(1=1):
    elif not(t[i-1]=t[i]) and t[i]=3 and not(already_done) then
      r[i]:=t[i]-1:
      already_done:=evalb(1=1):
    elif not(t[i-1]=t[i]) and t[i]=2 and not(already_done) then
      r[i]:=t[i]-1:
      already_done:=evalb(1=1):
    else
      r[i]:=t[i]:
    fi:
  od:
  if already_done then
    r[1]:=t[1]:
  else
    if t[1]-1 = 0 then
      r[1]:=t[1]:
    else
      r[1]:=t[1]-1:
    fi:
  fi:
elif g = 'G2' then
  for i from x to 2 by -1 do
    if not(t[i-1]=t[i]) and t[i]=21 and not(already_done) then
      r[i]:=20:
      already_done:=evalb(1=1):
    elif not(t[i-1]=t[i]) and t[i]=20 and not(member(t[i-1],{14})) and
      not(already_done) then
      r[i]:=19:
      already_done:=evalb(1=1):

```

```

elif not(t[i-1]=t[i]) and t[i]=19 and not(member(t[i-1],{8,10,11,12,13,14,18})) and
not(already_done) then
  r[i]:=18:
  already_done:=evalb(1=1):
elif not(t[i-1]=t[i]) and t[i]=18 and not(member(t[i-1],{8,10,11,12,13,14})) and
not(already_done) then
  r[i]:=17:
  already_done:=evalb(1=1):
elif not(t[i-1]=t[i]) and t[i]=17 and not(member(t[i-1],{6,7,8,9,10,11,12,13,14})) and
not(already_done) then
  r[i]:=16:
  already_done:=evalb(1=1):
elif not(t[i-1]=t[i]) and t[i]=16 and not(member(t[i-1],{3,4,5,6,7,8,9,10,11,12,13,14}))
and not(already_done) then
  r[i]:=15:
  already_done:=evalb(1=1):
elif not(t[i-1]=t[i]) and t[i]=15 and not(already_done) then
  r[i]:=t[i]:
  already_done:=evalb(0=1):
elif not(t[i-1]=t[i]) and t[i]=14 and not(already_done) then
  r[i]:=13:
  already_done:=evalb(1=1):
elif not(t[i-1]=t[i]) and t[i]=13 and not(member(t[i-1],{8,10,11,12})) and
not(already_done) then
  r[i]:=12:
  already_done:=evalb(1=1):
elif not(t[i-1]=t[i]) and t[i]=12 and not(member(t[i-1],{8,10,11})) and
not(already_done) then
  r[i]:=11:
  already_done:=evalb(1=1):
elif not(t[i-1]=t[i]) and t[i]=11 and not(already_done) then
  r[i]:=9:
  already_done:=evalb(1=1):
elif not(t[i-1]=t[i]) and t[i]=10 and not(member(t[i-1],{6,7,8})) and
not(already_done) then
  r[i]:=7:
  already_done:=evalb(1=1):
elif not(t[i-1]=t[i]) and t[i]=9 and not(member(t[i-1],{6,7,8})) and
not(already_done) then
  r[i]:=7:
  already_done:=evalb(1=1):
elif not(t[i-1]=t[i]) and t[i]=8 and not(member(t[i-1],{3,4,6})) and
not(already_done) then
  r[i]:=6:
  already_done:=evalb(1=1):
elif not(t[i-1]=t[i]) and t[i]=7 and not(already_done) then
  r[i]:=5:
  already_done:=evalb(1=1):
elif not(t[i-1]=t[i]) and t[i]=6 and not(member(t[i-1],{3,4})) and
not(already_done) then
  r[i]:=4:

```

```

        already_done:=evalb(1=1):
    elif not(t[i-1]=t[i]) and t[i]=5 and not(member(t[i-1],{3,4})) and
    not(already_done) then
        r[i]:=4:
        already_done:=evalb(1=1):
    elif not(t[i-1]=t[i]) and t[i]=4 and not(member(t[i-1],{3})) and
    not(already_done) then
        r[i]:=3:
        already_done:=evalb(1=1):
    elif not(t[i-1]=t[i]) and t[i]=3 and not(already_done) then
        r[i]:=2:
        already_done:=evalb(1=1):
    elif not(t[i-1]=t[i]) and t[i]=2 and not(already_done) then
        r[i]:=1:
        already_done:=evalb(1=1):
    else
        r[i]:=t[i]:
    fi:
od:
if already_done then
    r[1]:=t[1]:
else
    if t[1]-1 = 0 then
        r[1]:=t[1]:
    elif member(t[1],{6,7,8,9,11}) then
        r[1]:=t[1]-2:
    elif t[1]=10 then
        r[1]:=7:
    else
        r[1]:=t[1]-1:
    fi:
fi:
RETURN(convert(r,list)):
end:

```

The procedure **lub** takes as input a Lie Algebra \mathfrak{g} and two tableaux \mathbf{s} and \mathbf{t} . Whenever we call this procedure, the tableaux \mathbf{s} and \mathbf{t} are both \mathfrak{g} -tableaux of the same shape. The output of the procedure is the least upper bound in the lattice L_λ of the two tableaux, i.e. $\mathbf{s} \vee \mathbf{t}$.

