Computing Invariant Rings with Macaulay2

Takehiko Yasuda

This is lecture materials for an algebraic geometry class taught by Yasuda at the University of Osaka in the spring/summer semester of academic year 2024. (English translation from the Japanese original.)

1 Day 1

1.1 Reference Web Pages

- Macaulay2: `https://macaulay2.com/`
- Macaulay2 documentation: `https://macaulay2.com/doc/Macaulay2/share/doc/Macaulay2/Macaulay2Doc/html/`
- InvariantRing package documentation: `https://macaulay2.com/doc/Macaulay2/share/doc/Macaulay2/InvariantRing/html/index.html`

1.2 Let's Start with a Simple Example

First, let's start the program.

```
+ M2 --no-readline --print-width 94
Macaulay2, version 1.24.05
with packages: ConwayPolynomials, Elimination,
   IntegralClosure, InverseSystems,
Isomorphism, LLLBases, MinimalPrimes, OnlineLookup,
   PrimaryDecomposition,
ReesAlgebra, Saturation, TangentCone, Truncations,
   Varieties
```

Next, load the package for computing invariant rings.

```
i1 : loadPackage "InvariantRing"

o1 = InvariantRing

o1 : Package
```

This package is for computing invariant rings for linear actions of linear reductive groups. It works well for finite groups when the characteristic and order are coprime (tame case).

Let's start by computing a simple invariant ring. First, prepare a polynomial ring in two variables with rational coefficients.

```
i2 : R = QQ[x,y]

o2 = R

o2 : PolynomialRing
```

Next, prepare a matrix corresponding to the interchange of x and y, and define a group action on the polynomial ring R determined by this matrix.

```
i3 : M = permutationMatrix [2,1]

o3 = | 0 1 |
     | 1 0 |

                2        2
o3 : Matrix ZZ  <--- ZZ

i4 : myAction = finiteAction(M,R)

o4 = R <- {| 0 1 |}
           | 1 0 |

o4 : FiniteGroupAction
```

To find generators of the invariant ring, use the command `invariants`.

```
i12 : invariants myAction

Warning: stopping condition not met!
Output may not generate the entire ring of invariants.
Increase value of DegreeBound.

               2    2
o12 = {x + y, x  + y }

o12 : List
```

The generators of the invariant ring are found to be $x + y$ and $x^2 + y^2$. For some reason, there's an error message saying it's unclear if these generate the invariant ring.

Even if we increase the `DegreeBound` as suggested, the same error appears, so we'll ignore it. The generators seem to be correctly computed.

1.3 Exercises

1. For $n = 3, 4, 5$, find the invariant ring for the permutation action of the symmetric group of degree n on the polynomial ring in n variables. Hint: You can create matrices corresponding to $n-1$ transpositions that generate the symmetric group of degree n using `apply(n-1, i -> permutationMatrix(n, [i+1,i+2]))`.

2. Do the same with alternating groups instead of symmetric groups.

3. Find the relations between the generators of each invariant ring.

4. For each invariant ring found above, compute the Hilbert (Poincaré) series using the command `hilbertSeries`.

1.4 Extension of Coefficient Field

When using the rational number field as the coefficient field, the possible group actions are limited. To handle more group actions, we need to extend the coefficient field. In particular, to deal with diagonal actions, we need to adjoin roots of unity to the field.

Let's try adjoining a cubic root of unity to the rational number field.

```
i1 : L = toField(QQ[a]/(a^2+a+1)) -- treat the ring as a
   field

o1 = L

o1 : PolynomialRing

i3 : a^3

o3 = 1

o3 : L
```

Note that we should not use `toField(QQ[a]/(a^3-1))`. While `a^3-1` is not irreducible and won't define a field, no error message will appear. However, this will lead to incorrect results in later calculations. Let's input the correct irreducible polynomial.

