Affine.m – *Mathematica* package for computations in representation theory of finite-dimensional and affine Lie algebras

Anton Nazarov[a,b,*]

[a] *Department of High Energy Physics, Faculty of physics, SPb State University 198904, Sankt-Petersburg, Russia*
[b] *Chebyshev Laboratory, Faculty of Mathematics and Mechanics, SPb State University 199178, Saint-Petersburg, Russia*

Abstract

In this paper we present **Affine.m** – program for computations in representation theory of finite-dimensional and affine Lie algebras and describe implemented algorithms. Algorithms are based on the properties of weights and Weyl symmetry. The most important problems for us are the ones, concerning computation of weight multiplicities in irreducible and Verma modules, branching of representations and tensor product decomposition. These problems have numerous applications in physics and we provide some examples of these applications. The program is implemented in popular computer algebra system *Mathematica* and works with finite-dimensional and affine Lie algebras.

Keywords:
Mathematica; Lie algebra; affine Lie algebra; Kac-Moody algebra; root system; weights; irreducible modules, CFT, Integrable systems

PROGRAM SUMMARY

*Manuscript Title:***Affine.m** – *Mathematica* package for computations in representation theory of finite-dimensional and affine Lie algebras
*Authors:*Anton Nazarov
*Program Title:*Affine.m

*Corresponding author.
E-mail address: antonnaz@gmail.com

Journal Reference:
Catalogue identifier:
*Licensing provisions:*none
*Programming language:*Mathematica
*Computer:*i386-i686, x86_64
Operating system: Linux, Windows, MacOS, Solaris
RAM: 5-500 Mb
Keywords: Mathematica; Lie algebra; affine Lie algebra; Kac-Moody algebra; root system; weights; irreducible modules, CFT, Integrable systems
Classification: 5 Computer Algebra, 4.2 Other algebras and groups
Nature of problem:
Representation theory of finite-dimensional Lie algebras has many applications in different branches of physics, including elementary particle physics, molecular physics, nuclear physics. Representations of affine Lie algebras appear in string theories and two-dimensional conformal field theory used for the description of critical phenomena in two-dimensional systems. Also Lie symmetries play major role in study of quantum integrable systems.
Solution method:
We work with weights and roots of finite-dimensional and affine Lie algebras and use Weyl symmetry extensively. Central problems which are the computations of weight multiplicities, branching and fusion coefficients are solved using one general recurrent algorithm based on generalization of Weyl character formula. We also offer alternative implementation based on Freudenthal multiplicity formula which can be faster in some cases.
Restrictions:
Computational complexity grows fast with the rank of algebra, so computations for algebra of rank greater than 8 are non practical.
Unusual features:
We offer the possibility to use traditional mathematical notation for objects in representation theory of Lie algebras in computations if **Affine.m** is used in *Mathematica* notebook interface.
Running time:
From seconds to days depending on rank of algebra and complexity of representations.

1. Introduction

Representation theory of Lie algebras is of central importance for different areas of physics and mathematics. In physics Lie algebras are used to describe symmetries of quantum and classical systems. Computational methods in representation theory have a long history [1], there exist numerous software packages for computations related to Lie algebras [2], [3], [4, 5], [6], [7].

Most popular programs [2], [3], [6], [5] are created to study representation theory of simple finite-dimensional Lie algebras. The main computational problems are the following:

1. Construction of root system which determine the properties of Lie algebra including its commutation relations.
2. Weyl group traversal which is important due to Weyl symmetry of root system and characters of representations.
3. Calculation of weight multiplicities, branching and fusion coefficients, which are essential for construction and study of representations.

There are well-known algorithms for these tasks [8], [9], [1], [10]. The third problem is most computation-intensive. There are two different recurrent algorithms which are based on the Weyl character formula and the Freudenthal multiplicity formula. In this paper we analyze them both.

Infinite-dimensional Lie algebras also have growing number of applications in physics for example in conformal field theory and in a study of integrable systems. But infinite-dimensional algebras are much harder to investigate and the number of available computer programs is much smaller in this case.

Affine Lie algebras [11] constitute important and tractable class of infinite-dimensional Lie algebras. They are constructed as central extensions of loop algebras of (semi-simple) finite-dimensional Lie algebras and appear naturally in a study of Wess-Zumino-Witten and coset models in conformal field theory [12], [13], [14], [15].

The structure of affine Lie algebras allows to adapt for them computational algorithms created for finite-dimensional Lie algebras [7], [16], [17]. The book [17] with the tables of multiplicities and other computed characteristics of affine Lie algebras and representations was published in 1990. But we are not aware of software packages for popular computer algebra systems which can be used to extend these results. We address this issue and present **Affine.m** – *Mathematica* package for computations in representation theory of affine and finite-dimensional Lie algebras. We describe the features and limitations of the package in the present paper. We also provide representation-theoretical background of implemented algorithms and present examples of computations relevant to physical applications.

The paper starts with an overview of Lie algebras and their representation theory (Sec. 2). Then we describe datastructures of **Affine.m** used to present different objects related to Lie algebras and representations (Sec. 3) and discuss implemented algorithms (Sec. 4). Next section consists of physically interesting examples (Sec. 5). The paper is concluded with the discussion of possible extensions and refinements (Sec. 6).

2. Theoretical background

In this section we remind necessary definitions and present formulae used in computations.

2.1. Lie algebras of finite and affine types

Lie algebra $\mathfrak{g}$ is a vector space with bilinear operation $[\cdot,\cdot] : \mathfrak{g} \otimes \mathfrak{g} \to \mathfrak{g}$, which is called *Lie bracket*. If we choose the basis X_i in $\mathfrak{g}$ we can specify commutation relations by the *structure constants* C_{ijk}:

$$[X^i, X^j] = \sum_k C^{ij}_k X^k \tag{1}$$

Lie algebra is *simple* if it contains no non-trivial ideals with respect to commutator. *Semisimple* Lie algebra is a direct sum of simple Lie algebras. In the present paper we treat simple and semisimple Lie algebras.

Cartan subalgebra $\mathfrak{h}_{\mathfrak{g}}$ is a nilpotent subalgebra of $\mathfrak{g}$ that coincides with its normalizer. We denote the elements of basis of $\mathfrak{h}_{\mathfrak{g}}$ by H^i.

Killing form on $\mathfrak{g}$ gives a non-degenerate bilinear form $(\cdot,\cdot)$ on $\mathfrak{h}_{\mathfrak{g}}$ which can be used to identify $\mathfrak{h}_{\mathfrak{g}}$ with the subspace of the dual space $\mathfrak{h}^*_{\mathfrak{g}}$ of linear functionals on $\mathfrak{h}_{\mathfrak{g}}$. *Weights* are the elements of $\mathfrak{h}^*_{\mathfrak{g}}$ and are denoted by Greek letters $\mu, \nu, \omega, \lambda \ldots$

Special choice of basis gives compact description of commutation relations (1). This basis can be encoded by the root system which is discussed in Section 2.2 (See also [18, 19]).

Loop algebra $L\mathfrak{g} = \mathfrak{g} \otimes \mathbb{C}[t, t^{-1}]$, corresponding to semisimple Lie algebra $\mathfrak{g}$, has commutation relations

$$[X^i t^n, X^j t^m] = t^{n+m} \sum_k C^{ij}_k X^k \tag{2}$$

Central extension leads to the appearance of an additional term

$$[X^i t^n + \alpha c, X^j t^m + \beta c] = t^{n+m} \sum_k C_k^{ij} X^k + (X^i, X^j) n \delta_{n+m,0} c \tag{3}$$

This algebra $\hat{\mathfrak{g}} = \mathfrak{g} \otimes \mathbb{C}[t, t^{-1}] \oplus \mathbb{C}c$ is called non-twisted *affine Lie algebra* [11], [20, 21], [17].

We do not treat twisted affine Lie algebras in the present paper.

2.2. Modules, weights and roots

Let $\mathfrak{g}$ be a inite-dimensional or an affine Lie algebra.

Then $\mathfrak{g}$-module is a vector space V together with a bilinear map $\mathfrak{g} \times V \to V$ such that

$$[x, y] \cdot v = x \cdot (y \cdot v) - y \cdot (x \cdot v), \quad \text{for } x, y \in \mathfrak{g}, v \in V \tag{4}$$

Representation of algebra $\mathfrak{g}$ on a vector space V is the homomorphism $\mathfrak{g} \to gl(V)$ from $\mathfrak{g}$ to Lie algebra of endomorphisms of the vector space V with the commutator as a bracket.

