CoCoALib - Design #254

How to return a Hilbert Series in CoCoALib

08 Oct 2012 16:00 - Anna Maria Bigatti

Status:	Closed	**Start date:**	08 Oct 2012
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	New Function	**Estimated time:**	6.00 hours
Target version:	CoCoALib-0.99533 Easter14	**Spent time:**	5.10 hours

Description

Right now Hilbert series do not exist in CoCoALib.
In CoCoA-5 they are built in the package hp.cpkg5.

Now we need to clean it up at fix a common interface for the Hilbert Series in TmpHilbert and in ExternalLib-Normaliz.

We should make something like the **factorization** class and make a related **from** function for sending this class to CoCoA-5.

Related issues:

Related to CoCoALib - Feature #203: Function to get the Hilbert Polynomial ri...	**Closed**	**04 Jul 2012**
Related to CoCoA-5 - Feature #93: Hilbert series: global output rings for non...	**Closed**	**23 Feb 2012**
Related to CoCoA-5 - Design #376: Hilbert: Redesign hp.cpkg5 and public funct...	**In Progress**	**18 Jun 2013**
Related to CoCoALib - Feature #390: Store unique copy of QQ[t_1..t_n] (RingQQ...	**Closed**	**23 Jul 2013**
Related to CoCoALib - Feature #399: add myHilbertSeries member field to ideal...	**New**	**02 Aug 2013**
Related to CoCoA - Support #425: Osnabrueck 2014-01	**Closed**	**27 Jan 2014**
Related to CoCoA-5 - Design #493: Cleanup Hilbert package hp.cpkg5	**Closed**	**24 Mar 2014**

History

#1 - 12 Oct 2012 19:27 - Anna Maria Bigatti

- Tracker changed from Feature to Design

- Category set to New Function

- Target version set to CoCoALib-0.9953

#2 - 19 Mar 2013 10:35 - Christof Soeger

Any news on it?

In libnormaliz it is a vector for the coefficients of the numerator and a map for the exponents of (1-t^i)^e. However I do not suggest this as type in CoCoALib.

I think the type should offer the possibility to easily

- access the data, for example to check if the h-vector is palindromic,
- do arithmetic operations, like adding two series (as rational functions).

In connection to this we also need a type for Hilbert (quasi-)polynomials

I suggest to have a unique ring where all these polynomials could live in. For me a univariate polynomial ring over QQ is enough, but other applications could profit from such a multivariate ring.

Do you have any support for quasi-polynomials?

#3 - 20 Mar 2013 23:17 - John Abbott

- Status changed from New to In Progress

Here is a suggestion: a RECORD with two fields numer and denom where

- numer is a polynomial,
- and denom is a RECORD with the same structure as would be returned by a call to factor (*i.e.* fields factors, multiplicities and RemainingFactor).

There is also the matter *to which ring do these polynomials belong?*

#4 - 17 Jun 2013 11:59 - Anna Maria Bigatti

- % Done changed from 0 to 20

Design:
new class **HilbertSeries**: member fields
myNum RingElem and **myDenFactors** factorization.

Constructors:
(RingElem num, factorization<RingElem> DenFactors) -- general
(vector<BigInt> DenseRepr, vector<long> DenExponents) -- for univariate HS
e.g. DenExps = [1,3,2,2,3] --> (1-t) * (1-t^3)^2 * (1-t^2)^2

Guarantees:
HilbertSeriesPolyRing(long (or MachineInt?) n): ring for all num and den factors.
No guarantees on num or den (irreducibility, order of factors, ...)