Using the newly defined field L as the coefficient field, we can compute the invariant ring under the action of a cyclic group of order 3 as follows.

```
i20 : R = L[x,y]

o20 = R
```

```

o20 : PolynomialRing

i21 : myAction = finiteAction(matrix{{a,0},{0,a^2}},R)

o21 = R <- {| a 0     |}
            | 0 -a-1 |

o21 : FiniteGroupAction

i22 : invariantRing myAction

Warning: stopping condition not met!
Output may not generate the entire ring of invariants.
Increase value of DegreeBound.

o22 =          3   3
      L[x*y, y , x ]

o22 : RingOfInvariants
```

We can verify that the obtained invariant ring is normal (integrally closed) using the command `isNormal`.

```
i67 : T = invariantRing myAction;

Warning: stopping condition not met!
Output may not generate the entire ring of invariants.
Increase value of DegreeBound.

i69 : I = definingIdeal T

             3
o69 = ideal(u  - u u )
             1    2 3

o69 : Ideal of L[u ..u ]
                  1   3

i70 : T2 = (ring I)/I -- represent the invariant ring as a
   quotient ring

o70 = T2

```

```
o70 : QuotientRing

i72 : isNormal T2

o72 = true
```

Various commands like `isNormal` cannot be applied directly to instances of the `RingOfInvariants` class output by `invariantRing`. We need to create an instance of the `QuotientRing` class as shown above.

1.5 Exercises

1. Compute invariant rings for diagonal actions of cyclic groups with various numbers of variables and orders of cyclic groups.
2. For each case, verify the number of generators and relations.
3. Confirm that all obtained invariant rings are normal.
4. Calculate the Hilbert-Poincaré series and verify that Molien's formula holds.
5. (Advanced) Try to determine whether the invariant rings are Cohen-Macaulay and/or Gorenstein. (Hint: Look for packages needed to check these properties.)

Actually, for diagonal actions, we can compute invariant rings without extending the coefficient field. (Note: In scheme theory, this can be interpreted as the invariant ring under the action of the group scheme μ_l. The same monomials generate the ring regardless of the coefficient field.)

```
i31 : R = QQ[x,y]

o31 = R

o31 : PolynomialRing

i32 : A = diagonalAction(matrix{{1,2}},{3},R)

o32 = R <- ZZ/3 via

      | 1 2 |

o32 : DiagonalAction

i33 : invariantRing A

```

```
o33 =             3   3
      QQ[x*y, y , x ]

o33 : RingOfInvariants
```

1.6 Quotient Map

The embedding map from the invariant ring to the polynomial ring

$$k[x_1, \ldots, x_n]^G \hookrightarrow k[x_1, \ldots, x_n]$$

corresponds to the quotient map

$$\mathbb{A}^n_k \to \mathbb{A}^n_k/G$$

For the invariant ring `T2` created above, we can define the embedding map in M2 as follows.

The following calculations can only be performed over the rational number field, so let's prepare the invariant ring over the rational field again and represent it as a quotient ring.

```
i112 : R = QQ[x,y]

o112 = R

o112 : PolynomialRing

i113 : A = diagonalAction(matrix{{1,2}},{3},R)

o113 = R <- ZZ/3 via

       | 1 2 |

o113 : DiagonalAction

i115 : I = definingIdeal invariantRing A

                 3
o115 = ideal(u  - u u )
              1    2 3

o115 : Ideal of QQ[u ..u ]
                    1   3

```

```
i116 : T = (ring I)/I

o116 = T

o116 : QuotientRing

i117 : describe T

       QQ[u ..u ]
           1   3
o117 = ----------
         3
        u  - u u
         1    2 3
```

Next, define a map from T to R by specifying where the variables map to.

```
i120 : use R

o120 = R

o120 : PolynomialRing

i121 : f = map(R,T,{x*y,y^3,x^3})

                             3   3
o121 = map (R, T, {x*y, y , x })

o121 : RingMap R <-- T

i122 : isWellDefined f

o122 = true
```

When the domain is a quotient ring rather than a polynomial ring, there's no guarantee that the map will be well-defined, so we check just to be sure.

Next, take the point (1,2) in the affine plane $\mathbb{A}_k^2 = k^2$, and compute its image under f and its preimage.

```
i127 : m1 = ideal(x-1,y-2)

o127 = ideal (x - 1, y - 2)

o127 : Ideal of R

```

```
i128 : m2 = preimage(f,m1)

o128 = ideal (u  - 2, u  - 1, u  - 8)
               1       3       2

o128 : Ideal of T

i129 : m3 = f(m2)

                          3       3
o129 = ideal (x*y - 2, x  - 1, y  - 8)

o129 : Ideal of R

i130 : decompose m3

                                                     2
o130 = {ideal (y - 2, x - 1), ideal (2x + y + 2, y  + 2y +
   4)}

o130 : List
```

From the above calculation, we can see that the image of (1,2) under the quotient map is the point (2,8,1) in k^3. Also, we found that the preimage of the point (2,8,1) under the quotient map is given by the ideal $(xy-2, x^3-1, y^3-8)$. This ideal has two minimal associated primes $(x-1, y-2), (2x+y+2, y^2+2y+4)$. The second ideal cannot be further decomposed over the rational field, but corresponds to the pair of points $(\zeta, \zeta^2 2), (\zeta^2, \zeta 2)$.

