

CoCoA-5 - Support #548

Printing rings with ID

06 May 2014 12:09 - Anna Maria Bigatti

Status:	Closed	Start date:	06 May 2014
Priority:	Normal	Due date:	
Assignee:	Anna Maria Bigatti	% Done:	100%
Category:	Incomplete function	Estimated time:	7.50 hours
Target version:	CoCoA-5.1.2 summer 2015	Spent time:	7.55 hours

Description

for some rings there is a unique implementation: **RingQQ()**, **RingZZ()**, **RingQQt(n)**.
For all the others the constructor **NewRing..** creates a new copy of the ring (in some more elaborate ring constructions it would impossible to determine whether two rings are the same).

In CoCoA-5 this fact is hidden by the friendly syntax **K ::= ZZ/(3)[x,y]** which calls **NewRingFp(3)**.

I think it would be handy, when printing a ring, to print also its address, something like

```
/**/ Print (NewRingFp(3));  
FFp(3) ("1234567")  
/**/ Print (NewRingFp(3));  
FFp(3) ("1897634")
```

This would make it easier for the user to understand which rings are the same, and to highlight that is not usable as input.

Later suggestion: in CoCoALib rings have an ID, so could be printed as

```
/**/ Print NewRingFp(3);  
FFp(3) [ID=3]  
/**/ Print NewRingFp(3);  
FFp(3) [ID=4]
```

Latest suggestion:

```
# S ::= ZZ/(3) [x,y,z]; S;  
RingWithID(4, "RingID(3) [x,y,z]")  
# CoeffRing(S);  
RingID(3, "FFp(3)")
```

Related issues:		
Related to CoCoALib - Feature #420: Allow user to give a name to a ring	New	07 Jan 2014
Related to CoCoA-5 - Feature #274: InputForm for output readable as input	In Progress	20 Jun 2012
Related to CoCoA-5 - Feature #18: Printing matrices: I/O unified style for Co...	Closed	02 Nov 2011
Related to CoCoA-5 - Design #483: Unique copies of rings in CoCoA-5	New	19 Mar 2014
Related to CoCoA-5 - Feature #751: description of a RING	Closed	29 Jul 2015

History

#1 - 06 May 2014 12:28 - John Abbott

- Category set to Incomplete function

Each ring in CoCoA has an ID number (they start at 1 and are assigned incrementally). It'd be better to print those than the internal addresses!

#2 - 06 May 2014 14:08 - Anna Maria Bigatti

- Status changed from New to In Progress

- % Done changed from 0 to 10

I tried to see how it looks (just in RingFp.C).

This would be the output for test-RingHom1 not bad, I think

```
Composite homomorphism is RingHom(ZZ --> FFp(7) [ID=2])
```

```
Using composite hom: 1 maps to 1 in FFp(7) [ID=2]
```

```
Using composite hom: 2 maps to 2 in FFp(7) [ID=2]
```

```
Using composite hom: 3 maps to 3 in FFp(7) [ID=2]
```

#3 - 06 May 2014 14:19 - Anna Maria Bigatti

And in CoCoA-5 would look like this:

```
/**/ Print NewRingFp(3);  
FFp(3) [ID=3]  
/**/ Print NewRingFp(3);  
FFp(3) [ID=4]  
/**/ Print NewRingFp(3);  
FFp(3) [ID=5]
```

In case we do this change: how to make it elegantly so that it prints the ID for all rings except ZZ and QQ?

20140625 That should be easy because there are CoCoALib fns IsZZ and IsQQ

#4 - 06 May 2014 14:32 - John Abbott

Should this topic be in CoCoALib rather than CoCoA-5?

From the point of a user it would be very handy if the C5 expression ZZ/(3) always produced the same ring (and similarly for NewZZmod(3) in CoCoALib). This can be achieved using a global table of rings, though there is some risk that it might fill up with "junk"...