```

lub := proc(g,s,t)
    local r,i:
    r:=array(1..nops(s)):
    if g = 'A2' or g = 'B2' then
        for i from 1 to nops(s) do

```

```

    r[i]:=min(s[i],t[i]):
  od:
elif g = 'G2' then
  for i from 1 to nops(s) do
    if (s[i]=5 and t[i]=6) or (s[i]=6 and t[i]=5) then
      r[i]:=4:
    elif (s[i]=7 and t[i]=8) or (s[i]=8 and t[i]=7) then
      r[i]:=6:
    elif (s[i]=9 and t[i]=10) or (s[i]=10 and t[i]=9) then
      r[i]:=7:
    elif (s[i]=5 and t[i]=8) or (s[i]=8 and t[i]=5) then
      r[i]:=4:
    elif (s[i]=8 and t[i]=9) or (s[i]=9 and t[i]=8) then
      r[i]:=6:
    else
      r[i]:=min(s[i],t[i]):
    fi:
  od:
fi:
RETURN(convert(r,list)):
end:

```

The procedure **edge_set** takes as input a Lie Algebra \mathfrak{g} , an array of ordered tableaux S , and two non-negative integers a and b as described above. The procedure assumes that $S := \mathbf{total_order}(\mathfrak{g}, a, b)$ has been previously called. The output of the procedure varies according to certain additional optional arguments. The procedure iteratively determines all the directed edges (i.e. covering relations) between tableaux as well as the associated color (red, blue) of each edge. If no optional arguments are given as input, the output will be a set containing all the directed edges of the lattice and a table containing all the red edges and all the blue edges. If the optional argument **coeff** is included in the procedure call, edge coefficients $\pi_{s,t}$ are checked to verify that each one holds for the coefficients computed. If any diamond or crossing relations fail, the procedure will display an error message detailing where the failure occurs. If the optional argument **max** is given, then the output will be the maximal edge coefficient. If the optional argument **min** is given, then the output will be the minimal edge coefficient.


```

edge_set := proc(g,S,a,b)
local edges, pmin, pmax, edge_min, edge_max, edge_color, total_level, dimension, B, m,
  nlevel, i, j, k, l, temp, t, sum_above_vertex, sum_below_vertex, ndx_t, u, ndx_u,
  ndx_d, crossing_flag, diamond_flag:
edge_min[1]:=infinity:
edge_max[1]:=-infinity:
edge_min[2]:={}:
edge_max[2]:={}:
edges:=table():
edge_color:=table([(blue)={},(red)={}]):
dimension:=lattice_dimension(g,a,b):
total_level:=S[dimension][3]:
B:=bdry_index(S):
m:=array(1..dimension,1..2):
for i from 1 to dimension do
  t:=convert_tableaux(g,S[i][1]);
  if g = 'A2' then
    m[i,1]:=t[2]+t[4]-t[3]-t[5]:
    m[i,2]:=t[1]+t[5]-t[2]-t[6]:
  elif g = 'B2' then
    m[i,1]:=2*t[2]-2*t[4]+t[6]-t[7]+t[8]-t[9]:
    m[i,2]:=t[1]-t[2]+t[4]-t[5]+t[7]-t[8]:
  elif g = 'G2' then
    m[i,1]:=3*t[2]+t[3]-t[4]-3*t[5]+2*t[6]+3*t[9]-2*t[10]+t[11]-t[12]-
      3*t[13]+t[15]-t[16]+2*t[17]-2*t[19]+t[20]-t[21]:
    m[i,2]:=t[1]-t[2]+t[4]+2*t[5]-t[6]-2*t[9]+t[10]-t[11]+t[13]-t[14]+
      t[16]-t[17]+t[19]-t[20]:
  fi:
od:
sum_above_vertex:=array(1..dimension,1..2):
sum_below_vertex:=array(1..dimension,1..2):
for i from 1 to dimension do
  sum_above_vertex[i,1]:=0:
  sum_above_vertex[i,2]:=0:
  sum_below_vertex[i,1]:=0:
  sum_below_vertex[i,2]:=0:
od:
nlevel:=array(0..total_level):
for i from 0 to total_level do
  nlevel[i]:=0:
od:
for i from 1 to dimension do
  nlevel[S[i][3]]:=nlevel[S[i][3]]+1:
od:
for i from 0 to total_level-1 do
  for j from 0 to nlevel[i]-1 do
    for k from 0 to nlevel[i+1]-1 do
      crossing_flag[1]:=evalb(0=1):
      crossing_flag[2]:=evalb(0=1):
      diamond_flag:=evalb(0=1):
      if (S[B[i+1]+k][2] = S[B[i]+j][2] - 2)

```

