A Maple Package for Root Systems and Finite Coxeter Groups

Version 2.4[*]
February 28, 2004

John R. Stembridge[†]
Department of Mathematics
University of Michigan
Ann Arbor, Michigan 48109–1109 USA

email: jrs@umich.edu
Web: www.math.lsa.umich.edu/~jrs

Contents

1. Introduction

This article provides an introduction to Version 2.4 of `coxeter` and `weyl`, two packages of 50 Maple programs that provide assistance in the study of root systems, finite Coxeter groups, and Weyl characters. The `coxeter` package contains 38 basic procedures for manipulating roots, vectors and reduced expressions, for generating permutation representations and irreducible characters of Coxeter groups, and for retrieving the type of information one finds in the appendices of Bourbaki [**1**]. The `weyl` package is a supplement to `coxeter` containing 12 procedures for manipulating weights and characters of irreducible representations of semisimple Lie algebras (i.e., Weyl characters). It is capable of reproducing the book of tables by Bremner, Moody and Patera [**2**].

[*] Version 2.4 is the Alexandre-Théophile Vandermonde Birthday Edition.

[†] This work supported by NSF grants DMS–0070685 and DMS–0245385.

I do not claim that these packages will answer every question one might want to ask about a root system, Coxeter group, or Weyl character. However, even if it doesn't do exactly what you want, there is a good chance you can build the procedures you need out of the raw materials provided here. Alternatives include the *LiE* package of van Leeuwen, Cohen and Lisser (*et. al.*) [**6**], the *CHEVIE* package of Meinolf Geck (*et. al.*) [**7**], and the *Schur* package of Brian Wybourne [**8**].

The main improvements in the recent versions of `coxeter` and `weyl` are:

- Several new functions have been added to `weyl`, including three algorithms for decomposing products of Weyl characters (superseding the previously incomplete implementation of the `qtensor` algorithm in Version 2.1), two algorithms for restricting Weyl characters to reductive Lie subalgebras, and two functions `toM` and `toX` for manipulating expressions in the character ring of a semisimple Lie algebra.
- Several new examples have been added or improved, including procedures for computing "fake degrees" of finite reflection groups, for generating the crystal graphs of representations of quantum (Lie) algebras, and for traversing the elements of a Coxeter group or any of its parabolic quotients.
- A "Vanilla Edition" of the packages is now available. The Vanilla Edition should work with any OS that runs Maple.
- The package is now compatible with Maple 8 and Maple 9.

The main difference between Versions 2.4 and 2.3 involves a tweak to enable use of Maple 8 and 9; the only changes between 2.3 and 2.2 involve minor bug fixes, not functionality. For a detailed log of the changes and new features in each version, see §8.

I still get reports from sites that are using obsolete versions of these packages, including the versions in the Maple Share Library. These sites should upgrade to take advantage of the vast improvements that have been added since the release of the 1.x versions.

2. System Requirements

First, you must have Maple V, or Maple 6, 7, 8, or 9 ⟨`www.maplesoft.com`⟩.

The packages have been developed and tested on various Unix machines running Releases 3, 4, and 5 of Maple V and Maple 6, 7, 8, and 9. While I no longer have access to machines running the first releases of Maple V, the packages are designed so that they *should* be compatible with these versions as well.[1]

Version 2.4 of `coxeter` and `weyl` is packaged in two formats: one is a Unix `tar` file (the "Unix Edition"), and the second is a collection of plain text files that comprise the Vanilla Edition. Both editions are available for download from

⟨`http://www.math.lsa.umich.edu/~jrs/maple.html`⟩.

Users of Maple on Unix platforms have the choice of installing either the Unix or Vanilla Editions; the (minor) differences are discussed in §3. For everyone else, the Vanilla Edition is the only choice. In particular, users of the special edition packaged for the Macintosh in older versions of the packages will need to migrate to the Vanilla Edition.

[1] Congratulations to anyone who still has access to a working copy of Maple V Release 1 or 2.

3. Getting Started

Special Notes for the Unix Edition. All of the information needed to install the packages is provided in an accompanying `READ_ME` file. However, if a System Administrator has done the installation for you, and you aren't already a user of one of my packages, then you will need to find out the name of the directory where the packages have been stored and edit an initialization file.

For the purposes of this discussion, let us assume that the packages have been stored in the directory `/usr/local/maple/packages`. You can confirm this by verifying that `/usr/local/maple/packages` has subdirectories named `coxeter` and `weyl`. Next, you will need to create a file named `.mapleinit` in your home directory. Or add to it, if you already have one. In this file you should insert the following two lines:

```
HomeLib:=`/usr/local/maple/packages`:
libname:=libname, HomeLib:
```

Be careful to correctly type both of the left quote-marks (`). Each time Maple is invoked, these statements will define where the packages are located. (For Maple V Release 1, the syntax is slightly different—see the notes below.)

