CoCoALib - Feature #386

add resolution data type

23 Jul 2013 09:46 - Anna Maria Bigatti

Status:	In Progress	**Start date:**	23 Jul 2013
Priority:	High	**Due date:**	
Assignee:	Anna Maria Bigatti	**% Done:**	10%
Category:	New Function	**Estimated time:**	0.00 hour
Target version:	CoCoALib-1.0	**Spent time:**	1.50 hour

Description

Description:
work in progress

Related issues:

Related to CoCoALib - Feature #383: Resolution/morse: integrate Mario Albert'...	**In Progress**	**27 Jun 2013**
Related to CoCoALib - Feature #387: implement algorithm(s) for resolutions	**New**	**23 Jul 2013**
Related to CoCoA-5 - Feature #388: port optimized resolutions to CoCoA-5	**New**	**23 Jul 2013**
Related to CoCoALib - Design #550: add myResolution member field to ideal an...	**New**	**09 May 2014**
Related to CoCoALib - Feature #729: Betti numbers, betti diagram	**In Progress**	**08 Jun 2015**

History

#1 - 29 Oct 2013 14:59 - Anna Maria Bigatti

- Target version changed from CoCoALib-0.99534 Seoul14 to CoCoALib-0.99532

#2 - 01 Apr 2014 19:20 - Anna Maria Bigatti

- Target version changed from CoCoALib-0.99532 to CoCoALib-0.99533 Easter14

#3 - 08 Apr 2014 18:15 - Anna Maria Bigatti

- Target version changed from CoCoALib-0.99533 Easter14 to CoCoALib-0.99534 Seoul14

#4 - 30 Jul 2014 17:04 - Anna Maria Bigatti

- Target version changed from CoCoALib-0.99534 Seoul14 to CoCoALib-1.0

#5 - 10 Jun 2015 13:41 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

Mario and I have discussed some possible designs; neither of us has much experience with resolutions, so it is possible that the design is not completely sensible.

We think that a (free) resolution is made of two parts:

- **(A)** the rightmost "factor" which is the input module (or ideal);
- **(B)** the modules constructed to make the resolution.

We propose a structure for representing part **(B)** inspired by Anna's prototype implementation in CoCoA-5. The structure is just a vector<submodule> where each submodule (of a free module) contains both the degree shifts and the matrix of the homomorphism -- each submodule generator corresonds to a row of the matrix representing the homomorphism.

We think vector is appropriate because the resolution is a "dense" construct (without any "holes"); see also the discussion about the Betti construct in #729. Note that this design also allows an empty resolution (*e.g.* if the input is a zero ideal).

To handle part **(A)** we are thinking of 4 separate classes depending on the type of input:

- ideal I in poly ring P -- resolution is over P
- quotient ring P/I -- resolution is over poly ring P
- module (submodule of a free module over P) -- resolution is over poly ring P

- quotient module (free P--module quotiented by FG submodule) -- resolution is over poly ring P

Are any cases missing?

Since the C++ types of the input are different it seems simplest to have different classes for each case. Each class would contain the input object and a structure for part **(B)** and perhaps some other data (*e.g.* a Betti construct).

#6 - 11 Jun 2015 11:43 - Anna Maria Bigatti

> Since the C++ types of the input are different it seems simplest to have different classes for each case. Each class would contain the input object and a structure for part **(B)** and perhaps some other data (*e.g.* a Betti construct).

it seems a pity because they are all P-modules, but I think that cannot be expressed in C++ without using multiple-inheritance (aaaargh!).

Maybe ideal could inherit (easily and cleanly) from module.
But for ring and QuotientRing (P and P/I are P-modules) ummmmmmm...

#7 - 11 Jun 2015 11:57 - John Abbott

One possibility might be to have a "module view" much like MatrixView, but I'm far from certain whether such a thing is genuinely implementable.

I do note that the presentations I have seen of resolutions always distinguish the input module from all the others in the resolution, so perhaps it is not so "disrespectful" of the maths if the implementation also handles the rightmost "factor" in a special way.

What operations are permitted on a resolution?

- print out (prettyprint and normal print?)
- query the structure (*e.g.* length, constructed modules, homomorphisms)
- get the Betti numbers
- compare for equality? get ring?

Perhaps while I'm here in KL, I can engage in some "industrial espionage" (*e.g.* ask Hans what Singular does)

#8 - 11 Jun 2015 14:58 - John Abbott

Hans says that singular uses more or less the design we are considering; apparently in some cases they store two resolutions (a minimal one, and another with specific properties requested by the creator).

They also have another more complex (tree-like?) structure for representing the "resolution of singularities".

I did ask about the rightmost factor; Hans said that internally Singular uses essentially the same representation for ideals and modules.