CoCoALib - Bug #154

GCD normalization (e.g. monic)

07 May 2012 14:30 - John Abbott

Status:	In Progress	**Start date:**	07 May 2012
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	20%
Category:	Tidying	**Estimated time:**	0.00 hour
Target version:	CoCoALib-1.0	**Spent time:**	1.15 hour

Description

GCDs in a poly ring over a field are defined only upto a constant factor.
If the field is a FractionField then there's a fairly natural choice of normalized factor.
If it is not then the preferred choice of normalization is unclear. There are three options:

1. no normalization, the costant factor is whatever it turns out to be
2. make the factor monic (this is surely canonical)
3. some other normalization (but what???)

Note that making the factor monic can be "unnatural". Suppose the result is 7*x+8 in ZZ/(32003)[x]; making this monic produces the less comprehensible x+4573.

Related issues:

Related to CoCoALib - Feature #50: Polynomial content	**Closed**	**30 Nov 2011**
Related to CoCoALib - Feature #259: Squarefree(?) GCD-free basis	**Closed**	**09 Oct 2012**

History

#1 - 18 Dec 2013 14:20 - Anna Maria Bigatti

- Subject changed from GCD normalization to GCD normalization (e.g. monic)

- Category set to Tidying

- Target version set to CoCoALib-0.99534 Seoul14

- % Done changed from 0 to 10

I agree that it would be convenient having a guarantee on the result, but the operation might be uselessly costly (for example if I'm interested in knowing only the degree of the GCD).

It might be a good idea having two functions:
- gcd, GcdMonic (or whatever)
or
- GcdNonMonic, - gcd

#2 - 18 Dec 2013 14:30 - John Abbott

I think we should guarantee that if the gcd is trivial then the result is 1, so users can write clear code such as if (gcd(f,g) == 1) ...

The function called **gcd** should give this guarantee.

I prefer that the fn called gcd give a normalized result, and that the possibly faster function which may give unnormalized results have a "strange" name.

#3 - 18 Dec 2013 14:53 - Anna Maria Bigatti

John Abbott wrote:

> I think we should guarantee that if the gcd is trivial then the result is 1, so users can write clear code such as if (gcd(f,g) == 1) ...
> The function called **gcd** should give this guarantee.

There is a trivial solution for this case: when the final result of gcd is invertible we just return one(R).
That is cheap (just depends on IsInvertible which is usually fast) performs no other operation, and has no ambiguity in the definition!

#4 - 09 Jul 2014 18:09 - John Abbott

- Target version changed from CoCoALib-0.99534 Seoul14 to CoCoALib-1.0

#5 - 24 Nov 2016 13:56 - John Abbott

- Status changed from New to In Progress

- % Done changed from 10 to 20

I stumbled across this problem/issue again when trying to implement GCDFreeBasis (see #259).

In reference to my comment 2 (above), the code would actually be clearer if I wrote if (IsCoprime(f,g)) which ought to be marginally cheaper than if (gcd(f,g)==1) since the latter requires converting 1 to an element of the relevant ring.

We could try to implement two fns, "normalized" GCD and "unnormalized" GCD, and then do some tests to compare them. My expectation is that they are equally fast/slow in almost all cases.

Note that we can normalize a single element just by computing gcd(elem,0)

Here are some sugestions for what "normalized" could mean:

- in ZZ the result is non-negative.
- in ZZ[...] the result has positive leading coeff.
- in QQ[...] the result is ZZ-primitive with positive leading coeff (special case: gcd(0,0) = 0).
- in FrF[...] where FrF is a field of fractions, the result is primitive in "base ring" of k and coeff of LPP is normalized.
- in k[...] with k a field, the result is monic.
- in R[a,b][x,y] the result is the same as in R[a,b,x,y].

I am not actually sure that the above rules are exhaustive, but I think they are a reasonable starting point.

These rules do imply that we get the "hard to comprehend" result x+4573 instead of 7*x+8 in the example given in the initial description.

There are some other possibly "surprising" results: *e.g.* in QQ[x] we get gcd(x+1/2, x^2-1/4) = 2*x+1

Maybe there could be an extra rule that in k[...] if all args are monic then the result is monic?

Comments? Ideas? Criticisms?

#6 - 24 Nov 2016 13:57 - John Abbott

- Related to Feature #259: Squarefree(?) GCD-free basis added