You are now ready to run the packages. Simply enter the commands `with(coxeter)` or `with(weyl)` during a Maple session. Although `weyl` internally uses many of the procedures in `coxeter`, it is not necessary to explicitly load `coxeter` to use `weyl`. To access documentation about an individual function, use `?coxeter,`*function* or `?weyl,`*function*. The source code is located in

`/usr/local/maple/packages/coxeter/src`,
`/usr/local/maple/packages/weyl/src`,

the library of data for the exceptional root systems is located in

`/usr/local/maple/packages/coxeter/lib`,

and some additional files of examples and applications may be found in

`/usr/local/maple/packages/coxeter/examples`.

Special Notes Regarding Maple V Release 1. The syntax for specifying the locations of external libraries is slightly different in Maple V Release 1. The `.mapleinit` file should instead have the lines

```
HomeLib:=`/usr/local/maple/packages`:
_liblist:=[HomeLib]:
```

Be careful to correctly type the left quote-marks (`) and underscore character (_).

Also, when the Unix Edition of `coxeter` and `weyl` is installed with Release 1 of Maple V, the help texts are not integrated into the package. Instead, users should consult the HTML version of the help texts provided with the Vanilla Edition. (The mechanism for loading help for external functions is so primitive in Release 1 that it no longer makes sense to support it.)

Special Notes for the Vanilla Edition. The Vanilla Edition consists primarily of a single text file that may be loaded into a Maple session via the `read` command. It loads all of the

package functions in `coxeter` and `weyl` and the entire contents of the library (including conjugacy class data and character tables for the exceptional groups) into the workspace, whereas the Unix Edition keeps clutter to a minimum by loading functions and library elements only at the moment they are first used.

When the Vanilla Edition is first loaded, the package functions are initially useable only via the long forms

`coxeter[`*function*`](`*arguments*`)` and `weyl[`*function*`](`*arguments*`)`.

Although `coxeter` and `weyl` are not recognized by the Maple `with` command, the Vanilla Edition provides two special functions `withcoxeter` and `withweyl` so that the commands `withcoxeter()` and `withweyl()` enable the use of the short forms

function`(`*arguments*`)`.

Similarly, `withcoxeter(`*arguments*`)` and `withweyl(`*arguments*`)` enable the short forms for some subset of the package, modeling the expected behavior of `with(coxeter,`*arguments*`)` and `with(weyl,`*arguments*`)`.

Documentation for the individual functions is not loaded when the package is loaded. Instead there is a separate HTML file containing all of the help texts cross-referenced with hyperlinks. Similarly, the suite of examples is provided as a directory of plain text files available for download.

The remainder of this article presents an overview of the workings of `coxeter` and `weyl`, defines the fundamental objects of study, and explains the design of the data structures that model these objects. A one-line description of the purpose of each procedure and example is provided in §6. For more detailed information about the individual commands, consult the on-line help. For more information about some of the key algorithms used by the packages, see [**5**].

4. Definitions

We begin with some of the basic definitions in the theory of (finite) root systems and Coxeter groups. For more details, the standard reference is Bourbaki [**1**]. I also highly recommend the books by Humphreys on Lie algebras [**3**] and reflection groups and Coxeter groups [**4**]. In the following, we will need a real vector space with a symmetric, positive definite inner product $\langle \cdot , \cdot \rangle$ (i.e., a real Euclidean space). For simplicity, let us use $\mathbf{R}^d$ together with its standard inner product.

Root systems. Given a nonzero $\alpha \in \mathbf{R}^d$, the reflection corresponding to α is the linear transformation $\sigma_\alpha : \mathbf{R}^d \to \mathbf{R}^d$ defined by $\sigma_\alpha(\beta) = \beta - \langle \beta, \alpha^\vee \rangle \alpha$, where $\alpha^\vee := 2\alpha / \langle \alpha, \alpha \rangle$. A (reduced) *root system* Φ is a finite subset of $\mathbf{R}^d - \{0\}$ (whose members are called *roots*) satisfying the following pair of axioms:

1. For $\alpha \in \Phi$, $\Phi \cap \mathbf{R}\alpha = \{\pm\alpha\}$.
2. For $\alpha \in \Phi$, σ_α permutes Φ.

If Φ also satisfies

3. $\langle \alpha, \beta^\vee \rangle \in \mathbf{Z}$ for all $\alpha, \beta \in \Phi$,

then the root system is said to be *crystallographic*. In this case, the roots generate a lattice, which is a valuable property from the point of view of explicit computation, since it implies the existence of a basis relative to which the roots have integer coordinates.

