CoCoALib - Bug #1710

IsSqFree, IsIrred bugs in ZZ[x] and QQ[x]

16 Nov 2022 09:33 - John Abbott

Status:	Closed		**Start date:**	16 Nov 2022
Priority:	Urgent		**Due date:**	
Assignee:	John Abbott		**% Done:**	100%
Category:	Maths Bugs		**Estimated time:**	3.90 hours
Target version:	CoCoALib-0.99850		**Spent time:**	3.85 hours

Description

Nico Mexis reported by email:

```
use ZZ[x];
f := -10*x^2 -20*x -20;
println [IsSqFree(f), gcd(f, deriv(f,x))];
/**/ [false, 10]

use QQ[x];
f := -10*x^2 -20*x -20;
println [IsSqFree(f), gcd(f, deriv(f,x))];
/**/ [true, 1]
```

As far as I can tell, it could suffice to replace the IsCoprime calls in PolyRing.C (lines 82 and 97) by IsConstant calls.
However, I do not know which side effects that could have or if that is completely bogus.

Also, the same applies to IsIrred (and most likely many more functions).
It seems rather odd that IsIrred on -10*x^2 -20*x -20 returns false in QQ[x], but true in ZZ[x] ;)

Related issues:

Related to CoCoALib - Feature #1108: New fn: IsCoprime (whenever gcd makes se...	**Closed**	**17 Oct 2017**

History

#1 - 16 Nov 2022 16:29 - John Abbott

- % Done changed from 0 to 10

Nico also send the following by email:

```
use ZZ[x];
gcd(10, 10*x);
/**/ 10

use QQ[x];
gcd(10, 10*x);
/**/ 1
```

Returning 10 is a reasonable thing to do in ZZ[x], but then it should be the same in QQ[x], I think.
However, this obviously affects IsCoprime since it checks if the gcd is invertible (if I remember correctly).

#2 - 16 Nov 2022 16:34 - John Abbott

- Status changed from New to In Progress

In response to note-1:

Without doubt, in ZZ[x] we must have gcd(10, 10*x) = 10; conceivably it could also be -10, but JAA prefers a positive LC.

In QQ[x] the result is defined only up to an invertible factor (same as in ZZ[x], but the only invertibles there are 1 and -1). JAA would find it surprising if, in the ring QQ[x], the computation gcd(10, 10*x) gave any value other than 1.

#3 - 16 Nov 2022 16:56 - John Abbott

Here are some simple test cases:


```
use ZZ[x];
f := 2*x+2;
IsSqFree(f); // expect true
IsIrred(f);  // expect true (if we ignore non-invertible coeffs)
```


2022-11-16: both tests produce false

Must also clarify doc!

#4 - 16 Nov 2022 17:00 - John Abbott

- % Done changed from 10 to 20

Here is a test case where I am unsure what the correct result should be:


```
use ZZ[x];
f := -(2*x+2);
factor(f);
-->  record[RemainingFactor := -2, factors := [x +1], multiplicities := [1]]
SqFreeFactor(f);
-->  record[RemainingFactor := 2, factors := [-x -1], multiplicities := [1]]
```


Right now I have a preference for making leading coeffs positive (in this case).
What do you think?

#5 - 16 Nov 2022 17:24 - John Abbott

- Status changed from In Progress to Resolved

- % Done changed from 20 to 80

I have modified the source code (following Nico's suggestions, more or less)
I have added a caution to the doc. I have added a new exbug test for CoCoALib.
Will check in shortly!

#6 - 17 Nov 2022 06:50 - John Abbott

Since **IsCoprime** can behave "unexpectedly" in ZZ[x]...
Should we limit the applicability of **IsCoprime**?

For example, it could give an error if the ring is not ZZ or a polynomial ring over a field...

Removing IsCoprime completely seems too drastic.

#7 - 23 Nov 2022 18:25 - John Abbott

Anna will think about this.

#8 - 30 Nov 2022 19:21 - Anna Maria Bigatti

John Abbott wrote:

> Since IsCoprime can behave "unexpectedly" in ZZ[x]...
> Should we limit the applicability of IsCoprime?

I think that it is right to give an error because it might be ambiguous: IsCoprime(2*x, 2*(x-1)). In the manual we could suggest to run either
IsConstant(gcd(2*x, 2*(x-1))); or IsInvertible(gcd(2*x, 2*(x-1)));

#9 - 30 Nov 2022 21:33 - John Abbott

- Related to Feature #1108: New fn: IsCoprime (whenever gcd makes sense) added

#10 - 14 Dec 2022 21:05 - John Abbott

- Status changed from Resolved to Feedback

- % Done changed from 80 to 90

I have added a new "exbug" test. I have updated the code in CoCoALibSupplement.C.
I have revised the CoCoA-5 manual.

One small doubt: as it currently stands one can do IsCoprime(M,N) if M and N are
elements of the ring ZZ, but it does not work if they are of type INT. There are
two obvious ways to solve this inconsistency:

- disable IsCoprime for the ring ZZ
- enable IsCoprime for INT values

What do you think?

PS for integers we can always do gcd(M,N)=1
PPS CoCoALib does have IsCoprime for BigInt... that probably dictates the answer here!

#11 - 15 Dec 2022 09:15 - Nico Mexis

If all else fails, the issue in CoCoA 5 could be fixed in the same manner as factor by providing an extra IsCoprimeINT function.
However, I would agree that this is not an optimal solution.

#12 - 06 Mar 2023 22:41 - John Abbott

I have now extended **IsCoprime** so that it covers integers and ring elems,and a mixture of the two.
Added tests, and doc.
Will check in shortly.

#13 - 07 Mar 2023 20:55 - John Abbott

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

- Estimated time set to 3.90 h