For an arbitrary representation it is possible to diagonalize the operators corresponding to Cartan generators H^i simultaneously by a special choice of basis $\{v_j\}$ in V:

$$H^i \cdot v_j = \nu_j^i v_j \tag{5}$$

Eigenvalues ν_j^i of Cartan generators on an element of basis v_j determine a weight $\nu_j \in \mathfrak{h}_{\mathfrak{g}}^*$ such that $\nu_j(H^i) = \nu_j^i$. Vector $v \in V$ is called the weight vector of the weight λ if $Hv = \lambda_j(H)v, \ \forall H \in \mathfrak{h}$. The weight subspace consists of all weight vectors $V_\lambda = \{v \in V : Hv = \lambda_j(H)v, \ \forall H \in \mathfrak{h}\}$. The weight multiplicity $m_\lambda = \mathrm{mult}(\lambda) = \mathrm{dim} V_\lambda$ is the dimension of the weight subspace.

The structure of module is determined by the set of weights since the action of generators E^α on weight vectors is

$$E^\alpha \cdot v_\lambda \propto v_{\lambda+\alpha} \tag{6}$$

Module structure can be encoded by the formal character of a module

$$\mathrm{ch} V = \sum_\lambda m_\lambda e^\lambda \tag{7}$$

Character $\mathrm{ch}V \in \mathcal{E}$ is an element of algebra $\mathcal{E}$ generated by formal exponents of weights. A character can be specialized by taking its value on some element ξ of $\mathfrak{h}$.

Lie algebra is its own module with respect to a special kind of representation. The action that defines this representation is called adjoint and is given by the bracket $ad_X Y = [X, Y]$. *Roots* are weights of the adjoint representation of $\mathfrak{g}$. They encode the commutation relations of algebra in the following way. Denote by Δ the set of roots. For each $\alpha \in \Delta$ there exist a root $-\alpha \in \Delta$ and the generators $E^{\alpha}, E^{-\alpha}$ such that

$$[H^i, E^{\alpha}] = \alpha^i E^{\alpha} \tag{8}$$

$$\left[E^{\alpha}, E^{\beta}\right] = \begin{cases} N_{\alpha,\beta} E^{\alpha+\beta}, & \text{if } \alpha+\beta \in \Delta \\ \frac{2}{(\alpha,\alpha)} \sum_i \alpha^i H^i, & \text{if } \alpha = -\beta \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

Given the root system Δ we can choose the set of positive roots. This is a subset $\Delta^+ \subset \Delta$ such that for each root $\alpha \in \Delta$ exactly one of the roots $\alpha, -\alpha$ is contained in Δ^+ and for any two distinct positive roots $\alpha, \beta \in \Delta^+$ such that $\alpha + \beta \in \Delta$ their sum is also positive $\alpha + \beta \in \Delta^+$. Elements of $-\Delta^+$ are called negative roots.

A positive root is *simple* if it cannot be written as a sum of positive roots. The set of simple roots $\Phi = \{\alpha_i\}$ is a basis in $\mathfrak{h}^*_{\mathfrak{g}}$ and each root can be written as $\alpha = \sum_i n_i \alpha_i$ with all n_i non-negative or non-positive simultaneously. In case of a finite-dimensional Lie algebra $\mathfrak{g}$ simple roots are numbered from 1 to the rank of algebra $i = 1, \ldots, r, \quad r = \mathrm{rank}(\mathfrak{g})$. By numbering simple roots with an index i we introduce lexicographic ordering in the root system Δ. Highest root with respect to this ordering is denoted by $\theta = \sum_{i=1,\ldots,r} a_i \alpha_i$, coefficients a_i are called *marks*. θ is also the highest weight of the adjoint module (See section 2.3). *Comarcs* are the numbers equal to $a_i^{\vee} = \frac{(\alpha_i,\alpha_i)}{2} a_i$.

Although for affine Lie algebra $\hat{\mathfrak{g}}$ the set of roots Δ is infinite the set of simple roots Φ is finite and its elements are denoted by $\alpha_0, \ldots \alpha_r$ where $r = \mathrm{rank}(\mathfrak{g})$. The roots $\alpha_1, \ldots, \alpha_r$ are the roots of the underlying finite-dimensional Lie algebra $\overset{\circ}{\mathfrak{g}}$. The root $\alpha_0 = \delta - \theta$ is the difference of *imaginary root* δ and θ – the highest root of the algebra $\overset{\circ}{\mathfrak{g}}$. Note that root multiplicity $\mathrm{mult}(\alpha)$ for an affine Lie algebra root can be greater than one.

Subalgebra $\mathfrak{b}_+ \subset \mathfrak{g}$ spanned by the generators H^i, E^{α} for positive roots $\alpha \in \Delta^+$ is called the Borel subalgebra.

Parabolic subalgebra $\mathfrak{p}_I \supset \mathfrak{b}_+$ contains Borel subalgebra and is generated by some subset of simple roots $\{\alpha_j : j \in I, I \subset \{1 \ldots r\}\}$. It is spanned by the subset of generators $\{H^i\} \cup \{E^\alpha : \alpha \in \Delta^+\} \cup \{E^{-\alpha} : \alpha \in \Delta^+, \alpha = \sum_{j\in I} n_j \alpha_j\}$. *Regular subalgebra* $\mathfrak{a} \subset \mathfrak{g}$ is determined by the root system $\Delta_\mathfrak{a}$ with the set of simple roots $\{\beta_i, i = 1, \ldots, r_\mathfrak{a}\}$ being a subset of set of roots $\{\alpha_1, \ldots, \alpha_r\} \cup \{-\theta\}$.

The *Weyl group* $W_\mathfrak{g}$ is generated by reflections $\{s_i : \mathfrak{h}^*_\mathfrak{g} \to \mathfrak{h}^*_\mathfrak{g}\}$ corresponding to simple roots $\{\alpha_i\}$:

$$s_i \cdot \lambda = \lambda - \frac{2(\alpha_i, \lambda)}{(\alpha_i, \alpha_i)}\alpha_i \tag{10}$$

Root system and characters of representation are invariant with respect to the Weyl group action. Root system can be reconstructed from the set of simple roots by the Weyl group transformations.

Weyl groups are finite for finite-dimensional Lie algebras and finitely-generated for affine Lie algebras.

Consider an action of the element $s_\alpha s_{\alpha+\delta}$ of the Weyl group of affine Lie algebra $\hat{\mathfrak{g}}$ for α being a simple root of the underlying finite-dimensional Lie algebra $\mathfrak{g}$. Using definition (10) it is easy to see that $s_\alpha s_{\alpha+\delta} \cdot \lambda = \lambda + \frac{2}{(\alpha,\alpha)}\alpha + \left(\frac{(\alpha,\alpha)}{2(\lambda,\delta)} + (\lambda, \alpha)\right)\delta$. So the Weyl group can be presented as a semidirect product of the Weyl group $W_\mathfrak{g}$ of $\mathfrak{g}$ and the set of translations corresponding to roots of $\mathfrak{g}$.

A Weyl group element can be presented as a product of elementary reflections in multiple ways. Number of elementary reflections in the shortest sequence representing an element $w \in W_\mathfrak{g}$ is called the *length* of w and is denoted by $l(w)$. We also use the notation $\epsilon(w) = (-1)^{l(w)}$ for the parity of the number of Weyl reflections generating w.

Fundamental domain $\bar{C}$ for the Weyl group $W_\mathfrak{g}$ action on $\mathfrak{h}^*_\mathfrak{g}$ is determined by the requirement $\xi \in \bar{C} \Leftrightarrow (\xi, \alpha_i) \geq 0$ for all simple roots α_i. It is called a *main Weyl chamber*.

The *Cartan matrix* A is defined by products of simple roots

$$A_{ij} = \frac{2(\alpha_i, \alpha_j)}{(\alpha_j, \alpha_j)} \tag{11}$$

and can be used for a compact description of Lie algebra commutation relations in the Chevalley basis [18], [22], [23].

Form (11) induces a basis dual to the simple roots basis. It is called the *fundamental weights basis.* We denote its elements by ω_i:

$$\langle \omega_i, \alpha_j \rangle = \frac{2(\omega_i, \alpha_j)}{(\alpha_j, \alpha_j)} = \delta_{ij} \tag{12}$$

For a finite-dimensional Lie algebra there are r fundamental weights, $i = 1, \dots, r$. For affine Lie algebra we have additional fundamental weight $\omega_0 = \lambda,\ (\lambda, \delta) = 1,\ (\lambda, \lambda) = (\delta, \delta) = 0$. Other fundamental weights are equal to $\omega_i = a_i^v \lambda_0 + \overset{\circ}{\omega}_i$, where $\overset{\circ}{\omega}_i$ is the fundamental weight of the finite-dimensional Lie algebra $\mathfrak{g}$.

The sum of fundamental weights $\rho = \sum_i \omega_i$ is called a *Weyl vector.* It is an important tool in representation theory.

2.3. Highest weight modules

We consider finitely-generated $\mathfrak{g}$-modules V such that $V = \bigoplus_{\xi \in \mathfrak{h}_{\mathfrak{g}}^*} V_\xi$, where each V_ξ is finite-dimensional and there exists a finite set of weights $\lambda_1, \dots \lambda_s$ which generates the weight system of V, i.e. if $\mathrm{dim} V_\xi \neq 0$ then $\xi = \lambda_i - \sum_{k=1,\dots,r} n_k \alpha_k$ where $n_k \in \mathbb{Z}_+$ (See [24], [25]).

