CoCoALib - Design #846

IsIrred: correct design?

21 Mar 2016 13:46 - John Abbott

Status:	In Progress	**Start date:**	21 Mar 2016
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	20%
Category:	Improving	**Estimated time:**	0.00 hour
Target version:	CoCoALib-1.0	**Spent time:**	1.85 hour

Description

What is the correct way to design/implement IsIrred for a RingElem?

I would like a design that does not need use to implement "largely pointless" (member) functions.

Also note that the current impl produces an ERR::SERIOUS (rather than ERR::NYI)for unimplemented cases.

Related issues:

Related to CoCoALib - Feature #885: IsIrred3: fast 3-way irred test (returnin...	**In Progress**	**24 May 2016**

History

#1 - 21 Mar 2016 14:05 - John Abbott

- Category set to Improving

What are the exact semantics of IsIrred? Does it apply only to RingElem? [JAA: yes]

Wikipedia says that an **irreducible element** is a non-zero, non-invertible element of an integral domain which has no non-trivial factorization. So what should IsIrred do if:

- it is given an element of a non-integral-domain? [JAA: error]
- it is given zero in an integral domain? [JAA: false]
- it is given an invertible element in an integral domain? [JAA: false]

If the RingElem belongs to a field, should IsIrred return false or give error?

Note that the 3 "easy" cases above could all be handled by a generic function. An input which is not one of these easy cases requires handling depending on the ring (*i.e.* via a member function).

For which specific rings do we need to implement myIsIrred?

- RingZZ where it just tests primality (or pseudo-primality?)
- poly rings over a field
- any other cases?

JAA thinks we should continue with the policy that a ring is a poly ring iff it is implemented as such; for instance QQ[x,y]/ideal(x) is **not** a poly ring (even though it is canonically isomorphic to QQ[y])

#2 - 21 Mar 2016 14:07 - John Abbott

- Target version set to CoCoALib-0.99550 spring 2017

#3 - 24 May 2016 17:27 - John Abbott

- Related to Feature #885: IsIrred3: fast 3-way irred test (returning bool3) added

#4 - 25 May 2016 13:43 - John Abbott

- Status changed from New to In Progress

- Assignee set to John Abbott

- % Done changed from 0 to 20

I think a first version of IsIrred should give error if the ring is a field; if it turns out that this is inconvenient, then it will be easy to replace the error with false, and that should not break any existing code.

The main work of IsIrred will be done by ring mem fns called myIsIrred. Which sanity checks should be performed by IsIrred itself, and which by myIsIrred? The usual rule is that mem fns do not perform sanity checks (except when debugging is enabled), so that should be done by IsIrred.

Is there any real practical distinction between performing the sanity checks and handling the trivial cases? Maybe. I think that the mem fn IsIrred should always handle the trivial cases, then perhaps there should be a private mem fn for handling the non-trivial cases in a ring specific manner. What should this private mem fn be called? How about IsIrredNonTriv? Or IsIrredGeneralCase? Or maybe IsIrredPrivate?

Which fn should call IsIrred3? (see issue #885)

#5 - 25 May 2016 16:49 - John Abbott

The simplest design is to make IsIrred call IsIrred3; this somehow equates calling IsIrred3 to handling the trivial cases -- indeed this is quite reasonable because IsIrred3 effectively generalizes the notion of "trivial case". With this viewpoint, it could make sense to put the trivial checks inside IsIrred3 (if this can be done without having to copy the code for each ring which offers a specific IsIrred3 impl).

#6 - 16 Sep 2016 16:15 - John Abbott

- Target version changed from CoCoALib-0.99550 spring 2017 to CoCoALib-0.99560

#7 - 06 Nov 2017 14:54 - John Abbott

- Target version changed from CoCoALib-0.99560 to CoCoALib-0.99600

#8 - 09 Nov 2017 13:57 - John Abbott

What should **IsIrred** do in the following cases:

- arg is not in an integral domain [error?]
- arg is zero [error? or false?]
- arg is an invertible element [error? or false?]

What are your opinions? My current preference is error in all 3 cases (though I am a little uneasy about giving error if the arg is an invertible element)

#9 - 25 Jun 2018 12:05 - John Abbott

- Target version changed from CoCoALib-0.99600 to CoCoALib-0.99650 November 2019

#10 - 26 Feb 2019 16:32 - John Abbott

- Target version changed from CoCoALib-0.99650 November 2019 to CoCoALib-1.0