CoCoALib - Design #415

Remove AsPolyRing etc?

23 Nov 2013 17:02 - John Abbott

Status:	Closed	**Start date:**	23 Nov 2013
Priority:	High	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Tidying	**Estimated time:**	20.00 hours
Target version:	CoCoALib-0.99534 Seoul14	**Spent time:**	18.20 hours

Description

I'm fed up with having to make explicit "manual" casts from ring to PolyRing etc.

I've just fixed a line of code which looked correct but did not compile:

```
CoCoA_ASSERT(IsSparsePolyRing(owner(f)) && IsFiniteField(CoeffRing(owner(f))));
```

To make it compile I had to insert a call to AsSparsePolyRing inside CoeffRing. Frankly I find it irritating to have to do this, and I do not believe it improves the readability of the code.

I believe we can eliminate these fns:
AsPolyRing, AsSparsePolyRing, AsFractionField, AsQuotientRing, AsDenseUPolyRing, AsRingTwinFloat.

It is just possibile that eliminating them might produce an ambiguity; if so, I'm sure it can easily be resolve by a minor name change.

Perhaps also AsFGModule and AsFreeModule?

Comments? Opinions? Ideas?

Related issues:

Related to CoCoALib - Design #584: BaseRing for all rings	**Closed**	**08 Jul 2014**
Is duplicate of CoCoALib - Feature #139: Usefulness of ring casting fns (remo...	**Closed**	**27 Apr 2012**

History

#1 - 25 Nov 2013 11:42 - Anna Maria Bigatti

- Target version set to CoCoALib-0.99532

- % Done changed from 0 to 10

John Abbott wrote:

> Frankly I find it irritating to have to do this, and I do not believe it improves the readability of the code.

I agree: readability would improve without this "feature".
In fact we added it so that we get an error at compile time instead of run time, but the point is that just a "AsBlahRing" would make it compile .... and give the error at run time!
It is true that this forces the user to think before writing "AsBlahRing", but it an extra-safety that does not help "enjoying CoCoALib".

I think we can fix the ambiguities and make the code more readable and enjoyable.

#2 - 01 Apr 2014 19:38 - Anna Maria Bigatti

- Target version changed from CoCoALib-0.99532 to CoCoALib-0.99533 Easter14

#3 - 07 Apr 2014 18:24 - John Abbott

- Target version changed from CoCoALib-0.99533 Easter14 to CoCoALib-0.99534 Seoul14

#4 - 15 Apr 2014 15:07 - John Abbott

- Status changed from New to Closed

Closing because it is duplicated

#5 - 15 Apr 2014 15:09 - John Abbott

- Status changed from Closed to In Progress

- Assignee set to John Abbott

- Priority changed from Normal to High

Reopening because it was auto-closed when I closed #139 which this duplicates.

#6 - 11 Jun 2014 17:40 - John Abbott

- % Done changed from 10 to 20

I have managed to remove most of the need for AsPolyRing; will continue later this evening.

#7 - 12 Jun 2014 19:30 - John Abbott

Everything compiles & all tests pass :-)

It would be handy to have a ctor for RingElem which accepts a RingBase* instead of a ring; it would make some functions slightly neater and slightly faster (because it avoids incr/decr of a refcount, but you probably wouldn't notice the better speed). Maybe I'll try implementing, and see how I feel about it...

#8 - 13 Jun 2014 08:03 - Anna Maria Bigatti

John Abbott wrote:

> It would be handy to have a ctor for RingElem which accepts a RingBase* instead of a ring; it would make some functions slightly neater and slightly faster (because it avoids incr/decr of a refcount, but you probably wouldn't notice the better speed).

Does that mean that such a RingElem would have a ring that is not ref-counted (i.e. it should only be used as a temporary so that it does not live beyond its ring?)

JAA reply no, the RingElem would be normal (complete with ref count); what I'd like is to be able to write RingElem ans(RingPtr) rather than RingElem(ring(RingPtr)), this latter has an extra incr/decr of the ref count for the ring.

#9 - 08 Jul 2014 11:17 - John Abbott

- % Done changed from 20 to 40

- Estimated time set to 20.00 h

I've removed AsPolyRing, AsSparsePolyRing and AsFractionField.

So far so good. Some bits of code have become a bit neater, nothing has become less readable.

I do foresee a "problem". Consider the function(s) called InducedHom. At the moment there are two of them (with almost identical signatures), one for FractionField and one for QuotientRing. Once I introduce an automatic conversion from ring to QuotientRing we will get an ambiguity in a call like InducedHom(R, phi) since the compiler won't know whether to convert R into a FractionField or a QuotientRing.

I could define a new function also called InducedHom which then tests the actual type of its first arg and then calls the appropriate concrete InducedHom function; alternatively the user must explicitly write InducedHom(FractionField(R), phi) which disambiguates.

Similar comments apply to the functions BaseRing and EmbeddingHom.

#10 - 08 Jul 2014 17:49 - John Abbott

- % Done changed from 40 to 50

I've now removed AsQuotientRing, and revised the imple of BaseRing (see #584)

#11 - 09 Jul 2014 17:50 - John Abbott

- % Done changed from 50 to 70

I have removed AsDenseUPolyRing, AsFreeModule and AsFGModule.
The code compiles and all tests pass, so I have checked in.

It was running slowly on my computer (but that might be incidental).

No doc, no examples, no (specific) tests.

#12 - 10 Jul 2014 15:32 - John Abbott

- Status changed from In Progress to Feedback

- % Done changed from 70 to 90

I've updated the documentation (removed references to AsPolyRing etc.)

tests: I can't really test a function I've removed from the library :-)
examples: idem (mutatis mutandis)

--> feedback

#13 - 14 Jul 2014 15:07 - Anna Maria Bigatti

nice! :-)
some builtin functions are now oneliners
(cvs-ed)

#14 - 17 Jul 2014 14:29 - John Abbott

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

No problems after 1 week, so closing.