Highest weight module V^μ contains a single highest weight μ, all the other weights are obtained by subtractions of linear combinations of simple roots $\lambda = \mu - n_1 \alpha_1 - \cdots - n_r \alpha_r,\ n_k \in \mathbb{Z}_+$.

The most simple type of highest weight modules is the Verma module M^μ. Its space can be defined as a module

$$M^\mu = U(\mathfrak{g}) \underset{U(\mathfrak{b}_+)}{\otimes} D^\mu(\mathfrak{b}_+), \tag{13}$$

with respect to a multiplication in $U(\mathfrak{g})$ and $\underset{U(\mathfrak{b}_+)}{\otimes}$ means that the action of elements of $U(\mathfrak{b}_+)$ "falls through" the left part of tensor product onto the right part. Here $\mathfrak{b}_+$ is Borel subalgebra, $D^\mu(\mathfrak{b}_+)$ is a representation of $\mathfrak{b}_+$ such that $D(E^\alpha) = 0,\ D(H) = \mu(H)$ for any positive root α. Elements of $\mathfrak{g}$ act from the left and we should commute all the elements of $\mathfrak{b}_+$ to the right, so that they can act on the space $D^\lambda(\mathfrak{b}_+)$.

Weight multiplicities in Verma modules can be found by applying the Weyl character formula

$$\mathrm{ch} M^\mu = \frac{e^\mu}{\prod_{\alpha \in \Delta^+} (1 - e^{-\alpha})^{\mathrm{mult}(\alpha)}} = \frac{e^\mu}{\sum_{w \in W} \epsilon(w) e^{w\rho - \rho}} \tag{14}$$

Here we have used the Weyl denominator identity

$$R := \prod_{\alpha\in\Delta^+}\left(1-e^{-\alpha}\right)^{\mathrm{mult}(\alpha)} = \sum_{w\in W}\epsilon(w)e^{w\rho-\rho}, \tag{15}$$

and $\epsilon\left(w\right) := \det\left(w\right)$ is equal to the parity of the sequence of Weyl reflections generating w.

Verma module M^{μ} has the unique maximal submodule and the unique nontrivial simple quotient L^{μ} which is an *irreducible highest weight module.*

Irreducible highest weight modules have no non-trivial submodules. The Weyl character formula for irreducible highest weight modules is

$$\mathrm{ch}L^{\mu} = \frac{\sum_{w\in W}\epsilon(w)e^{w(\mu+\rho)-\rho}}{\sum_{w\in W}\epsilon(w)e^{w\rho-\rho}} = \sum_{w\in W}\epsilon(w)\ \mathrm{ch}M^{w(\mu+\rho)-\rho} \tag{16}$$

Thus the character of an irreducible highest weight module can be seen as a combination of characters of Verma modules. (This fact is a consequence of the Bernstein-Gelfand-Gelfand resolution ([26, 27], see also [24]).)

Construction of generalized Verma modules is analogous to (13), but representation of Borel subalgebra is substituted by a representation of parabolic subalgebra $\mathfrak{p}_I \supset \mathfrak{b}_+$ generated by some subset $\{\alpha_I\}$ of simple roots $I \subset \{1,\ldots,r\}$:

$$M_I^{\mu} = U\left(\mathfrak{g}\right)\otimes_{U(\mathfrak{p}_I)} L^{\mu}_{\mathfrak{p}_I}.$$

Introduce a formal element $R_I := \prod_{\alpha\in\Delta^+\setminus\Delta^+_{\mathfrak{p}_I}}\left(1-e^{-\alpha}\right)^{\mathrm{mult}(\alpha)}$. Then the character of a generalized Verma module can be written as

$$\mathrm{ch}M_I^{\mu} = \frac{1}{R_I}\mathrm{ch}L^{\mu}_{\mathfrak{p}_I}. \tag{17}$$

We can use the Weyl character formula to obtain recurrent relations for weight multiplicities – important tools for calculations [28, 29].

For irreducible highest-weight modules recurrent relations have the following form

$$m_{\xi} = -\sum_{w\in W\setminus e}\epsilon(w)m_{\xi-(w(\rho)-\rho)} + \sum_{w\in W}\epsilon(w)\delta_{(w(\mu+\rho)-\rho),\xi}. \tag{18}$$

Formulae for Verma and generalized Verma modules differ only in the second term on the right-hand side. In case of Verma module it is just $\delta_{\xi,\mu}$. For generalized Verma module the summation in the second term on the right-hand side of (18) is over the Weyl subgroup generated by reflections corresponding to roots $\{\alpha_I\}$.

Other recurrent formula can be obtained from a study of Casimir elements action on irreducible highest weight modules [18]:

$$m_\lambda = \frac{2}{(\mu+\rho)^2 - (\lambda+\rho)^2} \sum_{\alpha\in\Delta^+} \sum_{k\geq 1} (\lambda + k\alpha, \alpha) m_{\lambda+k\alpha}. \tag{19}$$

It is called the Freudenthal multiplicity formula. Note that it is applicable only to irreducible modules.

We discuss the use of formulae (18) and (19) for computations in section 4.

Now consider an algebra $\mathfrak{g}$ and a reductive subalgebra $\mathfrak{a} \subset \mathfrak{g}$. Simple roots β_i of the subalgebra $\mathfrak{a}$ can be presented as linear combinations of $\mathfrak{g}$-algebra roots α_j: $\beta_i = \sum_{j=1,\ldots,r_{\mathfrak{g}}} k_j \alpha_j,\ j = 1, \ldots, r_{\mathfrak{a}}$.

Each irreducible $\mathfrak{g}$-module is also an $\mathfrak{a}$-module, although $L^\mu_{\mathfrak{g}}$ is in general not irreducible as $\mathfrak{a}$-module. It can be decomposed into a direct sum of irreducible $\mathfrak{a}$-modules:

$$L^\mu_{\mathfrak{g}} = \bigoplus_\nu b^\mu_\nu L^\nu_{\mathfrak{a}} \tag{20}$$

Coefficients in such a decomposition are called branching coefficients.

It is possible to calculate branching coefficients by constructing and successively subtracting the submodules $L^\nu_{\mathfrak{a}}$. (The characters were not involved yet!) This traditional approach has serious limitations especially in case of affine Lie algebras. We discuss them in the end of section 4.

Now we describe an alternative approach which is based on recurrent properties of branching coefficients. But before we proceed to these recurrent relations we need several additional definitions.

For a subalgebra $\mathfrak{a} \subset \mathfrak{g}$ we introduce the subalgebra $\mathfrak{a}_\perp$. Consider the root subspace $\mathfrak{h}^*_{\perp\mathfrak{a}}$ orthogonal to $\mathfrak{h}_{\mathfrak{a}}$,

$$\mathfrak{h}^*_{\perp\mathfrak{a}} := \{\eta \in \mathfrak{h}^* | \forall h \in \mathfrak{h}_{\mathfrak{a}}; \eta(h) = 0\},$$

and the roots (correspondingly – positive roots) of $\mathfrak{g}$ orthogonal to roots of $\mathfrak{a}$,

$$\begin{aligned} \Delta_{\mathfrak{a}_\perp} &:= \{\beta \in \Delta_{\mathfrak{g}} | \forall h \in \mathfrak{h}_{\mathfrak{a}}; \beta(h) = 0\}, \\ \Delta^+_{\mathfrak{a}_\perp} &:= \left\{\beta^+ \in \Delta^+_{\mathfrak{g}} | \forall h \in \mathfrak{h}_{\mathfrak{a}}; \beta^+(h) = 0\right\}. \end{aligned} \tag{21}$$

Let $W_{\mathfrak{a}_\perp}$ be a subgroup of W generated by reflections w_β with the roots $\beta \in \Delta^+_{\mathfrak{a}_\perp}$. The subsystem $\Delta_{\mathfrak{a}_\perp}$ determines a subalgebra $\mathfrak{a}_\perp$ with the Cartan subalgebra $\mathfrak{h}_{\mathfrak{a}_\perp}$.

