CoCoALib - Support #1499

factorization: allow zero as exponent?

05 Oct 2020 13:23 - John Abbott

Status:	Closed	**Start date:**	05 Oct 2020
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Documentation	**Estimated time:**	1.81 hour
Target version:	CoCoALib-0.99800	**Spent time:**	1.90 hour

Description

A factorization object in CoCoALib contains a list factor-multiplicity pairs.
According to the doc the factors must be non-zero and not invertible, and the multiplicities must be positive.

Currently, the err mesg produced if a zero multiplicity is specified says "negative exponent" not allowed. This is incorrect/confusing.

If we want to use a factorization to represent the result of FactorMultiplicity (see #1463) then we would need to allow 0 as a multiplicity (**or** be prepared to handle the case that the list of factors found is empty)

Discuss, decide, implement.

Related issues:

Related to CoCoALib - Design #1463: SmoothFactor: use FactorMultiplicity	**Closed**	**16 Jun 2020**

History

#1 - 05 Oct 2020 13:24 - John Abbott

- Related to Design #1463: SmoothFactor: use FactorMultiplicity added

#2 - 05 Oct 2020 13:26 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

My current feeling is that I should simply change the error mesg, and leave the rest as is: namely, multiplicities must be (strictly) positive.

This would be quick and easy to achieve. But first I would like to hear if there any contrasting opinions.

#3 - 06 Oct 2020 10:12 - John Abbott

- Subject changed from factoization: allow zero as exponent? to factorization: allow zero as exponent?

#4 - 06 Oct 2020 10:22 - John Abbott

- % Done changed from 10 to 20

The current interface for FactorMultiplicity does not use a factorization.
Even if an alternative interface (like that to DivideOutMaxPower) is desired, using a factorization object is not especially helpful. And if we really want to use a factorization object, we can just skip factors with multiplicity zero.

In other words, I see no justification in allowing 0 as an exponent in the representation.

What might be reasonable is to allow factorization::myAppend to be called with 0 multiplicity (in which case it simply ignores the factor). What do you think?

#5 - 09 Oct 2020 11:11 - John Abbott

- Assignee set to John Abbott

If we allow 0 as a multiplicity when calling myAppend then it should also be allowed in the ctor call which accepts two vectors (factors & mults). This means that the ctor must be slightly rewritten... not really a problem, just a little tedious.

NOTE is seems that the main change would be havng to move the code from ourConsistencyCheck into the ctor itself; it seems that ourConsistencyCheck is not called from elsewhere.

#6 - 09 Oct 2020 13:27 - Anna Maria Bigatti

John Abbott wrote:

> My current feeling is that I should simply change the error mesg, and leave the rest as is: namely, multiplicities must be (strictly) positive.
>
> This would be quick and easy to achieve. But first I would like to hear if there any contrasting opinions.

I agree

#7 - 14 Oct 2020 11:17 - John Abbott

- % Done changed from 20 to 30

I now prefer not to allow 0 as a multiplicity because it could be confusing: *e.g.* someone might call myAppend(fac,0) and then be surprised that nothing was appended to the list of factors.

Also it was not wholly clear to me what myAppend(1,0) should do: the point is that the multiplicity is 0, so the call should be "ignored", but in this instance the factor is a forbidden one (factors may not be invertible or zero-divisors), so perhaps an error should be signalled???

In other words, **I agree with comment 2**.

#8 - 14 Oct 2020 21:38 - John Abbott

- Status changed from In Progress to Resolved

- % Done changed from 30 to 80

I have implemented as decided (*i.e.* mults must be positive)... effectively we are rejecting the original proposal.

Should factorization throw an exception if the ringelems are in a field?
It is possible to create a factorization object over a field, but there can be only a RemainingFactor (since all values are zero-divs or invertible).

NOTE factorization does not allow automatic ring conversion -- I think it is probably better this way (when would one want to make a factorization with factors in different rings??)

#9 - 26 Oct 2020 12:26 - John Abbott

- Description updated

- Status changed from Resolved to Closed

- % Done changed from 80 to 100

- Estimated time set to 1.81 h

FINAL DECISION:

- do not allow 0 multiplicity (will throw an exception)
- do not allow factorization objects over a field (will throw an exception)
- do not allow automatic ring conversion (with throw MixedRings)

Will test the new code shortly... 8-|