# Computational tools for representation theory of affine Lie algebras

Anton Nazarov

29 April 2009

**Abstract**

We describe computational algorithms for construction of representations of affine Lie algebras and computation of branching coefficients of representations of affine Lie algebra to representations of regular affine sub-algebra. Also we introduce the implementation of these algorithms as Maple package and present examples studied with these computational tools.

## 1 Introduction

Affine Lie algebras constitute the simplest and most studied class of infinite-dimesional Lie algebras [1, 2]. They are connected with theory of modular forms and other mathematical subjects, but their main applications in physics are in conformal field theory, where they are current algebras of WZW-models [3, 4].

Since the affine Lie algebras are infinite-dimensional extensions of simple finite-dimensional Lie algebras, their representation theory is well understood [2], [1], [5]. But infinite dimensionality poses a challenge for computational methods.

In this paper we discuss the tasks of construction of representations which is reduced to computation of weight multiplicities and of branching of representation of affine Lie algebra to representation of affine subalgebra.

The first task is usually solved using the algorithms based on the recursive Freudenthal formula for weight multiplicities (1). The summation is optimized by considering of Weyl symmetry [6], [7]. While these algorithms are reasonably fast, the use of Racah-like recurrent relation (6) is theoretically more efficient [8]. In practice speed depends upon the realisation. So we describe this alternative approach in section 2 and show results of practical speed comparisons.

The task of calculation of branching coefficients is usually solved by the reduction to finite-dimensional case [9]. In some physical applications this task is greatly simplified by the additional physical constraints, but for general case this approach is rather inconvenient since it is not easily formalisable for machine compuatation. We propose another approach based on recurrent relations for anomalous branching coefficients [10] in section 3 and discuss the method to optimize this computation. Then we describe our realisation of mentioned algorithms 4 and some applications 5.

## 2 Representations

We consider integrable modules $L^{\mu}$ with the highest weight $\mu$. As for simple Lie algebras the most computation-intensive task is to find weight multiplicities (dimensions of weight subspaces). These multiplicities are usually encoded in form of string functions and these functions are connected with generic theta-functions and modular forms [2].

It is possible to write general relations for string functions in order to express them as the combination of generic theta-functions [**?**]. But in these paper we limit ourselves to automated computation of weight multiplicities.

Weight multiplicities are essential for explicit construction of module and are also used for calculation of WZW fusion coefficients which are sufficient for calculation of all correlation functions in WZW-model [3].

There are several recursive algorithms for computation of weight multiplicities which are based on different recurrent relations between weight multiplicities. These relations follow from denominator identity and Weyl-Kac formula [2, 1].

Freudenthal formula is the most used formula for computation of weight multiplicities of simple finite-dimensional Lie algebras:

$$mult(\xi) = \frac{2}{(\mu+\rho|\mu+\rho)-(\xi+\rho|\xi+\rho)} \sum_{\alpha\in\Delta^{+}} mult(\alpha) \sum_{k=1}^{\infty} mult(\xi+k\alpha)(\xi+k\alpha|\alpha) \tag{1}$$

Here $\xi$ is the weight in question, $\mu$ - highest weight of the module, $\rho$ - Weyl vector of Lie algebra and $\Delta^{+}$ is the set of positive roots of the algebra.

Algorithms, based on Freudenthal formula (1) are implemented in several software packages, such as Coxeter/Weyl [11], LiE [12], LambdaTensor [13].

Freudenthal formula can be used for affine Lie algebras as well. It was the main tool for preparation of tables in the book [6], but the software used by the authors is not available. We introduce our implementation of this algorithm in section 4 and compare it ( 5) to the alternative approach, based on the formula (6), which is described below.

