Münster J. of Math. **4** (2011), 1–28
urn:nbn:de:hbz:6-32449575202
Münster Journal of Mathematics

Branching of representations to symmetric subgroups

Michael G. Eastwood and Joseph A. Wolf

(Communicated by Linus Kramer)

Abstract. Let $\mathfrak{g}$ be the Lie algebra of a compact Lie group and let θ be any automorphism of $\mathfrak{g}$. Let $\mathfrak{k}$ denote the fixed point subalgebra $\mathfrak{g}^\theta$. In this paper we present LiE programs that, for any finite dimensional complex representation π of $\mathfrak{g}$, give the explicit branching $\pi|_\mathfrak{k}$ of π on $\mathfrak{k}$. Cases of special interest include the cases where θ has order 2 (corresponding to compact Riemannian symmetric spaces G/K), where θ has order 3 (corresponding to compact nearly-Kaehler homogeneous spaces G/K), where θ has order 5 (which include the fascinating 5-symmetric space E_8/A_4A_4), and the cases where $\mathfrak{k}$ is the centralizer of a toral subalgebra of $\mathfrak{g}$.

1. Introduction

There are many situations where one wants to see the explicit branching of a particular representation from the Lie algebra $\mathfrak{g}$ of a compact Lie group to a Lie subalgebra $\mathfrak{k}$. In many cases the situation corresponds to a compact homogeneous space G/K of some geometric interest, such as the cases where G/K is a Riemannian symmetric space, a nearly-Kaehler manifold, or the compact group realization of a complex flag manifold. Here we follow the standard convention that G has Lie algebra $\mathfrak{g}$ and K has Lie algebra $\mathfrak{k}$. Most cases of geometric interest have the interesting property that $\mathfrak{k}$ is the fixed point set of an automorphism θ of $\mathfrak{g}$. In essentially all cases one can compute the branching by hand, but the time and effort involved may be extreme. This situation is greatly ameliorated by use of the public domain computer program LiE [9]. In this paper we produce the LiE routines that carry out the branching of representations from $\mathfrak{g}$ to $\mathfrak{k}$ explicitly when $\mathfrak{k}$ is the fixed point set of an automorphism θ of $\mathfrak{g}$.

MGE: Research supported by the Australian Research Council and by hospitality from the University of California, Berkeley.

JAW: Research partially supported by NSF Grant DMS 99-88643, by the Australian Research Council, and by hospitality from the University of Adelaide.

One might expect the built-in branch routine of LiE to do the job for us without any additional programming. One problem is that LiE reorders the simple roots of $\mathfrak{k}$ to put them in the Bourbaki order of $\mathfrak{k}$, not of $\mathfrak{g}$. This makes iteration of branching very difficult and causes serious problems for identifying the restriction in cases where there is a symmetry of the Dynkin diagram of $\mathfrak{k}$. That problem is exacerbated when one iterates restrictions. Worse, and this is crucial for many geometric applications, on each summand of the restricted representation LiE renormalizes the restriction to the center of $\mathfrak{k}$ in a complicated manner. This causes serious problems for geometric and analytic applications where negativity of vector bundles is needed and is controlled by restriction to the center of $\mathfrak{k}$. Another problem is that the LiE database of restriction matrices is limited to rank $\leqq 8$, and there it has gaps. Our LiE routines specifically address and solve those problems.

We developed many of these LiE routines for use in our work [5] on the range of the double fibration transform [6, Chap. 14], where we need explicit information on branching from the Levi component of a parabolic subgroup to its intersection with a maximal compact subgroup. These LiE programs are based on structural information on Lie algebras and automorphisms to be found in [2], [10], [11], [7] and [8].

In all cases the LiE routines are written so that one can copy and paste them from a pdf of this paper directly into an ascii file that can then be read by the LiE program.

We necessarily start out by describing the use of the LiE program and how its use varies with the properties of $(\mathfrak{g}, \theta)$. Thus in Section 2 we indicate root orderings and their role in computing LiE's restriction matrix. Then in Section 2A we reduce questions of branching to the cases where $\mathfrak{g}$ is simple and $\mathfrak{k}$ is a maximal θ-stable subalgebra of $\mathfrak{g}$, where there are three essentially different situations. The case where $\mathfrak{g}$ is simple and $\operatorname{rank} \mathfrak{k} < \operatorname{rank} \mathfrak{g}$ is described in Section 2B. It relies on information from [10], [11] and [7]. The case where $\mathfrak{g}$ is simple, $\operatorname{rank} \mathfrak{k} = \operatorname{rank} \mathfrak{g}$ and $\mathfrak{k}$ is not semisimple, is the subject of Section 2C. It relies on information from [2], [7], and the standard structure theory of parabolic subgroups. Then the case where $\mathfrak{g}$ is simple, $\operatorname{rank} \mathfrak{k} = \operatorname{rank} \mathfrak{g}$ and $\mathfrak{k}$ is semisimple, is indicated in Section 2D. This is the most delicate case, and it depends on methods from [2], [7] and [8].

In Section 3 we list all cases where $\mathfrak{g}$ is simple, $\mathfrak{k}$ is θ-maximal and $\operatorname{rank} k < \operatorname{rank} \mathfrak{g}$. For each of them we describe how to find the restriction matrix and we give the listing of a LiE program that computes branching from $\mathfrak{g}$ to $\mathfrak{k}$. The programs are (3.1), (3.2), (3.3), (3.4), (3.5), (3.6), (3.7) and (3.8). In all but two of these, $\theta^2 = 1$ so G/K is a Riemannian symmetric space, and in those two we have $\theta^3 = 1$.

In Section 4 we discuss the LiE programs for the cases where $\mathfrak{g}$ is simple, $\operatorname{rank} \mathfrak{k} = \operatorname{rank} \mathfrak{g}$ and $\mathfrak{k}$ is not semisimple. Those essentially are the cases where $\mathfrak{g}$ is simple and $\mathfrak{k}$ is the centralizer of a toral subalgebra, where the LiE programs are described in Section 2C.

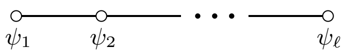
Section 5 gives the LiE branching programs for the cases where $\mathfrak{g}$ is simple, $\theta^2 = 1$ and $\operatorname{rank}\mathfrak{k} = \operatorname{rank}\mathfrak{g}$. The programs (5.1), (5.2) and (5.3) apply when $\mathfrak{g}$ is classical. There one has no surprises on the root orders, but when $\mathfrak{g}$ is exceptional the LiE program scrambles the root order going from $\mathfrak{g}$ to $\mathfrak{k}$. In (5.4), (5.5), (5.6), (5.12), (5.13) and (5.14) this is fairly straightforward, as there is not much flexibility for the location of $\mathfrak{k}$ inside $\mathfrak{g}$. However, in applications [5] we need to keep track of the various simple roots, and we must deal with the fact that there are three combinatorially distinct A_1A_5's in E_6 and two essentially distinct A_1D_6's in E_7. This results in more programs than one might expect, specifically in (5.7), (5.8), (5.9), (5.10) and (5.11).

Section 6 completes the results of Section 5, providing the LiE routines for the seven remaining cases, those where $\mathfrak{g}$ is simple, $\theta^3 = 1$ or $\theta^5 = 1$, and $\operatorname{rank}\mathfrak{k} = \operatorname{rank}\mathfrak{g}$. These routines are (6.1), (6.2), (6.3), (6.4) (6.5), (6.6) and (6.7). There, as in the exceptional group cases of Section 5, we label the simple roots of $\mathfrak{k}$ to minimize any departure from the root ordering of $\mathfrak{g}$.

As indicated in Section 2A, this completes the analysis of branching of finite dimensional irreducible representations from the Lie algebra $\mathfrak{g}$ of a compact Lie group to the fixed point set $\mathfrak{k}$ of any automorphism θ of $\mathfrak{g}$.

2. Restriction Matrices and Branching in LiE

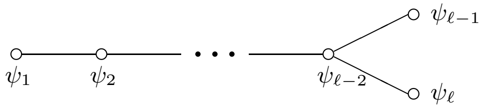
All our LiE routines are given by files with names of the form `branch_X_Y.lie` where `X` is the LiE designation of the type of $\mathfrak{g}$, e.g. `E6`, and `Y` is the LiE designation of the type of $\mathfrak{k}$, e.g. `F4`. They are called within the LiE program by first reading in the file, (> `read branch_X_Y.lie`) and then giving the command (> `branch_X_Y(v)`) where $\mathtt{v} = [v_1, \ldots, v_n]$ represents the highest weight $\sum v_i \xi_i$ of an irreducible representation π of $\mathfrak{g}$ to be branched on $\mathfrak{k}$. Here the ξ_i are the fundamental simple highest weights. Note that this depends on the ordering of the simple roots ψ_i. LiE uses (and therefore we use) Bourbaki order [3], given as follows on the Dynkin diagrams.



(type A_ℓ , $\ell \geqq 1$)

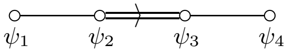
(type B_ℓ , $\ell \geqq 2$)

(type C_ℓ , $\ell \geqq 3$)

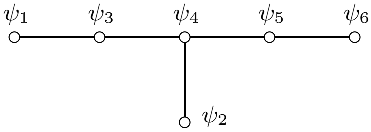


(type D_ℓ , $\ell \geqq 4$)

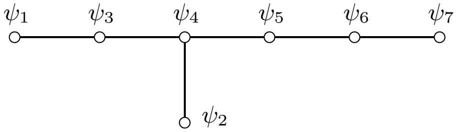
(type G_2)



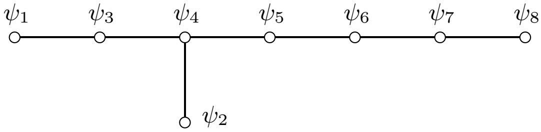
(type F_4)



(type E_6)



(type E_7)



(type E_8)

where, if there are two root lengths, the arrow points from the long roots to the short roots.

If $\mathfrak{k}$ is a subalgebra of $\mathfrak{g}$ then the LiE program computes branching of representations by use of a "restriction matrix". This is the matrix whose rows are the restrictions, from a Cartan subalgebra $\mathfrak{t}$ of $\mathfrak{g}$ to a Cartan subalgebra $\mathfrak{s} \subset \mathfrak{t}$ of $\mathfrak{k}$, of the fundamental simple weights of $\mathfrak{g}$ as linear combinations of the fundamental simple weights of $\mathfrak{k}$. Obviously this depends on the relation between our choices of simple root systems for $\mathfrak{g}$ and $\mathfrak{k}$.

2A. **Reduction to the cases where $\mathfrak{g}$ is simple and $\mathfrak{k}$ is θ-maximal.** We start with the Lie algebra $\mathfrak{g}$ of a compact connected Lie group G and an automorphism θ of $\mathfrak{g}$. The fixed point algebra is $\mathfrak{k} = \mathfrak{g}^\theta$, and K is the corresponding analytic subgroup of G. We start also with an irreducible finite dimensional representation π of $\mathfrak{g}$. We want to describe $\pi|_\mathfrak{k}$ explicitly.

We indicate how to reduce our branching questions to the case where $\mathfrak{g}$ is simple and $\mathfrak{k} = \mathfrak{g}^\theta$ is maximal among the θ-stable subalgebras of $\mathfrak{g}$. That done, we have three essentially different possibilities. The methods appropriate to those three situations are addressed in Sections 2B, 2C and 2D below, and carried out completely in the remainder of this paper.

Our branching procedures all use the LiE program. We give listings of the relevant LiE routines, and when the programming aspects are not so obvious we give an exposition of the mathematics behind our branching routines.

