CoCoALib - Feature #261

Review the utility of RefRingElem

10 Oct 2012 10:33 - John Abbott

Status:	Closed	**Start date:**	10 Oct 2012
Priority:	High	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Safety	**Estimated time:**	0.00 hour
Target version:	CoCoALib-0.9952	**Spent time:**	31.50 hours

Description

From studying issue #221 it has become apparent that the role of **RefRingElem** needs urgent review.

Related issues:

Related to CoCoALib - Feature #221: Better RingElems	**Closed**	**08 Aug 2012**

History

#1 - 10 Oct 2012 10:36 - John Abbott

JAA is currently assessing how useful/necessary RefRingElem is; the hope is that they can simply be eliminated (along with the associated difficulties -- see #221).

#2 - 10 Oct 2012 12:07 - John Abbott

I am considering eliminating the type RefRingElem. To do this I must find out where the type is used.

The type RefRingElem appears in many function signatures because the guidelines were to use it whenever a modifiable "ring elem" was to be passed; the understanding being that a true RingElem is automatically converted to RefRingElem by C++ automatically passing to a base class. If the class RefRingElem is eliminated then all these formal parameters would naturally be replaced by ones of type RingElem&.

The crucial question is *where are true RefRingElem objects created?*
And in each case *can a reasonable alternative implementation be found?*

The only places I have found where RefRingElem values are created are:

- **A** ModuleElem::operator[] for non-const values; also the myCompt member fn
- **B** <matrix>::myRefEntry
- **C** RefLC for DistrMPolyClean and DistrMPolyInlPP
- **D** RingWeyl::myMulByPP and RingWeyl::myReductionStep

Case **A** is probably a mistake in design; it is not used anywhere in CoCoALib/CoCoA-5, so can easily be eliminated (and probably replaced by a SetEntry procedure as for matrices).

Case **B** the result of myRefEntry is used only inside BlockMatrix::mySetEntry and ConcatVerImpl::mySetEntry; a member fn similar to myRefEntry is fairly essential, but it could return simply a RingElemRawPtr. Overall: it would not be too problematic to eliminate use of RefRingElem in this case.

Case **C** the result of RefLC is used in just two places, and in fact only the RingElemRawPtr is used. It would be easy to redefine RefLC to produce a RingElemRawPtr.

Case **D** the two uses of RefRingElem can be eliminated, but will lead to a slight uglification of the code.

#3 - 10 Oct 2012 15:51 - John Abbott

I have modified the source code in the 4 areas indicated in my previous post. It all compiles and the tests pass.

Case **A**: I simply commented out the code; no further changes were needed as the troublesome functions were never called anywhere.

Case **B**: I replaced all **myRefEntry** mem fns with **myRawEntry**; the only uses are in some mySetEntry mem fns, and I'm undecided whether the modified code is better or worse than before -- it is slightly harder to read, but also more explicit.

Case **C**: I replaced **RefLC** with **RawLC** which gives a raw ptr to the LC -- an ugly workaround. I'm hoping to find a cleaner solution.

Case **D**: the fn **RingWeylImpl::myMulByPP** is no better or worse than before; **RingWeylImpl::myReductionStep** is a little less readable than before (but still just about acceptable).

#4 - 11 Oct 2012 16:12 - John Abbott

I have implemented a reasonably clean solution to case **C**; it is exception safe but does make a "wasteful copy".

I have commented out the class **RefRingElem** and replaced it with a typedef for RingElem&. There were a few further manual changes too.
Everything compiles, and the tests all pass. Some dodgy code was highlighted by the compiler -- this shows that the 3 layer hierarchy ConstRefRingElem to RefRingElem to RingElem was in fact flawed!

Next step is to text-replace RefRingElem with RingElem& (and hope that everything still compiles afterwards).

#5 - 11 Oct 2012 18:34 - John Abbott

- % Done changed from 0 to 50

I have replaced all occurrences of **RefRingElem** (usually by **RingElem&**),
and made a few scattered minor changes so that everything compiles.

I'll do the CVS check in tomorrow -- it's too late, and I'm too tired to do it safely now.

The documentation will need to be updated too!

#6 - 15 Oct 2012 14:54 - John Abbott

- % Done changed from 50 to 60

CVS check-in blocked until new release of CoCoALib has been made.

I'll look at the documentation in the meantime.

#7 - 17 Oct 2012 20:10 - John Abbott

- % Done changed from 60 to 70

Checking in completed.
Documentation has still to be updated.

#8 - 24 Oct 2012 16:32 - John Abbott

- % Done changed from 70 to 80

I have renamed the class ConstRefRingelem into **RingElemAlias**, and introduced a new typedef for **ConstRefRingElem**. There were many consequential changes. It compiles and all tests pass, so I have checked in.

The documentation still needs to be updated.

#9 - 24 Oct 2012 16:56 - Anna Maria Bigatti

confirm: all compiles and runs smoothly :-)

#10 - 26 Oct 2012 17:13 - John Abbott

- Status changed from In Progress to Resolved

- % Done changed from 80 to 100

I have updated the doc for RingElem and friends.
It probably needs some more work, but I'm taking a rest over the weekend.

#11 - 18 Feb 2013 12:50 - John Abbott

- Status changed from Resolved to Closed

Since no problems have surfaced in 4 months, I regard this matter as fully resolved, so I am closing it.