We are mostly interested in the use of the new method proposed in the papers [14, 15, 10, 16]. The idea is to use recurrent relations for anomalous branching coefficients based on the summation over the special set of vectors $\Gamma_{\mathfrak{a}\subset\mathfrak{g}}$ called "fan". We need to introduce some notations. Let's consider the reduction of the representation of the affine Lie algebra $\mathfrak{g}$ to representations of affine Lie algebra $\mathfrak{a}$. By $\pi_{\mathfrak{a}}$ we denote the projection of the root space $\mathfrak{h}^*_{\mathfrak{g}}$ to $\mathfrak{h}^*_{\mathfrak{a}}$. The set $\Gamma_{\mathfrak{a}\subset\mathfrak{g}}$ is introduced as the combination of projection of positive roots $\Delta^+$ of algebra $\mathfrak{g}$ using formulae:

$$\prod_{\alpha\in(\pi_{\mathfrak{a}}\circ\Delta^+)} \left(1-e^{-\alpha}\right)^{\mathrm{mult}(\alpha)-\mathrm{mult}_{\mathfrak{a}}(\alpha)} = -\sum_{\gamma\in\Phi_{\mathfrak{a}\subset\mathfrak{g}}} s\left(\gamma\right) e^{-\gamma}. \tag{2}$$

$$\Phi_{\mathfrak{a}\subset\mathfrak{g}} = \left\{\gamma\in P_{\mathfrak{a}} \mid s\left(\gamma\right)\neq 0\right\}; \tag{3}$$

$$\Gamma_{\mathfrak{a}\subset\mathfrak{g}} = \left\{\xi-\gamma_0 | \xi\in\Phi_{\mathfrak{a}\subset\mathfrak{g}}\right\} \setminus \left\{0\right\}. \tag{4}$$

$$k_{\xi}^{(\mu)} = -\frac{1}{s\left(\gamma_0\right)}\left(\sum_{w\in W}\epsilon\left(w\right)\delta_{\xi,\pi_{\mathfrak{a}}\circ(w\circ(\mu+\rho)-\rho)+\gamma_0} + \sum_{\gamma\in\Gamma_{\mathfrak{a}\subset\mathfrak{g}}} s\left(\gamma+\gamma_0\right) k_{\xi+\gamma}^{(\mu)}\right) \tag{5}$$

Fan doesn't depend on the module, but is determined by the injection of sub-algebra into the algebra. If sub-algebra is Cartan sub-algebra, this relation gives recurrent relation for weight multiplicities

$$mult(\xi) = -\sum_{w\in W\setminus e} det(w)\ mult(\xi+\rho-w\rho) + \sum_{w\in W} det(w)\delta_{\xi,w(\mu+\rho)-\rho} \tag{6}$$

The algorithm works as follows:

- Compute the set of "anomalous" weights as the orbit of sum of highest weight $\mu$ and Weyl vector $\rho$ under the action of Weyl group, shifted by $-\rho$.
- Compute the set of vectors $\rho - w\rho$ for $w\in W\setminus e$
- Set the multiplicity of highest weight equal to 1 as the initial condition.
- Calculate multiplicities of weights grade-by-grade using formula (6).

We could illustrate the algorithm with the picture.

Weyl group of affine Lie algebra is richer than that of simple finite-dimensional Lie algebra. It can be presented as semi-direct product of finite Weyl group $W_0$ of simple Lie algebra $\mathfrak{g}_0 \subset \mathfrak{g}$ with infinite abelian group translations along the roots of $\mathfrak{g}_0$ which are non-imaginary roots of $\mathfrak{g}$: $W = W_0 \ltimes T$.

The Weyl group orbit of highest root bounds the weights with non-zero multiplicity.

Rich Weyl symmetry of affine Lie algebras allows further improvement of algorithm.

It can be shown that it is enough to compute multiplicities of the weights of the form $\lambda = (\overset{\circ}{\lambda}, k, n)$ where $\overset{\circ}{\lambda}$ is finite part, $k$ - level of the representation and $n$ - grade only for finite number of $\overset{\circ}{\lambda}$ lying in the intersection of main Weyl chamber with the subspace of grade zero. Multiplicities of all other weights could be obtained by Weyl group action.

So all the required information can be encoded in the form of string functions:

$$\sigma^{\mu}_{\lambda} = \sum_{k=0}^{\infty} mult(\mu - kd) q^k \tag{7}$$

Where $d$ is the element dual to imaginary root $\delta$.