Write $\mathfrak{g} = \mathfrak{g}' \oplus \mathfrak{z}$ where $\mathfrak{g}'$ is semisimple and $\mathfrak{z}$ is the center of $\mathfrak{g}$. Each summand is θ-stable, so $\mathfrak{k} = (\mathfrak{k} \cap \mathfrak{g}') \oplus (\mathfrak{k} \cap \mathfrak{z})$. Also $\pi = \pi' \boxtimes \chi$, exterior tensor product, where π' represents $\mathfrak{g}'$ and χ is a 1-dimensional representation of $\mathfrak{z}$. Now $\pi|_\mathfrak{k} = (\pi'|_{\mathfrak{k}\cap\mathfrak{g}'}) \boxtimes (\chi|_{\mathfrak{k}\cap\mathfrak{z}})$ and evaluation of the latter factor is just restriction of a linear functional to a linear subspace. Thus we need only worry about computing $\pi'|_{\mathfrak{k}\cap\mathfrak{g}'}$. That is the first reduction: it suffices to consider the case where $\mathfrak{g}$ is semisimple.

Decompose $\mathfrak{g}$ as a direct sum of simple ideals. Then θ gives a permutation on that set of ideals, and as such it is a product of disjoint cycles. In other words, we have a decomposition $\mathfrak{g} = \mathfrak{h}_1 \oplus \cdots \oplus \mathfrak{h}_r$ where θ preserves each $\mathfrak{h}_i$ and induces a cyclic permutation on its simple direct summands. Now $\mathfrak{k} = \mathfrak{g}^\theta = \mathfrak{h}_1^\theta \oplus \cdots \oplus \mathfrak{h}_r^\theta$. That is the second reduction: it suffices to consider the case where θ induces a cyclic permutation on the simple ideals of $\mathfrak{g}$.

Now we have reduced to the case $\mathfrak{g} = \mathfrak{g}_1 \oplus \cdots \oplus \mathfrak{g}_m$ where the $\mathfrak{g}_i$ are simple, $\theta(\mathfrak{g}_{i-1}) = \mathfrak{g}_i$ for $1 < i \leqq m$, and $\theta(\mathfrak{g}_m) = \mathfrak{g}_1$. We interpret the $\theta : \mathfrak{g}_{i-1} \cong \mathfrak{g}_i$ as identifications. That done,

$$\begin{aligned} \mathfrak{g} = &\mathfrak{g}_1 \oplus \cdots \oplus \mathfrak{g}_1 \ (m \text{ summands}) \\ &\text{where } \theta(\xi_1, \ldots, \xi_m) = (\gamma(\xi_m), \xi_1, \ldots, \xi_{m-1}) \text{ for } \xi_i \in \mathfrak{g}_1. \end{aligned} \tag{2.1}$$

Here γ is an automorphism on $\mathfrak{g}_1$. Now we have

$$\theta^m(\xi_1, \ldots, \xi_m) = (\gamma(\xi_1), \ldots, \gamma(\xi_m)). \tag{2.2}$$

Thus $\mathfrak{k} = \mathfrak{g}^\theta = (\mathfrak{g}_1^\gamma \oplus \cdots \oplus \mathfrak{g}_1^\gamma)^\theta$ where there are m summands $\mathfrak{g}_1^\gamma$. Denote $\mathfrak{k}_1 = \mathfrak{g}_1^\gamma$. From (2.1) and (2.2) we have

$$\mathfrak{k} = \mathfrak{g}^\theta = \{(\xi_1, \ldots, \xi_1) \mid \xi_1 \in \mathfrak{k}_1 = \mathfrak{g}_1^\gamma\} = \operatorname{diag} \mathfrak{k}_1 . \tag{2.3}$$

Now it suffices to consider the case where $\mathfrak{g}$ is simple.

We address the programming aspects. Suppose that we are given an irreducible representation π of $\mathfrak{g} = \mathfrak{g}_1 \oplus \cdots \oplus \mathfrak{g}_1$ (m summands). Then π is the exterior tensor product $\pi_1 \boxtimes \cdots \boxtimes \pi_m$ of irreducible representation π_i of $\mathfrak{g}_1$. In view of (2.3), $\pi|_\mathfrak{k}$ is the interior tensor product of the restrictions of the π_i to the $\mathfrak{k}_1 = \mathfrak{g}_1^\gamma$. We can do this in two stages. First we compute the restrictions $\pi_i|_{\mathfrak{k}_1}$, which only involves cases where we branch from a simple Lie algebra, and then we decompose the tensor product. In the latter setting we have reduced to the case where $\gamma = 1$ but $\mathfrak{g}_1^\gamma$ may no longer be simple. Still, $\mathfrak{g}^\theta$ decomposes under the action of θ in the setting of a cycle of simple ideals. This is the third reduction: the branching problem is reduced to the case where $\mathfrak{g} = \mathfrak{g}_1 \oplus \cdots \oplus \mathfrak{g}_1$, sum of m simple ideals, and θ acts by $\theta(\xi_1, \ldots, \xi_m) = (\xi_m, \xi_1, \ldots, \xi_{m-1})$. In this case $\pi = \pi_1 \otimes \cdots \otimes \pi_m$ and $\mathfrak{k}$ is the diagonal $\operatorname{diag} \mathfrak{g}_1$ in $\mathfrak{g}$.

We have reduced the case of branching from nonsimple $\mathfrak{g}$ to two parts: branching from simple proper subalgebras of $\mathfrak{g}$ and decomposing tensor products of irreducible representations of $\mathfrak{g}$. The latter is done in LiE as follows. Let `v` be a matrix of m rows, each row `v[i]` a vector of length equal to the rank n of $\mathfrak{g}_1$, where the row `v[i]` = [`v[i,1]`, ..., `v[i,n]`] describes the highest weight $\lambda_i = \sum_j v[i,j]\xi_j$ of π_i in terms of the fundamental simple weights ξ_j. If $m = 2$ and the default is set to the Cartan type of $\mathfrak{g}_1$ then we can use LiE's built-in function

```
tensor(v[2],v[1])
```

for the tensor product decomposition. If $m > 2$ we do this recursively, but we must first convert the `v[i]` to polynomials in the LiE sense,

```
w = null(m,n); for i = 1 to m do w[i] = tensor(v[i],null(n)) od
```

and then we can issue the LiE command

```
w = w[1]; for i = 2 to m do w = tensor(w[i],w) od
```

Here is a general LiE routine to systematize this. It is called in LiE by `branch_diag(v,g)` where g is the Lie type of a simple Lie algebra such as A3, C7, G2, F4 or E8, and where v is a matrix of nonnegative integers whose rows have length rank g representing highest weights of the representations of g to be tensored together.

(2.4)

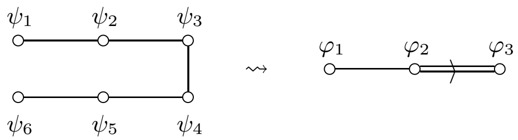
```
# file branch_diag.lie                                                    #
# usage: branch_diag(v,g) where g is a simple Lie algebra type  #
# (An, ..., E8) and v is a matrix of rank g columns, whose rows #
# specify the highest weights of reps $\pi_i$ of g;  It returns #
# the (interior) tensor product of the $\pi_i$.                       #
branch_diag(mat v; grp g) = setdefault(g);
loc u = tensor(null(Lie_rank),null(Lie_rank));
for r  row(v) do u = tensor(u,tensor(r,null(Lie_rank))) od;
print("the branching from product of "+n_rows(v)+" copies of "
+Lie_group(Lie_code[1],Lie_code[2])+" to the diagonal is"); u
```

2B. **Case $\mathfrak{g}$ simple and rank $\mathfrak{k}$ < rank $\mathfrak{g}$.** Suppose first that $\mathfrak{g}$ is simple and $\operatorname{rank}\mathfrak{k} < \operatorname{rank}\mathfrak{g}$. Choose respective Cartan subalgebras $\mathfrak{s} \subset \mathfrak{t}$. Then there is a simple root system $\Psi = \{\psi_1, \dots, \psi_n\}$ for $(\mathfrak{g}, \mathfrak{t})$ such that the restrictions $\psi_1|_{\mathfrak{s}}, \dots \psi_n|_{\mathfrak{s}}$ form a simple root system $\Phi = \{\varphi_1, \dots, \varphi_r\}$ for $\mathfrak{k}$. See [10]. In that case we have a root restriction matrix `res_rt` whose j^{th} row is given by `res_rt[j] =` $[m_{j,1}, \dots, m_{j,r}]$ where $\psi_j|_{\mathfrak{s}} = \sum_k m_{j,k}\varphi_k$. LiE however requires the corresponding restriction matrix of fundamental simple weights, and can compute it from `res_rt` as

(2.5) `res_wt = i_Cartan(`$\mathfrak{g}$`)*res_rt*Cartan(`$\mathfrak{k}$`)/det_Cartan(`$\mathfrak{g}$`)`

where `i_Cartan(`$\mathfrak{g}$`)/det_Cartan(`$\mathfrak{g}$`)` is the inverse of the Cartan matrix of $\mathfrak{g}$ using Ψ and `Cartan(`$\mathfrak{k}$`)` is the Cartan matrix of $\mathfrak{k}$ using Φ.

Here is an example. Let $\mathfrak{g} = \mathfrak{su}(7)$ and $\mathfrak{k} = \mathfrak{so}(7)$. The Dynkin diagram of $\mathfrak{k}$ is obtained by folding that of $\mathfrak{g}$,



In other words, the simple root restrictions are $\psi_1 \mapsto \varphi_1$, $\psi_2 \mapsto \varphi_2$, $\psi_3 \mapsto \varphi_3$, $\psi_4 \mapsto \varphi_3$, $\psi_5 \mapsto \varphi_3$ and $\psi_6 \mapsto \varphi_1$. Thus

$$\texttt{res\_rt} = \begin{bmatrix} 1&0&0\\ 0&1&0\\ 0&0&1\\ 0&0&1\\ 0&1&0\\ 1&0&0 \end{bmatrix}$$

Now (2.5) gives

$$\texttt{res\_wt} = \begin{bmatrix} 1&0&0\\ 0&1&0\\ 0&0&2\\ 0&0&2\\ 0&1&0\\ 1&0&0 \end{bmatrix}.$$

If the LiE default group is set to A6 for $\mathfrak{su}(7)$ then branching of the adjoint representation of $\mathfrak{su}(7)$ on $\mathfrak{so}(7)$ is given by `branch([1,0,0,0,0,1],B3,res_wt)`, resulting in `1X[0,1,0] +1X[2,0,0]`.