2 Day 2

Today, let's perform calculations related to Du Val singularities using Macaulay2!

2.1 Computing Invariant Rings

Du Val singularities are the singularities that appear in quotient varieties $\mathbb{C}^2/G$ by finite subgroups of $SL_2(\mathbb{C})$. The finite subgroups of $SL_2(\mathbb{C})$ have been classified, and we know exactly which matrices generate them.

Reference: Graham Leuschke, The McKay correspondence, p.15, `https://www.leuschke.org/uploads/McKay-total.pdf`

Let's calculate the coordinate ring of a Du Val singularity. As an example, let's consider type D_4. In this case, the corresponding finite group is generated by the following two matrices:

$$\begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix}, \quad \begin{pmatrix} 0 & i \\ i & 0 \end{pmatrix}$$

To calculate in M2, we prepare as follows:

```
i1 : loadPackage "InvariantRing"  -- Load InvariantRing package

o1 = InvariantRing

o1 : Package

i2 : L = toField(QQ[w]/(w^2+1))  -- Adjoin fourth root of unity to QQ

o2 = L

o2 : PolynomialRing

i3 : R = L[x,y]

o3 = R

o3 : PolynomialRing
```

Next, we define the group action and compute the invariant ring.

```
i21 : X1 = matrix{{w,0},{0,-w}}; X2 = matrix{{0,w},{w,0}};

                           2              2
o21 : Matrix (L[u..w])  <-- (L[u..w])

                           2              2
o22 : Matrix (L[u..w])  <-- (L[u..w])

i23 : action1 = finiteAction({X1,X2},R)  -- Define the action of finite group generated by two matrices

o23 = R <- {| w 0  |, | 0 w |}
            | 0 -w |  | w 0 |

o23 : FiniteGroupAction

i24 : A = invariantRing action1
```

```
o24 =     4     4    2 2    5        5
      L[x  + y , x y , x y - x*y ] -- May not display like this depending on version. In that case, you can display generators using "generators"

o24 : RingOfInvariants

i25 : definingIdeal A  -- The ideal when expressing the invariant ring as a quotient of polynomial ring

              2       3    2
o25 = ideal(u u  - 4u  - u )
             1 2      2    3

o25 : Ideal of L[u ..u ]
                  1   3
```

From the last calculation result, we found that the quotient variety in this case is isomorphic to an affine algebraic variety in $\mathbb{C}^3$ defined by the equation $u_1^2u_2 - 4u_2^3 - u_3^2 = 0$. After the coordinate transformation

$$x = u_3, \quad z = 4^{1/3}u_2, \quad y = (-1)^{1/2}4^{1/6}u_1$$

we get the well-known D_4 type singularity equation $x^2 + y^2z + z^3 = 0$.

2.2 Exercises

1. Find the invariant ring for type D_6 using M2 and verify that the quotient variety is isomorphic to an affine algebraic variety defined by $x^2 + y^2z + z^5 = 0$.

2. Similarly verify for types E_6 and E_7. (The calculation might take some time.)

2.3 Computing Blowups

First, let's copy and paste the following code into M2 and press Enter.

```
affineCharts = S ->(
      -- affine charts of a blowup without simplification
      T := (flattenRing S)_0;
      U := ambient T;
      I := ideal T;
      varsOfS := apply(flatten entries vars S,i->sub(i,U));
      apply(varsOfS, i->  U / (I + ideal(i - 1)))) ;

```

```
isBlowupSmooth = S ->(
    -- checks if the Rees algebra of an ideal is smooth.
    all(affineCharts(S),isSmooth2)  ) ;

isSmooth2 = R -> (
  -- checks if an affine ring is smooth
  if (isPolynomialRing R) then true else
    (
      S := ambient R;
      I := ideal R;
      (ideal singularLocus I) == ideal(1_S)
    )
  ) ;

isBlowupNormal = S ->(
   -- checks if the Rees algebra of an ideal is normal.
      all(affineCharts(S),isNormal)) ;
```

These are part of the Macaulay2 functions that Yasuda previously wrote for his research and published `http://www4.math.sci.osaka-u.ac.jp/~takehikoyasuda/codes/MyPackage.m2`. (However, the published function “affineCharts” stopped working, so “affineCharts2” was replaced with “affineCharts” this time. “isSmooth” was changed to “isSmooth2” to avoid conflict with the built-in function of the same name.)