#5 - 06 May 2014 14:39 - John Abbott

Here is an idea: simple to implement, but perhaps not so nice to use?

- a ring prints out simply as ring(1), ring(2) etc
- describe RingOf(M) produces a full description of the ring (similar to what is currently printed)

This would make it very clear when two rings are the same, but forces use of a "special" command to find out what the ring really is. How often does one really want to print out all the information about the structure of a ring?

Do we want printing of rings to be valid as input? (e.g. in the printed form of a matrix)

#6 - 06 May 2014 15:01 - Anna Maria Bigatti

John Abbott wrote:

Here is an idea: simple to implement, but perhaps not so nice to use?

- a ring prints out simply as ring(1), ring(2) etc
- describe RingOf(M) produces a full description of the ring (similar to what is currently printed)

I like it, it could be **RingWithID(6)**. Maybe that could also be used as input?

#7 - 24 Jun 2014 16:51 - Anna Maria Bigatti

- Subject changed from *Printing rings* to *Printing rings with ID*

#8 - 25 Jun 2014 13:09 - John Abbott

- % Done changed from 10 to 20

If we go with the idea of printing simply RingWithID(...) then I presume the describe command should fully describe the ring, rather than just the topmost level:

```
P := QQ[x,y,z];
Print P;
RingWithID(4);
describe P;
???
```

In this case the nicest output would be QQ[x,y,z].

Suppose the session continues like this:

```
Use P;
I := ideal(x^100+lots-of-terms);
PmodI := P/I;
P2 := PmodI[a,b,c];
describe P2;
???
```

Here we probably do not want one huge line of the form QQ[x,y,z]/(ideal(x^100+lots-of-terms))[a,b,c]

It would be more comprehensible to have an output of the form:

```
RingWithID(9): RingWithID(8)[a,b,c]
RingWithID(8): RingWithID(4)/(ideal(x^100+lots-of-terms))
RingWithID(4): RingWithID(2)[x,y,z]
RingWithID(2): QQ
```

What do you think?

#9 - 25 Jun 2014 13:14 - John Abbott

The name RingWithID is longer than I would like; I originally suggested ring(3) but that is not so clear. Maybe RingID(3).

We could even regard RingID as an indexable value rather than a function, so the user would write RingID[3]. Not sure if this is a good idea... maybe it makes no difference?

15:07 I think it'd be easier for the user if RingID is a function; to me it would feel like an "exceptional case" if it were like a (read-only) list...

#10 - 26 Jun 2014 15:14 - John Abbott

How will rings be printed in resolutions?

I recall that CoCoA used a syntax like ring#7; we could do this (but it would be a new syntax, and thus require changes to lexer/parser).

#11 - 02 Jul 2014 17:19 - John Abbott

Talked about this with Christof. Here's an idea we had:

allow RingID to have an optional second arg which is a string giving some info about the ring (this 2nd arg could be ignored when reading RingID(...)).

For instance we could get behaviour like this...

```
P := QQ[a,b];
P2 := P[x,y];
PrintLn P2;
RingID(7, "RingID(6)[x,y]");
```

#12 - 02 Jul 2014 17:21 - Anna Maria Bigatti

John Abbott wrote:

allow RingID to have an optional second arg which is a string giving some info about the ring (this 2nd arg could be ignored when reading RingID(...)).

YES!! I'll try it immediately!

[Later]

```
# S := ZZ/(3)[x,y,z];
# S;
RingID(4, "RingID(3)[x, y, z]")
# CoeffRing(S);
RingID(3, "FFp(3) ")
```

#13 - 02 Jul 2014 18:42 - Anna Maria Bigatti

- % Done changed from 20 to 40

mostly done.... now I should update all the failing tests.... tomorrow

#14 - 03 Jul 2014 08:40 - Anna Maria Bigatti

fixed all tests, but we should still discuss on the output.
For example

```
RingID(2, "RingID(0) [x, y] ")
```

would be more readable as

```
RingID(2, "ZZ[x, y] ")
```

so:

- 1 - write "by hand" the printing of the indets without spaces
- 2 - write ring "name" for rings with unique copies (there is something similar for printing matrices)

#15 - 03 Jul 2014 08:56 - John Abbott

Yesterday Christof suggested that ZZ and QQ should always be printed that way (rather than ring(0) and ring(1)). We were undecided about rings like ZZ/(3); one idea was to print it nicely if the nice form is sufficiently short (e.g. less than 10 chars).