2C. **Case $\mathfrak{g}$ simple, rank $\mathfrak{k}$ = rank $\mathfrak{g}$ and $\mathfrak{k}$ is not semisimple.** Suppose that $\mathfrak{g}$ is simple and $\mathfrak{k}$ is of equal rank but is not semisimple. Recall that $\mathfrak{k}$ is θ-maximal in the sense that it is maximal among the θ-stable proper subalgebras of $\mathfrak{g}$. It follows that $\mathfrak{k}$ is the centralizer of its center, so it is a compact real form of the reductive (Levi) component of a parabolic subalgebra of $\mathfrak{g}_{\mathbb{C}}$. We remark that the centralizer of a toral subalgebra $\mathfrak{e} \subset \mathfrak{t}$ of $\mathfrak{g}$ is always the fixed point of an automorphism $\theta \in \mathrm{Ad}(\exp(\mathfrak{e}))$, for example $\theta = \mathrm{Ad}(t)$ where the powers of t form a dense subgroup of the torus $\exp(\mathfrak{e})$. Now $\mathfrak{g}$ has a simple root system $\Psi = \{\psi_1, \dots, \psi_n\}$ such that some subset $\Phi \subset \Psi$ is a simple root system for $\mathfrak{k}$. We use the notation of Baston and Eastwood [1] to indicate these Levi components, i.e. to indicate these centralizers in $\mathfrak{g}$ of subtori of its Cartan subalgebra. Thus if $\psi \in \Psi \setminus \Phi$ we replace the circle $\circ$ for ψ by a cross $\times$. We refer to this as the *diagram* of the corresponding parabolic subalgebra $\mathfrak{q}_\Phi$ of $\mathfrak{g}_{\mathbb{C}}$, the corresponding parabolic subgroup Q_Φ of $G_{\mathbb{C}}$, and our algebra $\mathfrak{k} = \mathfrak{q}_\Phi \cap \mathfrak{g}$ which is a compact real form of the Levi component of $\mathfrak{q}_\Phi$. For example, the parabolic subalgebra $\mathfrak{q}_\Phi$ of $\mathfrak{sl}(n+1;\mathbb{C})$ that corresponds to the complex projective space $P_n(\mathbb{C}) = SL(n+1;\mathbb{C})/Q_\Phi$ is given by $\Psi = \{\psi_1, \dots, \psi_n\}$ and $\Phi = \{\psi_1, \dots, \psi_{n-1}\}$, so it has diagram ○——○— • • • —○——×, and the parabolic subalgebra for the Grassmannian of incident point, hyperplane pairs in $P_n(\mathbb{C})$ is given by $\Phi = \{\psi_2, \dots, \psi_{n-1}\}$ and has diagram ×——○— • • • —○——×. These correspond to the cases

$$\mathfrak{k} = \mathfrak{u}(n) \subset \mathfrak{su}(n+1) = \mathfrak{g} \text{ and}$$
$$\mathfrak{k} = \{x \in \mathfrak{u}(1) \oplus \mathfrak{u}(n-1) \oplus \mathfrak{u}(1) \mid \operatorname{trace} x = 0\} \subset \mathfrak{su}(n+1) = \mathfrak{g}.$$

Suppose that Φ consists of all but one element $\gamma = \psi_i$ of Ψ, in other words that $\mathfrak{q}_\Phi$ is a maximal parabolic subalgebra of $\mathfrak{g}_{\mathbb{C}}$. That is the case where there is only one $\times$ on the diagram of $\mathfrak{q}_\Phi$. Then there is a simple LiE routine (from the LiE manual [9]) that describes branching of representations from $\mathfrak{g}$ to $\mathfrak{k} = \mathfrak{q}_\Phi \cap \mathfrak{g}$:

(2.6)
```
# file Levi_branch.lie #
Levi_mat(int i) = fundam(id(Lie_rank) - i)
Levi_type(int i) = Cartan_type(Levi_mat(i))
Levi_diagram(int i) = diagram(Levi_type(i))
Levi_res_mat(int i) = res_mat(Levi_mat(i))
Levi_branch(vec v; int i) = loc m = Levi_mat(i);
r = res_mat(m);
branch(v, Cartan_type(m), r)
```

To use it, say with $\mathfrak{g} = E_7$ and $\mathfrak{k} = E_6T_1$, do `read(Levi_branch.lie)`, then `setdefault(E7)`, then `diagram` in LiE to see that $\gamma = \psi_7$, do `v = [v_1, v_2, v_3, v_4, v_5, v_6, v_7]` for the highest weight $\sum v_i\xi_i$ of π, and compute the restriction by `Levi_branch(v,7)`. The result is a sum of vectors with multiplicities, e.g.

$$1\mathtt{X}[0,0,0,0,0,2,-6]+2\mathtt{X}[0,0,0,0,0,2,-4]+4\mathtt{X}[0,0,0,0,0,2,-2]+\dots$$

where the last entries (`-6, -4, -2`) refer to the central torus. For the meaning of the others do `Levi_diagram(7)` in order to compare the root orderings (in the LiE program) for $\mathfrak{g}$ and $\mathfrak{k}$.

Suppose next that Φ consists of all but two elements ψ_i and ψ_j of Ψ, in other words that there are two $\times$'s on the diagram of $\mathfrak{q}_\Phi$. We modify the routine (2.6) to accommodate this. Here it is important that $i > j$ so that we remove rows i and j from a matrix by removing the i^{th} and then the j^{th} of that.

(2.7)

```
# file Levi_branch2.lie #
Levi_mat(int i, j) = fundam((id(Lie_rank) - i) - j)
Levi_type(int i, j) = Cartan_type(Levi_mat(i,j))
Levi_diagram(int i, j) = diagram(Levi_type(i,j))
Levi_res_mat(int i, j) = res_mat(Levi_mat(i,j))
Levi_branch2(vec v; int i, j) = loc m = Levi_mat(i,j);
r = res_mat(m);
branch(v, Cartan_type(m), r)
```

Similarly if Φ consists of all but three elements ψ_i, ψ_j and ψ_k of Ψ, $i > j > k$,

(2.8)

```
# file Levi_branch3.lie #
Levi_mat(int i, j, k) = fundam(((id(Lie_rank) - i) - j) - k)
Levi_type(int i, j, k) = Cartan_type(Levi_mat(i,j,k))
Levi_diagram(int i, j, k) = diagram(Levi_type(i,j,k))
Levi_res_mat(int i, j, k) = res_mat(Levi_mat(i,j,k))
Levi_branch3(vec v; int i, j, k) = loc m = Levi_mat(i,j,k);
r = res_mat(m);
branch(v, Cartan_type(m), r)
```

At this point the pattern is clear. For example, try

```
read Levi_branch3.lie
setdefault(E8)
Levi_branch3([1,0,0,0,0,0,0,1],8,6,4)
```

The first five entries in each resulting 8-tuple give the branching on the semisimple part $A_2A_1A_1A_1$ of $\mathfrak{k} = \mathfrak{q}_{\Psi\setminus\{\psi_8,\psi_6,\psi_4\}}$, but with some roots permuted. To see the permutation look at the restriction matrix

`Levi_res_mat(int 8, 6, 4)`

and remove rows 8, 6 and 4, and remove the last three columns. In LiE this can be implemented as

```
*(((*(((Levi_res_mat(8, 6, 4) - 8) - 6) - 4) -8) -7) -6)
```

In this way the `Levi_branch` LiE routines give the restriction to the semisimple part of $\mathfrak{k}$.

Of course, these routines also give the action of the center of $\mathfrak{k}$ on each irreducible summand but, unfortunately, this is implemented in a rather *ad hoc* fashion. We now explain how to specify the central action in a more systematic and useful manner. At the same time, we avoid having to deal with the permutations introduced by the programs `Levi_branch`, as above. The problems with these programs can be illustrated with the following simple examples. With `setdefault(F4)` in place we have LiE calculate the following matrices.

(2.9)

i_Cartan	Levi_res_mat(3)	Levi_res_mat(4)	Levi_res_mat(4,3)
[[2,3,4,2]	[[0,1,0,4]	[[1,0,0,2]	[[0,1,2,0]
,[3,6,8,4]	,[1,0,0,8]	,[0,1,0,4]	,[1,0,4,0]
,[2,4,6,3]	,[0,0,0,6]	,[0,0,1,3]	,[0,0,3,0]
,[1,2,3,2]	,[0,0,1,3]	,[0,0,0,2]	,[0,0,0,1]
]	]	]	]

In this particular case, the matrices `Levi_res_mat(i)` are easy to understand. The first three columns specify a permutation of the uncrossed nodes and the last column is the i^{th} column of the inverse Cartan matrix `i_Cartan`. It is easy to check that the element of the Cartan subalgebra $\mathfrak{t} \subset \mathfrak{g}$ defined by the i^{th} column of the inverse Cartan matrix with respect to the basis of fundamental weights is in the center of the corresponding Levi subalgebra $\mathfrak{k}$. (Indeed, this is minus the so-called 'grading element' of the corresponding maximal parabolic subalgebra [4].) Thus, the restriction matrix specifies a permutation of the uncrossed nodes and a particular element of the center. Here is the branching of the adjoint representation given by `Levi_branch([1,0,0,0],3)`.

$$
\begin{aligned}
&[0,1,0,4] \oplus [0,0,1,3] \oplus [1,0,2,2] \oplus [0,1,1,1] \\
&\qquad \oplus \big([0,0,2,0] \oplus [1,1,0,0] \oplus [0,0,0,0]\big) \\
&\qquad \oplus [1,0,1,-1] \oplus [0,1,2,-2] \oplus [0,0,1,-3] \oplus [1,0,0,-4],
\end{aligned}
$$

(where the ordering is given by the value of the grading element taken from [4] from -4 to 4, bearing in mind that this is minus the value naturally provided by LiE).

$$
\begin{aligned}
\mathfrak{g} = \mathfrak{g}_{-4} \oplus \mathfrak{g}_{-3} \oplus \mathfrak{g}_{-2} \oplus \mathfrak{g}_{-1} \oplus \mathfrak{g}_{0} \oplus \mathfrak{g}_{1} \oplus \mathfrak{g}_{2} \oplus \mathfrak{g}_{3} \oplus \mathfrak{g}_{4} \\
= [0,1,0,4] \oplus [0,0,1,3] \oplus \cdots \oplus [1,0,0,-4]
\end{aligned}
$$

(and this is exactly the realization of $\mathfrak{g}$ as the 4-graded Lie algebra corresponding to the parabolic subalgebra $\mathfrak{q}_\Phi = \mathfrak{g}_0 \oplus \cdots \oplus \mathfrak{g}_4$ as in [4, Theorem 3.2.1]).

The restriction matrix `Levi_res_mat(4,3)` in (2.9) is more difficult to understand. Certainly, we could use

```
[[0,1,4,2]
,[1,0,8,4]
,[0,0,6,3]
,[0,0,3,2]
]
```

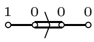
as a more easily understandable restriction matrix. It is obtained by using the j^{th} and i^{th} columns of `i_Cartan` to replace the last two columns of `r`, a change that is easily implemented in LiE by adding

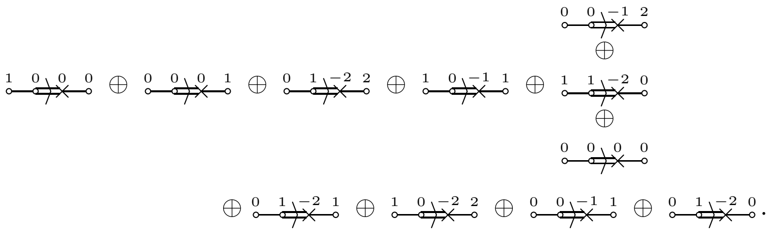
```
for k = 1 to Lie_rank do r[k,Lie_rank-1] = i_Cartan[k,j] od;
for k = 1 to Lie_rank do r[k,Lie_rank] = i_Cartan[k,i] od;
```

as the penultimate two lines of `Levi_branch2.lie`. In comparison with (2.9), the last two columns of `Levi_res_mat(4,3)` are some linear combination of the appropriate columns of the inverse Cartan matrix. Moreover, the case $\mathfrak{g} = F_4$ is deceptively simple because its Cartan matrix has unit determinant. In general, because LiE is restricted to integer arithmetic, it is only reasonable to use the appropriate columns from `i_Cartan`, as above. In particular, the grading element will not be simply minus the sum of these columns but, in addition, one must divide by `det_Cartan`. Although the grading element takes on integral values on the adjoint representation (from $-k$ to k where $\mathfrak{g}$ is $|k|$-graded by the parabolic subalgebra $\mathfrak{q}_\Phi$), for a general irreducible representation its values will be rational with `det_Cartan` as denominator. In any case, the raw instructions `Levi_res_mat(i,j)` and `Levi_res_mat(i,j,k)` produce rather bizarre changes of basis from the more natural normalization provided by the inverse Cartan matrix and even `Levi_res_mat(i)` is better modified by

```
for j = 1 to Lie_rank do r[j,Lie_rank] = i_Cartan[j,i] od;
```

to avoid spurious factors.