The *rank* of Φ is the dimension of its linear span.

The classification(s). A root system Φ is irreducible if it cannot be partitioned into two mutually orthogonal subsets. It is easily verified that the blocks of any orthogonal partition are themselves root systems, so every root system has a (unique) orthogonal decomposition into irreducible root systems.

The classification of irreducible root systems is well-known. However, implicit in the classification is that we agree on the answer to the question of when two root systems are isomorphic.

The "correct" notion of isomorphism depends on whether the context is crystallographic or not. For general root systems Φ and Φ', a reasonable definition of isomorphism is that there should be a bijection $\alpha \mapsto \alpha'$ such that the angle between $\alpha, \beta \in \Phi$ is the same as the angle between $\alpha', \beta' \in \Phi'$. If all roots have the same length (a hypothesis that can always be imposed but is not always convenient), this is equivalent to the existence of an isometry that maps Φ to Φ'. For crystallographic root systems, we need a more restrictive notion of isomorphism—there should be a bijection $\alpha \mapsto \alpha'$ such that $\langle \alpha, \beta^\vee \rangle = \langle \alpha', \beta'^\vee \rangle$. A given root system may not be isomorphic (in the general sense) to any crystallographic root system, or it may be isomorphic to two crystallographic root systems that are distinct (in the crystallographic sense).

Thus there are really two classifications: the isomorphism classes of general and crystallographic irreducible root systems. In the crystallographic case, we have the familiar A_n ($n \geq 1$), B_n ($n \geq 2$), C_n ($n \geq 3$), D_n ($n \geq 4$), E_6, E_7, E_8, F_4, and G_2 (subscripts indicate rank). In the general case, we have A_n ($n \geq 1$), B_n ($n \geq 2$), D_n ($n \geq 4$), E_6, E_7, E_8, F_4, H_3, H_4, and $I_2(m)$ ($m \geq 5$). It would be more precise to have (for example) two names for B_n, one for a crystallographic context and one for a general context, but tradition dictates otherwise.

Bases. Every root system has a *base*: a linearly independent set of roots $\Delta = \{\alpha_1, \ldots, \alpha_n\}$ with the property that every root α is a linear combination $\sum_i c_i \alpha_i$ with every $c_i \geq 0$ or every $c_i \leq 0$. (If Φ is crystallographic then every c_i is an integer, and conversely.) A root is said to be *positive* or *negative* (with respect to Δ) according to the signs of the base coordinates. The base roots are also referred to as *simple roots*.

The base generates the root system in the sense that Φ is the smallest root system containing Δ. Furthermore, all bases of Φ are equivalent, for if Δ and Δ' are bases of isomorphic root systems Φ and Φ' (in either sense of isomorphism), then there is an isomorphism that maps Δ to Δ'.

In the crystallographic case, it follows from the fact that Δ generates Φ that the Cartan integers $\langle \alpha, \beta^\vee \rangle$ for $\alpha, \beta \in \Phi$ are completed determined by the Cartan integers involving the simple roots. In other words, the isomorphism class of a crystallographic root system depends only on the so-called *Cartan matrix* $[\langle \alpha_i, \alpha_j^\vee \rangle]$.

Similarly, the fact that Δ generates Φ also implies that the isomorphism class of Φ (in the sense of isomorphism of general root systems) depends only on the matrix of angles between pairs of simple roots.

Coxeter groups. The reflections σ_α for $\alpha \in \Phi$ generate a finite group $W(\Phi)$ of isometries of $\mathbf{R}^d$ known as a *reflection group*. Moreover, the reflections $\sigma_1, \ldots, \sigma_n$ corresponding to the simple roots $\alpha_1, \ldots, \alpha_n$ (the so-called *simple reflections*) suffice to generate $W(\Phi)$. The

reflection groups corresponding to crystallographic root systems are the Weyl groups of semisimple Lie algebras. In general, reflection groups turn out to be precisely the finite *Coxeter groups*; i.e., the finite groups having presentations of the form

$$\langle s_1, \ldots, s_n : (s_i s_j)^{m_{ij}} = 1 \rangle,$$

where the m_{ij} are positive integers such that (1) $m_{ij} = m_{ji}$, and (2) $m_{ij} = 1$ iff $i = j$.

The symmetric matrix $[m_{ij}]$, known as the *Coxeter matrix*, completely determines the reflection group $W(\Phi)$ up to isomorphism. It also determines the isomorphism class of Φ (in the general, not crystallographic sense), since $(1 - 1/m_{ij})\pi$ is the angle between the simple roots α_i and α_j.