#16 - 03 Jul 2014 09:05 - Anna Maria Bigatti

John Abbott wrote:

Yesterday Christof suggested that ZZ and QQ should always be printed that way (rather than ring(0) and ring(1)). We were undecided about rings like ZZ/(3); one idea was to print it nicely if the nice form is sufficiently short (e.g. less than 10 chars).

OK, I'll make some experiments.
Meanwhile, we should also think of implementing RingID(N).
I think there is a table in GlobalManager. (wild guess)

#17 - 03 Jul 2014 11:29 - Anna Maria Bigatti

what about this: (verbose, but not more than before the ID change)
I like it! but there is the problem of too many quotes :-{

```
# NewFractionField(NewPolyRing(QQ,SymbolRange("t",3,5)));  
RingID(4,"FractionField(RingID(3,"QQ[t[3],t[4],t[5]]")"))
```

20140703-13:38 Your example seems too verbose to me; I think I would expect just `RingID(4,"FrF(RingID(3))")`. Hmm, this is tricky. I wonder whether this might be acceptable `RingID(4,"FrF(RingID(3),"QQ[3indets]"))`

#18 - 03 Jul 2014 11:41 - Anna Maria Bigatti

added function ID for RING

#19 - 03 Jul 2014 13:28 - John Abbott

I'm still not happy with the name of `RingID`; the name makes me think that it **gets** the ID from a ring, rather than gets the ring from its ID. `RingWithID` is clearer but is too long...
Puzzled.

#20 - 04 Jul 2014 10:50 - Anna Maria Bigatti

John Abbott wrote:

I'm still not happy with the name of `RingID`; the name makes me think that it **gets** the ID from a ring, rather than gets the ring from its ID. `RingWithID` is clearer but is too long...

In fact, it is not too long (and it's very expressive), because at the end it prints a shorter string than the "old-style" printing.
I think we need an extra function (I was trying to avoid that) saying whether the ring has a unique implementation (therefore it's better printed explicitly) or not (therefore printed with ID)

I had a look at `GlobalManager` and now **I think** that currently there is no way (yet) to see if `"RingWithID(5)"` exists and get its value.

#21 - 04 Jul 2014 11:39 - John Abbott

Should this issue be in `CoCoA-5` or `CoCoALib`?

You are right that there is no global register of rings currently. We would need one if we want to avoid creating two identical rings. As you know, I do not much like globals, though they are sometimes necessary. Some care will be needed to make a global work properly in a multithreaded environment.

Should **all** rings appear in the global registry? Even temporary rings which exist only inside a fn?

Am I correct in deducing that you want a fn which given a ring ID produces the corresponding ring?

#22 - 04 Jul 2014 12:12 - Anna Maria Bigatti

John Abbott wrote:

Should this issue be in CoCoA-5 or CoCoALib?

I think it can stay in CoCoA-5 because it is easier re-create rings.

You are right that there is no global register of rings currently. We would need one if we want to avoid creating two identical rings. As you know, I do not much like globals, though they are sometimes necessary. Some care will be needed to make a global work properly in a multithreaded environment.

I think that trying to catch identical constructions might be tricky, and if it works 90% of the cases it might be even more confusing. I think that printing with ID helps the user understand better what a ring is and so avoid to make unnecessary (and costly, if we need to do all checks every time) re-creation of rings

Should **all** rings appear in the global registry? Even temporary rings which exist only inside a fn?

hmhhh, now I'm not sure we should do this.