As an example, let me explain what the function ”isSmooth2” does. When it receives input `R` (assuming `R` is a quotient ring of a polynomial ring by an ideal), it first checks if `R` is a polynomial ring, and if so, returns `true`. If `R` is not a polynomial ring, it sets `S` as the polynomial ring used to define `R` and `I` as the ideal, then computes the defining ideal of the singular locus using `ideal singularLocus I` and checks if it equals the trivial ideal `ideal(1_S)`. For details on how to create functions in M2, refer to Section 5.1 “Creating Functions” in `http://www4.math.sci.osaka-u.ac.jp/~takehikoyasuda/pdfs/CompAG-en.pdf`. By creating your own functions, you can perform more complex calculations.

Now, let’s compute the blowup at the origin of an A_2 singularity.

```
i5 : R = QQ[x,y,z]/(x*y-z^3)

o5 = R

o5 : QuotientRing

i6 : A = reesAlgebra ideal(x,y,z)

o6 = A

```

```
o6 : QuotientRing
```

Explanation for those who know scheme theory: For an ideal I of ring R, the graded ring called the Rees algebra is defined as follows:

$$A := R[It] = \bigoplus_{n \geq 0} I^n$$

Its Proj is the blowup of $\operatorname{Spec} R$ along ideal I. When generators $f_1, \ldots, f_l$ of I are fixed, it can be written as a quotient ring $R[x_1, \ldots, x_l]/J$. The blowup is covered by l affine open sets, and for each $i \in \{1, \ldots, l\}$,

$$B_i := R[x_1, \ldots, x_l]/(J + (x_i - 1))$$

becomes the coordinate ring of an affine open set.

Let's check if the blowup at the origin of an A_2 singularity is smooth.

```
i7 : isBlowupSmooth A

o7 = true
```

This calculation shows that the blowup at the origin of an A_2 singularity is smooth and provides a resolution of singularities.

Each affine open chart can be computed as:

```
i8 : affineCharts A

                                QQ[w ..w , x..z]
                                    0 2
o8 =
   {-------------------------------------------------------------------------,

        3 2 2
     (- z + x*y, w x - w y, w y - w z, w x - w z, w z - w y, w z - w w , w -
         1)
                 1 2 0 1 0 2 0 2 0 1 2 0
    ----------------------------------------------------------------------------------

                               QQ[w ..w , x..z]
                                   0 2
    ------------------------------------------------------------------------,

       3 2 2
    (- z + x*y, w x - w y, w y - w z, w x - w z, w z - w y, w z - w w , w -
       1)
                1 2 0 1 0 2 0 2 0 1 2 1
    ----------------------------------------------------------------------------------
```

```
                                   QQ[w ..w , x..z]
                                       0 2
     -----------------------------------------------------------------------------}

         3 2 2
     (- z + x*y, w x - w y, w y - w z, w x - w z, w z - w y, w z - w w , w -
         1)
                   1 2 0 1 0 2 0 2 0 1 2 2

o8 : List

i9 : singularLocus o8_0 -- the singular locus of the 1st ring

      QQ[]
o10 = ---- -- The singular locus is empty.
        1

o10 : QuotientRing
```

The exceptional set can be calculated as follows:

```
i11 : specialFiber ideal(x,y,z)

        QQ[w ..w ]
            0   2
o11 = ----------
           w w
            0 1

o11 : QuotientRing
```

We found that the exceptional set is defined by

$$w_0 w_1 = 1$$

in the projective plane $\mathbb{P}^2$ with homogeneous coordinates w_0, w_1, w_2. From this, we can see that the exceptional set consists of two projective lines intersecting orthogonally at one point.

2.4 Exercises

1. Let's try similar calculations for A_3 and D_4 singularities.

2. If you have extra time, look at `http://www4.math.sci.osaka-u.ac.jp/~takehikoyasuda/pdfs/CompAG3.pdf` and try various calculations. If you think you might use M2 in the future, read Section 5.1 "Creating Functions" to learn how to create functions.