The MOP Manual

Ulf Jürgens and Gerhard Röhrle

November 26, 2001

Contents

Chapter 1

MOP Version 1 – a short introduction

MOP is a joint project of Ulf Jürgens and Gerhard Röhrle, written entirely in the GAP language. It is a GAP share package based on the CHEVIE package. The source file containing the program and data is built up in a similar fashion as the files in GAP, it contains a copyright notice and the names of the MOP authors.

MOP is designed to compute upper bounds for the modality of the action of parabolic subgroups P on $\mathfrak{p}_u$, the Lie algebra of the unipotent radical of P, for simple algebraic groups G over an algebraically closed field K of characteristic $\mathrm{char} K \geq 0$. Concerning the notion of the modality of an action of an algebraic group and other references hereto we refer to [PR97]. MOP can be applied provided G has a simply laced Dynkin diagram.

MOP has a single front end function `Modality`; see chapter 2. It computes a record which contains an upper bound for the modality of P.

In view of the classification results of modality zero parabolic subgroups P in classical groups [HR99] (i.e., those with a finite number of orbits on $\mathfrak{p}_u$), MOP was designed to specifically calculate the modality of parabolic subgroups of exceptional algebraic groups.

We have used MOP to partially classify parabolic subgroups of exceptional algebraic groups of modality zero, [JR]. Further applications of MOP where cases of parabolic groups of higher modality are determined can also be found in [JR].

This manual aims at a reader familiar with the mathematical aspects and the features of MOP as outlined in [JR].

October 2001

Chapter 2

MOP Functions

2.1 Modality

MOP has a single front end function

`Modality(`*type*`, `*rank*`, `*J*`, [rec(...)])`.

It computes from this input data a record which contains an upper bound for the modality of the action of P_J on the Lie algebra of its unipotent radical. Apart from that the output record contains various other pieces of relevant information, see section 2.2 below.

We allow for instances of type A, D, or E only.

The first three parameters are mandatory: *type* indicates the type of a Dynkin diagram (A, D, or E), the *rank* of G and the subset J of $\{1, ..., rank\}$ which defines the parabolic subgroup P. The fourth parameter is an optional record. It is used to overwrite existing items of the MOP record. Concerning the meaning of these features, see [JR].

The following optional parameters are available:

- LowerBound (default value 0)

  If a lower bound (other than zero) for the modality of the action of P on $\mathfrak{p}_u$ is known for theoretical reasons, this variable should be set to make the rank criterion effective; see the example in E_6 in section 2.3 below.

- UseInductionList (default value *true*)

  If this variable is set to *true*, the program uses the induction list.

- MOP.MaxEliminationTime (default value 60)

  This parameter determines the maximum number of seconds which are spent on one call of the internal function `MOP.Elimination`.

- SaveInterval (default value 20)

  This optional parameter defines the number of minutes after which the status of the program execution is supposed to be saved. The actual saving may occur considerably later, because the program execution must first return from all function calls and reach the main loop in `Modality`.

- startstring (default value rec())

  This variable defines a start string used to initialize the stack. It must be a record of the form rec(m:=*list1*,k:=*list2*), where the lists "list1"and "list2"are the numbers of the roots defining the positions labelled with "M"or "K", respectively. If this parameter is not provided, then it is computed from the following:

- m (default value [])

  k (default value `MOP.PositionsRootsInRadical`)

  These lists define the support of the start string by simple lists of numbers of roots which are labelled with "M"or "K", respectively, representing a corresponding start string. The default values of the lists represent $\mathfrak{p}_u$, the Lie algebra of the unipotent radical of the parabolic P.

- Verbose (default value 0)

  In the default setting the algorithm does not provide any intermediate results. Setting this parameter to positive integer values (between 1 and 4) forces MOP to print various pieces of information of the analysis. The higher the value, the more details are given. In principle this allows a complete check by the user; see chapter 3 for an example.

2.2 The output record

As illustrated in our example below, MOP computes a record with the following record fields that contain the result of the analysis; for further details see [JR]:

Counter is itself a record of various counters. For instance, here we keep track how frequently the various deletion criteria are applied, we count the number of splittings and the frequency of a call of the internal function `MOP.ExtendedOperation`.

LowerBound is the best lower bound for the modality of P known a priori.