The Cartan subalgebra $\mathfrak{h}$ can be decomposed in the following way: $\mathfrak{h} = \mathfrak{h}_{\mathfrak{a}} \oplus \mathfrak{h}_{\mathfrak{a}_\perp} \oplus \mathfrak{h}_\perp$

We also introduce the notations

$$\widetilde{\mathfrak{a}_\perp} := \mathfrak{a}_\perp \oplus \mathfrak{h}_\perp \qquad \widetilde{\mathfrak{a}} := \mathfrak{a} \oplus \mathfrak{h}_\perp. \tag{22}$$

For $\mathfrak{a}$ and $\mathfrak{a}_\perp$ we consider the corresponding Weyl vectors, $\rho_{\mathfrak{a}}$ and $\rho_{\mathfrak{a}_\perp}$ and compose the so called "defects" $\mathcal{D}_{\mathfrak{a}}$ and $\mathcal{D}_{\mathfrak{a}_\perp}$ of the injection:

$$\mathcal{D}_{\mathfrak{a}} := \rho_{\mathfrak{a}} - \pi_{\mathfrak{a}}\rho, \qquad \mathcal{D}_{\mathfrak{a}_\perp} := \rho_{\mathfrak{a}_\perp} - \pi_{\mathfrak{a}_\perp}\rho. \tag{23}$$

For $\mu \in P^+$ consider the linked weights $\{(w(\mu+\rho)-\rho)\,|w \in W\}$ and their projections to $h^*_{\mathfrak{a}_\perp}$ additionally shifted by the defect $-\mathcal{D}_{\mathfrak{a}_\perp}$:

$$\mu_{\mathfrak{a}_\perp}(w) := \pi_{\mathfrak{a}_\perp}[w(\mu+\rho)-\rho] - \mathcal{D}_{\mathfrak{a}_\perp}, \quad w \in W.$$

Among the weights $\{\mu_{\mathfrak{a}_\perp}(w)\,|w \in W\}$ one can always choose those located in the fundamental chamber $\overline{C_{\mathfrak{a}_\perp}}$. Let U be the set of representatives u for the classes $W/W_{\mathfrak{a}_\perp}$ such that

$$U := \left\{u \in W| \quad \mu_{\mathfrak{a}_\perp}(u) \in \overline{C_{\mathfrak{a}_\perp}}\right\} \quad . \tag{24}$$

Thus we can form the subsets:

$$\mu_{\widetilde{\mathfrak{a}}}(u) := \pi_{\widetilde{\mathfrak{a}}}[u(\mu+\rho)-\rho] + \mathcal{D}_{\mathfrak{a}_\perp}, \quad u \in U, \tag{25}$$

and

$$\mu_{\mathfrak{a}_\perp}(u) := \pi_{\mathfrak{a}_\perp}[u(\mu+\rho)-\rho] - \mathcal{D}_{\mathfrak{a}_\perp}, \quad u \in U. \tag{26}$$

Notice that the subalgebra $\mathfrak{a}_\perp$ is regular by definition since it is built on a subset of roots of the algebra $\mathfrak{g}$.

Denote by $k^{(\mu)}_\xi$ signed branching coefficients. If $\xi \in \bar{C}_{\mathfrak{a}}$ is in main Weyl chamber $k^{(\mu)}_\xi = b^{(\mu)}_\xi$ otherwise $k^{(\mu)}_\xi = \epsilon(w) b^{(\mu)}_{w(\xi+\rho_{\mathfrak{a}})-\rho_{\mathfrak{a}}}$ where $w \in W_{\mathfrak{a}}$ is such that $w(\xi+\rho_{\mathfrak{a}})-\rho_{\mathfrak{a}} \in W_{\mathfrak{a}}$.

Now we can use the Weyl character formula to write a recurrent relation [30] for signed branching coefficients $k^{(\mu)}_\xi$ corresponding to an injection $\mathfrak{a} \hookrightarrow \mathfrak{g}$:

$$\begin{aligned} k^{(\mu)}_\xi = -\tfrac{1}{s(\gamma_0)} \Big(& \textstyle\sum_{u\in U} \epsilon(u) \ \dim\left(L^{\mu_{\mathfrak{a}_\perp}(u)}_{\mathfrak{a}_\perp}\right) \delta_{\xi-\gamma_0,\pi_{\widetilde{\mathfrak{a}}}(u(\mu+\rho)-\rho)} + \\ & + \textstyle\sum_{\gamma\in\Gamma_{\widetilde{\mathfrak{a}}\to\mathfrak{g}}} s(\gamma+\gamma_0)\, k^{(\mu)}_{\xi+\gamma} \Big). \end{aligned} \tag{27}$$

The recursion is governed by the set $\Gamma_{\mathfrak{a}\to\mathfrak{g}}$ called the injection fan. The latter is defined by the carrier set $\{\xi\}_{\mathfrak{a}\to\mathfrak{g}}$ for the coefficient function $s(\xi)$

$$\{\xi\}_{\widetilde{\mathfrak{a}}\to\mathfrak{g}} := \{\xi \in P_{\widetilde{\mathfrak{a}}} | s(\xi) \neq 0\}$$

appearing in the expansion

$$\prod_{\alpha\in\Delta^+\setminus\Delta^+_\perp} \left(1-e^{-\pi_{\widetilde{\mathfrak{a}}}\alpha}\right)^{\mathrm{mult}(\alpha)-\mathrm{mult}_{\mathfrak{a}}(\pi_{\widetilde{\mathfrak{a}}}\alpha)} = -\sum_{\gamma\in P_{\widetilde{\mathfrak{a}}}} s(\gamma)e^{-\gamma}; \tag{28}$$

The weights in $\{\xi\}_{\widetilde{\mathfrak{a}}\to\mathfrak{g}}$ are to be shifted by γ_0 – the lowest vector in $\{\xi\}$ – and the zero element is to be eliminated:

$$\Gamma_{\mathfrak{a}\to\mathfrak{g}} = \{\xi-\gamma_0 | \xi \in \{\xi\}\} \setminus \{0\}\,. \tag{29}$$

Formula (18) is a particular case of recurrent relation for branching coefficients (27) in the case of Cartan subalgebra $\mathfrak{a}=\mathfrak{h}_{\mathfrak{g}}$.

If the root system of $\mathfrak{a}_\perp$ is generated by some subset of $\mathfrak{g}$ simple roots $\alpha_1,\ldots,\alpha_r$ then the recurrent relation (27) is connected with the generalized Bernstein-Bernstein-Gelfand resolution for parabolic Verma modules [31].

Another particular case of this formula is connected with tensor product decompositions. Consider the tensor product of two irreducible $\mathfrak{g}$-modules $L^\mu \otimes L^\nu$. It is also a $\mathfrak{g}$-module but not irreducible in general. So

$$L^\mu \otimes L^\nu = \bigoplus_\gamma f^{\mu\nu}_\gamma L^\gamma \tag{30}$$

The coefficients $f^{\mu\nu}_\gamma$ are called fusion coefficients. The problem of computation of fusion coefficients is equivalent to branching problem for the diagonal subalgebra $\mathfrak{g}\subset\mathfrak{g}\oplus\mathfrak{g}$ (see [32]). So our implementation of recurrent algorithm can be used to decompose tensor products (See Section 5.1).

In the case of affine Lie algebras $\mathfrak{g},\mathfrak{a}$ the multiplicities m_ν and the corresponding branching coefficients b_ν can be regarded as the coefficients in power series decomposition of string and branching functions correspondingly:

$$\sigma_\nu(q) = \sum_{n=0}^{\infty} m_{\nu-n\delta}q^n, \quad \nu=\sum_j c_j\omega_j, \quad c_j\geq 0 \tag{31}$$

$$b_\nu(q) = \sum_{n=0}^{\infty} b_{\nu-n\delta}q^n, \quad \nu=\sum_j c_j\omega_j, \quad c_j\geq 0 \tag{32}$$

String and branching functions have modular and analytic properties which are important for conformal field theory, especially in coset models and the study of CFT on higher genus surfaces [33], [13], [12], [34].

Affine.m calculates weight multiplicities and branching coefficients for affine Lie algebras up to some finite grade. We present examples of computations in Sections 5.3, 5.4. Now we proceed to the description of datastructures and algorithms implemented in **Affine.m**.

3. Core datastructures

Having introduced necessary mathematical objects, problems and relations we now describe the related datastructures of **Affine.m**. Although *Mathematica* is untyped language it is possible to create structured objects and do type checks with patterns [35], [36].

3.1. Weights

Weights are represented by two data-structures: `finiteWeight` for finite-dimensional Lie algebras and `affineWeight` for affine.

Internally the finite weight is a List with the Head `finiteWeight`, its components are the coordinates of the weight in the orthogonal Bourbaki basis [23].

Affine weight is an extension of a finite weight by supplying it with level and grade coordinates. There is a set of functions defined for finite and affine weights. The complete list can be found in online help of the package. The most important are definitions of addition, multiplication by a number and a scalar product (bilinear form) for weights. These definitions allow to use traditional notation when working with **Affine.m**:

```
w=makeFiniteWeight[{1,0,3}];
v=makeFiniteWeight[{3,2,1}];
2*w+v==makeFiniteWeight[{5,2,7}]
w.v==6
```

The use of orthogonal basis in the internal structure of weights allows us to work with weights without complete specification of root system which is useful for study of branching, since roots of the subalgebra can be specified by hand.