For many purposes, however, it is better to write all weights as linear combinations of the fundamental weights of $(\mathfrak{g}, \mathfrak{t})$ and, following [1], attach the resulting coefficients to the corresponding nodes of the Dynkin diagram. In our example, the adjoint representation  decomposes as

(2.10) 

The conversion between these two conventions is the definition of the restriction matrix. Therefore, no matter what restriction matrix is used, to convert back to the conventions of [1], one simply needs to invert the restriction matrix and apply this inverse matrix by right multiplication to each term obtained from `branch(v, Cartan_type(m), r)`. Since LiE allows only integer multiplication, inverting a matrix with integer entries requires some care. In the following program, the restriction matrix `r` is inverted by first extracting from it a permutation matrix `p`, noting that permutation matrices are orthogonal, and forming `(*p)*r`. The result necessarily has the form

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & * \\ 0 & 1 & \cdots & 0 & * \\ \vdots & \vdots & \ddots & \vdots & * \\ 0 & 0 & \cdots & 1 & * \\ 0 & 0 & 0 & 0 & * \end{bmatrix},$$

which is inverted by a dint of an explicit formula.

(2.11)

```
# file Levi_branch_improved.lie #
Levi_mat(int i) = fundam(id(Lie_rank) - i)
Levi_type(int i) = Cartan_type(Levi_mat(i))
Levi_diagram(int i) = diagram(Levi_type(i))
Levi_res_mat(int i) = res_mat(Levi_mat(i))
div(pol p;int k) = loc l = 0X(null(n_vars(p)));
for i=1 to length(p) do l = l+coef(p,i)X(expon(p,i)/k) od; l
Levi_branch(vec v; int i) = loc m = Levi_mat(i);
loc r = res_mat(m); loc p = r;
for j = 1 to Lie_rank do p[j,Lie_rank] = 0 od;
p[i,Lie_rank] = 1;
loc q = (*p)*r; loc det_q = q[Lie_rank,Lie_rank];
loc qq = q;
for j = 1 to Lie_rank do qq[j,Lie_rank] = -q[j,Lie_rank] od;
qq[Lie_rank,Lie_rank] = 1;
loc s = null(Lie_rank,Lie_rank);
for j = 1 to Lie_rank do s[j,j] = det_q od;
s[Lie_rank,Lie_rank] = 1; loc i_q = qq*s;
loc b = branch(v, Cartan_type(m), r); loc x = b*i_q*(*p);
div(x,det_q)
```

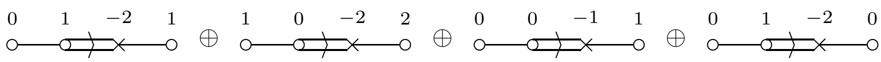
The program is used as before but the result is expressed using the diagrammatic conventions of [1]. For example,

```
read Levi_branch_improved.lie
setdefault(F4)
Levi_branch([1,0,0,0],3)
```

gives

```
1X[0,0,-1,1] +1X[0,0,-1,2] +1X[0,0, 0,0] +1X[0,0, 0,1] +
1X[0,1,-2,0] +1X[0,1,-2,1] +1X[0,1,-2,2] +1X[1,0,-2,2] +
1X[1,0,-1,1] +1X[1,0, 0,0] +1X[1,1,-2,0]
```

as in (2.10) (but devoid of the convenient ordering there. The ordering of (2.10) is essential when the action of the full parabolic $\mathfrak{q}_\Phi$ is considered rather than just its Levi factor. Representations of $\mathfrak{q}_\Phi$ are generally filtered. For example, the tail of (2.10),



is interpreted in [1, p. 135] as inducing the cotangent bundle on the corresponding generalized flag manifold.).

Levi_branch2.lie is similarly improved

(2.12)

```
# file Levi_branch2_improved.lie #
Levi_mat(int i, j) = fundam((id(Lie_rank) - i) - j)
Levi_type(int i, j) = Cartan_type(Levi_mat(i,j))
Levi_diagram(int i, j) = diagram(Levi_type(i,j))
Levi_res_mat(int i, j) = res_mat(Levi_mat(i,j))
div(pol p;int k) = loc l = 0X(null(n_vars(p)));
for i=1 to length(p) do l = l+coef(p,i)X(expon(p,i)/k) od; l
Levi_branch2(vec v; int i, j) = loc m = Levi_mat(i,j);
loc r = res_mat(m); loc p = r;
for k = 1 to Lie_rank do p[k,Lie_rank] = 0 od;
p[i,Lie_rank] = 1;
for k = 1 to Lie_rank do p[k,Lie_rank-1] = 0 od;
p[j,Lie_rank-1] = 1;
loc q = (*p)*r;
loc det_q = q[Lie_rank-1,Lie_rank-1]*q[Lie_rank,Lie_rank] \
           -q[Lie_rank,Lie_rank-1]*q[Lie_rank-1,Lie_rank];
qq = q;
for j = 1 to Lie_rank do \
           qq[j,Lie_rank] = -q[j,Lie_rank] od;
qq[Lie_rank-1,Lie_rank] = 0; qq[Lie_rank,Lie_rank] = 1;
for j = 1 to Lie_rank do \
           qq[j,Lie_rank-1] = -q[j,Lie_rank-1] od;
qq[Lie_rank-1,Lie_rank-1] = 1; qq[Lie_rank,Lie_rank-1] = 0;
loc s = null(Lie_rank,Lie_rank);
for j = 1 to Lie_rank do s[j,j] = det_q od;
s[Lie_rank-1,Lie_rank-1] = (q*qq)[Lie_rank,Lie_rank];
s[Lie_rank,Lie_rank-1] = -(q*qq)[Lie_rank,Lie_rank-1];
s[Lie_rank-1,Lie_rank] = -(q*qq)[Lie_rank-1,Lie_rank];
s[Lie_rank,Lie_rank] = (q*qq)[Lie_rank-1,Lie_rank-1];
loc i_q = qq*s;
loc b = branch(v, Cartan_type(m), r); loc x = b*i_q*(*p);
div(x,det_q)
```

and to improve Levi_branch3.lie is left as an exercise (which implicitly requires incorporating the formula for the inverse of a general 3×3 matrix).

2D. **Case $\mathfrak{g}$ simple, rank $\mathfrak{k}$ = rank $\mathfrak{g}$ and $\mathfrak{k}$ is semisimple.** This is the most delicate case: $\mathfrak{g}$ is simple, θ is an inner automorphism, and $\mathfrak{k}$ is semisimple. Let us assume that $\mathfrak{k}$ is θ-maximal. Then the Borel-de-Siebenthal structure theory [2] provides a simple root system $\Psi = \{\psi_1, \ldots, \psi_n\}$ and a simple root $\gamma = \psi_r \in \Psi$ such that $\Psi_{\mathfrak{k}} = (\Psi \setminus \{\gamma\}) \cup \{-\beta_{\mathfrak{g}}\}$ is a simple root system for $\mathfrak{k}$, and θ has order n_r, where the maximal root $\beta_{\mathfrak{g}} = \sum n_i \psi_i$.

Let $\mathfrak{s}$ denote the maximal rank subalgebra of $\mathfrak{g}$ with simple root system $\Psi_{\mathfrak{s}} = (\Psi \setminus \{\gamma\})$. Let $w_{\mathfrak{g}}$ and $w_{\mathfrak{s}}$ denote, respectively, the longest elements of the Weyl groups $W_{\mathfrak{g}}$ and $W_{\mathfrak{s}}$. We write $\Sigma^+(\mathfrak{g}, \mathfrak{t})$ for the positive root system of $\mathfrak{g}$ relative to $\mathfrak{t}$ defined by Ψ.

Lemma 2.13. *The transformation* $-w_{\mathfrak{s}}$ *preserves* $\Phi_{\mathfrak{s}}$ *and sends* $-\beta_{\mathfrak{g}}$ *into* $\Sigma^+(\mathfrak{g}, \mathfrak{t})$.

Proof. In general, the longest element of the Weyl group sends the positive Weyl chamber to its negative, so $-w_{\mathfrak{s}}$ preserves $\Phi_{\mathfrak{s}}$. But $-w_{\mathfrak{s}}(-\beta_{\mathfrak{g}}) = w_{\mathfrak{s}}(\beta_{\mathfrak{g}})$ is obtained from $\beta_{\mathfrak{g}} = \sum m_i \psi_i$ by a series of simple root reflections $s_\psi : \xi \mapsto \xi - \frac{2\langle \psi, \xi \rangle}{\langle \psi, \psi, \rangle} \psi$ with $\psi \neq \gamma$. Thus the coefficient of γ in $w_{\mathfrak{s}}(\beta_{\mathfrak{g}})$ is the same as that in $\beta_{\mathfrak{g}}$, which is $n_r > 0$, so $w_{\mathfrak{s}}(\beta_{\mathfrak{g}}) \in \Sigma^+(\mathfrak{g}, \mathfrak{t})$. □

We now indicate how the LiE program uses $w_{\mathfrak{s}}$ and $\beta_{\mathfrak{g}}$ to compute the restriction matrix `res_wt`, which it uses to calculate restrictions of representations of $\mathfrak{g}$ to $\mathfrak{k}$. First, we use $-w_{\mathfrak{s}}$ to carry the simple root system $\Psi_{\mathfrak{k}} = \Psi_{\mathfrak{s}} \cup \{-\beta_{\mathfrak{g}}\}$ of $\mathfrak{k}$ to another simple root system $\Phi := \Psi_{\mathfrak{s}} \cup \{w_{\mathfrak{s}}(\beta_{\mathfrak{g}})\}$. The point is that Φ then consists of positive roots for $\mathfrak{g}$, all but one of them simple, by Lemma 2.13. The LiE program assumes Bourbaki root order for both $\mathfrak{g}$ and $\mathfrak{k}$. It permutes the roots of Φ in a somewhat arbitrary way in order to do this when it computes the restriction matrix and applies it to branching of representations from $\mathfrak{g}$ to $\mathfrak{k}$. We will try to do this in a way that involves minimal permutation.

We start by computing $w_{\mathfrak{s}}$ within the LiE program. It is the long word for $W_{\mathfrak{s}}$, but there LiE orders the roots in a manner that loses track of the simple roots of $\mathfrak{k}$ and so is not appropriate for our applications. Instead we use the routine

```
ws = reduce(long_word^r_reduce(long_word,[1,2,...,r-1,r+1,...,n-1,n]))
```

Here `long_word` is the longest element $w_{\mathfrak{g}}$ of the Weyl group $W_{\mathfrak{g}}$, so the shortest element of the coset $w_{\mathfrak{g}} W_{\mathfrak{s}}$ is

```
r_reduce(long_word,[1,2,...,r-1,r+1,...,n-1,n]).
```

Then we set up the new simple root system Φ for $\mathfrak{k}$ as the rows of a matrix RR in which $w_{\mathfrak{s}}(\beta_{\mathfrak{g}})$ replaces γ and the roots are re-ordered (minimally) to Bourbaki order. Then there are three ways to compute the restriction matrix `res_wt`.

The first is to note that RR is the inverse of the matrix `res_rt`, so one can compute

```
res_wt = i_Cartan(𝔤)*res_rt*Cartan(𝔨)/det_Cartan(𝔤),
```

which is (2.5). The second is just to use the LiE assignment

```
res_wt = res_mat(RR).
```

And the third, which is in fact the way that LiE implements `res_mat`, is to set $\mathfrak{g}$ as the default by `setdefault(𝔤)` (putting in the Lie type of $\mathfrak{g}$), initialize `res_wt` as the $n \times n$ identity matrix, `res_wt = id(n)`, and then fill it in by

```
for j = 1 to n do
   for i = 1 to n do
      res_wt[i,j] = Cartan(i_Cartan[i],RR[j])/det_Cartan
   od
od
```

In the next few sections we will run through the various basic cases cases of $(\mathfrak{g}, \mathfrak{k})$ and then reduce the general case to these basic cases.

3. Cases: $\mathfrak{g}$ is simple, $\mathfrak{k}$ is θ-maximal and $\operatorname{rank}\mathfrak{k} < \operatorname{rank}\mathfrak{g}$

Recall the automorphism θ of $\mathfrak{g}$ with $\mathfrak{k} = \mathfrak{g}^\theta$. In this section we assume that $\mathfrak{k}$ is θ-maximal, in other words that it is maximal among the proper θ-invariant subalgebras of $\mathfrak{g}$, and we apply the methods of Section 2B.