CannotAnalyzeList contains all strings that did not satisfy any of the elimination criteria. The cardinality of this list is printed in the output in the line `unresolved strings`. If after the analysis is complete this list is empty, then the `LowerBound` and the upper bound for the modality of P computed by MOP agree. Else MOP computes an upper bound for the modality of P from the strings in this list.

factorlist is a list of primes. The result of the modality analysis is valid provided the characteristic of the underlying algebracially closed field K is not in this list. These primes stem from solving certain systems of equations involving root operators. This set of primes is listed in the output in the line `characteristic restrictions`.

Type is the type of G.

J is the set of simple roots defining P.

W is the record consisting of the Weyl group of G.

WJ is the record consisting of the Weyl group of P.

MaxModality is the upper bound for the modality of the action of P computed by MOP.

SubDiagrams is a list of configurations of subsystems stemming from the use of the induction list.

SubDiagramsDetail the list of all embeddings of subsystems leading to the configurations recorded in SubDiagrams.

operations contains the Print function.

2.3 An Example

We illustrate a function call with an example where G is of type E_6 and the parabolic subgroup $P = P_J$ is defined by the set of simple roots with the labels $J = \{1, 2, 6\}$. It follows from conceptual results in [Röh97] that in this instance the modality of P is at least 2. Consequently, when calling the function `Modality`, we call it with the optional parameter `LowerBound:=2`. As it turns out the modality in this instance is in fact exactly 2, as computed by MOP.

After the run of MOP we show the different record fields of the record that is computed by MOP, and show part of the contents of the `Subdiagrams` record field in this instance. Here the distinct occurring types are listed together with their parabolic subgroups, again labelled by `J`, e.g., `A1xA5 J=[ 1, 3, 5 ]`, see the section on induction in [JR].

```
gap> RequirePackage("mop");

   WELCOME to the CHEVIE package, Version 3 (Dec 1996)

      Meinolf  Geck,  Frank Luebeck,  Gerhard  Hiss,  Gunter Malle,  Jean
      Michel,  and  Goetz Pfeiffer,  Lehrstuhl  D  fuer Mathematik,  RWTH
      Aachen, IWR der Universitaet  Heidelberg, University of St. Andrews
      and Universite Paris VII

   This replaces the former weyl package. For first help type

          ?CHEVIE Version 3 -- a short introduction


   WELCOME to the MOP package, Version 1 (Oct 2001)
      Ulf Juergens, Gerhard Roehrle

gap> r:=Modality("E", 6, [1,2,6], rec(LowerBound:=2));
option LowerBound:=2
Modality analysis for type E6, J = [ 1, 2, 6 ]
E6          2
            |
    1 - 3 - 4 - 5 - 6
```

```
--------------------------------------------
1545 strings analyzed
772 splittings
--------------------------------------------
31 induction list matches
163 already done
579 occurrences of rank condition
0 J-Height criterion invoked
--------------------------------------------
0 extended operations
--------------------------------------------
0 unresolved strings
characteristic restrictions: [ 2, 3 ]
The modality of P is 2.
--------------------------------------------

gap> RecFields(r);
[ "Counter", "CannotAnalyzeList", "factorlist", "Type",  "J",
"LowerBound", "W", "WJ", "MaxModality", "SubDiagrams",
"SubDiagramsDetail", "operations" ]

gap> RecFields(r.Counter);
[ "Rank", "Extended", "Redundancy", "Split", "String",
"InductionList", "JHeight" ]

gap> r.SubDiagrams;
[ A1xA5 J=[  ]
    , D5 J=[ 1 ]
    , D5 J=[ 1, 2, 4 ]
    , A1xA5 J=[ 1, 2, 5 ]
    , D5 J=[ 1, 2, 5 ]
    , A1xA5 J=[ 1, 3, 5 ]
    , A1xA5 J=[ 1, 3, 6 ]
    , D5 J=[ 1, 4 ]
    , D5 J=[ 1, 5 ]
    , D5 J=[ 2 ]
    , A1xA5 J=[ 2, 4 ]
    , D5 J=[ 2, 4 ]
    , A1xA5 J=[ 2, 4, 6 ]
    , A1xA5 J=[ 2, 5 ]
...
```