3.2. Root systems

To specify an algebra of finite or affine type it is enough to fix its root system. Root systems are represented by two datatypes `finiteRootSystem` and affineRootSystem. The latter is an extension of the former. We offer several different constructors for these datastructures. It is possible to specify the set of simple roots explicitly, for example to study the subalgebra $B_2 \subset B_4$ we can use the definition

```
b2b4=makeFiniteRootSystem[ { {1,−1,0,0}, {0,1,0,0} } ]
```

There are constructors for root systems of simple finite-dimensional Lie algebras:

```
b2=makeSimpleRootSystem[B,2]
```

We use typographic features of *Mathematica* frontend to offer mathematical notation for simple Lie algebras:

```
B₂ == makeFiniteRootSystem[ { {1, −1}, {0, 1} }]
```

Non-twisted affine root systems can be created as affine extensions of finte root systems, e.g.

```
b2affine = makeAffineExtension[b2]
```

In notebook interface this can be written simply as $\hat{B}_2$.

Semisimple Lie algebras can be created as sums of simple:

```
A₁ ⊕ A₁ == finiteRootSystem[2, 2, {finiteWeight[2, {1, 0}], finiteWeight[2, {0, 1}]}]
```

Predicate rootSystemQ checks if the object is root systems of finite or affine type.

List of simple roots is the property of root system so it is accessed as rs[simpleRoots].

We have implemented several functions to get major properties of root systems. Weyl vector is given by the function rho[rs_?rootSystemQ]:

```
In[1]  =  rho[b2]
Out[1]  =  finiteWeight[2, {3/2, 1/2}]
```

Positive roots can be constructed with positiveRoots[rs_?rootSystemQ]. For an affine Lie algebra this and related functions return the list up to fixed grade. This grade limit is set through the value of rs[gradeLimit] which is equal to 10 by default. List of roots (up to gradeLimit) is returned by roots[rs]. Cartan matrix and fundamental weights are calculated by the functions cartanMatrix and fundamentalWeights correspondingly.

It is possible to specify a weight of a Lie algebra by its Dynkin labels

```
weight[b2][1,2,3] == makeFiniteWeight[{2, 1}]
```

The function dynkinLabels[rs_?rootSystemQ][wg_?weightQ] returns Dynkin labels of weight wg in the root system rs.

Elements of Weyl group are specified by the set of indices of reflections, so the element $w = s_1 s_2 s_1$ of the Weyl group of algebra B_2 is constructed with `weylGroupElement[b2][1,2,1]`. Then it can be applied to the weights:

```
w = weylGroupElement[b2][1,2,1];
w @ makeFiniteWeight[{1,0}] == makeFiniteWeight[{-1,0}]
```

Computation of lexicographically minimal form [10, 37] for Weyl group elements can be conveniently implemented through the use of pattern-matching in *Mathematica*. In [38] rewrite rules for simple finite dimensional and affine Lie algebras are presented as *Mathematica* patterns. Our presentation of Weyl group elements is compatible with the code of [38]

```
In[1]  = <<A3reduce;
         reduce[s[1,2,1,2,1,3,2,1,1]]
Out[1] = s[2, 3, 2]

In[2]  = (weylGroupElement[A3] @@ reduce[s[1,2,1,2,1,3,2,1,1]]) @ weight[A3][-1,-2,-1]
Out[2] = finiteWeight[4, {-2, 2, 1, -1}]

In[3]  = dynkinLabels[A3][Out[2]]
Out[3] = {-4, 1, 2}
```

3.3. Formal elements

We represent formal characters of representations by special structure `formalElement`. This structure is a hash-table implemented with DownValues. The keys are weights at the exponents and the values are the corresponding multiplicities. `makeFormalElement[{`$\gamma_1,\dots,\gamma_n$`},{`$m_1,\dots,m_n$`}]` creates data-structure which represents the element $\sum_{i=1}^{n} m_i e^{\gamma_i}$ of the formal algebra $\mathcal{E}$. The operations in $\mathcal{E}$ are implemented for `formalElement` data-type: formal elements can be added, multiplied by a number or by an exponent of a weight. There exists also a multiplication of formal elements but no division.

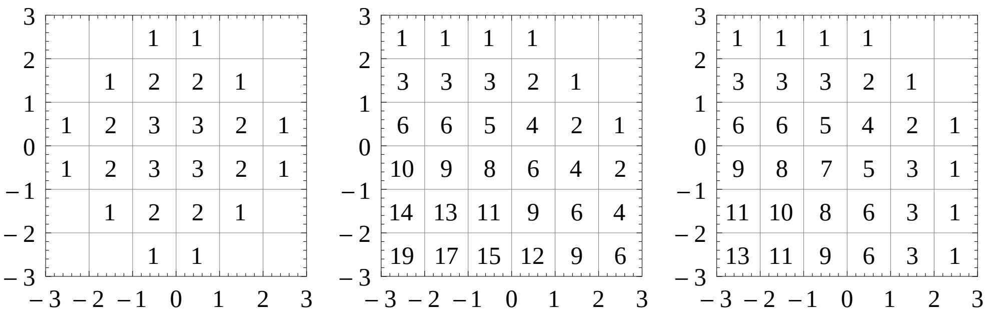
```
In[1]  = makeFormalElement[{makeFiniteWeight[{1,1}],makeFiniteWeight[{0,0}]},{1,2}] *
           (2 * Exp[makeFiniteWeight[{1,0}]] *
           makeFormalElement[{makeFiniteWeight[{1,1}],makeFiniteWeight[{0,0}]},{1,2}]);
In[2]  = In[1][weights]
Out[2] = {finiteWeight[2, {1, 0}], finiteWeight[2, {2, 1}], finiteWeight[2, {3, 2}]}

In[3]  = In[1][multiplicities]
Out[3] = {8, 8, 2}
```

3.4. Modules

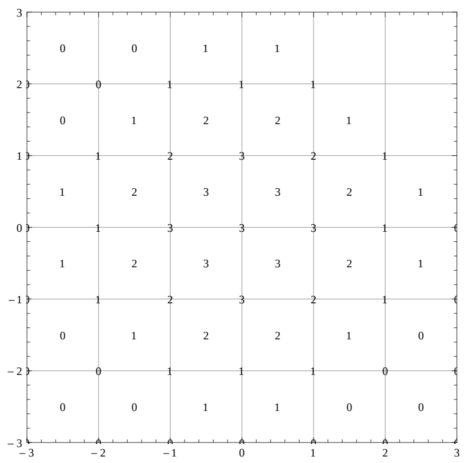
Affine.m can be used to study different kinds of modules, i.e. Verma modules, irreducible modules and parabolic Verma modules. We need datastructure `module` to represent generic module of Lie algebra $\mathfrak{g}$. Module properties can be deduced from its set of singular weights using Weyl character formulae (14),(15),(17),(16). Set of singular weights can have Weyl symmetry. It can be a symmetry with respect to the Weyl group $W_{\mathfrak{g}}$ or with respect to some subalgebra $W_{\mathfrak{a}}$ as in the case of parabolic Verma modules. Then it is possible to study only the main Weyl chamber $C_{\mathfrak{a}}$. To use this symmetry generic constructor for `module` datastructure accepts several parameters `makeModule[rs_?rootSystemQ][singWeights_formalElement,subs_?rootSystemQ|emptyRootSystem[],limit:10`. Here `rs` is the root system of the Lie algebra $\mathfrak{g}$,`singWeights` is the set of singular weights, `subs` is the root system corresponding to the Weyl group $W_{\mathfrak{a}}$ which is the (anti-)symmetry of the set of singular weights. Parameter `limit` limits computation for infinite-dimensional modules such as Verma or parabolic Verma. There are several specialized constructors for different types of highest weight modules:

```
vm=makeVermaModule[B2][{2,1}];
pm=makeParabolicVermaModule[B2][weight[B2][2,1],{1}];
im=makeIrreducibleModule[B2][2,1];
GraphicsRow[textPlot/@{im,vm,pm}]
```



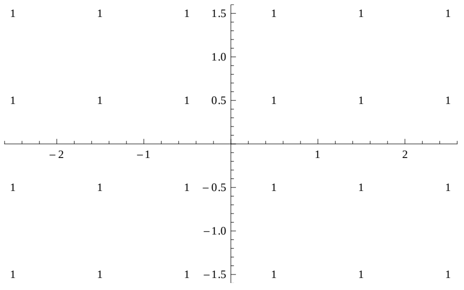
As we already stated properties of a module are encoded by its singular element. Function `singularElement[m_module]` returns singular element of a module as `formalElement` datastructure. Character (up to `limit` for (parabolic) Verma modules) is returned by the function `character[m_module]`. Direct sum of modules is a module and we use natural notation

```
im1=makeIrreducibleModule[B2][weight[B2][2,1]];
im2=makeIrreducibleModule[B2][weight[B2][1,2]];
textPlot[im1⊕ im2]
```



The tensor product is also implemented but only for finite-dimensional Lie algebras, since tensor product of affine Lie algebra modules leads to rich new structures [39, 40, 41] which are out of the scope of the present paper.

```
textPlot[makeIrreducibleModule[A1][5]⊗ makeIrreducibleModule[A1][3]];
```



4. Computational algorithms

As we have already stated in section 2.3 there exist two recurrent relations which can be used to calculate weight multiplicities in irreducible modules. Both algorithms proceed in the following way to calculate weight multiplicities:

1. Create the list of weights in the main Weyl chamber by subtracting all possible combinations of simple roots from the highest weight (e.g. for finite-dimensional algebra subtract α_1 from μ while inside $\bar{C}$, then subtract α_2 from all the weights already obtained etc).
2. Sort the list of weights by their product with the Weyl vector.