```

or (S[B[i+1]+k][2] = S[B[i]+j][2])
or (S[B[i+1]+k][2] = S[B[i]+j][2] + 2) then
  if tableaux_distance(g,S[B[i+1]+k][1],S[B[i]+j][1]) = 1 then
    temp:=(S[B[i+1]+k][1]-S[B[i]+j][1]):
    for l from 1 to nops(temp) do
      if not(temp[l] = 0) then
        if g = 'A2' then
          if member(S[B[i]+j][1][1],{1,5}) then
            edge_color[blue]:=edge_color[blue] union {[B[i+1]+k,B[i]+j]}:
          elif member(S[B[i]+j][1][1],{2,4}) then
            edge_color[red]:=edge_color[red] union {[B[i+1]+k,B[i]+j]}:
          fi:
        elif g = 'B2' then
          if member(S[B[i]+j][1][1],{1,4,7}) then
            edge_color[blue]:=edge_color[blue] union {[B[i+1]+k,B[i]+j]}:
          elif member(S[B[i]+j][1][1],{2,3,6,8}) then
            edge_color[red]:=edge_color[red] union {[B[i+1]+k,B[i]+j]}:
          fi:
        elif g = 'G2' then
          if member(S[B[i]+j][1][1],{1,5,10,13,16,19}) then
            edge_color[blue]:=edge_color[blue] union {[B[i+1]+k,B[i]+j]}:
          elif member(S[B[i]+j][1][1],{2,3,6,8,9,11,12,15,17,18,20}) then
            edge_color[red]:=edge_color[red] union {[B[i+1]+k,B[i]+j]}:
          elif member(S[B[i]+j][1][1],{4}) then
            if member(S[B[i+1]+k][1][1],{5}) then
              edge_color[red]:=edge_color[red] union {[B[i+1]+k,B[i]+j]}:
            elif member(S[B[i+1]+k][1][1],{6}) then
              edge_color[blue]:=edge_color[blue] union {[B[i+1]+k,B[i]+j]}:
            fi:
          elif member(S[B[i]+j][1][1],{7}) then
            if member(S[B[i+1]+k][1][1],{10}) then
              edge_color[red]:=edge_color[red] union {[B[i+1]+k,B[i]+j]}:
            elif member(S[B[i+1]+k][1][1],{9}) then
              edge_color[blue]:=edge_color[blue] union {[B[i+1]+k,B[i]+j]}:
            fi:
          fi:
        fi:
      if member('coeff',[args]) then
        t:=rightmost_decrease(g,S[B[i+1]+k][1]):
        ndx_t:=find_index(g,S,B,t);
        if not(evalb(S[ndx_t][1]=S[B[i]+j][1])) then
          u:=lub(g,S[ndx_t][1],S[B[i]+j][1]):
          ndx_u:=find_index(g,S,B,u):
          edges[B[i+1]+k,B[i]+j]:=
            (edges[ndx_t,ndx_u]*edges[B[i]+j,ndx_u])/edges[B[i+1]+k,ndx_t]:
          diamond_flag:=evalb(1=1):
        else
          if member([B[i+1]+k,B[i]+j],edge_color[red]) then
            edges[B[i+1]+k,B[i]+j]:=
              m[B[i]+j,1]+sum_above_vertex[B[i]+j,1]-sum_below_vertex[B[i]+j,1]:
            crossing_flag[1]:=evalb(1=1):

```