It is often convenient to regard elements of a Coxeter group W as words in the alphabet of the generators $\{s_1, \ldots, s_n\}$. A word $w = s_{i_1} \cdots s_{i_l}$ is said to be *reduced* if the length l is as small as possible among all such representations of a given element $w \in W$.

Weights. In a crystallographic root system Φ with simple roots $\alpha_1, \ldots, \alpha_n$, an *integral weight* (or more briefly, a *weight*) is a vector λ with the property that the quantities $m_i = \langle \lambda, \alpha_i^\vee \rangle$ $(1 \leq i \leq n)$ are integers. The weight λ is *dominant* if $m_1, \ldots, m_n \geq 0$. The Weyl group $W(\Phi)$ permutes the set of weights, and each $W(\Phi)$-orbit of weights has a unique dominant member.

There are unique weights $\omega_1, \ldots, \omega_n \in \operatorname{Span} \Phi$ satisfying $\langle \omega_i, \alpha_j^\vee \rangle = \delta_{ij}$; these are the so-called *fundamental weights*. Every weight in $\operatorname{Span} \Phi$ is an integer linear combination of the fundamental weights; the dominant ones are those in the nonnegative span.

There is a semisimple Lie algebra $L = L(\Phi)$ associated to Φ (unique up to isomorphism), and for each dominant integral weight $\lambda \in \operatorname{Span} \Phi$, there is an irreducible finite-dimensional L-representation V_λ. Moreover, the V_λ's form a complete set of irreducible, non-isomorphic irreducible representations of L.

Orbit sums and Weyl characters. Given a semisimple Lie algebra L with (crystallographic) root system Φ and Weyl group $W = W(\Phi)$, the characters of representations of L may be regarded as elements of a ring R with a $\mathbf{Z}$-basis consisting of formal exponentials e^ν, where ν ranges over integral weights in $\operatorname{Span} \Phi$, subject to the rules $e^\mu \cdot e^\nu = e^{\mu+\nu}$. Moreover, for each dominant λ, the character $\chi(\lambda)$ of V_λ is given by the Weyl Character Formula; viz.,

$$\chi(\lambda) = \Sigma(\lambda + \rho)/\Sigma(\rho),$$

where $\Sigma(\nu) = \sum_{w \in W} \operatorname{sgn}(w) e^{w\nu}$ and 2ρ denotes the sum of all positive roots.

We refer to $\chi(\lambda)$ as a *Weyl character*.

It is easily shown that the Weyl characters $\chi(\lambda)$ form a $\mathbf{Z}$-basis for the W-invariant part of R (the W-action is inherited from the W-action on weights). On the other hand, there is another natural $\mathbf{Z}$-basis for the W-invariant part of R formed by the *orbit sums*

$$m(\lambda) := \sum_{\nu \in W\lambda} e^\nu = \frac{1}{N_\lambda} \sum_{w \in W} e^{w\lambda},$$

where λ ranges over dominant integral weights in $\operatorname{Span} \Phi$ and $N_\lambda = |\{w \in W : w\lambda = \lambda\}|$ denotes the order of the stabilizer of λ.

5. Data Structures

Let me now explain (and in some cases justify) the choices I have made in modeling root systems and Coxeter groups in `coxeter` and `weyl`.

Vectors. Perhaps the most important decision one has to make is how to represent vectors. Many of the efficient algorithms in this subject rely on the geometry of reflections acting on vectors, and the innermost core of many of the procedures in `coxeter` and `weyl` amount to computations of inner products and sums of vectors.

In these packages, a *vector* is defined to be a linear combination of a standard orthonormal basis `e1,e2,e3,...`.[2] The coefficients should be rational or floating-point. Thus `2*e2-e4/3` is a vector, but `sqrt(5)*e1+x*e2` is not. One consequence of this is that `coxeter` and `weyl` have reserved `e1,e2,e3,...` as global variables, and so these should not be assigned values by the user.

There are three reasons for treating vectors this way. First, the basic operations of addition and scalar multiplication are now trivial and fast for Maple. If `u` and `v` are vectors and `c` is a scalar then `u+v` is a vector sum and `c*v` is a scalar multiplication. Second, one frequently needs to compute inner products `iprod(r,v)`, where `r` is a root and `v` is arbitrary. Most root systems can be constructed so that the roots have relatively few nonzero coordinates. In fact, most roots have only two nonzero coordinates. (The worst offenders are the root systems of type E, which in their usual construction have some roots with 8 nonzero coordinates.) With this method of representing vectors, we can take advantage of the "sparseness" of the roots so that the cost of computing an inner product with `r` is proportional to the number of nonzero coordinates in `r`. Third and last is the fact that when one sees an expression such as `2*e1-e2-e3`, there is no ambiguity as to what it means—it is simply a vector in standard coordinates. A list of coordinates conveys less information and can lead to confusion when changes of basis occur.