Chapter 3

The Verbose Feature

3.1 An example in the verbose mode

We illustrate and explain the output of MOP in the verbose mode for the case when G is of type E_6 and $P = P_J$, where $J = \{1, 2, 4, 6\}$. This is one of the crucial modality zero parabolic instances needed for the classification, see Theorem 3.1 in [JR]. Most of the output is more or less self-explanatory. We chose to run this case with the verbose setting `Verbose:=3`. This case requires a higher setting for the time spent on the internal function `MOP.Elimination`; here we set `MaxEliminationTime:=600`.

```
gap> RequirePackage("mop");

   WELCOME  to  the  CHEVIE  package,  Version 3  (Dec  1996)

      Meinolf Geck,  Frank Luebeck,   Gerhard Hiss,
      Gunter Malle,  Jean Michel, and Goetz Pfeiffer,
          Lehrstuhl D fuer Mathematik, RWTH Aachen,
          IWR der Universitaet Heidelberg,
          University of St. Andrews and
          Universite Paris VII

   This replaces the former weyl package. For first help type

          ?CHEVIE Version 3 -- a short introduction


   WELCOME to the MOP package, Version 1 (Oct 2001)
      Ulf Juergens, Gerhard Roehrle

gap> r:=Modality("E", 6, [1,2,4,6], rec(MaxEliminationTime:=600, Verbose:=3));
gap> option MaxEliminationTime:=600
option  Verbose:=3
```

```
1: [ 1, 0, 0, 0, 0, 0 ]
2: [ 0, 1, 0, 0, 0, 0 ]
3: [ 0, 0, 1, 0, 0, 0 ]
4: [ 0, 0, 0, 1, 0, 0 ]
5: [ 0, 0, 0, 0, 1, 0 ]
6: [ 0, 0, 0, 0, 0, 1 ]
7: [ 1, 0, 1, 0, 0, 0 ]
8: [ 0, 1, 0, 1, 0, 0 ]
9: [ 0, 0, 1, 1, 0, 0 ]
10: [ 0, 0, 0, 1, 1, 0 ]
11: [ 0, 0, 0, 0, 1, 1 ]
12: [ 1, 0, 1, 1, 0, 0 ]
13: [ 0, 1, 1, 1, 0, 0 ]
14: [ 0, 1, 0, 1, 1, 0 ]
15: [ 0, 0, 1, 1, 1, 0 ]
16: [ 0, 0, 0, 1, 1, 1 ]
17: [ 1, 1, 1, 1, 0, 0 ]
18: [ 1, 0, 1, 1, 1, 0 ]
19: [ 0, 1, 1, 1, 1, 0 ]
20: [ 0, 1, 0, 1, 1, 1 ]
21: [ 0, 0, 1, 1, 1, 1 ]
22: [ 1, 1, 1, 1, 1, 0 ]
23: [ 1, 0, 1, 1, 1, 1 ]
24: [ 0, 1, 1, 2, 1, 0 ]
25: [ 0, 1, 1, 1, 1, 1 ]
26: [ 1, 1, 1, 2, 1, 0 ]
27: [ 1, 1, 1, 1, 1, 1 ]
28: [ 0, 1, 1, 2, 1, 1 ]
29: [ 1, 1, 2, 2, 1, 0 ]
30: [ 1, 1, 1, 2, 1, 1 ]
31: [ 0, 1, 1, 2, 2, 1 ]
32: [ 1, 1, 2, 2, 1, 1 ]
33: [ 1, 1, 1, 2, 2, 1 ]
34: [ 1, 1, 2, 2, 2, 1 ]
35: [ 1, 1, 2, 3, 2, 1 ]
36: [ 1, 2, 2, 3, 2, 1 ]
computing InductionList
number of subsystems of type [ D5 ]: 27
number of elements: 27
number of subsystems of type [ A1, A5 ]: 36
number of elements: 36
Length of induction list = 63
string with 31 entries
M [  ]
K [ 3, 5, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,
  25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36 ]
```

```
analyzing
M [  ]
K [ 3, 5, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,
  25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36 ]

splitting    3
analyzing
M [  ]
K [ 5, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
  26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36 ]

splitting    5
...
```

As shown, first MOP assigns root numbers to all the positive roots; all subsequent information refers to this numbering. Next it computes the induction list and initializes the stack with the string corresponding to $\mathfrak{p}_u$, the first string printed, rather its support, here the first string with 31 entries labelled "K". Then MOP proceeds with splittings. When it takes a string off the stack, MOP prints `analyzing` followed by the root numbers of the support of the string.

