CoCoALib - Bug #1199

GCD bug with high degree arg

28 Jun 2018 10:54 - John Abbott

Status:	Closed	**Start date:**	28 Jun 2018
Priority:	Normal	**Due date:**	
Assignee:	Anna Maria Bigatti	**% Done:**	100%
Category:	Maths Bugs	**Estimated time:**	3.33 hours
Target version:	CoCoALib-0.99600	**Spent time:**	3.50 hours

Description

```
p := 1000003; // prime
use P ::= ZZ/(p)[x,y];
f := x*(x-2);
gcd(f,x^p-x); // gives wrong ans, namely just x
```

Requires prime p >= 32749; ring must have at least 2 indets.

Related issues:

Related to CoCoALib - Design #707: MatrixOrderingMod32749Impl: test and write...	**In Progress**	**15 May 2015**

History

#1 - 28 Jun 2018 11:00 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 20

I wrote a program to try all primes in increasing order. None up to about 12000. Then restarted from 30000, and found the first failing case at p=32749 which is the same prime as is used in matrix-based term-orderings.

I have also simplified Robbiano's original example, and have put my simplified example in the issue description.

We are still investigating.

The code should detect that the polys are univariate, and then use the univariate code. Doing this properly may take some time; for instance if the input polys are (x+1)*(y+1) and (x+1)*(z+1) it should be possible to notice that the only indet in common is x, so the problem should reduce to a univariate one.

#2 - 28 Jun 2018 11:01 - John Abbott

- Related to Design #707: MatrixOrderingMod32749Impl: test and write documentation! added

#3 - 28 Jun 2018 11:36 - Anna Maria Bigatti

- Assignee set to Anna Maria Bigatti

After discovering the *magic prime* it became clear that the problem is in the computation of the syzygies (the polynomial being univariate is not, yet, detected) uses a **matrix-defined ordering**.
This poses the hard limit to the exponents of 32749-1 in order to have machine-long computations modulo 32749.

Then the problem was: "where should we have the check?"
(unfortunately the checks in the code allow bigger exponents)

So I added the check in ApplySPRCodomain (and also in GeneralPPMonoidHomImpl::myApply).
That does detect the overflow in this case.

A better solution would be to have the checks done by OrdvArith in the concrete class, in place of the looser checks.

#4 - 28 Jun 2018 11:37 - Anna Maria Bigatti

- % Done changed from 20 to 40

#5 - 02 Aug 2018 18:38 - Anna Maria Bigatti

- Status changed from In Progress to Feedback

- % Done changed from 40 to 90

This is resolved.
The exponent check in the homomorphism calls is already quite good in detecting extreme examples like this (with exponents far beyond the limit).

A better refined check is a more general problem, to be dealt by issue #707

#6 - 03 Aug 2018 14:31 - Anna Maria Bigatti

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

#7 - 03 Aug 2018 15:47 - John Abbott

- Estimated time set to 3.33 h