```

elif member([B[i+1]+k,B[i]+j],edge_color[blue]) then
  edges[B[i+1]+k,B[i]+j]:=
    m[B[i]+j,2]+sum_above_vertex[B[i]+j,2]-sum_below_vertex[B[i]+j,2]:
  crossing_flag[2]:=evalb(1=1):
fi:
fi:
if member([B[i+1]+k,B[i]+j],edge_color[red]) then
  sum_above_vertex[B[i+1]+k,1]:=
    sum_above_vertex[B[i+1]+k,1]+edges[B[i+1]+k,B[i]+j]:
  sum_below_vertex[B[i]+j,1]:=
    sum_below_vertex[B[i]+j,1]+edges[B[i+1]+k,B[i]+j]:
elif member([B[i+1]+k,B[i]+j],edge_color[blue]) then
  sum_above_vertex[B[i+1]+k,2]:=
    sum_above_vertex[B[i+1]+k,2]+edges[B[i+1]+k,B[i]+j]:
  sum_below_vertex[B[i]+j,2]:=
    sum_below_vertex[B[i]+j,2]+edges[B[i+1]+k,B[i]+j]:
fi:
if diamond_flag then
  for ndx_d from ndx_t+1 to B[i]+j-1 do
    if tableaux_distance(g,S[ndx_d][1],S[B[i+1]+k][1]) = 1 then
      u:=lub(g,S[ndx_d][1],S[B[i]+j][1]):
      ndx_u:=find_index(g,S,B,u):
      if not(edges[ndx_d,ndx_u]*edges[B[i]+j,ndx_u]=
        edges[B[i+1]+k,ndx_d]*edges[B[i+1]+k,B[i]+j]) then
        print('Diamond Condition fails for a='.a.'
          and b='.b.' at edge '.(B[i]+j).->'.(B[i+1]+k)):
        RETURN(0,0);
      fi:
    fi:
  od;
fi:
if edge_min[1]>=edges[B[i+1]+k,B[i]+j] then
  if edge_min[1]=edges[B[i+1]+k,B[i]+j] then
    edge_min[2]:=edge_min[2] union {[S[B[i+1]+k][1],S[B[i]+j][1]]}:
  else
    edge_min[1]:=edges[B[i+1]+k,B[i]+j]:
    edge_min[2]:={S[B[i+1]+k][1],S[B[i]+j][1]}:
  fi:
fi:
if edge_max[1]<=edges[B[i+1]+k,B[i]+j] then
  if edge_max[1] = edges[B[i+1]+k,B[i]+j] then
    edge_max[2]:=edge_max[2] union {[S[B[i+1]+k][1],S[B[i]+j][1]]}:
  else
    edge_max[1]:=edges[B[i+1]+k,B[i]+j]:
    edge_max[2]:={S[B[i+1]+k][1],S[B[i]+j][1]}:
  fi:
fi:
fi:
fi:
od:
fi:

```