Note that θ is an outer automorphism of $\mathfrak{g}$ because $\operatorname{rank}\mathfrak{k} < \operatorname{rank}\mathfrak{g}$. If some power $\theta^m \neq 1$ is an inner automorphism then its fixed point set is θ-invariant and satisfies $\mathfrak{k} \subsetneqq \mathfrak{g}^{\theta^m} \subsetneqq \mathfrak{g}$. As $\mathfrak{k}$ is θ-maximal we conclude that every power $\theta^m \neq 1$ is an outer automorphism of $\mathfrak{g}$. All possibilities are listed in [7, Theorem 5.10(3)]. There θ has prime order $p = 2$ or $p = 3$. If $p = 2$ then G/K is one of the Riemannian symmetric spaces $SU(n)/SO(n)$, $SU(2n)/Sp(n)$, $SO(2p+2+2q)/\{SO(2p+1) \times SO(2q+1)\}$, E_6/F_4, and E_6/C_4 .

If $p = 3$ then G/K is one of the nearly-Kaehler spaces $Spin(8)/G_2$ and $Spin(8)/SU(3)$.

It is useful to note that either the Dynkin diagram of $\mathfrak{k}$ is obtained by folding the diagram of $\mathfrak{g}$ as in [11]—all possibilities are listed in the tables of [11, pp. 245, 247]—or θ has form $\theta' \circ \operatorname{Ad}(g)$ where $\mathfrak{g}^{\theta'}$ is obtained by folding and $g \in G^{\theta'}$. For example $Spin(8)/G_2$ is obtained by folding and $Spin(8)/SU(3)$ is derived from it as just described; and E_6/F_4 is obtained by folding and E_6/C_4 is derived from it as just described. For details of the latter see [12, p. 291]. We now run through that list.

3A. **Case $G/K = SU(2m)/SO(2m)$.** In order to find the restriction matrix used by the LiE program, we consider the Cartan subalgebras

$$\mathfrak{s} = \{\operatorname{diag}\{u_1,\dots,u_m,-u_m,\dots,-u_1\} \mid u_i \in \sqrt{-1}\,\mathbb{R}\} \text{ of } \mathfrak{k} \text{ and}$$
$$\mathfrak{t} = \{\operatorname{diag}\{u_1,\dots,u_{2m}\} \mid u_i \in \sqrt{-1}\,\mathbb{R}, u_1+\cdots+u_{2m} = 0\} \text{ of } \mathfrak{g}.$$

The simple roots of $\mathfrak{g}$ are the $\psi_i = \varepsilon_i - \varepsilon_{i+1}$ for $1 \leqq i < 2m$, and the simple roots of $\mathfrak{k}$ are the $\varphi_i = \varepsilon_i - \varepsilon_{i+1}$ (for $1 \leqq i < m$) and $\varphi_m = \varepsilon_{m-1} + \varepsilon_m$. Thus the simple roots of $\mathfrak{g}$ have restriction to $\mathfrak{s}$ given by $\psi_i|_\mathfrak{s} = \varphi_i$ and $\psi_{m+i}|_\mathfrak{s} = \varphi_{m-i}$ for $1 \leqq i < m$ and $\psi_m|_\mathfrak{s} = 2\varepsilon_m = \varphi_m - \varphi_{m-1}$. Here is the relevant LiE routine `branch_A_D.lie` for branching from $SU(2m)$ to $SO(2m)$. It takes arguments `(m,v)`, where $m > 0$ is an integer and `v` is a vector of length $2m-1$ consisting of nonnegative integers, and branches `v`.

(3.1)
```
# file branch_A_D.lie                                           #
# usage: branch_A_D(m,v) branches v from SU(2m) to SO(2m)   #
branch_A_D(int m;vec v) = setdefault(Lie_group(1,2*m - 1));
res_rt = null(2*m-1,m);
for i=1 to m do res_rt[i,i] = 1 od;
res_rt[m,m-1] = -1; res_rt[m,m] = 1;
for i=1 to m-1 do res_rt[m+i,m-i] = 1 od;
res_wt = i_Cartan*res_rt*Cartan(Lie_group(4,m))/det_Cartan;
print("the branching of "+v+" from SU("+2*m+")");
print("                          to SO("+2*m+") is");
branch(v,Lie_group(4,m),res_wt)
```

Here is an example of its use:

```
read branch_A_D.lie
branch_A_D(8,[1,1,0,0,0,0,0,0,0,0,0,0,0,0,0])
    the branching of [1,1,0,0,0,0,0,0,0,0,0,0,0,0,0] from SU(16)
    to SO(16) is 1X[1,0,0,0,0,0,0,0] +1X[1,1,0,0,0,0,0,0]
```

If one wishes to vary v while holding m constant he can define m in LiE, e.g. as in `m = 7` , and replace the line

```
branch_A_D(int m;vec v) = setdefault(Lie_group(1,2*m - 1));
```

by

```
branch_A_D(vec v) = setdefault(Lie_group(1,2*m - 1));
```

.

3B. **Case $G/K = SU(2m+1)/SO(2m+1)$.** In order to find the restriction matrix we consider the Cartan subalgebras

$$\mathfrak{s} = \{\mathrm{diag}\{u_1, \dots, u_m, 0, -u_m, \dots, -u_1\} \mid u_i \in \sqrt{-1}\,\mathbb{R}\} \text{ of } \mathfrak{k} \text{ and}$$

$$\mathfrak{t} = \{\mathrm{diag}\{u_1, \dots, u_{2m+1}\} \mid u_i \in \sqrt{-1}\,\mathbb{R}, u_1 + \cdots + u_{2m+1} = 0\} \text{ of } \mathfrak{g}.$$

The simple roots of $\mathfrak{g}$ are the $\psi_i = \varepsilon_i - \varepsilon_{i+1}$ for $1 \leqq i \leqq 2m$, and the simple roots of $\mathfrak{k}$ are the $\varphi_i = \varepsilon_i - \varepsilon_{i+1}$ (for $1 \leqq i < m$) and $\varphi_m = \varepsilon_m$ (short simple root). Thus the simple roots of $\mathfrak{g}$ have restriction to $\mathfrak{s}$ given by $\psi_i|_{\mathfrak{s}} = \varphi_i$ and $\psi_{m+i}|_{\mathfrak{s}} = \varphi_{m+1-i}$ for $1 \leqq i \leqq m$. Here is the relevant LiE routine `branch_A_B.lie` for branching from $SU(2m+1)$ to $SO(2m+1)$. It takes arguments `(m,v)`, where $m > 0$ is an integer and `v` is a vector of length $2m$ consisting of nonnegative integers, and branches `v`.

(3.2)

```
# file branch_A_B.lie                                         #
# usage: branch_A_B(m,v) branches v from SU(2m+1) to SO(2m+1) #
branch_A_B(int m;vec v) = setdefault(Lie_group(1,2*m));
res_rt = null(2*m,m);
for i=1 to m do res_rt[i,i] = 1 od;
for i=1 to m do res_rt[m+i,m+1-i] = 1 od;
res_wt = i_Cartan*res_rt*Cartan(Lie_group(2,m))/det_Cartan;
print("the branching of "+v+" from SU("+(2*m+1)+")");
print("                           to SO("+(2*m+1)+") is");
branch(v,Lie_group(2,m),res_wt)
```

The same comment as just after (3.1) holds here when one wishes to set m in LiE.

3C. **Case $G/K = SU(2m)/Sp(m)$.** This case is quite similar to the case of $SU(2m)/SO(2m)$ above. We consider the Cartan subalgebras

$$\mathfrak{s} = \{\mathrm{diag}\{u_1, \dots, u_m, -u_m, \dots, -u_1\} \mid u_i \in \sqrt{-1}\,\mathbb{R}\} \text{ of } \mathfrak{k} \text{ and}$$

$$\mathfrak{t} = \{\mathrm{diag}\{u_1, \dots, u_{2m}\} \mid u_i \in \sqrt{-1}\,\mathbb{R}, u_1 + \cdots + u_{2m} = 0\} \text{ of } \mathfrak{g}.$$

The simple roots of $\mathfrak{g}$ are the $\psi_i = \varepsilon_i - \varepsilon_{i+1}$ for $1 \leqq i < 2m$, and the simple roots of $\mathfrak{k}$ are the $\varphi_i = \varepsilon_i - \varepsilon_{i+1}$ (for $1 \leqq i < m$) and $\varphi_m = 2\varepsilon_m$. Thus the simple roots of $\mathfrak{g}$ have restriction to $\mathfrak{s}$ given by $\psi_i|_{\mathfrak{s}} = \varphi_i$ and $\psi_{m+i}|_{\mathfrak{s}} = \varphi_{m-i}$ for $1 \leqq i < m$ and $\psi_m|_{\mathfrak{s}} = 2\varepsilon_m = \varphi_m$. Here is the relevant LiE routine `branch_A_C.lie` for branching from $SU(2m)$ to $Sp(m)$. It takes arguments (m,v), where $m > 0$ is an integer and v is a vector of length $2m-1$ consisting of nonnegative integers, and branches v.

(3.3)

```
# file branch_A_C.lie                                       #
# usage: branch_A_C(m,v) branches v from SU(2m) to Sp(m) #
branch_A_C(int m;vec v) = setdefault(Lie_group(1,2*m - 1));
res_rt = null(2*m-1,m);
for i=1 to m-1 do res_rt[i,i] = 1 od;
res_rt[m,m] = 1;
for i=1 to m-1 do res_rt[m+i,m-i] = 1 od;
res_wt = i_Cartan*res_rt*Cartan(Lie_group(3,m))/det_Cartan;
print("the branching of "+v+" from SU("+2*m+")");
print("                          to Sp("+m+") is");
branch(v,Lie_group(3,m),res_wt)
```

The same comment as just after (3.1) holds here when one wishes to set m in LiE.

3D. **Cases $G/K = SO(2p+2+2q)/\{SO(2p+1)\times SO(2q+1)\}$ and p,q not both 0.** We use the Cartan subalgebras

$$\mathfrak{t} = \{u := \mathrm{diag}\{u_1,\dots u_{p+q+1}, -u_{p+q+1},\dots - u_1\} \mid u_i \in \sqrt{-1}\,\mathbb{R}\} \text{ and}$$
$$\mathfrak{s} = \{u \in \mathfrak{t} \mid u_{p+q+1} = 0\}.$$

Now $\mathfrak{g}$ has simple roots $\psi_i = \varepsilon_i - \varepsilon_{i+1}$ for $1 \leqq i \leqq p+q$ and $\psi_{p+q+1} = \varepsilon_{p+q} + \varepsilon_{p+q+1}$. The subalgebra $\mathfrak{k}$ has simple roots $\varphi_i = \varepsilon_i - \varepsilon_{i+1}$ for $1 \leqq i < p$ and for $p+1 \leqq i < p+q$, $\varphi_p = \varepsilon_p$, and $\varphi_{p+q} = \varepsilon_{p+q}$. Thus the simple roots of $\mathfrak{g}$ have restriction to $\mathfrak{s}$ given by $\psi_i|_{\mathfrak{s}} = \varphi_i$ for $1 \leqq i < p$ and $p < i < p+q$, $\psi_p|_{\mathfrak{s}} = \varphi_p - \sum_1^q \varphi_{p+j}$, $\psi_{p+q}|_{\mathfrak{s}} = \varphi_{p+q}$, and $\psi_{p+q+1}|_{\mathfrak{s}} = \varphi_{p+q}$. Here is the relevant LiE routine branch_D_BB.lie for branching from $SO(2p+2q+2)$ to $SO(2p+1)\times SO(2q+1)$. It takes arguments (p,q,v), where $p,q \geqq 0$ are integers not both zero and v is a vector of length $p+q+1$ consisting of nonnegative integers, and it branches v.

