The SINGULAR 'Resolution of Singularities' Package: Getting started

Anne Frühbis-Krüger
FB Mathematik
Universität Kaiserslautern,
67653 Kaiserslautern, Germany

July 1, 2005

1 Installation

If SINGULAR is not installed on your (UNIX-type) computer, ask your system administrator to download and install it (version $\geq$ 2.0.6) on your computer. Also ask for the installation of the additional packages surf and graphviz which are used for producing graphical output, but are not part of the SINGULAR distribution. If all this is already installed on your computer, you are ready to start.

2 Working with the '`resol.lib`' Package

2.1 First Steps

The first step is starting SINGULAR for interactive use by specifying the command `Singular` at the system prompt of the computer. SINGULAR then starts with a message like

```
                     SINGULAR                             /
 A Computer Algebra System for Polynomial Computations   /   version 2-0-6
                                                        0<
     by: G.-M. Greuel, G. Pfister, H. Schoenemann        \   December 2004
FB Mathematik der Universitaet, D-67653 Kaiserslautern    \
```

where, of course, the version number and month and year might be different. On the next line the system then awaits input; to indicate this, the character > is displayed at the beginning of the line.

Before we can start using the resolution of singularities package, we need to load it:

```
LIB"resolve.lib";                 // load the resolution algorithm
LIB"reszeta.lib";                 // load its application algorithms
LIB"resgraph.lib";                // load the graphical output routines
```

Here the content of the line following the characters '//' is a comment. It can be omitted from the input; it is only specified here to explain the corresponding line to the reader.

Now, we are ready to use the resolution package. Currently, the object to be resolved can only be specified by means of an ideal contained in an affine chart. Let us illustrate this using the A_6-singularity as an example:

```
ring R=0,(x,y,z),dp;              // define the ring Q[x,y,z]
ideal I=x7+y2-z2;                 // an A6 surface singularity
```

To actually compute a resolution of singularities, we use the command `resolve`:

```
list L=resolve(I);               // compute the resolution
```

Since computation of the resolution might take quite some time, there is the possibility to view some debug output during the resolution by specifying an additional second parameter '1':

```
list L=resolve(I,1);             // compute the resolution in debug mode
```

At the same time this debug mode will perform additional sanity checks which only serve debug purposes and might slow down the algorithm. (In future releases, there will be a switch to only produce output without running in debug-mode.) If the user would like to stop at each blow-up step, the optional second argument may be set to '32'. In both cases, the debug output will be of the following form (the text after the '//' is just a comment which does not appear in the output):

```
⋮

++++++++++++++    BO    +++++++++++++++++++++++
3                             //
9                             // debugging information: progress
8                             //
+++++++++++++++++++++++++++++++++++++++++++++++

==== W:
_[1]=0                        // ideal of ambient space

==== J:
_[1]=x(3)*y(1)^2-x(3)-y(2)    // ideal of strict transform

==== E:
[1]:
   _[1]=1                      // strict transform of first exc. div.
[2]:
   _[1]=1                      // strict transform of second one
[3]:
   _[1]=y(2)                   // strict transform of third one
[4]:
   _[1]=x(3)                   // newborn exceptional divisor

==== Intersection
0,1,0,0,                       // debugging information
0,0,1,0,
0,0,0,1,
0,0,0,0

-------  Center ------------
_[1]=y(2)                      // center for upcoming blow-up
_[2]=x(3)                      //
----------------------------
```

⋮

When considering this output, it is important to observe that information is printed out for each chart occurring in the resolution process. In particular, not all exceptional divisors are, in general, present in a given chart; e.g. the exceptional divisors one and two in the above example did appear in an ancestor of the current chart, but they do not meet the current one. It may also occur that the total number of exceptional divisors arising in the whole resolution process is higher than the highest number of exceptional divisors in any of the charts.

As soon as the system finished computing the tree of charts of the resolution process, we can start identifying the exceptional divisors in the various charts:

```
list coll=collectDiv(L);        // identify the divisors
coll;                           // show the output
[1]:
   0,0,0,0,0,
   1,0,0,0,0,
   1,0,0,0,0,
   1,2,0,0,0,
   0,2,0,0,0,
   0,2,3,0,0,
   0,0,3,0,0,
   0,0,3,4,0,
   0,0,3,4,0,
   0,0,0,4,5,
   0,0,3,0,5,
   0,0,0,4,5,
   0,0,3,0,5
[2]:
   [1]:
      [1]:
         2,1
      [2]:
         3,1
      [3]:
         4,1
```

⋮

The output of this command is a list, whose first entry is a matrix with integer entries. The entry k in the i-th row and j-th column of this matrix identifies the j-th divisor (if it is visible in the chart) in the history of the i-th chart as the execeptional divisor numbered by k. The other output data is explained in the online-help for the command.

We can also draw the tree of charts:

```
ResTree(L,coll[1]);              // draw tree of charts
```

In a different window, the image 1 is produced:

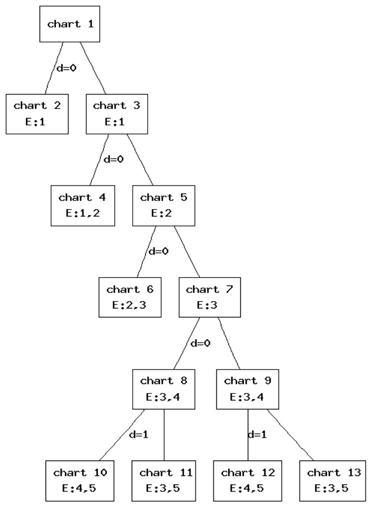

Figure 1: Tree of charts of the resolution of an A_6 surface singularity as computed by the implemented algorithm. Each of the boxes represents one chart, the list 'E' specifies the exceptional divisors visible in this chart in the order in which they arose. The lines connecting the boxes represent a blow-up linking the chart containing the center to the new charts, the number 'd' specifies the dimension of the center of this blow-up.