Root system names. The following names are reserved for the irreducible root systems:

```
A1, A2, A3,...; B1, B2, B3,...; C1, C2, C3,...; D2, D3, D4,...;
E3, E4, E5, E6, E7, E8, F4, G2, H3, H4, I2[2], I2[3], I2[4],...
```

In particular `coxeter` and `weyl` use these as global variables, so they should not be assigned values by the user. To name a reducible root system, one merely forms a monomial out of the names of the irreducible components. Thus `A1*A1*C4` specifies a root system of rank 6 whose irreducible components consist of two copies of A_1 and one copy of C_4. The integer `1` is used to designate the empty root system of rank 0.

Note that although the above names all specify well-defined root systems, there are several isomorphisms among them (in some cases depending on whether the context is crystallographic), and not all of them are irreducible. The redundancy is for the convenience of the user.

The root system data structure. Most of the procedures of `coxeter` and `weyl` include the specification of a root system as all or part of the input. This specification may given

[2] This is not the same as the vector data structure in Maple's `linalg` package.

in any of four ways: by name (see above), by a list of base vectors (e.g., `[e2-e1,e3-e2]` specifies a root system isomorphic to A_2), by a Coxeter matrix, or by a Cartan matrix. Furthermore, the procedures

`base, cartan_matrix, cox_matrix, name_of`

may be used to convert among these four ways of specifying a root system. For example, if `R` is any of the four forms of a root system data structure, then `base(R)` returns a list of simple roots for some realization of `R`, and `name_of(R)` returns the name of the isomorphism class of `R`.

The fact that we have two notions of isomorphism creates some complications. For example, if `R` is a Coxeter matrix, then there may either be no crystallographic root system with this Coxeter matrix, or more than one. In such cases, `cartan_matrix(R)` will generate either an error message (in the former case), or choose one of the Cartan matrices (in the latter case).

Another problem is in determining the `name_of` a root system. As discussed in §4, it would be preferable to have a distinct name for a "not necessarily crystallographic" root system of type (say) B_3. For if `R` is a list of base vectors, and `name_of(R)` returns `B3` as the result, then `R` may or may not be a crystallographic base. In rank two, we do have two names for the ambiguous cases: `name_of(R)` returns `B2` or `G2` in most (appropriate) cases; the only inputs for which it returns the names `I2[4]` or `I2[6]` involve lists of base vectors with floating-point coordinates.

Group elements. In `coxeter`, elements of a reflection (or Coxeter) group W are represented primarily as words in the simple reflections, and to a lesser extent as permutations.

If $\sigma_1, \ldots, \sigma_n$ are the simple reflections, the group element (or word) $w = \sigma_{i_1} \cdots \sigma_{i_l}$ is represented in `coxeter` as the list `[i_1,i_2,...,i_l]`. In particular, `[]` represents the identity element. It is easy to multiply or invert words. For example, if `w1` and `w2` are two words, then `[op(w1),op(w2)]` is their product, and the inverse of `w` is expressible as `[seq(w[-i],i=-nops(w)..-1)]`. The function `reduce` may be used to decide if two words represent the same group element.

There is a compact way to label the elements of a reflection group, based on the fact that every vector v in general position (not orthogonal to any of the roots) has a trivial stabilizer. Given such a vector v, there is a one-to-one correspondence between elements in the W-orbit of v and W itself; i.e., one may use the vector $w.v$ as a "label" for w. This trick is used by some of the procedures in `coxeter`, and you may find it useful if you write your own programs. In `coxeter`, one can do this by using the function `interior_pt` to produce a vector `v` in the interior of the fundamental chamber (i.e., on the positive side of the basic hyperplanes). Then, given a word `w`, one can use `reflect` to obtain the vector `u` that serves to label `w`. To invert this procedure and recover the word labeled by the vector `u`, use `vec2fc`. Care must be exercised when floating point coordinates are used, since two labeling vectors must be presumed equal if the distance separating them is small.

There are also facilities in `coxeter` for working with group elements represented as permutations. The main tools for this purpose are

`multperm, perm2word, perm_rep, stab_chain`.

The function `perm_rep` produces the permutation representation of a reflection group acting on the cosets of a parabolic subgroup. In the irreducible case, it is faithful if and

only if the subgroup is proper. The format of the output is compatible with Maple's `group` package, but can also produce output formatted for use with the *GAP* system [**9**]. The functions `multperm` and `perm2word` may be used to convert back-and-forth between the word representation of a group element and a permutation.

