Affine.m – *Mathematica* package for Lie algebras of finite, affine and extended affine type

A A Nazarov[1,2]

[1] Theoretical Department, SPb State University

198904, Sankt-Petersburg, Russia

[2] Chebyshev Laboratory,

Department of Mathematics and Mechanics, SPb State University

199178, Saint-Petersburg, Russia

email: antonnaz@gmail.com

April 15, 2011

Abstract

We present *Mathematica* package for computations in the representation theory of Lie algebras of finite, affine and extended affine types.

Program summary

Title of program: Affine.m
Catalogue identifier:
Program obtainable from: (submitted to Computer Physics Communications)
`http://github.com/naa/Articles/AffineLieAlgebras/Mathematica/`
Reference in CPC to previous version:
Catalogue identifier of previous version:
Does the new version supersede the original program?:
Computers: Any computer running Mathematica
Operating systems under which the new version has been tested: Linux, Unix, Windows XP, MacOS
Programming language: Mathematica (version 7.0 or higher)

Memory required to execute with typical data: 20 Mbyte
No. of bits in a word: 64 or 32
No. of processors used: 1
No. of bytes in distributed program, including test data, etc.: 0.15 Mbyte
Distribution format: Unencoded compressed tar file
Nature of physical problem: TODO: Finite dimensional Lie algebras are important for physics. Affine for CFT/

Symmetry has been a very important fundamental principle under- lying human knowledge about our physical world. Among several mathematical formulations of symmetry, the Lie algebras and their corresponding Lie groups are probably the ones most explored. They were discovered by Sophus Lie and Wilhelm Killing during the last two decades of the 19th century. Lies work on Lie groups was inspired by Galois work in 1832 in which he discovered the nite groups. Independently, Killing had started a classication of Lie groups which was the starting point to the lie Cartans doctoral thesis in the beginning at the 20th century. Cartan was able to make a complete classication of Lie groups. Since Cartans classica- tion, the theory of Lie groups has been utilized in many branches of physics, including molecular physics, atomic physics, nuclear physics and particle physics.
Method of solution: The Butler-Portugal algorithm.
Restrictions on the complexity of the problem: Calculations up to finite grade
Typical running time: A few seconds with generic expressions

1 Introduction

Representation theory of finite, affine and extended affine Lie algebras is of central importance for different areas of mathematical and theoretical physics. Simple Lie algebras appear as the symmetries of classical and quantum systems, e.g. of gauge quantum field theories. Tensor product decomposition problem for simple Lie algebras appears in classification of particles, branching problem – in the study of symmetry breaking, for example in great unification models.

Example.

Major role of representation theory for the study of quantum and classical integrable systems is well known. Here, for example, tensor product decomposition is required for the computation of spectra and eigenstates of Hamiltonian. Quantum deformation of simple Lie algebras appears naturally in the study of integrable systems. It is possible to introduce q-analogues of multiplicities and branching coefficients.

Definition.

Example.

Affine Lie algebras, WZW and coset models. Denominator identity

Multiplicities, branching and tensor product problems are essentially different faces of the same problem. One possible solution is recurrent computation from corresponding generalization of Weyl character formula. Such a solution is applicable in many cases. For example if q is not root of unity category of $U_q(\mathfrak{g})$-modules is equivalent to the non-deformed case. Since Weyl character formula is preserved by q-deformation in this case, recurrent approach is applicable to quantum groups. etc.

General recurrent relations for branching coefficients of non-maximal embeddings were proposed in [1]. In [2] it was shown that these relations demonstrate the connection of branching problem with BGG resolution. Also it was noticed, that weight multiplicities and tensor product decomposition coefficients can be obtained in the special cases of Lie algebra embeddings ($\mathfrak{h} \subset \mathfrak{g}$ and $\mathfrak{g} \subset \mathfrak{g} \otimes \mathfrak{g}$ correspondingly).

Example.

In this paper we present *Mathematica* package for computations in representation theory of affine Lie algebras.

Main features:

- Root systems, fundamental weights, Weyl reflections for finite dimensional and affine semi-simple Lie algebras.
- Recurrent calculation of weight multiplicities and branching coefficients for irreducible representations.
- Construction of closed algebraic expressions for string and branching functions for low-rank affine Lie algebras.

For (1) we use fairly standard algorithms. For simple finite-dimensional Lie algebras these algorithms were already implemented in several software packages ([3], [4], [5], [6]). We have made them available to the users of popular computer algebra system *Mathematica*. Flexibility of *Mathematica* language has allowed us to represent Weyl group elements in the form convenient for simplification with rules [].

There exists well-known fast recursive formula for recurrent computation of weight multiplicities of Freudenthal [7]. It is used in most software packages ([3], [4], [5]). We present our own implementation. Also we present another recurrent computation routine based on general approach to branching [1007]. In case of finite-dimensional Lie algebras this procedure is equivalent to Racah formula, which is faster for weight multiplicities in case of low rank algebras and big representations. (See the table with run times).

Branching and tensor-product decomposition is carried out by the same recurrent process, which is implemented for arbitrary pair of algebra $\mathfrak{g}$ and subalgebra $\mathfrak{a}$.

For tensor product decompositions it is sometimes possible to give close algebraic answer for arbitrary tensor powers (see [Lyakhovsky, Postnova]). This formulae are also included in our package.