3. Use a recurrent formula. If the weight required for recurrent computation is outside the main chamber use the Weyl symmetry.

The difference in the performance of algorithms is in the number of previous values required to get the multiplicity of weight under consideration. For the Weyl formula based recurrent relation (18) it is constant and equal to the number of elements in the Weyl group (if we are far from the boundary of representation diagram). When the Freudenthal formula (19) is used number of previous values grows with the distance from the external border of representation. So the Freudenthal formula is faster if the weight is close to the border or the rank of the algebra and the size of Weyl group is large [8]. Note that the Freudenthal formula is valid for irreducible modules only, so it can not be used to study (generalized) Verma modules.

We have made some experiments with our implementations of the Freudenthal formula and formula (19) and have got Figure 1, which depicts the dependence of the computation time on the number of weights in a module.

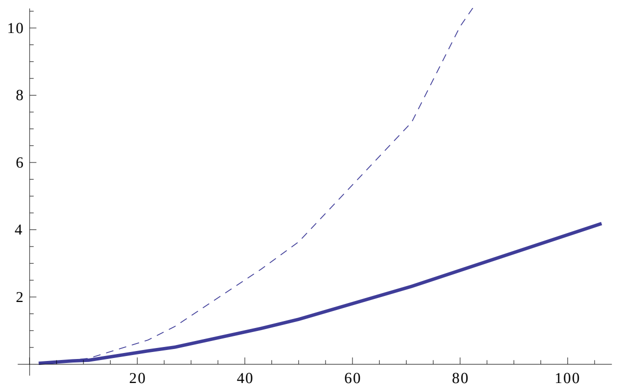


Figure 1: Running time of algorithms based on the Freudenthal formula (19) (dashed) and recurrent relation (18) (solid) with the number of weights in $\bar{C}$ for calculation of multiplicities in representations of B_2.

In the calculation of branching coefficients the application of the Freudenthal formula requires a complete construction of formal characters of an algebra module and all the representations of a subalgebra. It is impractical when the rank of algebra and subalgebra are big, for example for maximal subalgebras.

Alternative algorithm which was presented in the paper [30] contains the following steps:

1. Construct the root system $\Delta_{\mathfrak{a}}$ for the embedding $\mathfrak{a} \to \mathfrak{g}$.
2. Select all the positive roots $\alpha \in \Delta^+$ orthogonal to $\mathfrak{a}$, i.e. form the set $\Delta^+_{\mathfrak{a}_\perp}$.
3. Construct the set $\Gamma_{\mathfrak{a}\to\mathfrak{g}}$. Relation (2) defines the sign function $s(\gamma)$ and the set $\Phi_{\mathfrak{a}\subset\mathfrak{g}}$ where the lowest weight γ_0 is to be subtracted to get the fan: $\Gamma_{\mathfrak{a}\to\mathfrak{g}} = \{\xi - \gamma_0 | \xi \in \Phi_{\mathfrak{a}\subset\mathfrak{g}}\} \setminus \{0\}$.
4. Construct the set $\widehat{\Psi^{(\mu)}} = \{w(\mu+\rho)-\rho;\ w \in W\}$ of singular weights for the $\mathfrak{g}$-module $L^{(\mu)}$.
5. Select the weights $\left\{\mu_{\widetilde{\mathfrak{a}_\perp}}(w) = \pi_{\widetilde{\mathfrak{a}_\perp}}\left[w(\mu+\rho)-\rho\right] - \mathcal{D}_{\mathfrak{a}_\perp} \in \overline{C_{\widetilde{\mathfrak{a}_\perp}}}\right\}$. Since the set $\Delta^+_{\mathfrak{a}_\perp}$ is fixed we can easily check wether the weight $\mu_{\widetilde{\mathfrak{a}_\perp}}(w)$ belongs to the main Weyl chamber $\overline{C_{\widetilde{\mathfrak{a}_\perp}}}$ (by computing its scalar product with the fundamental weights of $\mathfrak{a}^+_\perp$).
6. For the weights $\mu_{\widetilde{\mathfrak{a}_\perp}}(w)$ calculate dimensions of the corresponding modules, $\dim\left(L^{\mu_{\widetilde{\mathfrak{a}_\perp}}(w)}_{\widetilde{\mathfrak{a}_\perp}}\right)$, using the Weyl dimension formula and construct the singular element $\Psi^{(\mu)}_{(\mathfrak{a},\mathfrak{a}_\perp)}$.
7. Calculate the anomalous branching coefficients using recurrent relation (27) and select among them those corresponding to the weights in the main Weyl chamber $\overline{C_{\mathfrak{a}}}$.

We can speed up the algorithm by one-time computation of the representatives of the conjugate classes $W/W_{\mathfrak{a}_\perp}$.

Consider the regular embedding $B_2 \subset B_4$. In this case the fan consists of 24 elements. In order to decompose B_4 module we need to construct the subset of singular weights of the module which projects to the main Weyl chamber of the subalgebra B_2. The full set of singular weights consists of 384 elements. The required subset contains at most 48 elements. So the time for the construction of this required subset is negligible if the number of branching coefficients is greater than that. We may estimate the total number of required operations for the computation of branching coefficients as the product of the number of elements in the main Weyl chamber of a subalgebra with non-zero branching coefficients and the number of elements in the fan. In the case of a direct algorithm we need to compute the multiplicities for each module in the decomposition, so the number of operations grows faster than the square of the number of elements in the main Weyl chamber of a subalgebra with non-zero branching coefficients.

To further illustrate this performance issue we include Figure 2 where we show the time required for computations of branching coefficients for

$B_3 \subset B_4$.

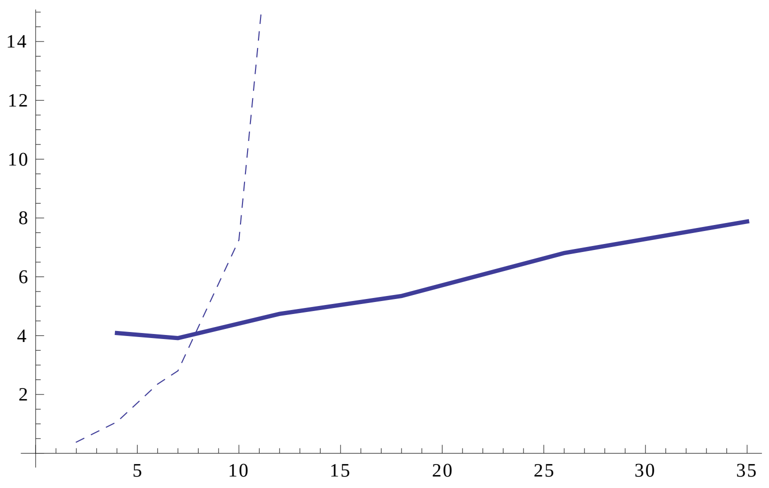


Figure 2: Running time of algorithms based on the Freudenthal formula (19) (dashed) and recurrent relation (27) (solid) with the number of weights in $\bar{C}$ for calculation of branching coefficients for $B_3 \subset B_4$.

5. Examples

In this section we present some examples of computations available with **Affine.m** with the code required to produce these results.

5.1. Tensor product decompositon for finite-dimensional Lie algebras

Computation of fusion coefficients for the decomposition of tensor product of highest-weight modules to the direct sum of irreducible modules has numerous applications in physics. For example, we can consider the spin of a composite system such as an atom. Another interesting example is integrable spin chain consisting of N particles with the spins living in some representation L of a Lie algebra $\mathfrak{g}$ with a $\mathfrak{g}$-invariant Hamiltonian H, describing the nearest-neighbour spin-spin interaction. In order to solve such a system, i.e. to find eigenstates of the Hamiltonian, we need to decompose $L^{\otimes N}$ into the direct sum of irreducible $\mathfrak{g}$-modules of lower dimension and diagonalize the Hamiltonian on these modules.

For fundamental representations of simple Lie algebras it is sometimes possible to get an analytic result describing the dependence of decomposition coefficients on N (See [32]). Our code provides numerical values and can be used to check this analytic results.