It is possible to “fold” the star, so that the summation in formula (6) involves only weights from main Weyl chamber. Also this computation can be rewritten as infinite set of equations on string functions coefficient, which can be solved recursively. See [16] for details.

This method is theoretically more efficient than the use of Freudenthal formula (1) [8], but our implementation is rather inefficient. Also we don’t have independent realisation of algorithms, based on (1) for affine Lie algebras, so we haven’t done any practical speed comparisons yet.

## 3 Branching rules

Say we want to obtain decomposition of module $L^{\mu}$ of algebra $\mathfrak{g}$ into modules of sub-algebra $\mathfrak{a}$

$$L^{\mu}_{\mathfrak{g}\downarrow\mathfrak{a}} = \bigoplus_{\nu \in P^{+}_{\mathfrak{a}}} b^{(\mu)}_{\nu} L^{\nu}_{\mathfrak{a}} \tag{8}$$

Here $\nu$ goes through weight lattice of $\mathfrak{a}$.

Branching coefficients can be computed in similar way using formula (23).

Notice that to obtain the branching rules we need only the projected singular element $\pi_{\mathfrak{a}} \circ \Psi^{(\mu)}$ of this module and the fan $\Gamma_{\mathfrak{a}\subset\mathfrak{g}}$ and do not need any other properties of the module itself.

Singular element can be computed the same way as the set of “anomalous” weights in section 2 and then projected onto the weight lattice of $\mathfrak{a}$.

The fan $\Gamma_{\mathfrak{a}\subset\mathfrak{g}}$ is computed as the combination of projection of positive roots $\Delta^+$ of algebra $\mathfrak{g}$ using formula:

$$\Phi_{\mathfrak{a}\subset\mathfrak{g}} = \{\gamma \in P_{\mathfrak{a}} \mid s(\gamma) \neq 0\}; \tag{9}$$

$$\prod_{\alpha\in(\pi_{\mathfrak{a}}\circ\Delta^+)} \left(1-e^{-\alpha}\right)^{\mathrm{mult}(\alpha)-\mathrm{mult}_{\mathfrak{a}}(\alpha)} = -\sum_{\gamma\in\Phi_{\mathfrak{a}\subset\mathfrak{g}}} s(\gamma)\, e^{-\gamma}. \tag{10}$$

$$\Gamma_{\mathfrak{a}\subset\mathfrak{g}} = \{\xi-\gamma_0 | \xi \in \Phi_{\mathfrak{a}\subset\mathfrak{g}}\} \setminus \{0\}. \tag{11}$$

Then branching coefficients are computed recursively using formula (23), which gives anomalous branching coefficients which are equal to usual branching coefficients for weights from main Weyl chamber.

## 3.1 Folding

To facilitate computation of branching coefficients we can use the idea similar to one used for computation of weight multiplicities. There we were able to fold the star so that all the recurrent computation took place in main Weyl chamber.

We use the formula (6). The summation in the first term of right-hand side goes through the set of weights $\{\xi+\pi-\omega\rho \mid \omega \in W\backslash e\}$. But for each weight $\zeta$ outside the main Weyl chamber there exists weight $\zeta_0$ in main Weyl chamber and $\omega \in W$ such that $\zeta = \omega\zeta_0$. So to calculate the multiplicity of $\xi$ we sum over the set of weights inside the main Weyl chamber.

The question is if it is possible to introduce similar procedure for computation of branching coefficients.

We start with the formula

$$L^{\mu}_{\mathfrak{g}\downarrow\mathfrak{a}} = \bigoplus_{\nu\in P^+_{\mathfrak{a}}} b^{(\mu)}_{\nu} L^{\nu}_{\mathfrak{a}} \tag{12}$$

In terms of characters it has the following form

$$\pi_{\mathfrak{a}}(chL^{\mu}_{\mathfrak{g}}) = \sum_{\nu\in P^+_{\mathfrak{a}}} b^{(\mu)}_{\nu} chL^{\nu}_{\mathfrak{a}} \tag{13}$$

It is well-known (see for example [1], [2]) that characters are Weyl-invariant.

