CoCoALib - Feature #1559

Default ctor for rings

09 Jan 2021 18:20 - John Abbott

Status:	In Progress	**Start date:**	09 Jan 2021
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	10%
Category:	Improving	**Estimated time:**	0.00 hour
Target version:	CoCoALib-0.99880	**Spent time:**	0.75 hour

Description

Winfried Bruns sent the following by email:

"I wanted to define a class that contains two sparesepoly rings as members. The base rings could be fixed, but not the number of indeterminates.
It is my impression that this is impossible due to the lack of a default constructor.
Could you make one? The lack of it really restricts the flexibility of programming."

Discuss; if appropriate, design & implement.

History

#1 - 15 Jan 2021 10:02 - John Abbott

The class **ring** does have a default ctor (which assigns RingZZ() as initial value).
In contrast, SparsePolyRing does not have a default ctor.

If I recall correctly, all rings are implemented as ref-counting pointers to an implementation class; SparsePolyRing simply checks that the impl class is an instance of a class derived from SparsePolyRingBase. When I write programs, I almost always use **ring** rather than SparsePolyRing; may I use SparsePolyRing only for function args (so that the compiler automatically checks that the supplied arg has the right properties).

So the question is: can Bruns's code use ring instead of SparsePolyRing?

#2 - 07 Mar 2024 20:35 - John Abbott

- Target version changed from CoCoALib-0.99850 to CoCoALib-0.99880

It is not completely clear to me what is desired. Winfried, can you clarify? Maybe give an example?
Were you able to solve your problem some other way?

#3 - 13 Mar 2024 16:49 - Anna Maria Bigatti

- Description updated

By Winfried Bruns:

I have forgotten the context, but the following is a scenario that has really come up. Normaliz allows polynomial constraints for lattice points. These polynomials must be stored in the class Cone. I am using a Normaliz specific polynomial format for this purpose. In fact, the (almost) only computation with these polynomials is evaluation on vectors. But there could be really more, and then CoCoALib could be the right choice.
But how can I store RingElem in the cone? They must belong to a polynomial ring, and this would require such a ring to be a member of Cone. How can we do all this?

#4 - 13 Mar 2024 16:58 - Anna Maria Bigatti

- % Done changed from 0 to 10

Anna Maria Bigatti wrote:

> By Winfried Bruns:
> But how can I store RingElem in the cone? They must belong to a polynomial ring, and this would require such a ring to be a member of Cone.
> How can we do all this?

I expect you want rational coefficients.
In this case you could create the polynomials in the (uniquely defined) ring QQ[t[1],...,t[n]] with RingQQt(long n).
But maybe you don't even need those polynomials to be in the same ring?
In this case just make the RingElem field which will be created by default in ZZ, then you may redefine it in any ring.

Have I answered your question?

#5 - 13 Mar 2024 20:42 - John Abbott

- Status changed from New to In Progress

I think Anna may be proposing to store in the cone object a RingElem rather than a ring; you can always obtain the ring of a RingElem using the function owner. A good default value for the RingElem could be zero in the ring you want.

Maybe the three of us can chat via Skype (or other similar system) to understand and resolve the problem?