Consider a tensor power of B_2 the first fundamental representation $\left(L^{[1,0]}\right)^{\otimes 4}$. Decomposition coefficients are just the branching coefficients for the tensor power module reduced to the diagonal subalgebra $B_2 \subset B_2 \oplus B_2 \oplus B_2 \oplus B_2$. So the following code calculates these coefficients:

```
fm = makeIrreducibleModule[B2][1, 0];
tp = ((fm⊗ ]fm)⊗ fm)⊗]fm;
subs = makeFiniteRootSystem[
  {1/4*{1, −1, 1, −1, 1, −1, 1, −1},
   1/4*{0, 1, 0, 1, 0, 1, 0, 1}}];
bc = branching[tp, subs];
{bc[#], dynkinLabels[subs][#]} & /@ bc[weights]
```

It produces a list of highest weights and tensor product decomposition coefficients:

```
{{1, {4, 0}}, {3, {2, 2}}, {0, {3, 0}},
{2, {0, 4}}, {3, {1, 2}}, {6, {2, 0}},
{6, {0, 2}}, {1, {1, 0}}, {3, {0, 0}}}]
```

Returning to the problem of spin chain Hamiltonian diagonalization we can see that instead of diagonalizing operator in space of dimension 625 we can diagonalize operators in the spaces of dimensions $55, 81, 30, 35, 35, 14, 10, 5, 1$.

5.2. Branching and parabolic Verma modules

We illustrate the generalized BGG-resolution by the diagrams of G_2 parabolic Verma modules which appear in the decomposition of irreducible module $L_{G_2}^{[1,1]}$:

$$\mathrm{ch}\left(L^{\mu}\right)=\sum_{u\in U} e^{\mu_{\tilde{\mathfrak{a}}}(u)}\epsilon(u)\mathrm{ch}M_I^{\mu_{\mathfrak{a}_\perp}(u)}. \tag{33}$$

The character of $L^{[1,1]}$ is presented in Figure 3, characters of the generalized Verma modules in decomposition (33) are shown in Figure 4. Characters in the upper row appear in (33) with positive sign and in the lower row – with negative.

5.3. String functions of affine Lie algebras and CFT models

String functions can be used to present formal characters of affine Lie algebra highest weight representation. They have interesting analytic and modular properties [11, 33, 42].

Affine.m produces power series decomposition for string functions. Consider an affine Lie algebra $\hat{sl(3)} = \hat{A}_2$ and its highest weight module $L^{(1,0,0)}$. To get the string functions we can use the code:

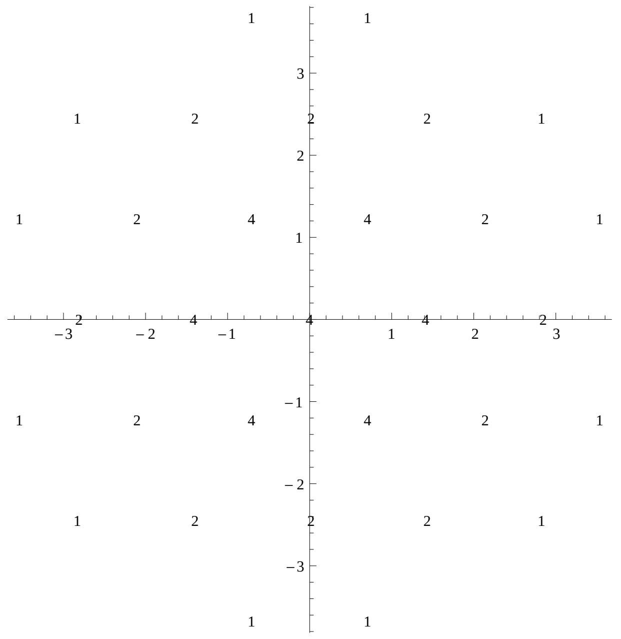


Figure 3: Character of the irreducible G_2-module $L^{[1,1]}$

```
stringFunctions[$\hat{A}_2$,{1,1,2}]
{{0, 0, 4},
  $2q + 10q^2 + 40q^3 + 133q^4 + 398q^5 + 1084q^6 + 2760q^7 + 6632q^8 + 15214q^9 + 33508q^{10}$},
{{0, 3, 1},
  $2q + 12q^2 + 49q^3 + 166q^4 + 494q^5 + 1340q^6 + 3387q^7 + 8086q^8 + 18415q^9 + 40302q^{10}$},
{{1, 1, 2},
  $1 + 6q + 27q^2 + 96q^3 + 298q^4 + 836q^5 + 2173q^6 + 5310q^7 + 12341q^8 + 27486q^9 + 59029q^{10}$},
{{2, 2, 0},
  $1 + 8q + 35q^2 + 124q^3 + 379q^4 + 1052q^5 + 2700q^6 + 6536q^7 + 15047q^8 + 33248q^9 + 70877q^{10}$},
{{3, 0, 1},
  $2 + 12q + 49q^2 + 166q^3 + 494q^4 + 1340q^5 + 3387q^6 + 8086q^7 + 18415q^8 + 40302q^9 + 85226q^{10}$}
```

Similarly for the affine Lie algebra $\hat{G}_2$ we get

```
stringFunctions[$\hat{G}_2$,{1,1,0}]
{{2, 0, 0},
  $1 + 8q + 37q^2 + 138q^3 + 431q^4 + 1227q^5 + 3208q^6 + 7901q^7$},
{{0, 0, 1},
  $3q + 18q^2 + 73q^3 + 247q^4 + 736q^5 + 2000q^6 + 5070q^7$},
{{1, 1, 0},
  $1 + 7q + 32q^2 + 117q^3 + 370q^4 + 1055q^5 + 2780q^6 + 6880q^7$},
{{0, 2, 0},
  $3q + 15q^2 + 63q^3 + 210q^4 + 633q^5 + 1725q^6 + 4407q^7$}
```

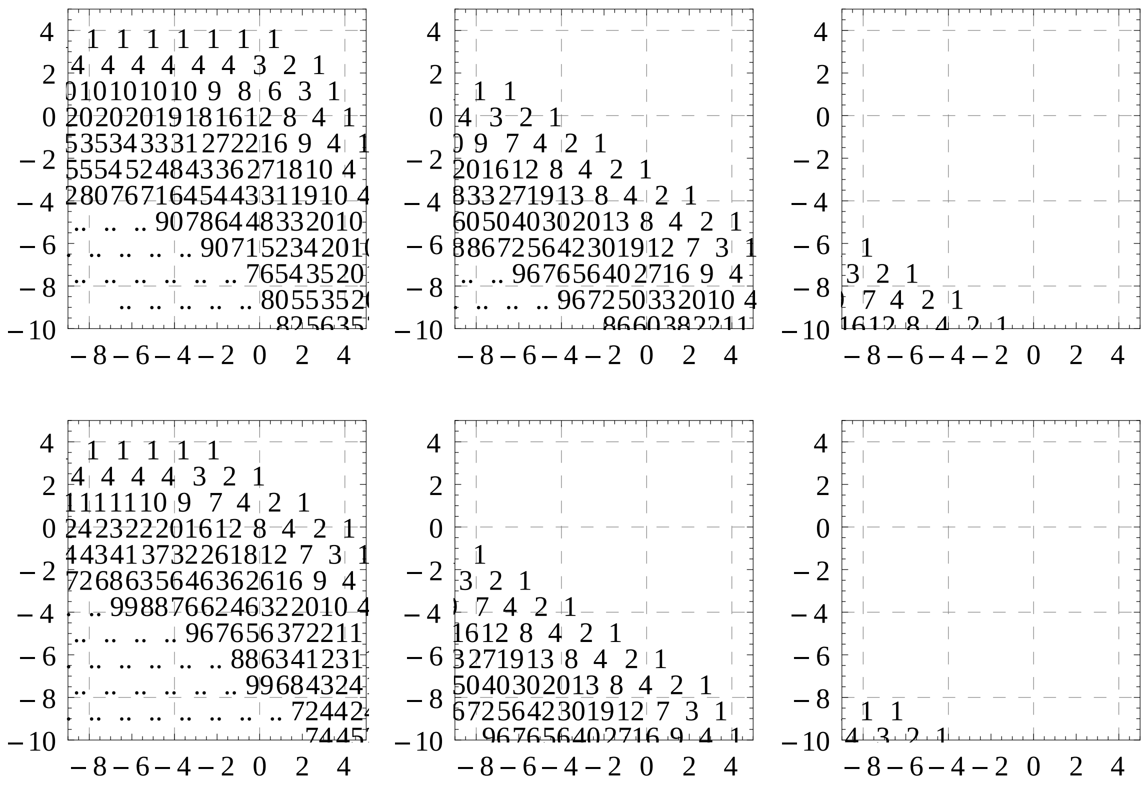


Figure 4: Characters of G_2 generalized Verma modules appearing in the decomposition of $L^{[1,1]}$. Parabolic Verma modules in the upper row appear in the decomposition with a positive sign, in the lower row – with a negative.