$$ch(L^{\omega\mu}) = ch(L^{\mu}) \tag{14}$$

Also Weyl group of the subalgebra is subgroup of Weyl group of algebra $W_{\mathfrak{a}} \subset W_{\mathfrak{g}}$. Using Weyl-Kac character formula

$$ch(L^{\mu}) = \frac{\sum_{\omega\in W} \epsilon(\omega) e^{\omega(\mu+\rho)-\rho}}{\prod_{\alpha\in\Delta^+}(1-e^{-\alpha})^{mult(\alpha)}} \tag{15}$$

the equation (13) can be rewritten as

$$\pi_{\mathfrak{a}}(chL^{\mu}_{\mathfrak{g}}) = \frac{\sum_{\nu\in P^+_{\mathfrak{a}}} b^{(\mu)}_{\nu} \Psi^{(\nu)}_{\mathfrak{a}}}{R_{\mathfrak{a}}} \tag{16}$$

where we have introduced the singular weight elements $\Psi^{(\nu)}_{\mathfrak{a}} = \sum_{\omega\in W_{\mathfrak{a}}} \epsilon(\omega) e^{\omega(\nu+\rho_{\mathfrak{a}})-\rho_{\mathfrak{a}}}$ for $\mathfrak{a}$-modules $L^{\nu}_{\mathfrak{a}}$ and the denominator $R_{\mathfrak{a}} = \prod_{\alpha\in\Delta^+_{\mathfrak{a}}}(1-e^{\alpha})^{mult_{\mathfrak{a}}(\alpha)}$. In the paper [10] following notation was introduced

$$\sum_{\nu\in P^+_{\mathfrak{a}}} b^{(\mu)}_{\nu} \Psi^{(\nu)}_{\mathfrak{a}} = \sum_{\nu\in P^+_{\mathfrak{a}}} b^{(\mu)}_{\nu} \sum_{\omega\in W_{\mathfrak{a}}} \epsilon(\omega) e^{\omega(\nu+\rho_{\mathfrak{a}})-\rho_{\mathfrak{a}}} \equiv \sum_{\lambda\in P_{\mathfrak{a}}} k^{(\mu)}_{\lambda} e^{\lambda} \tag{17}$$

The numbers $k^{(\mu)}_{\lambda}$ are called anomalous branching coefficients and are equal to the multiplicities of the submodules $L^{\nu}_{\mathfrak{a}}$ times the determinants $\epsilon(\omega)$.

Now we can rewrite formula (16) as

$$\pi_{\mathfrak{a}}(chL^{\mu}_{\mathfrak{g}}) = \frac{\sum_{\lambda\in P_{\mathfrak{a}}} k^{(\mu)}_{\lambda} e^{\lambda+\rho_{\mathfrak{a}}}}{e^{\rho_{\mathfrak{a}}} R_{\mathfrak{a}}} \tag{18}$$

Here we have multiplied the numerator and the denominator by $e^{\rho_{\mathfrak{a}}}$. Now we multiply both sides of equation by $e^{\rho_{\mathfrak{a}}} R_{\mathfrak{a}}$.

$$\pi_{\mathfrak{a}}(chL^{\mu}_{\mathfrak{g}}) e^{\rho_{\mathfrak{a}}} R_{\mathfrak{a}} = \sum_{\lambda\in P_{\mathfrak{a}}} k^{(\mu)}_{\lambda} e^{\lambda+\rho_{\mathfrak{a}}} \tag{19}$$

Then left-hand side of equation is anti-invariant under Weyl transformations of group $W_{\mathfrak{a}}$, since the character of $L^{\mu}_{\mathfrak{g}}$ is invariant, projection $\pi_{\mathfrak{a}}$ is orthogonal and $e^{\rho_{\mathfrak{a}}} R_{\mathfrak{a}}$ is anti-invariant ($\omega(e^{\rho_{\mathfrak{a}}} R_{\mathfrak{a}}) = \epsilon(\omega) e^{\rho_{\mathfrak{a}}} R_{\mathfrak{a}}$, see [1] for the proof). So we have

$$\omega\left(\sum_{\lambda\in P_{\mathfrak{a}}} k^{(\mu)}_{\lambda} e^{\lambda+\rho_{\mathfrak{a}}}\right) = \epsilon(\omega) \sum_{\lambda\in P_{\mathfrak{a}}} k^{(\mu)}_{\lambda} e^{\lambda+\rho_{\mathfrak{a}}} \tag{20}$$