Later in the analysis we encounter the following.

```
...
analyzing
M [  ]
K [ 30, 31, 32, 33, 34, 35, 36 ]

splitting    30
analyzing
M [  ]
K [ 31, 32, 33, 34, 35, 36 ]

rank condition
done:
M [  ]
K [ 31, 32, 33, 34, 35, 36 ]

done:
M [  ]
K [ 29, 31, 33, 34, 35, 36 ]
...
```

Here MOP performs a number of splitting operations and one of the resulting strings satisfies the rank criterion, i.e., in this case the six roots 31 to 36 are linearly independent. The conjugates of this string under the generators of the small Weyl group W_J may also be considered as analyzed. Following the comment `done`, they are printed after the successful analysis of the string itself, here there are two such strings. They are then added to the `RedundancyList`.

Further along we get

```
...
analyzing
M [ 22, 24, 25 ]
K [ 31, 32, 33, 34, 35, 36 ]

operators: [ 11 ]
apply [ 11 ]
eliminating  [ 31 ]
analyzing
M [ 22, 24, 25 ]
K [ 32, 33, 34, 35, 36 ]

operators: [ 12 ]
apply [ 12 ]
eliminating  [ 32 ]
analyzing
M [ 22, 24, 25 ]
K [ 33, 34, 35, 36 ]

operators: [ 16 ]
apply [ 16 ]
eliminating  [ 33 ]
analyzing
M [ 22, 24, 25 ]
K [ 34, 35, 36 ]

induction list
done:
M [ 22, 24, 25 ]
K [ 31, 32, 33, 34, 35, 36 ]

done:
M [ 18, 21, 24 ]
K [ 31, 32, 33, 34, 35, 36 ]
...
```

This is a typical situation illustrating the way MOP works on strings. First we see the string S which is studied, indicated by the comment `analyzing` followed by the string, more precisely its support. Then MOP attempts to remove the existing positions labelled with a "K" from the support of S (here MOP always proceeds to remove the one with the lowest root number, here 31, first and then progressing to ones with higher numbers). A suitable operator (or a list of operators) to remove 31 from the support of S is chosen. MOP indicates that by printing `operators:[ 11 ]`. Note that the roots add according to the numbering as follows: "$24 + 11 = 31$". If an elimination is possible, MOP performs it and indicates that by printing `apply [ 11 ]` followed by `eliminating [ 31 ]`. The resulting string is the one with 31 removed and which is then taken from the stack analyzed next.

MOP proceeds to remove the "K" at position 32, using the operator 12 ("$25 + 12 = 32$"), and then the one at position 33 with the operator 16 ("$22+16 = 33$"). The resulting string

```
analyzing
M [ 22, 24, 25 ]
K [ 34, 35, 36 ]
```

turns out to be a substring of a member of the induction list and thus we continue the analysis by considering the next string on the stack. Before continuing its analysis, MOP prints out all new strings that are added to the `RedundancyList` as a consequence of this successful analysis, just two in this instance.

At a later stage we encounter

```
...
analyzing
M [ 22, 23 ]
K [ 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36 ]

operators: [ 4 ]
apply [ 4 ]
eliminating  [ 26 ]
analyzing
M [ 22, 23 ]
K [ 27, 28, 29, 30, 31, 32, 33, 34, 35, 36 ]

already done:
M [  ]
K [ 19, 22, 23, 25, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36 ]


done:
M [ 22, 23 ]
K [ 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36 ]

done:
M [ 19, 21 ]
K [ 24, 25, 28, 29, 30, 31, 32, 33, 34, 35, 36 ]

done:
M [ 18, 27 ]
K [ 23, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36 ]

done:
M [ 23, 26 ]
K [ 22, 25, 27, 29, 30, 31, 32, 33, 34, 35, 36 ]

done:
M [ 18, 27 ]
K [ 22, 24, 26, 29, 30, 31, 32, 33, 34, 35, 36 ]
```

Here MOP removes the first "K" in the list at position 26 using the operator 4 and the resulting string turns out to be a substring of a member of the `RedundanyList`. This illustrates the effectiveness of this criterion, as here MOP removes a string from the stack whose support has cardinality 12. Immediately following the comment `already done`, MOP prints the string from the `RedundanyList` which contains S as a substring. This is followed again by the list of all new stings added to the `RedundanyList`, as a consequence of this successful analysis.