(3.4)

```
# file branch_D_BB.lie                                        #
# usage: branch_D_BB(p,q,v) branches v from SO(2p+2q+2)      #
#         to SO(2p+1)xSO(2q+1)                                #
branch_D_BB(int p,q;vec v) = setdefault(Lie_group(4,p+q+1));
res_rt = null(p+q+1,p+q);
for i=1 to p-1 do res_rt[i,i] = 1 od;
res_rt[p,p] = 1; for j=p+1 to p+q do res_rt[p,j] = -1 od;
for i=p+1 to p+q-1 do res_rt[i,i] = 1 od;
res_rt[p+q,p+q] = 1; res_rt[p+q+1,p+q] = 1;
res_wt = i_Cartan*res_rt*Cartan(Lie_group(2,p)*\
         Lie_group(2,q))/4;
print("the branching of "+v+" from SO("+(2*p+2*q+2)+")");
print("                  to SO("+(2*p+1)+")xSO("+(2*q+1)+") is");
branch(v,Lie_group(2,p)*Lie_group(2,q),res_wt)
```

The same comment as just after (3.1) holds here when one wishes to set p and q in LiE: do that and replace the line

```
branch_D_BB(int p,q;vec v) = setdefault(Lie_group(4,p+q+1));
```

by the line

```
branch_D_BB(vec v) = setdefault(Lie_group(4,p+q+1)); .
```

3E. **Case $G/K = Spin(8)/G_2$.** Here $\mathfrak{g}$ has simple root system $\{\psi_1, \psi_2, \psi_3, \psi_4\}$ numbered as in the Introduction, and $\mathfrak{k}$ has simple root system $\{\varphi_1, \varphi_2\}$ as follows. The Cartan subalgebra $\mathfrak{s}$ of $\mathfrak{k}$ is the subspace of the Cartan subalgebra $\mathfrak{t}$ of $\mathfrak{g}$ given by $\psi_1 = \psi_3 = \psi_4$. See [10]. The root restrictions are $\psi_1|_\mathfrak{s} = \psi_3|_\mathfrak{s} = \psi_4|_\mathfrak{s} = \varphi_1$ (the short simple root of $\mathfrak{k}$) and $\psi_2|_\mathfrak{s} = \varphi_2$ (the long simple root of $\mathfrak{k}$). Now the Lie routine for branching from $Spin(8)$ to G_2 is

(3.5)
```
# file branch_D4_G2.lie #
branch_D4_G2(vec v) = setdefault(D4);
res_rt = [[1,0],[0,1],[1,0],[1,0]];
res_wt = (i_Cartan*res_rt*Cartan(G2))/det_Cartan;
print("the branching of "+v+" from Spin(8) to G2 is");
branch(v,G2,res_wt)
```

3F. **Cases $G/K = Spin(8)/SU(3)$, $G/K = E_6/F_4$ and $G/K = E_6/C_4$.** There the restriction matrices can be computed as in the case of $Spin(8)/G_2$, or one can use the small database of maximal subalgebras built into LiE. That database is accessible by the commands

`res_mat(A2,D4)`, `res_mat(F4,E6)` and `res_mat(C4,E6)`

The corresponding LiE routines are

(3.6)
```
# file branch_D4_A2.lie #
branch_D4_A2(vec v) = setdefault(D4);
print("the branching of "+v+" from D4 to the triality A2 is");
branch(v,A2,res_mat(A2,D4))
```

and

(3.7)
```
# file branch_E6_F4.lie #
branch_E6_F4(vec v) = setdefault(E6);
print("the branching of "+v+" from E6 to F4 is");
branch(v,F4,res_mat(F4,E6))
```

and

(3.8)
```
# file branch_E6_C4.lie #
branch_E6_C4(vec v) = setdefault(E6);
print("the branching of "+v+" from E6 to C4 is");
branch(v,C4,res_mat(C4,E6))
```

4. CASES: $\mathfrak{g}$ IS SIMPLE AND $\mathfrak{k}$ IS THE CENTRALIZER OF A TORAL SUBALGEBRA

These cases were covered in Section 2C. If $\mathfrak{k}$ is the centralizer of a toral subalgebra of $\mathfrak{g}$ it is the fixed point set of an automorphism. For if G is a connected Lie group with Lie algebra $\mathfrak{g}$ and K is the analytic subgroup for $\mathfrak{k}$ we can choose $g \in G$ such that the powers $\{g^n \mid n \in \mathbb{Z}\}$ form a dense subgroup of the identity component of the center of K, and then $\mathfrak{k}$ is the fixed point set of $\mathrm{Ad}(g)$.

5. Cases: $\mathfrak{g}$ is simple, $\theta^2 = 1$ and $\operatorname{rank}\mathfrak{k} = \operatorname{rank}\mathfrak{g}$

In this section we apply the method of Section 2D and run through the cases where $n_r = 2$, i.e. the cases where G/K is a Riemannian symmetric space. The other cases of equal rank will be considered in the next section.

It turns out that, in the classical group symmetric space cases, we do not have to renumber the roots of $\Psi_{\mathfrak{s}}$, while the renumbering is needed for most of the exceptional cases.

5A. **Case $(G, K) = (SO(2p + 2q + 1), SO(2p) \times SO(2q + 1))$ where $p \geq 2$ and $q \geq 0$.** Here $w_{\mathfrak{s}}$ reverses the order of $\psi_1, \ldots, \psi_{r-1}$ but does not move ψ_i for $i > r$, so $w_{\mathfrak{s}}(\beta_{\mathfrak{g}})$ attaches to the diagram of $\mathfrak{s}$ at ψ_{r-2}. Now the simple roots of $\mathfrak{k}$ in Bourbaki root order are given by $\{\psi_1, \ldots, \psi_{r-1}, w_{\mathfrak{s}}(\beta_{\mathfrak{g}}); \psi_{r+1}, \ldots, \psi_n\}$. Here is the LiE program `branch_B_DB.lie` that branches the representation of $SO(2p + 2q + 1)$, specified by a vector v of length $p + q$, to $SO(2p) \times SO(2q + 1)$. Usage is `branch_B_DB(p,q,v)`.

(5.1)
```
# file branch_B_DB.lie #
# usage: branch_B_DB(p,q,v) branches v from SO(2p+2q+1)    #
#                                        to SO(2p)xSO(2q+1) #
branch_B_DB(int p,q; vec v) = setdefault(Lie_group(2,p+q));
loc JJ = A1*A1; loc KK = A1; loc LL = Lie_group(4,p+q);
u = null(p+q-1); for i = 1 to p+q-1 do u[i]=i od;
if q > 0 then for i = p to p+q-1 do u[i]=i+1 od fi;
ws = reduce(long_word^r_reduce(long_word,u));
RR = id(p+q); RR[p] = W_rt_action(high_root,ws);
if p >= 3 then JJ = Lie_group(4,p) fi;
if q >= 2 then KK = Lie_group(2,q) fi;
if q > 0 then LL = JJ*KK fi;
print("the branching of "+v+" from SO("+(2*p+2*q+1)+")");
print("                 to SO("+2*p+")xSO("+(2*q+1)+") is ");
branch(v,LL,res_mat(RR))
```

Again, the comment just after (3.1) indicates the modification when p and q are already set in LiE.

5B. **Case $(G, K) = (SO(2p + 2q), SO(2p) \times SO(2q))$ where $p, q \geq 2$.** Again $w_{\mathfrak{s}}$ reverses the order of $\psi_1, \ldots, \psi_{r-1}$ but does not move ψ_i for $r < i \leqq n - 2$, so $w_{\mathfrak{s}}(\beta_{\mathfrak{g}})$ attaches to the diagram of $\mathfrak{s}$ at ψ_{r-2} (or doesn't attach, if $r = 2$). The simple roots of $\mathfrak{k}$ in Bourbaki root order are $\{\psi_1, \ldots, \psi_{r-1}, w_{\mathfrak{s}}(\beta_{\mathfrak{g}}), \psi_{r+1}, \ldots, \psi_n\}$. Here is the LiE program for restriction of representations from $SO(2p + 2q)$ to $SO(2p) \times SO(2q)$.

```
# file branch_D_DD.lie                                    #
# usage: branch_D_DD(p,q,v) branches v from SO(2p+2q)     #
#                                       to SO(2p)xSO(2q)  #
```

(5.2)

```
branch_D_DD(int p,q; vec v) = setdefault(Lie_group(4,p+q));
loc JJ = A1*A1; loc KK = A1*A1;
u = null(p+q-1); for i = 1 to p+q-1 do u[i]=i od;
for i = p to p+q-1 do u[i]=i+1 od;
ws = reduce(long_word^r_reduce(long_word,u));
RR = id(p+q); RR[p] = W_rt_action(high_root,ws);
if p >= 3 then JJ = Lie_group(4,p) fi;
if q >= 3 then KK = Lie_group(4,q) fi;
print("the branching of "+v+" from SO("+(2*p+2*q)+")");
print("              to SO("+2*p+")xSO("+(2*q)+") is ");
branch(v,JJ*KK,res_mat(RR))
```

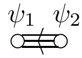
Again, the comment just after (3.1) indicates the modification when p and q are already set in LiE.

5C. **Case $(G, K) = (Sp(p+q), Sp(p)\times Sp(q))$ where $p, q \geq 1$.** Again $w_{\mathfrak{s}}$ reverses the order of $\psi_1, \ldots, \psi_{r-1}$ but does not move ψ_i for $i > r$, so $w_{\mathfrak{s}}(\beta_{\mathfrak{g}})$ attaches to the diagram of $\mathfrak{s}$ at ψ_{r-1} (or doesn't attach, if $r = 1$). Now the simple roots of $\mathfrak{k}$ in Bourbaki root order are $\{\psi_1, \ldots, \psi_{r-1}, w_{\mathfrak{s}}(\beta_{\mathfrak{g}}), \psi_{r+1}, \ldots, \psi_n\}$. The LiE program for restriction of representations from $Sp(p+q)$ to $Sp(p)\times Sp(q)$ is

(5.3)

```
# file branch_C_CC.lie                                       #
# usage: branch_C_CC(p,q,v) branches v from Sp(p+q)          #
#                                        to Sp(p)xSp(q),     #
branch_C_CC(int p,q; vec v) = setdefault(Lie_group(3,p+q));
loc JJ = A1; loc KK = A1;
u = null(p+q-1);
if p == 1 then for i = 1 to q do u[i]=i+1 od fi;
if p >= 2 then for i = 1 to p+q-1 do u[i]=i od fi;
if p >= 2 then for i = p to p+q-1 do u[i]=i+1 od fi;
ws = reduce(long_word^r_reduce(long_word,u));
RR = id(p+q); RR[p] = W_rt_action(high_root,ws);
if p >= 2 then JJ = Lie_group(3,p) fi;
if q >= 2 then KK = Lie_group(3,q) fi;
print("the branching of "+v+" from Sp("+(p+q)+")");
print("               to Sp("+p+")xSp("+q+") is ");
branch(v,JJ*KK,res_mat(RR))
```

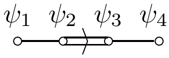
Again, the comment just after (3.1) indicates the modification when p and q are already set in LiE.