A possible point of confusion is that in `coxeter`, reflections act on the left but permutations act on the right. The latter is necessary to maintain compatibility with Maple's `group` package.

Weights. In keeping with the vector philosophy described earlier, an integral weight, according to `weyl`, is just another vector. It is up to the user to be sure that it has integer coordinates relative to the fundamental weights. Using the tools provided in `weyl`, this is quite easy to do. For example, to compute weight multiplicities or tensor product multiplicities for $sp(8)$ (the Lie algebra of C_4), one could use the `weights` function to produce a list of the fundamental weights for C_4, and then take integer combinations of this list of vectors to create the desired weights; e.g., after loading `weyl`,

```
w:=weights(C4);
weight_mults(2*w[1]+w[3],C4);
tensor(2*w[2],w[1]+w[2],C4);
```

The `weight_coords` function computes the coordinates of a vector with respect to the fundamental weights.

Orbit sums and Weyl characters. Several of the procedures in `weyl`, notably `branch`, `tensor`, `toM`, `toX`, and `weight_mults` produce as output, or accept as input, elements of the character ring of some semisimple Lie algebra $L = L(\Phi)$. The package recognizes the orbit sums $m(\lambda)$ and Weyl characters $\chi(\lambda)$ (see the discussion in §4) as primitive objects in this ring; more complicated elements may be constructed by taking sums and products of the primitive elements.

The orbit sum $m(\lambda)$ is represented as a Maple expression of the form `M[a,b,...,c]`, where `[a,b,...,c]` denotes the fundamental weight coordinates of λ; similarly, the Weyl character $\chi(\lambda)$ is represented as a Maple expression of the form `X[a,b,...,c]`. Thus for example, the command

```
toX(X[1,0]*M[0,1]^2, G2);
```

would convert $\chi(\omega_1)m(\omega_2)^2$ into a linear combination of Weyl characters for G_2.

6. Short Synopses of the Procedures

Here is a brief indication of the purpose of each function in `coxeter` and `weyl`. For the full details, including syntax, definitions and examples, consult the on-line help via the commands `?coxeter[`*function*`]` and `?weyl[`*function*`]`, or use the HTML document provided in the Vanilla Edition.

The `coxeter` package.

`base`	simple roots of a root system
`cartan_matrix`	Cartan matrix of a crystallographic root system
`char_poly`	characteristic polynomial of a group element
`class_rep`	determine conjugacy class representatives

`class_size`	determine sizes of conjugacy classes
`co_base`	simple co-roots of a root system
`cox_matrix`	Coxeter matrix of a root system or Coxeter group
`cox_number`	Coxeter number of a Coxeter group
`cprod`	inner product of characters or class functions
`degrees`	degrees of the basic polynomial invariants
`diagram`	Dynkin diagram of a root system
`exponents`	exponents of a root system
`highest_root`	highest root in an irreducible root system
`index`	index of connection for a crystallographic root system
`induce`	induce characters from reflection subgroups
`interior_pt`	find an interior point of the fundamental chamber
`iprod`	inner product of vectors
`irr_chars`	irreducible characters of a Coxeter group
`length_gf`	length generating function for a Coxeter group
`longest_elt`	longest element of a Coxeter group
`multperm`	multiply permutations
`name_of`	name of a root system or Coxeter group
`num_refl`	number of reflections in a Coxeter group
`orbit`	orbit of a vector under the action of a reflection group
`orbit_size`	size of the orbit of a vector under the action of a reflection group
`perm2word`	convert a permutation to a reduced word
`perm_char`	permutation character induced by a reflection subgroup
`perm_rep`	permutation representation of a Coxeter group
`pos_roots`	positive roots of a root system
`presentation`	generators and relations for a Coxeter group
`rank`	rank of a root system or Coxeter group
`reduce`	find a reduced expression for a group element
`reflect`	apply reflections to a vector
`restrict`	restrict characters to reflection subgroups
`root_coords`	coordinates of a vector with respect to the simple roots
`size`	size of a Coxeter group
`stab_chain`	stabilizer chain for a Coxeter group
`vec2fc`	map a vector to the fundamental chamber by reflections

The `weyl` package.