It can be rewritten as

$$\sum_{\lambda\in P_{\mathfrak{a}}} k^{(\mu)}_{\omega^{-1}(\lambda+\rho_{\mathfrak{a}})-\rho_{\mathfrak{a}}} e^{\lambda+\rho_{\mathfrak{a}}} = \epsilon(\omega) \sum_{\lambda\in P_{\mathfrak{a}}} k^{(\mu)}_{\lambda} e^{\lambda+\rho_{\mathfrak{a}}} \tag{21}$$

Then we can equate coefficients of identical exponents

$$k_{\lambda}^{(\mu)} = \epsilon(\omega) k_{\omega(\lambda+\rho_{\mathfrak{a}})-\rho_{\mathfrak{a}}}^{(\mu)} \tag{22}$$

We see that Weyl symmetry gives us all the values of $k_{\lambda}^{(\mu)}$ from the values inside the shifted by $-\rho_{\mathfrak{a}}$ fundamental Weyl chamber $C_{\mathfrak{a}} - \rho_{\mathfrak{a}}$.

In the paper [10] following recurrent formula for computation of $k_{\lambda}^{(\mu)}$ was proved

$$k_{\lambda}^{(\mu)} = -\frac{1}{s(\gamma_0)} \left( \sum_{w \in W} \epsilon(w)\, \delta_{\lambda, \pi_{\mathfrak{a}} \circ (w \circ (\mu+\rho) - \rho) + \gamma_0} + \sum_{\gamma \in \Gamma_{\mathfrak{a} \subset \mathfrak{g}}} s(\gamma + \gamma_0)\, k_{\lambda+\gamma}^{(\mu)} \right) \tag{23}$$

Here $k_{\lambda}^{(\mu)}$ is computed as the sum over special set $\Gamma_{\mathfrak{a} \subset \mathfrak{g}}$ (11).

Using relation (22) we can restrict computation to the shifted by $-\rho_{\mathfrak{a}}$ fundamental Weyl chamber $C_{\mathfrak{a}} - \rho_{\mathfrak{a}}$. Then branching coefficients inside the fundamental Weyl chamber are given by anomalous branching coefficients $b_{\lambda}^{(\mu)} = k_{\lambda}^{(\mu)}$ for $\lambda \in C_{\mathfrak{a}}$ [10].

## 4 Details of implementation

The algorithms are implemented as Maple programs. Since theory of affine Lie algebras is based upon the theory of simple finite-dimensional Lie algebras we use Coxeter/Weyl package [11] as the foundation of our work.

This design choice imposed some performance constraints but allowed us to keep the code concise by the lack of need of reimplementation of algorithms for computation of roots, weights and finite Weyl reflection. So there is less than 1000 lines of code in our package.

Currently implementation allows to calculate roots, dominant weights, weight multiplicities and branching coefficients for non-twisted affine Lie algebras of series $A, B, C, D$ and for twisted algebras of series $A^{(2)}$ and $D^{(2)}$.

Our program can be freely downloaded from `http://anton.nazarov.name/affine`

## 5 Applications

Weight multiplicities, calculated with presented algorithms can be used for explicit construction of irreducible highest-weight modules. For example, let's consider $A_3$-module of level 2 with highest weight $\alpha_1 + \alpha_2 + \alpha_3$.

String function coefficients are equal to

```
> al:=algebra_roots(A3);
              al := [e2 - e1, e3 - e2, e4 - e3, delta + e1 - e4]

> mu:=2*lambda0+sum(al[i],i=1..3);
                          mu := -e1 + e4 + 2 lambda0

> string_function(mu,mu,A3,30);
[1, 7, 32, 117, 371, 1063, 2819, 7029, 16660, 37836, 82836, 175658, 362153,

    728159, 1431447, 2757188, 5212989, 9689739, 17730751, 31977358, 56899365,

    99981354, 173632536, 298236157, 506980545, 853456334, 1423520148,

    2353697410, 3859547524, 6279119909, 10139144298]
```

Weight multiplicities can be used to calculate fusion coefficients of Wess-Zumino-Witten models, which are critical for computation of multipoint correlation functions that determine physical quantities.