5D. **Case G_2/A_1A_1.** Now we run through the exceptional group cases. First suppose that $G = G_2$. Then $K = A_1A_1$ with simple roots 

. Here $\beta_{\mathfrak{g}} = 3\psi_1 + 2\psi_2$ and $\gamma = \psi_2$, $\Psi_{\mathfrak{s}} = \{\psi_1\}$, and $w_{\mathfrak{s}}(\beta_{\mathfrak{g}}) = \beta_{\mathfrak{g}}$. Thus the LiE program (if one wants to bother with it in this case) is

(5.4)

```
# file branch_G2_A1A1.lie #
branch_G2_A1A1(vec v) = setdefault(G2);
ws = reduce(long_word^r_reduce(long_word,[1]));
RR = id(2); RR[2] = W_rt_action(high_root,ws);
print("the branching of "+v+" from G2 to A1A1 is");
branch(v,A1A1,res_mat(RR))
```

5E. **Cases F_4/A_1C_3 and F_4/B_4.** Next suppose that $G = F_4$. The simple root system is 

and the negative of the maximal root $\beta_{\mathfrak{g}} = 2\psi_1 + 3\psi_2 + 4\psi_3 + 2\psi_4$ attaches at ψ_1. Thus there are two possibilities: $\gamma = \psi_1$ and $K = A_1C_3$, or $\gamma = \psi_4$ and $K = B_4$. The corresponding LiE programs are given by

(5.5)

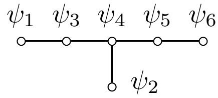
```
# file branch_F4_A1C3.lie #
branch_F4_A1C3(vec v) = setdefault(F4);
ws = reduce(long_word^r_reduce(long_word,[2,3,4]));
RR = id(4); RR[1] = W_rt_action(high_root,ws);
RR[2] = id(4)[4]; RR[4] = id(4)[2];
print("the branching of "+v+" from F4 to A1C3 is");
branch(v,A1C3,res_mat(RR))
```

and

(5.6)

```
# file branch_F4_B4.lie #
branch_F4_B4(vec v) = setdefault(F4);
ws = reduce(long_word^r_reduce(long_word,[1,2,3]));
RR = id(4); RR[1] = W_rt_action(high_root,ws);
RR[2] = id(4)[1]; RR[3] = id(4)[2]; RR[4] = id(4)[3];
print("the branching of "+v+" from F4 to B4 is");
branch(v,B4,res_mat(RR))
```

5F. **Cases E_6/A_1A_5 and E_6/A_5A_1.** Now suppose that $G = E_6$. Then there are three simple roots of coefficient 2 in the maximal root. All of them differ by automorphisms of the extended Dynkin diagram, so the corresponding subalgebras $\mathfrak{k}$ differ by an automorphism, but in [5] we will need to distinguish between them, so we treat them separately. The simple root system is



and the negative of the maximal root $\beta_{\mathfrak{g}} = \psi_1 + 2\psi_2 + 2\psi_3 + 3\psi_4 + 2\psi_5 + \psi_6$ attaches at ψ_2. There are three equivalent possibilities: (i) $\gamma = \psi_3$ and $K = A_1A_5$, (ii) $\gamma = \psi_5$ and $K = A_5A_1$, and (iii) $\gamma = \psi_2$ and $K = A_5A_1$. For the first, the LiE routine is

(5.7)
```
# file branch_E6_A1A5.lie #
branch_E6_A1A5(vec v) = setdefault(E6);
ws = reduce(long_word^r_reduce(long_word,[1,2,4,5,6]));
RR = id(6); RR[6] = W_rt_action(high_root,ws);
RR[3] = id(6)[4]; RR[4] = id(6)[5]; RR[5] = id(6)[6];
print("the branching of "+v+" from E6 to A1A5 is");
branch(v,A1A5,res_mat(RR))
```

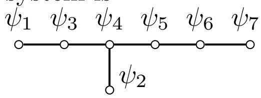
The second is quite similar,

(5.8)
```
# file branch_E6_A5A1.lie #
branch_E6_A5A1(vec v) = setdefault(E6);
ws = reduce(long_word^r_reduce(long_word,[1,2,3,4,6]));
RR = id(6); RR[1] = W_rt_action(high_root,ws);
RR[2] = id(6)[1]; RR[5] = id(6)[2];
print("the branching of "+v+" from E6 to A5A1 is");
branch(v,A5A1,res_mat(RR))
```

and the third is a bit different,

(5.9)
```
# file branch_E6_A5A1a.lie #
branch_E6_A5A1a(vec v) = setdefault(E6);
ws = reduce(long_word^r_reduce(long_word,[1,3,4,5,6]));
RR = id(6); RR[6] = W_rt_action(high_root,ws);
RR[2] = id(6)[3]; RR[3] = id(6)[4]; RR[4] = id(6)[5];
RR[5] = id(6)[6];
print("the branching of "+v+" from E6 to A5A1 is");
branch(v,A5A1,res_mat(RR))
```

5G. **Cases E_7/A_1D_6, E_7/D_6A_1 and E_7/A_7.** Next, let $G = E_7$. Then there are three simple roots of coefficient 2 in the maximal root. Two of them differ by an automorphism of the extended Dynkin diagram, so the corresponding subalgebras $\mathfrak{k}$ differ by an automorphism, but we will need to distinguish between them in [5]. So, as in some of the E_6 cases above, we treat them separately. The simple root system is



and the negative of the maximal root $\beta_{\mathfrak{g}} = 2\psi_1+2\psi_2+3\psi_3+4\psi_4+3\psi_5+2\psi_6+\psi_7$ attaches at ψ_1. The possibilities are (i) $\gamma = \psi_1$ and $K = A_1D_6$, (ii) $\gamma = \psi_6$ and $K = D_6A_1$, and (iii) $\gamma = \psi_2$ and $K = A_7$. For the first of these the LiE routine is

(5.10)

```
# file branch_E7_A1D6.lie #
branch_E7_A1D6(vec v) = setdefault(E7);
ws = reduce(long_word^r_reduce(long_word,[2,3,4,5,6,7]));
RR = id(7); RR[1] = W_rt_action(high_root,ws);
RR[2] = id(7)[7]; RR[3] = id(7)[6]; RR[4] = id(7)[5];
RR[5] = id(7)[4]; RR[6] = id(7)[3]; RR[7] = id(7)[2];
print("the branching of "+v+" from E7 to A1D6 is");
branch(v,A1D6,res_mat(RR))
```

For obvious reasons the second is similar

(5.11)

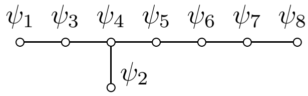
```
# file branch_E7_D6A1.lie #
branch_E7_D6A1(vec v) = setdefault(E7);
ws = reduce(long_word^r_reduce(long_word,[1,2,3,4,5,7]));
RR = id(7); RR[1] =  W_rt_action(high_root,ws);
RR[2] = id(7)[1]; RR[6] = id(7)[2];
print("the branching of "+v+" from E7 to D6A1 is");
branch(v,D6A1,res_mat(RR))
```

and the third is a bit different

(5.12)

```
# file branch_E7_A7.lie #
branch_E7_A7(vec v) = setdefault(E7);
ws = reduce(long_word^r_reduce(long_word,[1,3,4,5,6,7]));
RR = id(7); RR[7] = W_rt_action(high_root,ws);
RR[2] = id(7)[3]; RR[3] = id(7)[4]; RR[4] = id(7)[5];
RR[5] = id(7)[6]; RR[6] = id(7)[7];
print("the branching of "+v+" from E7 to A7 is");
branch(v,A7,res_mat(RR))
```

5H. **Cases E_8/D_8 and E_8/E_7A_1.** Finally, suppose that $G = E_8$. The simple root system is



and the negative of the maximal root $\beta_{\mathfrak{g}} = 2\psi_1 + 3\psi_2 + 4\psi_3 + 6\psi_4 + 5\psi_5 + 4\psi_6 + 3\psi_7 + 2\psi_8$ attaches at ψ_8. The possibilities are (i) $\gamma = \psi_1$ and $K = D_8$ and (ii) $\gamma = \psi_8$ and $K = E_7A_1$. In the first case the LiE routine is

(5.13)

```
# file branch_E8_D8.lie #
branch_E8_D8(vec v) = setdefault(E8);
ws = reduce(long_word^r_reduce(long_word,[2,3,4,5,6,7,8]));
RR = id(8); RR[1] = W_rt_action(high_root,ws);
RR[2] = id(8)[8]; RR[3] = id(8)[7]; RR[4] = id(8)[6];
RR[6] = id(8)[4]; RR[7] = id(8)[3]; RR[8] = id(8)[2];
print("the branching of "+v+" from E8 to D8 is");
branch(v,D8,res_mat(RR))
```

and in the second it is

(5.14)
```
# file branch_E8_E7A1.lie #
branch_E8_E7A1(vec v) = setdefault(E8);
ws = reduce(long_word^r_reduce(long_word,[1,2,3,4,5,6,7]));
RR = id(8); RR[8] = W_rt_action(high_root,ws);
print("the branching of "+v+" from E8 to E7A1 is");
branch(v,E7A1,res_mat(RR))
```

Taking into account the results of Sections 3 and 4, now for every compact connected, simply connected symmetric space G/K, G simple, we have shown how to compute the restriction to K of any irreducible finite dimensional representation of G. The irreducible compact connected, simply connected symmetric spaces G/K with $\mathfrak{g}$ not simple are the simply connected simple Lie group manifolds, which were covered in Section 2A.

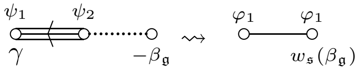
6. Cases: $\mathfrak{g}$ is simple, $\theta^3 = 1$ or $\theta^5 = 1$, and $\operatorname{rank}\mathfrak{k} = \operatorname{rank}\mathfrak{g}$

The maximal connected subgroups of maximal rank in a compact connected Lie group were described by A. Borel and J. de Siebenthal [2]. Most of them are symmetric subgroups, and their classification can be used in the classification of symmetric spaces [12]. The ones that are symmetric were considered in Section 5. The others correspond to the simple roots γ whose coefficient in the maximal root $\beta_{\mathfrak{g}}$ is an odd prime, necessarily 3 or 5. The ones for prime 3 are given by $(G,K) = (G_2, A_2)$, (F_4, A_2A_2), $(E_6, A_2A_2A_2)$, (E_7, A_2A_5), (E_7, A_5A_2), (E_8, A_8), and (E_8, E_6A_2). The one for prime 5 is given by $(G,K) = (E_8, A_4A_4)$. In all cases a simple root system for $\mathfrak{k}$ is given by $\Psi_{\mathfrak{k}} = (\Psi_{\mathfrak{g}} \setminus \{\gamma\}) \cup \{-\beta_{\mathfrak{g}}\}$, so we can use the methods of Section 2C as in Section 5.

In all of these cases one can rely on a LiE database to produce a restriction matrix `res_mat(Y,X)`. However the applications in [5] require that we keep track of which root of `Y` comes from which root of `X`, and LiE scrambles the root order, so we generally have to do this by hand. In each case we indicate which elements of the simple root system $\Phi = \{\varphi_1, \dots, \varphi_n\}$ of $\mathfrak{k}$ come from which elements of $(\Psi \setminus \{\gamma\}) \cup \{w_{\mathfrak{s}}(\beta_{\mathfrak{g}})\}$, where we try to use the least complicated correspondence.

6A. Case $G/K = G_2/A_2$.