`branch`	restrict representations to reductive subalgebras
`co_rho`	half the sum of the positive co-roots
`minuscule`	list minuscule and quasi-minuscule weights
`rho`	half the sum of the positive roots
`tensor`	decompose tensor products of representations
`toM`	convert polynomial expressions into orbit sums
`toX`	convert polynomial expressions into Weyl character sums
`weight_coords`	coordinates of a vector with respect to the fundamental weights
`weight_mults`	weight multiplicities in irreducible representations

`weight_sys`	dominant weights below a given dominant weight
`weights`	fundamental weights of a (crystallographic) root system
`weyl_dim`	dimension of an irreducible representation

Here is a short summary of the contents of the `examples` directory. These files must be explicitly loaded by the user and contain both documentation and source code.

`bruhat_order`	Bruhat ordering of a finite Coxeter group
`coset_reps`	minimum-length coset representatives for a parabolic subgroup
`ct_redex`	count reduced expressions for a group element
`dcoset_reps`	minimum-length double coset representatives for parabolic subgroups
`descent_gf`	descent generating function for a Coxeter group
`fake_degrees`	fake degrees of a finite reflection group
`list_weights`	list dominant weights in various sorted orders
`poincare`	Poincaré series (length g.f.) for *arbitrary* Coxeter groups
`qmult`	q-analogue of weight multiplicities in representations
`redex`	list all reduced expressions for a group element
`refl_rep`	matrices for the reflection representation of a Coxeter group
`root_poset`	the partial order of positive roots
`traverse`	iterate over the elements of a Coxeter group or parabolic quotient
`weak_order`	weak ordering (and Cayley graph) of a Coxeter group
`weight_order`	the partial order of dominant weights
`xtal_graph`	generate the crystal graph of a representation

Several of the examples use the `posets` package[3] for visualization.

7. The Non-Crystallographic Kludge

The non-crystallographic root systems pose a special challenge for machine computations. The crux of the problem is that in these cases, there is no lattice containing the roots. Thus any computation involving the roots cannot avoid vectors with irrational coordinates. In the cases of H_3 and H_4, the minimum extension of $\mathbf{Q}$ required is $\mathbf{Q}[\sqrt{5}]$, and for the dihedral cases $I_2(m)$, the most convenient extension is $\mathbf{Q}[\cos(\pi/m), \sin(\pi/m)]$.[4]

Maple does have facilities for working with algebraic extensions of $\mathbf{Q}$, so in principle it would be possible to write the core procedures of `coxeter` so that vectors with algebraic coefficients are correctly processed. However, in order to be able to recognize when two vectors are equal (for example), it would be necessary to frequently convert vectors to a normal form. This would impose a severe time penalty for `coxeter` calculations.

Sacrificing precision for speed, `coxeter` is designed to use floating-point coordinates for non-crystallographic root systems. This introduces a few unpleasant side effects, but these are largely hidden from the user. The main problems occur in deciding when two vectors are equal or orthogonal, or if a vector is on the positive side of a hyperplane. For this purpose, `coxeter` has a global variable

```
`coxeter/default`[epsilon]
```

[3] This package is available for download at ⟨`http://www.math.lsa.umich.edu/~jrs/maple.html`⟩.

[4] This is not necessarily the smallest extension; e.g., $I_2(5)$ is constructible over $\mathbf{Q}[\sqrt{5}]$.

whose value is used to resolve these problems. That is, inner products and distances[5] that are less in absolute value than `'coxeter/default'[epsilon]` are assumed to be zero. When the `coxeter` package is loaded, this parameter is assigned the value `0.001`. It may be subsequently modified by the user. Obviously this approach is not foolproof (computations involving $I_2(1000)$ would be problematic), but it works well in practice and it is unlikely that the user will ever need to change the default value.

8. Changelog

Version 2.4. There were no changes in functionality.

- Minimal internal changes were introduced for compatibility with Maple 8 and 9.
- The example `dcoset_reps` was rewritten to take advantage of a new (faster, more space-efficient) algorithm for generating double coset representatives.

Version 2.3. No changes in functionality.

- A bug in `stab_chain` (present since 2.0) has been fixed. This bug could be exposed only for certain non-default permutation representations. Thanks to Kenneth Gray of the University of Western Ontario for spotting this.
- A superficial bug in the function `perm_rep` has been fixed—when exporting to 'gap' format, the group name is once again printed correctly.
- An error in the documentation for `minuscule` has been fixed—this function is intended to generate only those minuscule weights that are *fundamental* weights.
- The formatting of the HTML version of the help texts has been improved—the symbols '`<`' and '`>`' should now display correctly in all browsers. Thanks to Brendan McKay of Australian National University for bringing this to my attention.