```

    fi;
od:
if member('coeff',[args]) then
  for l from 1 to 2 do
    if not(crossing_flag[l]) then
      if not(sum_below_vertex[B[i]+j,l]-sum_above_vertex[B[i]+j,l]=m[B[i]+j,l]) then
        print('Crossing Condition fails for a=',a,
          and b=',b,' at edge ',B[i+1]+k,', ',B[i]+j);
        RETURN(0,0);
      fi;
    fi;
  od;
od:
fi:
od:
od:
if member('min',[args]) and member('max',[args]) then
  lprint('min',a,b,a+b,edge_min[1],nops(edge_min[2]),edge_min[2]);
  lprint('max',a,b,a+b,edge_max[1],nops(edge_max[2]),edge_max[2]);
elif member('max',[args]) then
  lprint('max',a,b,a+b,edge_max[1],nops(edge_max[2]),edge_max[2]);
elif member('min',[args]) then
  lprint('min',a,b,a+b,edge_min[1],nops(edge_min[2]),edge_min[2]);
else
  RETURN(edges,edge_color);
fi;
end:

```

APPENDIX B

DATA FOR THE RANK TWO SIMPLE LIE ALGEBRAS

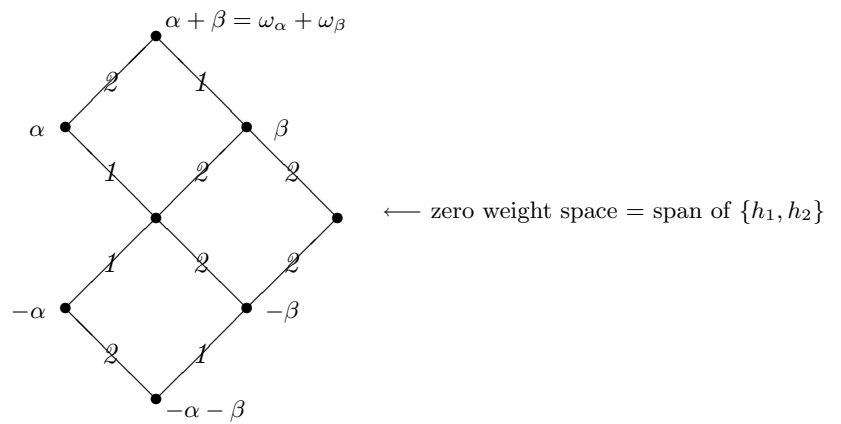
Data for A_2

Dynkin diagram: $\begin{array}{cc} \bullet & \bullet \\ \alpha & \beta \\ \omega_\alpha & \omega_\beta \end{array}$ Cartan Matrix: $\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$

Fundamental weights: w_α, w_β Simple roots: $\alpha = [2, -1] = 2\omega_\alpha - \omega_\beta$

Dimension of A_2 : 8 $\beta = [-1, 2] = -\omega_\alpha + 2\omega_\beta$

“Adjoint” representation: $\lambda = \begin{array}{|c|c|} \hline & \\ \hline \hline & \\ \hline \end{array} = \omega_\alpha + \omega_\beta$



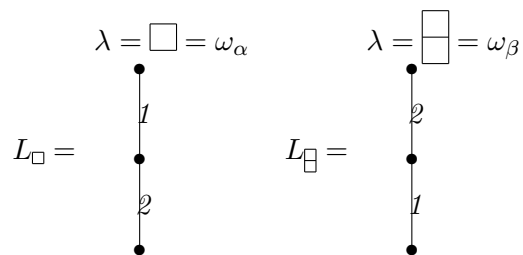
Positive roots: $\alpha, \beta, \alpha + \beta$

Negative roots: $-\alpha, -\beta, -\alpha - \beta$

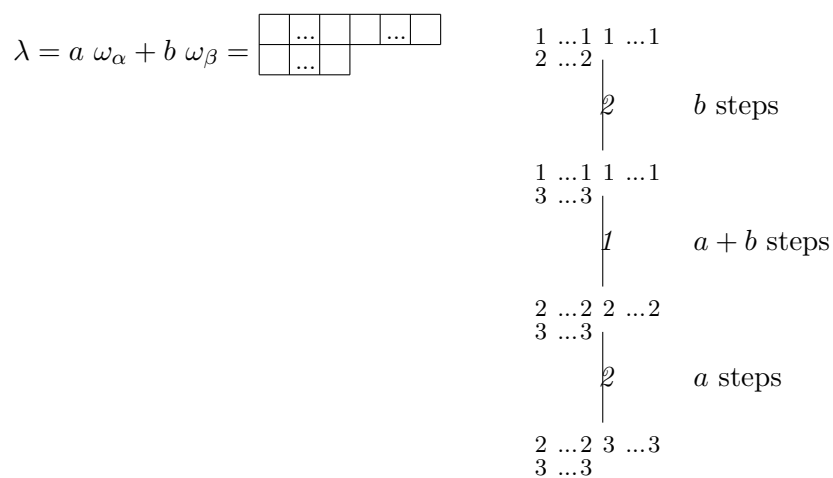
Dimension formula for irreducible representation with highest weight $\lambda = a \omega_\alpha + b \omega_\beta$

$$\dim(\lambda) = \frac{1}{2}(a+1)(b+1)(a+b+2)$$

Fundamental representations:



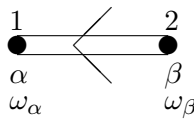
Boundary for semistandard lattice L_λ :



Length of L_λ : $2a + 2b$