2.2 Accessing the Resolution Data

The list which is returned by the command `resolve` contains all charts which were created during the resolution process. These charts can be accessed in the following way:

```
def Chart8=L[2][8];               // assign name Chart8 to the 8th chart
setring Chart8;                   // access Chart8
BO;                               // look at the object in detail
```

The last command produces the following output:

```
[1]:
   _[1]=0                         // ideal of the ambient space

[2]:
   _[1]=x(3)*y(1)^2-x(3)-y(2)     // ideal of the strict transform

[3]:
   1,1                            // internal data

[4]:
   [1]:
      _[1]=1                      // strict transform of 1st exc. div.
   [2]:
      _[1]=1                      // strict transform of 2nd one
   [3]:
      _[1]=y(2)                   // strict transform of 3rd one
   [4]:
      _[1]=x(3)                   // newborn exceptional divisor

[5]:                              // ideal describing sequence of blow-ups
   _[1]=x(3)*y(2)                 // image of 1st variable of orig. chart
   _[2]=x(3)^4*y(2)^3             // image of 2nd variable of orig. chart
   _[3]=x(3)^4*y(1)*y(2)^3        // image of 3rd variable of orig. chart

[6]:
   1,1,0,0                        // internal data

[7]:
   3,-1                           // internal data

[8]:                              // internal data
   _[1,1]=0
   _[1,2]=1
   _[1,3]=0
   _[1,4]=0
   _[2,1]=0
   _[2,2]=0
   _[2,3]=1
   _[2,4]=0
   _[3,1]=0
   _[3,2]=0
   _[3,3]=0
   _[3,4]=1
   _[4,1]=0
   _[4,2]=0
   _[4,3]=0
   _[4,4]=0

[9]:                               // internal data
   1,1
```

Additionally there is some more information encoded in this chart. The ideal `lastMap` specifies the images of the variables of the parent chart under the last blow-up; the ideal `cent` describes the center of the upcoming blow-up. The matrix `path` encodes the history of the resolution process leading to this chart:

```
0, 1,3,5,7,
-1,2,2,2,1
```

The last column specifies that this chart is the first one of the blow-up of chart 7 in the center specified there; chart 7 in turn arose as the second chart from the blow-up of chart 5 according to the second column from the right. The first column is only there for technical reasons, it does not have any meaning.

3 Working with the Applications Package '`zeta.lib`'

Currently, all applications are only available for surface singularities:
As a first application, we can compute the intersection matrix and genus of the exceptional curves on the blown-up surface:

```
list iD=intersectionDiv(L);       // compute intersection properties
iD;                               // show the output
```

The output of this command is a list whose first entry contains the intersection matrix of the exceptional divisors (each being a $\mathbb{C}$–irreducible curve) and the second entry is the list of genera of these divisors. The third and fourth entry help identifying the corresponding divisors in the respective charts:

```
[1]:
   -2,0,1,0,0,0,                  // intersection matrix
   0,-2,0,1,0,0,
   1,0,-2,0,1,0,
   0,1,0,-2,0,1,
   0,0,1,0,-2,1,
   0,0,0,1,1,-2

[2]:
   0,0,0,0,0,0                  // genera

[3]:
   [1]:                         // first exceptional divisor
      [1]:                      // can be found in
         2,1,1                  // chart 2: BO[4][1], first component
      [2]:
         4,1,1                  // chart 4: BO[4][1], first component
   [2]:                         // second exceptional divisor
      [1]:                      // can be found in
         2,1,2                  // chart 2: BO[4][1], second component
      [2]:
         4,1,2                  // chart 4: BO[4][1], second component
   [3]:                         // third exceptional divisor
      [1]:                      // ...
         4,2,1
      [2]:
         6,2,1
   [4]:
      [1]:
         4,2,2
      [2]:
         6,2,2
   [5]:
      [1]:
         6,3,1
      [2]:
         7,3,1
   [6]:
      [1]:
         6,3,2
      [2]:
         7,3,2
[4]:
   1,1,1,1,1,1                  // number of C-components of each
                                // Q-component listed in entry [3]
                                // (C complex numbers, Q rationals)
```

Given these data we are able to display the dual graph of the resolution:

```
InterDiv(iD[1]);                // draw dual graph of resolution
```

Caution! This feature is still in its testing phase. It might still undergo significant changes!

For people who would like to see nice pictures, we can also draw the final charts:

```
finalCharts(L,iD,abstractR(L)[1]); // draw pictures of the final charts
```

The graphical output is not always absolutely correct due to the fact that the images are produced by means of a ray-tracer which prepares the images symbolic-numerically.

As a last set of applications, we can compute the negative spectral numbers and the (local and global) Denef-Loeser zeta function for the given singularity:

```
spectralNeg(L);                     // negative spectral numbers
zetaDL(L,1);                        // global zeta function for d=1
zetaDL(L,2);                        // global zeta function for d=2
zetaDL(L,1,"local");                // local zeta function for d=1
```

Of course, there are no negative spectral numbers in this very simple example. The global and local zeta function (type of return value: string) coincide for isolated singularities. But this is only a syntax example anyway.