CoCoALib - Bug #1641

gcd does not recognize univariate input

20 Dec 2021 13:43 - John Abbott

Status:	Closed	**Start date:**	20 Dec 2021
Priority:	High	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Improving	**Estimated time:**	5.33 hours
Target version:	CoCoALib-0.99850	**Spent time:**	5.35 hours

Description

Determinants where you would not expect them!

```
/**/ SetVerbosityLevel(100);
/**/ use ZZ/(19)[x,y[1..100]];
/**/ f1 := x^4 -x^2 +7*x -7;
/**/ f2 := x^4 +4*x^3 -6*x^2 -6*x +7;
/**/ gcd(f1,f2);
```

As soon as there is more than 1 indet in the ring CoCoA "stupidly" uses the syzygy method to compute the gcd even if the input polys are univariate!

Improve!

Related issues:

Related to CoCoALib - Feature #1197: IsZeroDet: new fn	**In Progress**	**26 Jun 2018**
Related to CoCoALib - Design #1649: Add file SparsePolyOps-vector.C	**Closed**	**21 Jan 2022**
Related to CoCoALib - Slug #1057: Slug: Polynomial ring contructor slow with ...	**In Progress**	**04 May 2017**
Related to CoCoA-5 - Slug #1068: PolyRing constructor: NewOrdvArith computed ...	**In Progress**	**17 May 2017**
Related to CoCoALib - Design #1798: Computing in sub polyring	**New**	**22 Mar 2024**

History

#1 - 04 Jan 2022 11:18 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

I have modified the code to detect univariate inputs and to handle them "cleverly". The result is noticeably faster!

Now to see what it does with inputs like gcd(x*y[1], (x+1)*y[2]) which obviously involves only x and neither of y[1] and y[2].

NOTE: my sudoku program now takes about 3s for one example where previously it took about 40s!

#2 - 07 Jan 2022 20:23 - John Abbott

- Assignee set to John Abbott

- % Done changed from 10 to 20

I have checked in my first change (so that Anna can experiment with it).
There is more to come (if/when I find time).

#3 - 21 Jan 2022 12:36 - John Abbott

- Related to Feature #1197: IsZeroDet: new fn added

#4 - 21 Jan 2022 13:07 - Anna Maria Bigatti

- Related to Design #1649: Add file SparsePolyOps-vector.C added

#5 - 16 Mar 2024 21:44 - John Abbott

- Priority changed from Normal to High

This may well be length to resolve properly, but I really should look at it soon.

#6 - 19 Mar 2024 20:18 - John Abbott

The problem code is in SparsePolyOps-RingElem.C around line 718 (search for SyzOfGens or maybe just syz).

I think I need a combination of ContentWRT and maybe ContentFreeFactor. Time to look these up in the CoCoALib doc - I do hope I actually wrote some doc for those two fns!

#7 - 19 Mar 2024 22:10 - John Abbott

- % Done changed from 20 to 30

Made some progress. This is more tedious than I thought... the doc for CoCoALib could be better... (and the design too)

#8 - 20 Mar 2024 22:03 - John Abbott

- Status changed from In Progress to Resolved

- % Done changed from 30 to 70

The new code seems to work now, and is faster if the polys are recognized as univariate.

What I do not understand is why syz is so slow:

```
/**/ use ZZ/(19)[u[1..100],x,y[1..100]];
/**/ f1 := x^4 -x^2 +7*x -7;
/**/ f2 := x^4 +4*x^3 -6*x^2 -6*x +7;
/**/ g1 := subst(f1,x,u[1]-y[1]);
/**/ g2 := subst(f2,x,u[1]-y[1]);
/**/ t0 := CpuTime(); syz([g1,g1]); TimeFrom(t0);
--> TAKES 11s on my computer
```

BUT if I do the computation in the smallest suitable ring (with indets u[1],x,y[1]) then it takes 0.02s
Probably this should be a new issue!

#9 - 21 Mar 2024 10:01 - Anna Maria Bigatti

John Abbott wrote:

> The new code seems to work now, and is faster if the polys are recognized as univariate.
>
> What I do not understand is why syz is so slow:
> [...]
> **BUT** if I do the computation in the smallest suitable ring (with indets u[1],x,y[1]) then it takes 0.02s

On my computer this takes 0.222s (0.001 with indets u[1],x,y[1]).
Do you have debugging on?

#10 - 21 Mar 2024 10:25 - John Abbott

Ah yes, I do have debugging on.
Do you get a measurable time difference if the ring contains just 3 indets or if it contains 201 indets? Or even 2001?
It could be that there is some debugging check which takes a lot of time...

I'll investigate further this evening.

#11 - 21 Mar 2024 11:31 - Anna Maria Bigatti

It seems it's the "high number of variables" problem, and syz itself is quite fast: this examples takes ~4s on my computer.
Try with verbosity (first check out by verbosity additions):

```
/**/ SetVerbosityLevel(130);
/**/ /**/ use ZZ/(19)[u[1..400],x,y[1..400]];
/**/ f1 := x^4 -x^2 +7*x -7;
/**/ f2 := x^4 +4*x^3 -6*x^2 -6*x +7;
/**/ g1 := subst(f1,x,u[1]-y[1]);
/**/ g2 := subst(f2,x,u[1]-y[1]);
/**/ t0 := CpuTime(); syz([g1,g1]); TimeFrom(t0);
```

#12 - 21 Mar 2024 11:33 - Anna Maria Bigatti

- Related to Slug #1057: Slug: Polynomial ring contructor slow with (big) matrix ordering added

#13 - 21 Mar 2024 11:54 - Anna Maria Bigatti

- Related to Slug #1068: PolyRing constructor: NewOrdvArith computed twice added

#14 - 21 Mar 2024 12:02 - Anna Maria Bigatti

The problem for multivariate syz with many indets (from note-8 on) seems to be considered in #1057 and #1068, so I would just close this issue.

#15 - 21 Mar 2024 20:18 - John Abbott

- Status changed from Resolved to Closed

- % Done changed from 70 to 100

- Estimated time set to 5.33 h

I have added some new tests to test-SparsePolyRing1.C.
I conform that Anna's example from comment 11 does seems to spend a long time "doing not much (apparently)" then whoosh the GB computation is over in a flash!
The original example is now completed in about 0.001s (on my Linux laptop).
Closing.

#16 - 22 Mar 2024 09:25 - John Abbott

- Related to Design #1798: Computing in sub polyring added