2 Core algorithms and data structures

Consider Lie algebra $\mathfrak{g}$ of finite or affine type. We denote its Cartan subalgebra by $\mathfrak{h}$ and root system by Δ. Roots $\alpha_i \in \Delta$ and weights $\mu, \nu \in \mathfrak{h}^*$ of the algebra are represented by the data-structures `finiteWeight` and `affineWeight` of *Affine.m* package. Root system can be reconstructed from the set of simple roots $\alpha_1, \ldots, \alpha_r$, where r is the rank of the algebra, by the action of Weyl group $W_{\mathfrak{g}}$.

The set of simple roots can be created using different methods. The coordinates of simple roots can be entered by hand, for example direct sum $sl_2 \oplus sl_2$ can be constructed by the code

```
makeFiniteRootSystem[{{1,0}, {0,1}}]
```

For simple finite-dimensional Lie algebras there are built-in constructors

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

```
Out[6]= finiteRootSystem[2, 2, {finiteWeight[2, {1, -1}], finiteWeight[2, {0, 1}]}]
```

It is possible to specify direct sum of algebras:

```
b2+b2
```

```
Out[6]= finiteRootSystem[4, 4, {finiteWeight[4, {1, -1, 0, 0}], finiteWeight[2, {0, 1}]}]
```

Weights can be specified by coordinates in standard basis or by Dynkin labels. The latter way is preferable.

```
makeFiniteWeight[{1,0}]
```

```
weight[b2][1,1]
```

```
weight[b2a][1,0,1]
```

Addition, multiplication by number and scalar product are defined for roots and weights:

```
makeFiniteWeight[{1,1}]+2*makeFiniteWeight[{1,-1}]
```

```
Out[7]=finiteWeight[2, {3, -1}]
```

It is possible to compute Cartan matrix or draw Dynkin diagram of the algebra or specify an algebra with Cartan matrix or Dynkin diagram. TODO

Action of Weyl group $W_{\mathfrak{g}}$ generated by simple reflections $s_\alpha, \alpha \in \Delta$ is important for computations. We denote by $C_{\mathfrak{g}} \subset \mathfrak{h}^*$ the main Weyl chamber (fundamental domain for the action of $W_{\mathfrak{g}}$) and its closure by $\bar{C}_{\mathfrak{g}}$. In many computations it is possible to work inside the main Weyl chamber due to Weyl symmetry.

Sometimes it can be helpful to consider Weyl group elements abstractly withouth the reference to root system. For example, there exists package for rule-based simplification of Weyl group elements based on finite automata [cite!]. We offer the function `weylGroupElement[x__Integer][rs_?rootSystemQ]` which produces the object, representing element of Weyl group encoded by the set of reflections, that can act upon the weights. It can be used as follows:

```
<<simplification.m;
(weylGroupElement @@ simplify[s[1,2,3,1,3,1,2,1,4,2,1]])[b2+b2][makeFiniteWeight[{1,1,0,0}]]
```

```
Out[7]= finiteWeight[4, {-1, 1, 0, 0}]
```

Following functions can be used to compute objects related to Weyl group and its action on roots and weights

- $\forall \mu \in \mathfrak{h}^* \exists \lambda \in \bar{C}_{\mathfrak{g}}, w \in W_{\mathfrak{g}} : \mu = w\lambda$, the function `toFundamentalChamber` can be used to find such λ.
- Orbit of weight under the action of Weyl group is computed with the function `orbit`, the elements of the orbit are grouped by the length of Weyl group element.

2.1 Representation theory

Weights of the irreducible highest weight representation are calculated using the function `weightSystem`

```
weightSystem[b2][fundamentalWeights[b2][[1]]*2]
```

3 Examples

3.1 Finite-dimensional Lie algebras

Consider the algebra B_4 and regular subalgebra B_2. This embedding can be specified by the following code:

$$B_2 \subset B_4 \tag{1}$$

Branching coefficients can be calculated using the formula () with the code:

$$calculate_{branching} \tag{2}$$

It is interesting to note that the subalgebra $\mathfrak{a}_\perp$ is equal to B_2 in this case and not to A_1 as it can be thought from the Dynkin diagram. To visualize the diagrams and branching coefficients the following instructions were used:

$$plot \tag{3}$$

4 Conclusion

References

[1] V. Lyakhovsky and A. Nazarov, "Recursive algorithm and branching for nonmaximal embeddings," *Journal of Physics A: Mathematical and Theoretical* **to appear** , `arXiv:1007.0318 [math.RT]`.

[2] V. Lyakhovsky and A. Nazarov, "Recursive properties of branching and BGG resolution," `arXiv:1102.1702 [math.RT]`.

[3] 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/`.

[4] J. Stembridge, "Computational aspects of root systems, Coxeter groups, and Weyl characters, Interaction of combinatorics and representation theory, MSJ Mem., vol. 11," *Math. Soc. Japan, Tokyo* (2001) 1–38.

[5] T. Fischbacher, "Introducing LambdaTensor1. 0-A package for explicit symbolic and numeric Lie algebra and Lie group calculations," `hep-th/0208218`.

[6] T. Nutma, "Simplie." `http://code.google.com/p/simplie/`.

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