CoCoALib - Feature #587

port to CoCoALib: Homomorphism pkg (ker, IsInjective, IsSurjective..)

14 Jul 2014 10:39 - Anna Maria Bigatti

Status:	Closed	**Start date:**	14 Jul 2014
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	New Function	**Estimated time:**	18.80 hours
Target version:	CoCoALib-0.99600	**Spent time:**	18.60 hours

Description

Translate the package HomomorphismFns.cpkg5 into CoCoALib.
The related "utility representation" of a homomorphism as an ideal into a bigger ring should be stored inside the RingHom as a "mutable".

Related issues:

Related to CoCoA-5 - Feature #273: Package for Polynomial Algebra Homomorphisms	**Closed**	**12 Nov 2012**
Related to CoCoA-5 - Bug #1140: ImplicitModular: too many bad primes	**Closed**	**14 Dec 2017**
Related to CoCoA-5 - Support #242: CoCoA-5 Projects for students (e.g. credit...	**In Progress**	**28 Sep 2012**

History

#1 - 15 Dec 2014 12:45 - Anna Maria Bigatti

- Estimated time changed from 4.00 h to 16.00 h

#2 - 17 Jul 2017 14:20 - John Abbott

- Status changed from New to In Progress

- Assignee set to John Abbott

- % Done changed from 0 to 30

Since I had to assign some student projects after the CoCoALib mini-course (here in Kassel),
I gave the task of translating HomomorphismFns.cpkg5 into C++ to two students.

They seem to have done a good job (including finding some test cases).

Outstanding matters:

1. incorporate the RichHom structure into the data structure(s) representing the homomorphism
2. the students noticed that KerRichHom sometimes produces an answer with redundant generators (if domain is a quotient)
3. the students also implemented a new fn PreimageAndKer
4. the internal fn MakePreimageRichHom always computes the ker even when the arg is not in the image, right?

For point **(2)** we could take CanonicalRepr of the generators, add the gens of the DefiningIdeal of the quotient, compute RGB then take those elements of RGB which are not in the DefiningIdeal. This would give a more "canonical" answer, but it could also be "wasted effort" in some cases??

#3 - 17 Jul 2017 14:21 - John Abbott

Should I log the time the students spent on this project? If so, how? (and where?)

#4 - 17 Jul 2017 14:22 - John Abbott

- Description updated

#5 - 23 Jul 2017 16:02 - John Abbott

- % Done changed from 30 to 60

My "clever idea" in comment 2 for cleaning the gens of the kernel does not work well.

I have cleaned the students' code, and it is almost ready for public release, but... I have a question.

The return value of **preimage** does not convince me much: currently it returns a flag IsInImage, and possible 2 other fields (a particular preimage, and the ker). I think the argument is that determining whether the value lies in the image involves computing the RGB of "idealRS" (since an NF must be computed), and a set of generators for the ker can be read off this RGB.

If we plan to store the "RichHom" structure inside the datastructure for the homomorphism then the RGB of "idealRS" should be remembered from one call to another. So there is no real gain in returning a compound result... after the integration has been performed.

I propose a new fn **preimage0** which returns either a preimage or the zero element of the domain. The only "ambiguity" is when asking for the preimage of zero, but that is easy for a user to handle.

What do you think?

#6 - 24 Jul 2017 11:10 - Anna Maria Bigatti

John Abbott wrote:

> I propose a new fn **preimage0** which returns either a preimage or the zero element of the domain. The only "ambiguity" is when asking for the preimage of zero, but that is easy for a user to handle.

I like this idea. In fact I had a similar problem in another situation.
The "real" function preimage returns error when input is unsuitable, whether preimage0 does something special with/to 0 (and the user is warned by the 0).

This approach could be useful in other situations.

#7 - 24 Nov 2017 15:28 - John Abbott

- Target version changed from CoCoALib-1.0 to CoCoALib-0.99560

What is the status of this issue? Isn't it nearly finished?

Related to thi is the use of PreImage in some CoCoA-5 packages: I think we need to revise subalgebra.cpkg5 and also obsolescent.cpkg5 (which prints out a misleading advisory message).

#8 - 30 Nov 2017 17:32 - Anna Maria Bigatti

Improve the code adding weights and homogenization...
should be trivial...
(I will do it)

#9 - 07 Dec 2017 17:42 - Anna Maria Bigatti

After doing some experiments we (Lorenzo Robbiano and I) are now convinced that it is always worth giving weights and homogenizing (as for ImplicitHypersurface/ImplicitModular).

We discussed the mathematics (all ok) and implemented the first function ker_H modifying minimally (even though painfully ;-) the code for ker.
The point is that the ideal representing the homomorphism is homogenous (with an extra homogenizing indet), so all functions should deal with what this implies: ker_H just dehomogenizes the output (and will be eventually replace ker).

#10 - 14 Dec 2017 14:29 - Anna Maria Bigatti

ker should use implicit, and implicit should return (if not otherwise indicated) the answer in a ring with a compatible ordering, so that the result is a GBasis.

#11 - 14 Dec 2017 14:31 - Anna Maria Bigatti

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

#12 - 14 Dec 2017 16:26 - Anna Maria Bigatti

- Related to Bug #1140: ImplicitModular: too many bad primes added

#13 - 20 Dec 2017 08:51 - Anna Maria Bigatti

- Subject changed from port to CoCoALib: ker, IsInjective, IsSurjective.. to port to CoCoALib: Homomorphism pkg (ker, IsInjective, IsSurjective..)

#14 - 20 Dec 2017 08:53 - Anna Maria Bigatti

bug in IsInImage (from cocoa5).

```
/**/  QQxyz ::= QQ[x,y,z];
/**/  QQab  ::= QQ[a,b];

/**/  use QQab;
/**/  phi := PolyAlgebraHom(QQxyz, QQab, [a+1, a*b+3, b^2]);
/**/  IsInImage(phi, b);

Assertion failed: (px != 0), function operator->, file ../../configuration/ExternalLibs/include/boost/smart_pt
r/intrusive_ptr.hpp, line 174.

Process cocoa5 abort trap: 6
```

#15 - 20 Dec 2017 09:06 - Anna Maria Bigatti

Anna Maria Bigatti wrote:

> bug in IsInImage (from cocoa5).
> [...]

working on it, very strange: code looks honest

#16 - 20 Dec 2017 09:37 - Anna Maria Bigatti

- Related to Support #242: CoCoA-5 Projects for students (e.g. crediti F and tesi) added

#17 - 20 Dec 2017 09:39 - Anna Maria Bigatti

Fixed: it was just an "ARG (3)" instead of "ARG (1)" in BuiltinFunctions...

#18 - 03 Aug 2018 18:11 - John Abbott

- Status changed from In Progress to Closed

- % Done changed from 60 to 100

- Estimated time changed from 16.00 h to 18.80 h