Our next glimpse of MOP's output shows that sometimes only a combination of root operators leads to a successful elimination.

```
...
analyzing
M [ 22, 24, 25 ]
K [ 28, 29, 30, 31, 32, 33, 34, 35, 36 ]

operators: [ 6 ]
operators: [ 4 ]
operators: [ 6, 1 ]
operators: [ 4, 1 ]
operators: [ 6, 1, 4 ]
apply [ 6, 1, 4 ]
eliminating  [ 28, 26, 27 ]
analyzing
M [ 22, 24, 25 ]
K [ 29, 30, 31, 32, 33, 34, 35, 36 ]
...
```

Here MOP tries to eliminate position 28. This can be achieved using the operator 6 ("$24+6 = 28$"), this in turn introduces 27 ("$22+6 = 27$"), which was not in the support of the original string. However, the entry at 27 can be removed again by operator 1 creating a new entry at position 26 ("$24 + 1 = 26$") which can finally be removed again by applying operator 4 ("$22 + 4 = 26$"). Ultimately, all three roots, 28, 26, and 27 can be removed in this process, as shown and confirmed by MOP. That means that the system of equations stemming from these three operating roots and the roots operated on is solvable. Here the order in which the operators, 6, 1, and 4 are applied does matter.

Here is an example for an occurrence of an extended operation:

```
...
Extended
M [ 12, 13, 15, 16, 20, 22, 27 ]
K [  ]

try operator 37
try string M [ 9, 12, 13, 15, 16, 19, 20, 22, 25, 27 ]
K [  ]

analyzing
M [ 9, 12, 13, 15, 16, 19, 20, 22, 25, 27 ]
K [  ]
```

```
...

analyzing
M [ 9, 12, 13, 16, 20, 22, 25 ]
K [  ]

operators: [ 3 ]
operators: [ 11 ]
operators: [ 3, 11 ]
operators: [ 11, 7 ]
operators: [ 11, 3 ]
operators: [ 3, 38 ]
apply [ 3, 38 ]
eliminating  [ 25, 9 ]
polluting  [ 16, 18, 21 ]
analyzing
M [ 12, 13, 20, 22 ]
K [ 16, 18, 21 ]

already done:
M [ 12, 13 ]
K [ 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
  34, 35, 36 ]

extended analysis finished successfully
...
```

Here we see that MOP has encountered a string S only having positions labelled with an "M" in its support and no elimination is possible. In the extended operation MOP starts applying operators to S systematically starting with the negative roots in P. This creates new strings. Once all these new ones have been successfully eliminated, MOP confirms this with an affirmative message. This also shows an instance how MOP may chose to eliminate some entry for the expense of having to create new ones. This is indicated here by the comment `polluting [ 16, 18, 21 ]`.

The final lines of this run are as follows, illustrating the effectiveness of the redundancy criterion in a paradigmatic fashion. Here a string whose support is of cardinality 30 can be removed from the stack. The dots in the middle are in place of a final list of strings added to the `RedundancyList`.

```
...
analyzing
M [ 3, 5 ]
K [ 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
  27, 28, 29, 30, 31, 32, 33, 34, 35, 36 ]

operators: [ 1 ]
apply [ 1 ]
eliminating  [ 7 ]
```

```
analyzing
M [ 3, 5 ]
K [ 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
  27, 28, 29, 30, 31, 32, 33, 34, 35, 36 ]

already done:
M [  ]
K [ 3, 5, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
  26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36 ]

...

Modality analysis for type E6, J = [ 1, 2, 4, 6 ]
E6          2
            |
    1 - 3 - 4 - 5 - 6
--------------------------------------------
659 strings analyzed
328 splittings
--------------------------------------------
115 induction list matches
135 already done
85 occurrences of rank condition
0 J-Height criterion invoked
--------------------------------------------
2 extended operations
--------------------------------------------
0 unresolved strings
characteristic restrictions: [ 2, 3 ]
The modality of P is 0.
--------------------------------------------
```

Bibliography

[HR99] L. Hille and G. Röhrle. A classification of parabolic subgroups of classical groups with a finite number of orbits on the unipotent radical. *Transformation Groups*, 4:35–52, 1999.

[JR] U. Jürgens and G. Röhrle. MOP – algorithmic modality analysis for parabolic group actions. Preprint 00–123, SFB 343, Bielefeld (2000); to appear.

[PR97] V. Popov and G. Röhrle. On the number of orbits of a parabolic subgroup on its unipotent radical. *Australian Mathematical Society Lecture Series*, 9:297–320, 1997. Algebraic Groups and Lie Groups, Cambridge University Press.

[Röh97] G. Röhrle. A note on the modality of parabolic subgroups. *Indag. Math.*, 8:549–559, 1997.