In the paper [3] following equation for fusion coefficient is derived.

$$
{}^{(k)}N_{\lambda,\mu}^{\nu} \;=\; \sum_{w\in W} (\det\, w)\, \mathrm{mult}\,(\mu; w\nu-\lambda)\ . \tag{24}
$$

Here $\mathrm{mult}\,(\mu; w\nu-\lambda)$ denotes multiplicity of weight $w\nu-\lambda$ in the module with highest weight $\mu$.

Branching coefficients of affine Lie algebras are used crucial for coset construction of new models from WZW-models (see chapter 18 in [9] for details).

Although the task of computation of branching coefficients by hand is tedious, it is easily done by the computer.

For example, for branching of module $L^{[1,0,0]}$ of $A_2$ ($sl_3$ for complex algebras and $su_3$ for real) to sub-algebra $A_1$ ($sl_2$ or $su_2$) we have:

```
> wg:=weights(A2);
         e2    2 e1    e3                   e2      e1    2 e3
 wg := [---- - ---- + ---- + lambda0, - ---- - ---- + ---- + lambda0, lambda0]
         3      3      3                    3       3      3

> tt1:=branching_rules(wg[-1],A2,A1,{e1=alpha[1]},15);
                                    tt1 := res

> tt1[lambda0];
```

```
             2       3       4        5        6        7        8        9        10
1 + q + 2 q  + 5 q  + 7 q  + 11 q  + 17 q  + 25 q  + 36 q  + 52 q  + 72 q

              11         12         13         14         15
     + 100 q    + 139 q    + 187 q    + 251 q    + 336 q

> tt1[lambda0+e1/2];
          2       3       4        5        6        7        8        9        10
2 q + 2 q  + 4 q  + 6 q  + 10 q  + 14 q  + 24 q  + 32 q  + 48 q  + 66 q

             11         12         13         14         15
     + 94 q    + 126 q    + 176 q    + 232 q    + 314 q
```

First several terms are in agreement with the example in section 14.7.2 of the book [9].

For branching of the same module to special sub-algebra $A_2^{(2)}$ we get

```
> tt1:=branching_rules(wg[-1],A2,tA2,{e1=alpha[1]/2},15);
                                  tt1 := res

> tt1[lambda0];
                  4       6       8       10       12       14
             1 + q  + 2 q  + 3 q  + 4 q   + 7 q   + 8 q

> tt1[lambda0+e1/2];
                                       e1
                                  res[---- + lambda0]
                                       2

> tt1[lambda0+e1];
                   3       5       7       9       11        13        15
            q + 2 q  + 2 q  + 4 q  + 5 q  + 8 q   + 11 q   + 16 q
```

We see that this result is in agreement with the results from the paper [10].

## 6 Conclusion

We described new algorithms for representations of affine Lie algebras and implementation of that algorithms.

This implementation is work in progress, current version can be downloaded from `http://anton.nazarov.name/affine`.

Further study is required for better performance of implementation. Although big performance gain can be achieved for computation of branching coefficient using the idea similar to folding of fan 2, used for efficient computation of string functions coefficients.

Also some additional algorithms for physical applications can be implemented, for example for computation of fusion coefficients of WZW-models (24). It is important to note the existence of “Kac” program [17] especially designed for computations in conformal field theory models. The possibility of interoperability with “Kac” program should be studied.

## References

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

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

[3] M. Walton, “Affine Kac-Moody algebras and the Wess-Zumino-Witten model,” `arXiv:hep-th/9911187`.

[4] E. Witten, “Non-abelian bosonization in two dimensions,” *Communications in Mathematical Physics* **92** (1984) no. 4, 455–472.

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

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

[7] R. Moody and J. Patera, “Fast recursion formula for weight multiplicities,” *AMERICAN MATHEMATICAL SOCIETY* **7** (1982) no. 1, .