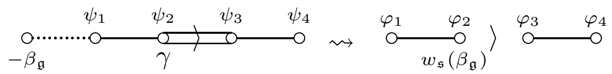
Here $\gamma = \psi_1$, $\mathfrak{k}$ is of type $A_2 = \mathfrak{su}(3)$, and $\Phi = \{\varphi_1, \varphi_2\}$ where φ_1 comes from $w_{\mathfrak{s}}(\beta_{\mathfrak{g}})$ and φ_2 comes from ψ_2. This is indicated in terms of the Dynkin diagrams, by



The corresponding LiE routine is

(6.1)
```
# file branch_G2_A2.lie #
branch_G2_A2(vec v) = setdefault(G2);
ws = reduce(long_word^r_reduce(long_word,[2]));
RR = id(2); RR[1] = W_rt_action(high_root,ws);
print("the branching of "+v+" from G2 to A2 is ");
branch(v,A2,res_mat(RR))
```

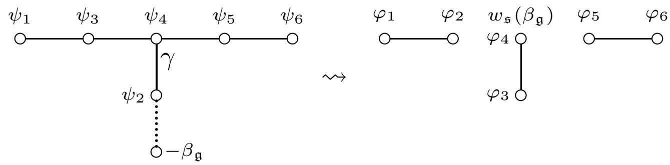
6B. **Case $G/K = F_4/A_2A_2$.** Here $\gamma = \psi_2$, $\mathfrak{k}$ is of type A_2A_2, and $\Phi = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4\}$ where φ_1 comes from ψ_1, φ_2 comes from $w_{\mathfrak{s}}(\beta_{\mathfrak{g}})$, φ_3 comes from ψ_3, and φ_4 comes from ψ_4. This is indicated in terms of Dynkin diagrams by



Here the isolated $\rangle$ indicates that φ_1 and φ_2 are long while φ_3 and φ_4 are short. The corresponding LiE routine is

(6.2)
```
# file branch_F4_A2A2.lie #
branch_F4_A2A2(vec v) = setdefault(F4);
ws = reduce(long_word^r_reduce(long_word,[1,3,4]));
RR = id(4); RR[2] = W_rt_action(high_root,ws);
print("the branching of "+v+" from F4 to A2A2 is ");
branch(v,A2A2,res_mat(RR))
```

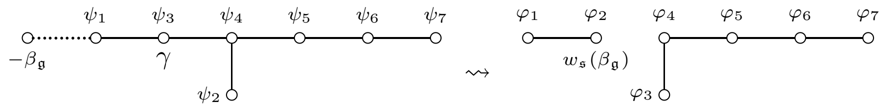
6C. **Case $G/K = E_6/A_2A_2A_2$.** Here $\gamma = \psi_4$, $\mathfrak{k}$ is of type $A_2A_2A_2$, and $\Phi = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6\}$ where φ_1 comes from ψ_1, φ_2 comes from ψ_3, φ_3 comes from ψ_2, φ_4 comes from $w_{\mathfrak{s}}(\beta_{\mathfrak{g}})$, φ_5 comes from ψ_5, and φ_6 comes from ψ_6. This is indicated in terms of Dynkin diagrams by



The corresponding LiE routine is

(6.3)
```
# file branch_E6_A2A2A2.lie #
branch_E6_A2A2A2(vec v) = setdefault(E6);
ws = reduce(long_word^r_reduce(long_word,[1,2,3,5,6]));
RR = id(6); RR[4] = W_rt_action(high_root,ws);
RR[2] = id(6)[3]; RR[3] = id(6)[2];
print("the branching of "+v+" from E6 to A2A2A2 is");
branch(v,A2A2A2,res_mat(RR))
```

6D. **Case $G/K = E_7/A_2A_5$.** Here $\gamma = \psi_3$, $\mathfrak{k}$ is of type A_2A_5, and $\Phi = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \varphi_7\}$ where φ_1 comes from ψ_1, φ_2 comes from $w_{\mathfrak{s}}(\beta_{\mathfrak{g}})$, φ_3 comes from ψ_2, φ_4 comes from ψ_4, φ_5 comes from ψ_5, φ_6 comes from ψ_6, and φ_7 comes from ψ_7. This is indicated in terms of Dynkin diagrams by

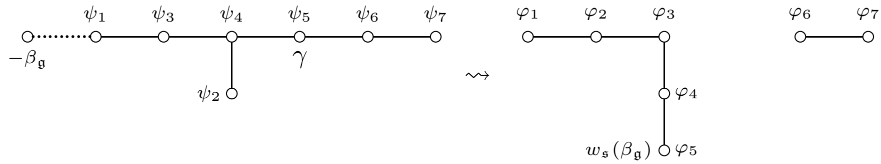


The corresponding LiE routine is

(6.4)

```
# file branch_E7_A2A5.lie #
branch_E7_A2A5(vec v) = setdefault(E7);
ws = reduce(long_word^r_reduce(long_word,[1,2,4,5,6,7]));
RR = id(7); RR[2] = W_rt_action(high_root,ws);
    RR[3] = id(7)[2];
print("the branching of "+v+" from E7 to A2A5 is");
branch(v,A2A5,res_mat(RR))
```

6E. **Case $G/K = E_7/A_5A_2$.** Here $\gamma = \psi_5$, $\mathfrak{k}$ is of type A_5A_2, and $\Phi = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \varphi_7\}$ where φ_1 comes from ψ_1, φ_2 comes from ψ_3, φ_3 comes from ψ_4, φ_4 comes from ψ_2, φ_5 comes from $w_{\mathfrak{s}}(\beta_{\mathfrak{g}})$, φ_6 comes from ψ_6, and φ_7 comes from ψ_7. This is indicated in terms of Dynkin diagrams by

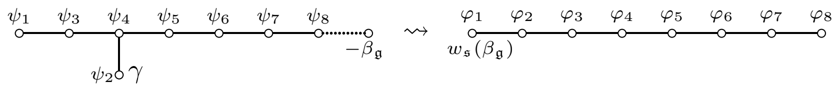


The corresponding LiE routine is

(6.5)

```
# file branch_E7_A5A2.lie #
branch_E7_A5A2(vec v) = setdefault(E7);
ws = reduce(long_word^r_reduce(long_word,[1,2,3,4,6,7]));
RR = id(7); RR[5] = W_rt_action(high_root,ws);
RR[2] = id(7)[3]; RR[3] = id(7)[4]; RR[4] = id(7)[2];
print("the branching of "+v+" from E7 to A5A2 is");
branch(v,A5A2,res_mat(RR))
```

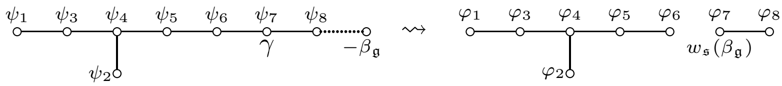
6F. **Case $G/K = E_8/A_8$.** Here $\gamma = \psi_2$, $\mathfrak{k}$ is of type A_8. The simple root system of $\mathfrak{k}$ is $\Phi = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \varphi_7, \varphi_8\}$ where φ_1 comes from $w_{\mathfrak{s}}(\beta_{\mathfrak{g}})$, φ_2 comes from ψ_1, φ_3 comes from ψ_3, φ_4 comes from ψ_4, φ_5 comes from ψ_5, φ_6 comes from ψ_6, φ_7 comes from ψ_7, and φ_8 comes from ψ_8. This is indicated in terms of Dynkin diagrams by



The corresponding LiE routine is

(6.6)
```
# file branch_E8_A8.lie #
branch_E8_A8(vec v) = setdefault(E8);
ws = reduce(long_word^r_reduce(long_word,[1,3,4,5,6,7,8]));
RR = id(8); RR[1] = W_rt_action(high_root,ws);
RR[2] = id(8)[1];
print("the branching of "+v+" from E8 to A8 is");
branch(v,A8,res_mat(RR))
```

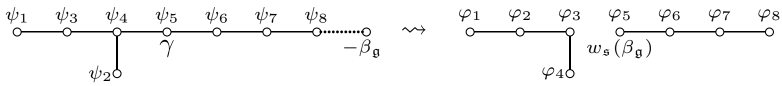
6G. **Case $G/K = E_8/E_6A_2$.** Here $\gamma = \psi_7$, $\mathfrak{k}$ is of type E_6A_2, and $\Phi = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \varphi_7, \varphi_8\}$ where φ_1 comes from ψ_1, φ_2 comes from ψ_2, φ_3 comes from ψ_3, φ_4 comes from ψ_4, φ_5 comes from ψ_5, φ_6 comes from ψ_6, φ_7 comes from $w_\mathfrak{s}(\beta_\mathfrak{g})$, and φ_8 comes from ψ_8. This is indicated in terms of Dynkin diagrams by



The corresponding LiE routine is

(6.7)
```
# file branch_E8_E6A2.lie #
branch_E8_E6A2(vec v) = setdefault(E8);
ws = reduce(long_word^r_reduce(long_word,[1,2,3,4,5,6,8]));
RR = id(8); RR[7] = W_rt_action(high_root,ws);
print("the branching of "+v+" from E8 to E6A2 is");
branch(v,E6A2,res_mat(RR))
```

6H. **Case $G/K = E_8/A_4A_4$.** Here $\gamma = \psi_5$, $\mathfrak{k}$ is of type A_4A_4, and $\Phi = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \varphi_7, \varphi_8\}$ where φ_1 comes from ψ_1, φ_2 comes from ψ_3, φ_3 comes from ψ_4, φ_4 comes from ψ_2, φ_5 comes from $w_\mathfrak{s}(\beta_\mathfrak{g})$, φ_6 comes from ψ_6, φ_7 comes from ψ_7, and φ_8 comes from ψ_8. This is indicated in terms of Dynkin diagrams by



The corresponding LiE routine is

(6.8)
```
# file branch_E8_A4A4.lie #
branch_E8_A4A4(vec v) = setdefault(E8);
ws = reduce(long_word^r_reduce(long_word,[1,2,3,4,6,7,8]));
RR = id(8); RR[5] = W_rt_action(high_root,ws);
RR[2] = id(8)[3]; RR[3] = id(8)[4]; RR[4] = id(8)[2];
print("the branching of "+v+" from E8 to A4A4 is");
branch(v,A4A4,res_mat(RR))
```

This completes our branching project as described in Section 2.

References

[1] R. J. Baston and M. G. Eastwood, *The Penrose transform*, Oxford Mathematical Monographs, Oxford Univ. Press, New York, 1989. MR1038279 (92j:32112)

[2] A. Borel and J. de Siebenthal, Les sous-groupes fermés de rang maximum des groupes de Lie clos, Comment. Math. Helv. **23** (1949), 200–221. MR0032659 (11,326d)

[3] N. Bourbaki, *Éléments de mathématique. Fasc. XXXIV. Groupes et algèbres de Lie. Chapitre IV: Groupes de Coxeter et systèmes de Tits. Chapitre V: Groupes engendrés par des réflexions. Chapitre VI: systèmes de racines*, Actualités Scientifiques et Industrielles, No. 1337 Hermann, Paris, 1968. MR0240238 (39 #1590)

[4] A. Čap and J. Slovák, *Parabolic geometries. I.* Background and general theory. Mathematical Surveys and Monographs, **154**. American Mathematical Society, Providence, RI, 2009. MR2532439 (2010j:53037)

[5] M. G. Eastwood and J. A. Wolf, The range of the double fibration transform. To appear.

[6] G. Fels, A. T. Huckleberry and J. A. Wolf, *Cycle spaces of flag domains.* A complex geometric viewpoint. Progress in Mathematics, **245**. Birkhäuser Boston, Inc., Boston, MA, 2006. MR2188135 (2006h:32018)

[7] A. Gray and J. A. Wolf, Homogeneous spaces defined by Lie group automorphisms, I, J. Differential Geometry **2** 1968, 77–114. MR0236328 (38 #4625a)

[8] A. Gray and J. A. Wolf, Homogeneous spaces defined by Lie group automorphisms, II, J. Differential Geometry **2** 1968, 115–159. MR0236329 (38 #4625b)

[9] LiE program, Computer Algebra Group of CWI, software project headed by Arjeh M. Cohen, now maintained by Marc van Leeuwen. Version 2.2 of LiE is available at `http://wwwmathlabo.univ-poitiers.fr/~maavl/LiE/`

[10] J. de Siebenthal, Sur les sous-groupes fermés connexes d'un groupe de Lie clos, Comment. Math. Helv. **25** (1951), 210–256. MR0043790 (13,319a)

[11] J. de Siebenthal, Sur les groupes de Lie compacts non connexes, Comment. Math. Helv. **31** (1956), 41–89. MR0094408 (20 #926)

[12] J. A. Wolf, *Spaces of constant curvature*, sixth edition, AMS Chelsea Publishing, Providence, RI, 2011. MR2742530 (2011j:53001)

Received September 11, 2009; accepted May 21, 2010

Michael G. Eastwood
Mathematical Sciences Institute
Australian National University
ACT 0200, Australia
E-mail: `meastwoo@member.ams.org`

Joseph A. Wolf
Department of Mathematics
University of California
Berkeley, CA 94720–3840, USA
E-mail: `jawolf@math.berkeley.edu`