CoCoALib - Feature #1267

Ideal equality

28 Mar 2019 16:49 - John Abbott

Status:	In Progress	**Start date:**	28 Mar 2019
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	10%
Category:	Improving	**Estimated time:**	10.00 hours
Target version:	CoCoALib-1.0	**Spent time:**	1.05 hour

Description

Long in Passau lamented that sometimes testing two ideals for equality was too slow.

It might help to perform a few "fast heuristic" tests first:
given ideals I and J,
first test whether I+ideal(x,z) = J+ideal(x,z) where x,z is a random subset of indets,
if these are unequal then surely I and J are unequal;
perhaps repeat for some other subsets of indets.

The hope is that computing GBases for I+ideal(x,z) and for J+ideal(x,z) is much cheaper than computing the GBases for I and J. So even if we get an inconclusive heuristic test, the overall extra cost is negligible.

Is this worth trying?

One could also consider adding other ideals such as ideal(x-1,z) or ideal(x,z+1).

Related issues:

Related to CoCoALib - Slug #725: Example database: Slow ideal equality test	**New**	**06 Jun 2015**

History

#1 - 29 Mar 2019 09:18 - Anna Maria Bigatti

- % Done changed from 0 to 10

- Estimated time set to 10.00 h

It is implemented in ideals.C like this:

```
    return IsContained(I, J) && IsContained(J, I);
```

This means that IsContained(I, J) is evaluated first, even though (J, I) might be easier.
Maybe we should have a concrete implementation for SparsePolyRing ideals checking if we already have the GBasis for I or J.

Questions:
1 - in his computations, is Long expecting to have equality or not? if yes, there is no real shortcut...
2 - are I and J homogenous? if yes, we could compute a truncated GBasis

#2 - 04 Apr 2019 15:16 - John Abbott

- Status changed from New to In Progress

Long sent me one example which wanted to test if a large ideal (in a ring with 36 indets) contained 1. He actually did this by testing ideal(one(P)) = J.

In that instance the result is false, and this could quickly and easily be checked by adjoining all indets as generators (effectively setting them all to zero).

My thought is that if the ideals are unequal then there is some chance of discovering this quickly by the heuristic test (not sure how many "atomic" tests should be made). Conversely, if the ideals are equal then sooner or later we must actually check the double inclusion (potentially computing 2 costly GBases). Assuming the heuristic checks are much faster than actually computing the full GBs, it should not matter that they are "a small waste of time".

#3 - 05 Apr 2019 12:30 - John Abbott

A similar trick could be used for ideal membership: if x is not in I+J then it is certainly not in I.

#4 - 09 Apr 2019 17:48 - Anna Maria Bigatti

I'm very uneasy in doing an automatic choice: the "trick" might work out as a slow overhead in a long loop with many easy checks.
Moreover, there are many tricks one can apply, but they are most effective if they are chosen knowing the kind of problem.
E.g. one could add the squares, or some powers of all indeterminates.

#5 - 10 Dec 2023 20:46 - John Abbott

- Related to Slug #725: Example database: Slow ideal equality test added