5.4. Branching functions and coset models of conformal field theory

It is believed that most rational models of CFT can be obtained from cosets G/A corresponding to the embedding $\mathfrak{a} \subset \mathfrak{g}$. These models can be studied as gauge theories [43, 44].

Branching functions for an embedding $\mathfrak{a} \subset \mathfrak{g}$ are the partition functions of CFT on the torus (see [13]).

As a first example we show how to construct branching functions for the embedding $\hat{A}_1 \to \hat{B}_2$ up to the tenth grade:

```
branchingFunctions[B̂2,makeAffineExtension[makeFiniteRootSystem[{{1, 1}}]], {1, 1, 1}]
```

```
{{3, 0},
 2 + 14q + 52q^2 + 154q^3 + 410q^4 + 994q^5 + 2248q^6 + 4832q^7 + 9934q^8 + 19680q^9 + 37802q^10},
 {{2, 1},
  4 + 20q + 72q^2 + 220q^3 + 584q^4 + 1424q^5 + 3248q^6 + 7012q^7 + 14488q^8 + 28844q^9 + 55616q^10} ,
 {{0, 3},
  4q + 20q^2 + 68q^3 + 200q^4 + 516q^5 + 1224q^6 + 2736q^7 + 5808q^8 + 11820q^9 + 23236q^10} ,
 {{1, 2},
  2 + 14q + 54q^2 + 168q^3 + 462q^4 + 1148q^5 + 2656q^6 + 5812q^7 + 12130q^8 + 24358q^9 + 47328q^10}
```

Another example demonstrates the computation of branching functions for the regular embedding $\hat{B}_2 \subset \hat{C}_3$:

```
sub=makeAffineExtension[parabolicSubalgebra[C3][2,3]];
branchingFunctions[Ĉ3,sub, {2, 0, 0, 0}]
```

```
{{0, 1, 0},
  2q − 20q^3 + 24q^4 + 82q^5 − 320q^6 + 108q^7} ,
{{1, 0, 0},
  1 − q − 8q^2 + 19q^3 + 16q^4 − 156q^5 + 205q^6 + 640q^7} ,
{{0, 0, 1},
  q − 5q^3 + 7q^5}
```

6. Conclusion

We have presented the package **Affine.m** for computations in representation theory of finite-dimensional and affine Lie algebras. It can be used to study Weyl symmetry, root systems, irreducible, Verma and parabolic Verma modules of finite-dimensional and affine Lie algebras. In the present paper we have also discussed main ideas used for an implementation of the package and described most important notions of representation theory required to use **Affine.m**.

We have demonstrated that recurrent approach based on the Weyl character formula is not only useful for calculations but also allows to establish connections with the (generalized) Bernstein-Bernstein-Gelfand resolution.

Also we have presented examples of computations with this package connected with problems of physics and mathematics.

In future versions of our software we are going to treat twisted affine Lie algebras, extended affine Lie algebras and provide more direct support for tensor product decompositions.

Acknowledgements

I thank V.D. Lyakhovsky and O.Postnova for discussions and helpful comments.

The work is supported by the Chebyshev Laboratory (Department of Mathematics and Mechanics, Saint-Petersburg State University) under the grant 11.G34.31.0026 of the Government of the Russian Federation.

Appendix A. Software package

The package can be freely downloaded from `http://github.com/naa/Affine`. To get the development code use the command

```
git clone git://github.com/naa/Affine.git
```

Contents of the package:

```
Affine/                              root folder
  demo/                                demonstrations
    demo.nb                              demo notebook
    paper.nb                             code for the paper
  doc/                               documentation folder
    figures/                           figures in paper
      timing.pdf                         diagram showing performance
      branching-timing.pdf               ...  for branching coefficients
      irrep-sum.pdf                      sum of B2 irreps
      irrep-verma-pverma.pdf             irrep, Verma, (p)Verma for B2
      G2-irrep.pdf                       irrep for G2
      G2-pverma.pdf                      parabolic Verma for G2
      tensor-product.pdf                 tensor product of A1-modules
    bibliography.bib                   bibliographic database
    paper.pdf                          present paper
    paper.tex                          paper source
    TODO.org                           list of issues
  src/                               source folder
    affine.m                           main software package
  tests/                             unit tests folder
    tests.m                            unit tests
  README.markdown                    installation and usage notes
```

References

[1] Belinfante, J. and Kolman, B., *A survey of Lie groups and Lie algebras with applications and computational methods*, Society for Industrial Mathematics, 1989.

[2] Nutma, T., Simplie.

[3] Van Leeuwen, M., Euromath Bull **1** (1994) 83.

[4] Stembridge, J., Journal of Symbolic Computation **20** (1995) 755.

[5] Stembridge, J., Coxeter/weyl packages for maple.

[6] Fischbacher, T., (2002).

[7] Fuchs, J., Schellekens, A. N., and Schweigert, C., Nucl. Phys. **B473** (1996) 323.

[8] Moody, R. and Patera, J., AMERICAN MATHEMATICAL SOCIETY **7** (1982).

[9] Stembridge, J., Math. Soc. Japan, Tokyo (2001) 1.

[10] Casselman, W., Inventiones mathematicae **116** (1994) 95.

[11] Kac, V., *Infinite dimensional Lie algebras*, Cambridge University Press, 1990.

[12] Walton, M., (1999).

[13] Di Francesco, P., Mathieu, P., and Senechal, D., *Conformal field theory*, Springer, 1997.

[14] Goddard, P., Kent, A., and Olive, D., Physics Letters B **152** (1985) 88 .

[15] Dunbar, D. C. and Joshi, K. G., Int. J. Mod. Phys. **A8** (1993) 4103.

[16] Gannon, T., (2001).

[17] Kass, S., Moody, R., Patera, J., and Slansky, R., *Affine Lie algebras, weight multiplicities, and branching rules*, Sl, 1990.

[18] Humphreys, J., *Introduction to Lie algebras and representation theory*, Springer, 1997.

[19] Humphreys, J., *Reflection groups and Coxeter groups*, Cambridge Univ Pr, 1992.

[20] Wakimoto, M., *Infinite-dimensional Lie algebras*, American Mathematical Society, 2001.

[21] Wakimoto, M., *Lectures on infinite-dimensional Lie algebra*, World scientific, 2001.

[22] Fulton, W. and Harris, J., *Representation theory: a first course*, volume 129, Springer Verlag, 1991.

[23] Bourbaki, N., *Lie groups and Lie algebras*, Springer Verlag, 2002.

[24] Humphreys, J., *Representations of semisimple Lie algebras in the BGG category O*, Amer Mathematical Society, 2008.

[25] Carter, R., *Lie algebras of finite and affine type*, Cambridge University Press, 2005.

[26] Bernstein, J., Gel'fand, I., and Gel'fand, S., Funktsional'nyi Analiz i ego prilozheniya **10** (1976) 1.

[27] Bernstein, I., Gel'fand, I., and Gel'fand, S., Functional Analysis and Its Applications **5** (1971) 1.

[28] Il'in, M., Kulish, P., and Lyakhovsky, V., Zapiski Nauchnykh Seminarov POMI **374** (2010) 197.

[29] Kulish, P. and Lyakhovsky, V., Symmetry, Integrability and Geometry: Methods and Applications **4**.

[30] Lyakhovsky, V. and Nazarov, A., Journal of Physics A: Mathematical and Theoretical **44** (2011) 075205.

[31] Lyakhovsky, V. and Nazarov, A., (2011).

[32] Kulish, P. P., Lyakhovsky, V. D., and Postnova, O. P., ArXiv e-prints (2011).

[33] Kac, V. and Wakimoto, M., Advances in mathematics(New York, NY. 1965) **70** (1988) 156.

[34] Walton, M., Nuclear Physics B **322** (1989) 775.

[35] Shifrin, L., *Mathematica programming: an advanced introduction.*

[36] Maeder, R., *Computer science with Mathematica*, Cambridge University Press, 2000.

[37] Casselman, W., Automata to perform basic calculations in coxeter groups, in *Representations of groups: Canadian Mathematical Society annual seminar, June 15-24, 1994, Banff, Alberta, Canada*, volume 16, page 35, Canadian Mathematical Society, 1995.

[38] van der Kallen, W., Computing shortlex rewite rules with mathematica.

[39] Kazhdan, D. and Lusztig, G., AMERICAN MATHEMATICAL SOCIETY **7** (1994).

[40] Kazhdan, D. and Lusztig, G., Journal of the American Mathematical Society **6** (1993) 905.

[41] Kazhdan, D. and Lusztig, G., AMERICAN MATHEMATICAL SOCIETY **6** (1993).

[42] Kac, V. and Peterson, D., Adv. in Math **53** (1984) 125.

[43] Hwang, S. and Rhedin, H., Mod. Phys. Lett. **A10** (1995) 823.

[44] Hwang, S. and Rhedin, H., Nuclear Physics B **406** (1993) 165.