Version 2.2. In addition to the debut of the Vanilla Edition, the significant changes and new features in `coxeter` and `weyl` are:

- The `qtensor` function has been replaced with the more powerful `tensor`. The original `qtensor` was only a partial implementation of the algorithm described in §7 of [**5**]; the new `tensor` function provides a complete implementation of three tensor product algorithms, including `qtensor`.
- There are five new functions: `branch`, `co_base`, `minuscule`, `toM`, and `toX`.
- Two new examples have been added: `fake_degrees` and `xtal_graph`.
- The example `dim_list` has been replaced with the more versatile `list_weights`.
- The `traverse` example has been redesigned, and now supports searching through arbitrary parabolic quotients and Coxeter group orbits.
- The function `perm_rep` now generates permutation representations of Coxeter groups relative to any parabolic subgroup.
- The name `export` is now a Maple keyword, so the `export` flag used by `perm_rep` and `presentation` has been renamed `gap`.
- The function `multperm` now supports additional input formats.
- The output of `weight_sys` is now sorted by height.

[5] Starting with Version 2.2, most of these distances are computed using the (cheaper) L^1-norm.

- The function `name_of` now returns the names `I2[4]` and `I2[6]` only when the input data is a list of simple roots with floating-point coordinates.
- Bugs were fixed in the example `refl_rep` (a minor issue related to the trivial root system), in `root_coords` (triggered by unlikely floating-point data), and in `name_of` (triggered only by certain inputs in the form of Cartan matrices).

Version 2.1. There were no changes in functionality in Version 2.1.

Version 2.0. Aside from improvements in the documentation, the major changes that affect users upgrading to Version 2.0 are:

- The function `descent_gf` has been moved from the `coxeter` package to the `examples` directory, and has been improved.
- The `store` function has been removed from `weyl`. It became superfluous with the improved algorithms in Version 2.0.
- The 120 point (unfaithful) permutation representation of $W(E_8)$ is no longer available. Also, permutation representations for the exceptional groups have been deleted from the library and are now generated on demand.
- The conjugacy class representatives produced by `class_rep`, and the order they appear in, has changed in some cases. In all cases, the representative produced now has minimum length. (In $W(E_8)$ there are a few classes for which I have not verified minimality, but the representatives are "probably" minimal.)
- The Coxeter matrix and Cartan matrix of the empty root system is now represented as the empty list `[]` rather than `NULL`.
- The contents of `weyl/examples` have been moved to `coxeter/examples`.

9. Miscellany

Global warning. The following names are reserved for use by `coxeter` and `weyl`:

1. `e1,e2,e3,...` (the standard orthonormal basis).
2. `A1,A2,...,B1,B2,...,I2` (root system names).
3. `s1,s2,s3,...` (names for the generators used in permutation representations).
4. `M, X` (table names used by `weyl` for orbit sums and Weyl characters).

Bugs. Please send reports of reproducible bugs to my e-mail address on page 1. Be sure to include an example, preferably as small as possible. Note that in their present form, `coxeter` and `weyl` do only a small amount of error-trapping for illegal input.

Compensation. If `coxeter` and `weyl` prove to be useful in your own research, I would appreciate an acknowledgment of it in publications derived from that research.

Mailing list. I maintain a low-traffic, private mailing list for users of `coxeter`, `weyl`, and my other packages `SF` (symmetric functions) and `posets` (partially ordered sets). If you would like to be kept informed of new versions, new packages, or (gasp) bugs, please send me a request to be added to the list.

Acknowledgment. I would like to thank Arjeh Cohen for a number of valuable discussions we had that suggested to me the feasibility of creating a Maple package for root systems and Coxeter groups.

10. Copyleft

11. References

[**1**] N. Bourbaki, "Groupes et Algèbres de Lie, Chp. IV–VI," Hermann, Paris, 1968.

[**2**] M. R. Bremner, R. V. Moody, J. Patera, "Tables of dominant weight multiplicities for representations of simple Lie algebras," Marcel Dekker, New York, 1985.

[**3**] J. E. Humphreys, "Introduction to Lie Algebras and Representation Theory," Springer Verlag, Berlin-New York, 1972.

[**4**] J. E. Humphreys, "Reflection groups and Coxeter groups," Cambridge Univ. Press, Cambridge, 1990.

[**5**] J. R. Stembridge, Computational aspects of root systems, Coxeter groups, and Weyl characters, *in* "Interactions of Combinatorics and Representation Theory," *MSJ Memoirs* **11**, Math. Soc. Japan, Tokyo, 2001, pp.1–38.

[**6**] LiE: `http://young.sp2mi.univ-poitiers.fr/~marc/LiE/`

[**7**] CHEVIE: `http://www.math.rwth-aachen.de/~CHEVIE/`

[**8**] Schur: `http://smc.vnet.net/schur.html`

[**9**] GAP: `http://www.ccs.neu.edu/mirrors/GAP/NEU/`