CoCoA-5 - Bug #446

intersection fails with zero ideal

20 Feb 2014 11:40 - John Abbott

Status:	Closed	**Start date:**	20 Feb 2014
Priority:	Urgent	**Due date:**	
Assignee:	Anna Maria Bigatti	**% Done:**	100%
Category:	Incomplete function	**Estimated time:**	3.00 hours
Target version:	CoCoA-5.0.9	**Spent time:**	3.25 hours

Description

The following input wrongly produces an error

```
Use P ::= QQ[x,y,z];
I := ideal(x,y);
J := ideal(zero(P));
Intersection(I,J); --> gives error!
```

Related issues:

Related to CoCoALib - Design #455: Which sets of generators in an ideal?	**New**	**03 Mar 2014**
Related to CoCoA-5 - Bug #1080: intersect: problem with zero generators	**Closed**	**16 Jun 2017**

History

#1 - 20 Feb 2014 11:46 - John Abbott

Same problem with radical(J).

#2 - 20 Feb 2014 15:53 - John Abbott

Bad news: the problem occurs somewhere inside Max's code :-(

The problem is that EmbedPolyList tries to create a GPoly starting from a zero polynomial, but the ctor wants to compute the LPP of the starting poly.

I tried hacking EmbedPolyList (lines 1489-1501 in TmpGReductor.C) so that the main loop skips zero polys; this caused another problem in myPrepareGBasis because the list myPairs is empty, but an assert says it should be non-empty.

What to do?

#3 - 21 Feb 2014 00:36 - John Abbott

radical seems happy so long as there is at least one non-zero generator.
However, radical(ideal(one(R))) gives a very strange error message!

intersect appears to exhibit the problem if either ideal has a zero generator;
I tried ideal(x,0) and ideal(y,0) where 0 represents zero(R)

Anna says she knows a fix for intersect.

Strictly, the issue belongs to CoCoALib.

#4 - 21 Feb 2014 00:40 - John Abbott

Should there be a separate issue about putting some proper "extreme case" tests for all ideal operations in the standard test suite(s)? Probably best in the CoCoALib tests!

The tests should cover char 0 and char p. Perhaps some non-trivial coeffs rings too?

#5 - 21 Feb 2014 11:39 - Anna Maria Bigatti

- Category set to Incomplete function

- Target version set to CoCoA-5.0.9

- % Done changed from 0 to 10

I had fixed this kind of problemi in SparsePolyRing.C for colon

```
      RingElem Z(zero(myAmbientRing()));
      myGensValue.erase(remove(myGensValue.begin(), myGensValue.end(),Z),
                        myGensValue.end());
```

... in fact.... I'm not sure why gens do not change...

```
/**/  Use R ::= QQ[x,y];  I := ideal(x*y, zero(R), x^2);  I : ideal(x);
ideal(y, x)
/**/ I;
ideal(x*y, 0, x^2)
```

#6 - 21 Feb 2014 11:59 - John Abbott

The mem fn myColon is called by colon (see source ideal.C:280) which makes a copy of I and then "plays" with the gens of that copy -- this does not change the original arg (which is const).

What I tried to suggest yesterday, is that an ideal could contain **two** lists of generators: one is the list the user supplied, the other is a "cleaned list" *e.g.* without zero values, maybe without duplicates, maybe made monic, perhaps put into a good order, etc. I can see one problem: which list should be used when printing the ideal? Presumably the original user-supplied list should be returned by the mem fn myGens (or whatever it is called). Mmmm, it could be confusing to have two lists... Unless the "cleaned list" is used only when calling ideal operations...???

#7 - 21 Feb 2014 12:16 - Anna Maria Bigatti

John Abbott wrote:

> The mem fn myColon is called by colon (see source ideal.C:280) which makes a copy of I and then "plays" with the gens of that copy -- this does not change the original arg (which is const).

I expect we could do the same for intersect

> What I tried to suggest yesterday, is that an ideal could contain **two** lists of generators: one is the list the user supplied, the other is a "cleaned list" *e.g.* without zero values, maybe without duplicates, maybe made monic, perhaps put into a good order, etc. I can see one problem: which list should be used when printing the ideal? Presumably the original user-supplied list should be returned by the mem fn myGens (or whatever it is called). Mmmm, it could be confusing to have two lists... Unless the "cleaned list" is used only when calling ideal operations...???

I think the cleaned list could be hidden to the user, a bit like the gbasis.
For monomial ideals it would be handy to have the list of PPs in a similar fashion.
A template member field? (boggle)

#8 - 22 Feb 2014 10:49 - John Abbott

We must remember to add CoCoALib tests for all these "trivial cases" -- they should be in CoCoALib rather than CoCoA-5, I believe.

#9 - 21 Mar 2014 09:53 - Anna Maria Bigatti

- Status changed from New to Feedback

- Assignee set to Anna Maria Bigatti

- % Done changed from 10 to 90

done and added to test-ideal1.

Added the zero ideal case to

```
const ideal intersect(const ideal& I, const ideal& J)
```

using **myIntersect** is still dangerous (as all the my functions)

#10 - 21 Mar 2014 12:32 - John Abbott

Do the myIntersect functions have assertions that their args are valid?

#11 - 21 Mar 2014 12:56 - Anna Maria Bigatti

ehm.... most of them yes ;-) except SparsePolyRing ideals (I think).

Looking at all of them I'm getting convinced we should make a "myAssignZero" which knows, for any implementation, all the fields to be cleared.

#12 - 21 Mar 2014 14:15 - Anna Maria Bigatti

added test in cocoalib. closing.

#13 - 25 Mar 2014 17:36 - John Abbott

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

#14 - 02 Apr 2014 18:35 - Anna Maria Bigatti

- Estimated time set to 3.00 h

#15 - 16 Jun 2017 19:03 - John Abbott

- Related to Bug #1080: intersect: problem with zero generators added