Convention Adopted for Code	
Tableau \dots	is denoted by \dots
$\begin{array}{ c } \hline 1 \\ \hline 2 \\ \hline \end{array}$	1
$\begin{array}{ c } \hline 1 \\ \hline 3 \\ \hline \end{array}$	2
$\begin{array}{ c } \hline 2 \\ \hline 3 \\ \hline \end{array}$	3
$\begin{array}{ c } \hline 1 \\ \hline \end{array}$	4
$\begin{array}{ c } \hline 2 \\ \hline \end{array}$	5
$\begin{array}{ c } \hline 3 \\ \hline \end{array}$	6

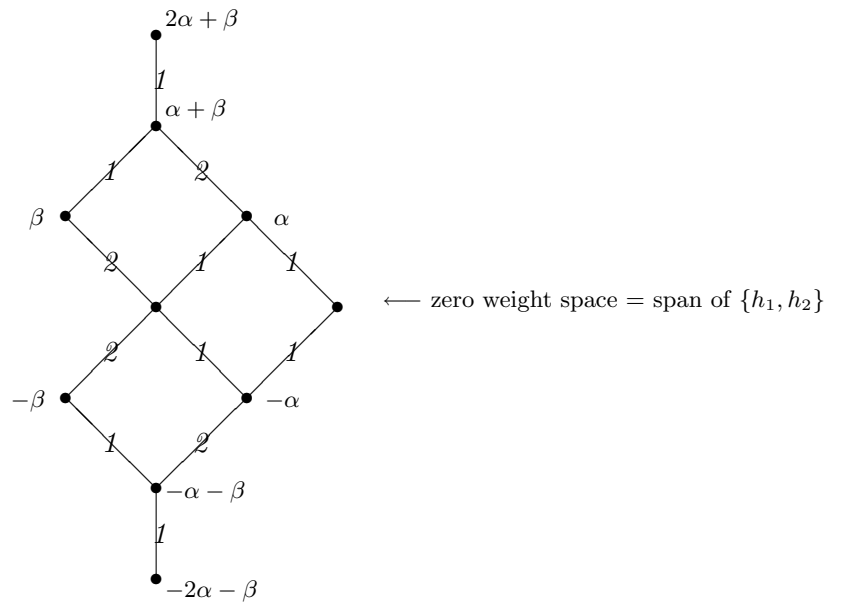
Data for B_2

Dynkin diagram:  Cartan Matrix:
$$\begin{pmatrix} 2 & -1 \\ -2 & 2 \end{pmatrix}$$

Fundamental weights: w_α, w_β Simple roots: $\alpha = [2, -1] = 2w_\alpha - w_\beta$
 $\beta = [-2, 2] = -2w_\alpha + 2w_\beta$

Dimension of B_2 : 10

“Adjoint” representation: $\lambda = \square\square = 2w_\alpha$



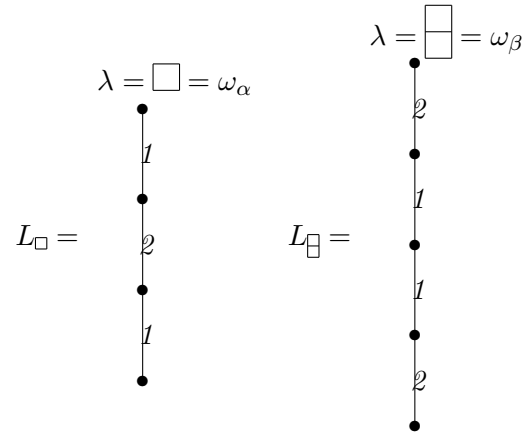
Positive roots: $\alpha, \beta, \alpha + \beta, 2\alpha + \beta$

Negative roots: $-\alpha, -\beta, -\alpha - \beta, -2\alpha - \beta$

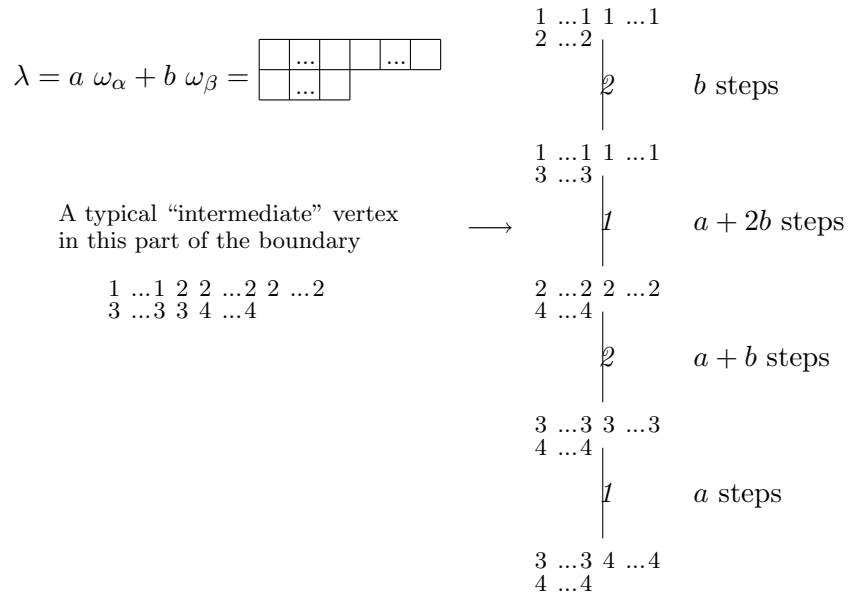
Dimension formula for irreducible representation with highest weight $\lambda = a \omega_\alpha + b \omega_\beta$

$$\dim(\lambda) = \frac{1}{3!}(a+1)(b+1)(a+b+2)(a+2b+3)$$

Fundamental representations:



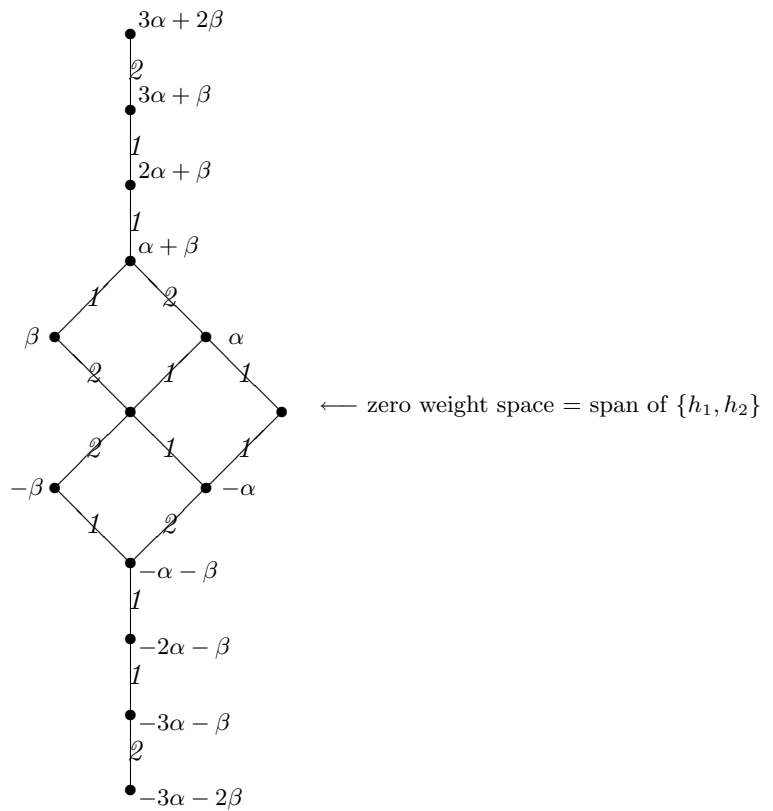
Boundary for semistandard lattice L_{λ} :



Length of L_{λ} : $3a + 4b$

Data for G_2

Dynkin diagram:		Cartan Matrix:	$\begin{pmatrix} 2 & -1 \\ -3 & 2 \end{pmatrix}$
Fundamental weights:	w_α, w_β	Simple roots:	$\alpha = [2, -1] = 2w_\alpha - w_\beta$
Dimension of G_2 :	14		$\beta = [-3, 2] = -3w_\alpha + 2w_\beta$
“Adjoint” representation:	$\lambda = \begin{matrix} \square \\ \square \end{matrix} = \omega_\beta$		



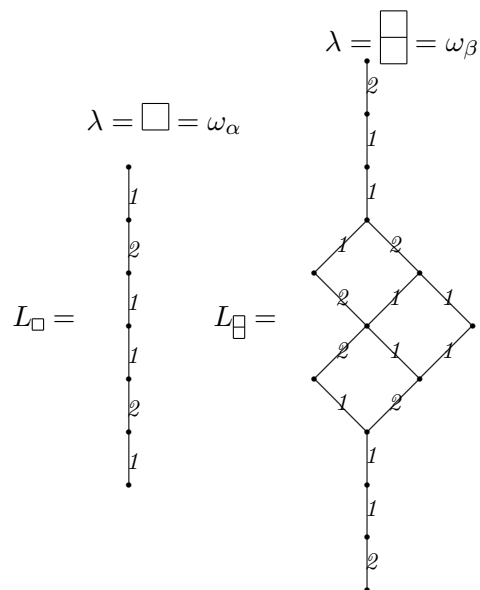
Positive roots: $\alpha, \beta, \alpha + \beta, 2\alpha + \beta, 3\alpha + \beta, 3\alpha + 2\beta$

Negative roots: $-\alpha, -\beta, -\alpha - \beta, -2\alpha - \beta, -3\alpha - \beta, -3\alpha - 2\beta$