[8] A. Nazarov, “Comparison of algorithms for construction of representations of lie algebras,” in *Physics and Progress*, Physics and Progress, SPbSU. 2008. `http://anton_nazarov.fatal.ru/static/articles/2008/physics-and-progress/article.pdf`.

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

[10] M. Ilyin, P. Kulish, and V. Lyakhovsky, "On a property of branching coefficients for affine Lie algebras," *Algebra i Analiz, to appear, arXiv* **812** , `arXiv:0812.2124 [math.RT]`.

[11] J. Stembridge, "A Maple package for symmetric functions," *Journal of Symbolic Computation* **20** (1995) no. 5-6, 755–758. `http://www.math.lsa.umich.edu/~jrs/maple.html`.

[12] M. Van Leeuwen, "LiE, a software package for Lie group computations," *Euromath Bull* **1** (1994) no. 2, 83–94. `http://www-math.univ-poitiers.fr/~maavl/LiE/`.

[13] T. Fischbacher, "Introducing LambdaTensor1. 0-A package for explicit symbolic and numeric Lie algebra and Lie group calculations," *Arxiv preprint hep-th/0208218* (2002) .

[14] V. Lyakhovsky, S. Melnikov, *et al.*, "Recursion relations and branching rules for simple Lie algebras," *Journal of Physics A-Mathematical and General* **29** (1996) no. 5, 1075–1088, `q-alg/9505006`.

[15] Lyakhovsky, "Recurrent properties of affine lie algebra representations." Supersymmetry and Quantum Symmetry, Dubna, August, 2007.

[16] P. Kulish and V. Lyakhovsky, "String Functions for Affine Lie Algebras Integrable Modules," *Symmetry, Integrability and Geometry: Methods and Applications* **4** , `arXiv:0812.2381 [math.RT]`.

[17] J. Fuchs, A. N. Schellekens, and C. Schweigert, "A matrix S for all simple current extensions," *Nucl. Phys.* **B473** (1996) 323–366, `arXiv:hep-th/9601078`. `http://www.nikhef.nl/~t58/kac.html`.

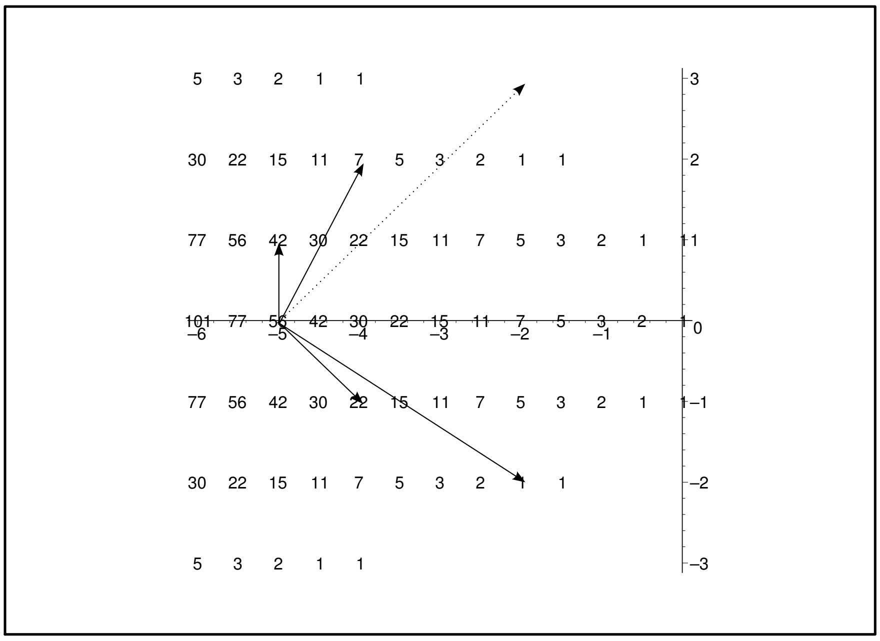

Figure 1: Computation of weight multiplicities of level 1 representation of algebra $A_1$