CoCoA-5 - Bug #870

GBasis of product of ideals is wrong (Vadim Tropashko) --> I.myReset()

26 Apr 2016 09:31 - John Abbott

Status:	Closed	**Start date:**	26 Apr 2016
Priority:	Urgent	**Due date:**	
Assignee:	Anna Maria Bigatti	**% Done:**	100%
Category:	enhancing/improving	**Estimated time:**	2.01 hours
Target version:	CoCoA-5.2.0 spring 2017	**Spent time:**	1.60 hour

Description

Vadim Tropashko reported an incorrect result. Robbiano has simplified it. Here is the simplified version.

```
use QQ[x,y];
I1 := ideal(x,y);
I2 := ideal(x-1,y-1);
J := I1*I2;
J; -->  ideal(x^2 -x, x*y -x, x*y -y, y^2 -y)  prints out OK
GBasis(J); --> plainly wrong!!
GBasis(ideal(gens(J))); --> correct!
```

Related issues:

Related to CoCoALib - Design #871: Redesign ideals	**New**	**26 Apr 2016**
Related to CoCoALib - Design #924: FlagManager for bool/bool3 flags	**New**	**19 Sep 2016**
Related to CoCoALib - Feature #899: IsMaximal, IsPrimary for IDEAL (in cocoa...	**Closed**	**27 Jun 2016**

History

#1 - 26 Apr 2016 09:35 - John Abbott

Since this was reported by a CoCoA-5 user i have put it in the CoCoA-5 category even though I'm almost certain that the problem lies in CoCoALib.

My guess is that the function which creates a product of ideals incorrectly sets the GBasis field (with the wrong value).

I don't have much desire to delve in Max's code; Anna do you?

We should get this sorted out ASAP -- it's an embarrassing bug! Also it should be added to the CoCoA tests (preferably CoCoALib).

#2 - 26 Apr 2016 09:39 - John Abbott

Try looking around SparsePolyRing.C:2029

#3 - 26 Apr 2016 10:16 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

Here is the equivalent CoCoALib code -- it exhibits the problem.

```
  void program()
  {
    GlobalManager CoCoAFoundations;

    const ring Qxy = NewPolyRing(RingQQ(), symbols("x,y"));
    const RingElem x = indet(Qxy,0);
    const RingElem y = indet(Qxy,1);
    vector<RingElem> v;
    v.push_back(x); v.push_back(y);
    ideal I1 = ideal(v);
    v.clear();
    v.push_back(x-1); v.push_back(y-1);
    ideal I2 = ideal(v);
    I1 = I1*I2;
    cout << "GB: " << GBasis(I1) << endl;
    ideal J = ideal(gens(I1));
    cout << "GB2: " << GBasis(J) << endl;
  }
```

#4 - 26 Apr 2016 13:38 - John Abbott

I continue to think that the current design of ideals is completely wrong (but I cannot find the redmine task about this to mark as a related issue). I am convinced that it is poor design to make the function myMul modify one of the args; I expect it would be cleaner if it took two args and produced a new ideal.

The problem is almost certainly around line 2025. The fields IhaveMonomialGens3Value and other similar ones should be reset always (unless both ideals are monomial). For some reason they are reset only if char is non-zero -- why??

I am also mystified as to why IamPrime3Flag is not reset (commented out).

#5 - 26 Apr 2016 15:08 - John Abbott

- Related to Design #871: Redesign ideals added

#6 - 26 Apr 2016 15:19 - John Abbott

I suggest that Anna try to fix the functions already in SparsePolyRing.C.

Even though I'm hoping for a proper redesign of ideals (in the not too distant future?), I think it makes sense to fix the functions we already have, since I'm hoping they can be largely copied as they are to make the new impls... which means that time spent debugging these functions is not (entirely) wasted.

Should there be a procedure to "reset all 3-way flags" for ideals? Requiring each function to mention explicitly every internal flag is not "scalable" (as we have just seen in issue #870). Also a "reset all flags" procedure would mimic well the creation of new ideal for the result (which is the direction I think we should probably move in).

#7 - 27 Apr 2016 08:57 - Anna Maria Bigatti

John Abbott wrote:

> I suggest that Anna try to fix the functions already in SparsePolyRing.C.

done! (cvs-ing)

> Should there be a procedure to "reset all 3-way flags" for ideals? Requiring each function to mention explicitly every internal flag is not "scalable" (as we have just seen in issue #870). Also a "reset all flags" procedure would mimic well the creation of new ideal for the result (which is the direction I think we should probably move in).

Indeed, and even more: I implemented myReset() which resets **all** member fields but the generators (and removed myClearGBasis).
Any info which can be maintained/deduced has to be done explicitely.

We had the similar (by implementation) bug:

```
/**/ I := ideal(x+y, x-y);  MinGens(I);
[x, -y]

/**/ MinGens(I+ideal(z));
[x, -y]
```

#8 - 27 Apr 2016 19:05 - John Abbott

- Assignee set to Anna Maria Bigatti

- % Done changed from 10 to 50

I have added a new CoCoALib test (test-bug6.C) which tests for this specific problem.

It should later be expanded or replaced by a more thorough test on ideals.

Anna will add a similar test to the CoCoA-5 suite.

#9 - 14 Jul 2016 20:35 - Anna Maria Bigatti

- Subject changed from GBasis of product of ideals is wrong (Vadim Tropashko) to GBasis of product of ideals is wrong (Vadim Tropashko) --> I.myReset()

#10 - 01 Mar 2017 13:33 - Anna Maria Bigatti

- Status changed from In Progress to Feedback

#11 - 01 Mar 2017 14:16 - Anna Maria Bigatti

- Related to Design #924: FlagManager for bool/bool3 flags added

#12 - 01 Mar 2017 14:17 - Anna Maria Bigatti

- Related to Feature #899: IsMaximal, IsPrimary for IDEAL (in cocoalib) added

#13 - 26 Apr 2017 15:56 - Anna Maria Bigatti

- Status changed from Feedback to Closed

- % Done changed from 50 to 100

- Estimated time set to 2.01 h