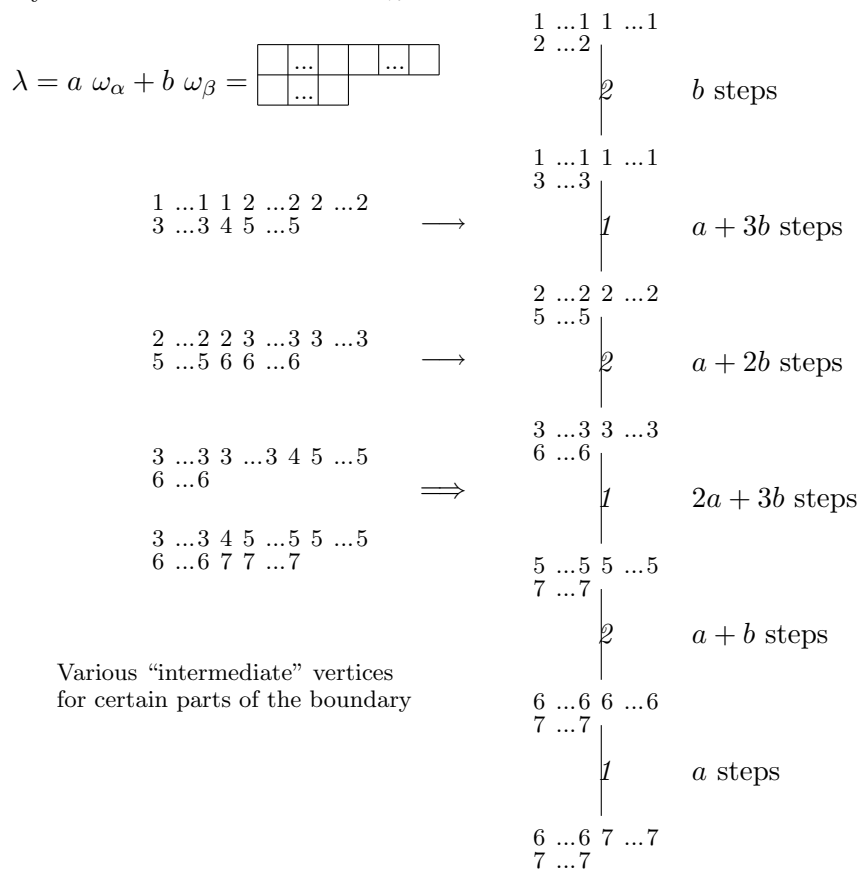
Dimension formula for irreducible representation with highest weight $\lambda = a \omega_\alpha + b \omega_\beta$

$$\dim(\lambda) = \frac{1}{5!} (a + 1)(b + 1)(a + b + 2)(a + 2b + 3)(a + 3b + 4)(2a + 3b + 5)$$

Fundamental representations:



Boundary for semistandard lattice L_λ :



Length of L_λ : $6a + 10b$

Convention Adopted for Code	
Tableau ...	is denoted by ...
$\begin{array}{ c } \hline 1 \\ \hline 2 \\ \hline \end{array}$	1
$\begin{array}{ c } \hline 1 \\ \hline 3 \\ \hline \end{array}$	2
$\begin{array}{ c } \hline 1 \\ \hline 4 \\ \hline \end{array}$	3
$\begin{array}{ c } \hline 1 \\ \hline 5 \\ \hline \end{array}$	4
$\begin{array}{ c } \hline 2 \\ \hline 5 \\ \hline \end{array}$	5
$\begin{array}{ c } \hline 1 \\ \hline 6 \\ \hline \end{array}$	6
$\begin{array}{ c } \hline 2 \\ \hline 6 \\ \hline \end{array}$	7
$\begin{array}{ c } \hline 1 \\ \hline 7 \\ \hline \end{array}$	8
$\begin{array}{ c } \hline 3 \\ \hline 6 \\ \hline \end{array}$	9
$\begin{array}{ c } \hline 2 \\ \hline 7 \\ \hline \end{array}$	10
$\begin{array}{ c } \hline 3 \\ \hline 7 \\ \hline \end{array}$	11
$\begin{array}{ c } \hline 4 \\ \hline 7 \\ \hline \end{array}$	12
$\begin{array}{ c } \hline 5 \\ \hline 7 \\ \hline \end{array}$	13
$\begin{array}{ c } \hline 6 \\ \hline 7 \\ \hline \end{array}$	14
$\begin{array}{ c } \hline 1 \\ \hline \end{array}$	15
$\begin{array}{ c } \hline 2 \\ \hline \end{array}$	16
$\begin{array}{ c } \hline 3 \\ \hline \end{array}$	17
$\begin{array}{ c } \hline 4 \\ \hline \end{array}$	18
$\begin{array}{ c } \hline 5 \\ \hline \end{array}$	19
$\begin{array}{ c } \hline 6 \\ \hline \end{array}$	20
$\begin{array}{ c } \hline 7 \\ \hline \end{array}$	21

BIBLIOGRAPHY

- [Don] R. G. Donnelly, “Extremal properties of bases for representations of semisimple Lie algebras,” to appear in *J. Algebraic Comb.*