A ring occupies lots of memory and we'd better get rid of the unused ones (everytime we compute a Groebner basis we create a temporary ring!)

Maybe the function "RingWithID(n)" should return an error saying something like:

"this is not a function: give names to rings or use RingOf(object)" ;-)

(I like that! it's an easy way to tell the user how to be more careful)

If we allow the user to write RingWithID(7), that will have a totally different meaning next time cocoa is started. So it fails the utility of being "re-usable" and saved output cannot be read again anyway.

Am I correct in deducing that you want a fn which given a ring ID produces the corresponding ring?

more than "want" was a "wondering". Now I'm getting convinced it should not be implemented.

#23 - 02 Sep 2014 11:38 - John Abbott

- % Done changed from 40 to 50

#24 - 03 Sep 2014 13:35 - John Abbott

- Target version changed from CoCoA-5.1.1 Seoul14 to CoCoA-5.1.2 summer 2015

I think the current implementation is adequate (rather than fully satisfactory); but I'd like time to think more about the issue before declaring it fully resolved. So, I'm changing the target version to 5.1.2.

#25 - 23 Oct 2014 11:08 - Anna Maria Bigatti

I have implemented in CoCoALib

```
void ***RingBase::myOutputSelfLong(std::ostream& out) const
```

I think it should be called by describe in CoCoA-5, which currently returns only

A value of type RING

Have I just forgotten do it, or did we decide not to? (and why?)
... maybe I just didn't find how to access the interpreter code for describe...

#26 - 23 Oct 2014 12:21 - John Abbott

I think it has just been forgotten; it seems like a good idea to me. Indeed now that direct printing produces a very compact(incomplete) form, CoCoA-5 really needs a way to get the full details of a ring.

I note that describe could produce several lines of output if we want it to. For instance, the description of a ring could be like this: (copied from comment 8)

```
RingWithID(9): RingWithID(8)[a,b,c]
RingWithID(8): RingWithID(4)/(ideal(x^100+lots-of-terms))
RingWithID(4): RingWithID(2)[x,y,z]
RingWithID(2): QQ
```


#27 - 20 May 2015 13:10 - Anna Maria Bigatti

- *Status changed from In Progress to Resolved*
- *Assignee set to Anna Maria Bigatti*

John Abbott wrote:

I think it has just been forgotten; it seems like a good idea to me. Indeed now that direct printing produces a very compact(incomplete) form, CoCoA-5 really needs a way to get the full details of a ring.

I note that describe could produce several lines of output if we want it to. For instance, the description of a ring could be like this: (copied from comment 8)
[...]

I'm thinking how to achieve this. We haven't yet designed a describe mechanism for cocoalib.
The easiest (without messing too much with the Interpreter) is to write a description function in cocoalib returning a string (i.e. which does not print). Then the cocoa-5 describe would just print that string.
Should I do it like that? (then I could also do it for RINGHOM)

#28 - 29 Jul 2015 12:28 - Anna Maria Bigatti

- *Description updated*

Anna Maria Bigatti wrote:

Maybe the function "RingWithID(n)" should return an error saying something like:
"this is not a function: give names to rings or use RingOf(object)" ;-)
(I like that! it's an easy way to tell the user how to be more careful)

done

#29 - 29 Jul 2015 12:43 - Anna Maria Bigatti

- *Status changed from Resolved to Closed*
- *% Done changed from 50 to 100*
- *Estimated time set to 7.50 h*

All the relevant code is done. I find it quite satisfactory (even though I'm not entirely convinced with the design of the functions myOutputSelfShort and myOutputSelfLong)

The function `R->myOutputSelfLong(out)` gives a more complete description of the ring, and should be the code for describe R in CoCoA-5, so I move the discussion on details and refinements to a dedicated issue.