- [DLP1] R. G. Donnelly, S. J. Lewis, and R. Pervine, “Constructions of representations of $\mathfrak{o}(2n + 1, \mathbb{C})$ that imply Molev and Reiner-Stanton lattices are strongly Sperner,” *Discrete Math.* **263** (2003), 61-79.
- [DLP2] R. G. Donnelly, S. J. Lewis, and R. Pervine, personal communication, July 2003.
- [DW] R. G. Donnelly and N. J. Wildberger, personal communication, July 2003.
- [Eve] J. Eveland, “Using Pictures to Classify Solutions to Certain Systems of Matrix Equations,” Undergraduate honors thesis, Murray State University, 2001.
- [GT] I. M. Gelfand and M. L. Tsetlin, “Finite-dimensional representations of the group of unimodular matrices,” *Dokl. Akad. Nauk. USSR* **71** (1950), 825-828. [Russian]
- [How] R. Howe, “Very Basic Lie Theory,” *Amer. Math. Monthly* **90** (1983), 600-623.
- [Hum] J. E. Humphreys, *Introduction to Lie Algebras and Representation Theory*, Springer, New York, 1972.
- [Kna] A. W. Knapp, Review of *Matrix Groups: An Introduction to Lie Group Theory* by Andrew Baker and *Lie Groups: An Introduction Through Linear Groups* by Wolf Rossmann, *Amer. Math. Monthly* **110** (2003), 446-455.
- [Lit] P. Littelmann, “A generalization of the Littlewood-Richardson rule,” *J. Algebra* **130** (1990), 328-336.
- [Mc] M. McClard, “Picturing Representations of Simple Lie algebras of rank two,” Master’s thesis, Murray State University, 2000.
- [Pr] R. A. Proctor, “Solution of two difficult combinatorial problems with linear algebra,” *Amer. Math. Monthly* **89** (1982), 721-734.
- [Row] D. E. Rowe, Review of *Emergence of the Theory of Lie Groups* by Thomas Hawkins, *Notices Amer. Math. Soc.* **50** (2003), 668-677.
- [Sta] R. P. Stanley, *Enumerative Combinatorics, Vol. 1*, Wadsworth and Brooks/Cole, Monterey, CA, 1986.
- [Stem] J. R. Stembridge, “The Home Page for SF, posets, and coxeter/weyl”, <http://www.math.lsa.umich.edu/~jrs/maple.html>, 2002.