Operations:
constructors (copy ctor)
assignment
operator<<
operator== (H1==H2 iff they are the same in FrFld(P))
operator!=
num(HS)   (same as internally stored: no guarantees on simplification)
DenFactors(HS)
wish list: convertion into rational function (needs global FractionField)

#5 - 17 Jun 2013 16:08 - Christof Soeger

I spoke with W.Bruns about a name for the ring. One suggestion is PolynomialInvarianceRing (or a shortened form) instead of HilbertSeriesPolyRing. We think it is a better description and allows a more flexible use.
In fact even in the Hilbert series and the Hilbert polynomial the meanings of the variables are different. For the Hilbert polynomial the variable is a degree in the monoid, algebra, ... and for the Hilbert series it is a auxiliary variable to form the series. Comparison between these to objects as elements in one polynomial ring is not useful.
But we don't think there should be two different rings.

#6 - 18 Jun 2013 08:54 - Anna Maria Bigatti

Christof Soeger wrote:

> I spoke with W.Bruns about a name for the ring. One suggestion is PolynomialInvarianceRing (or a shortened form) instead of HilbertSeriesPolyRing. We think it is a better description and allows a more flexible use.

... what about **RingQQt(n)**? That is pretty self-explanatory and flexible ;-)

#7 - 18 Jun 2013 14:15 - Christof Soeger

Anna Maria Bigatti wrote:

> ... what about **RingQQt(n)**? That is pretty self-explanatory and flexible ;-)

Yes that is flexible :) A good possibility.

In CoCoA 5 we will also have this ring(s)?

#8 - 18 Jun 2013 18:40 - John Abbott

- % Done changed from 20 to 50

New name implemented & checked-in.

#9 - 18 Jun 2013 19:12 - Anna Maria Bigatti

- Assignee set to John Abbott

- % Done changed from 50 to 80

Christof Soeger wrote:

> Anna Maria Bigatti wrote:
>
> > ... what about **RingQQt(n)**? That is pretty self-explanatory and flexible ;-)
>
> In CoCoA 5 we will also have this ring(s)?

Yes, checked in

#10 - 23 Jul 2013 18:16 - Anna Maria Bigatti

- Subject changed from How to return a Hilbert Series to How to return a Hilbert Series in CoCoALib

#11 - 29 Oct 2013 13:39 - Anna Maria Bigatti

- Target version changed from CoCoALib-0.9953 to CoCoALib-0.99532

#12 - 28 Jan 2014 18:04 - Christof Soeger

- % Done changed from 80 to 90

Anna Maria Bigatti wrote:

> Constructors:
> (RingElem num, factorization<RingElem> DenFactors) -- general
> (vector<BigInt> DenseRepr, vector<long> DenExponents) -- for univariate HS
> e.g. DenExps = [1,3,2,2,3] --> (1-t) * (1-t^3)^2 * (1-t^2)^2

I implemented the constructor for univariate HS. It is used in the new function Normaliz::HilbertSeries(cone c).

#13 - 29 Jan 2014 19:50 - Anna Maria Bigatti

- Status changed from In Progress to Resolved

Now all essentially set.
One thing to decide: factorization of denom is now arbitrary:
Normaliz uses a compact form (i.e. (1-t)^2*(1-t^3)^4)
CoCoA uses the long form with all multiplicities == 1.
Should we set for a specific form?
Should we offer conversion from compact to/from long form?

Which functions should be available on class **HPSeries**?
Surely a **simplify**, to get HVector and dimension (with standard grading)

#14 - 01 Apr 2014 17:28 - Anna Maria Bigatti

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

#15 - 17 Apr 2014 00:50 - John Abbott

I'm not an expert in the field, and in my ignorance find the "compact" form more satisfactory. I could even envisage a mem fn for factorization which makes the factors compact (*i.e.* gathers together equal factors and sums their multiplicities).

It could be nice if the "compact" form had the factors in increasing order of degree (first 1-t then 1-t^2 and so on).

#16 - 17 Apr 2014 09:24 - Anna Maria Bigatti

- Estimated time set to 6.00 h

#17 - 17 Apr 2014 15:19 - Anna Maria Bigatti

- Status changed from Resolved to Closed

- % Done changed from 90 to 100

(for now) we have decided that factorization of denom is arbitrary.
There will be functions to clean up a factorization.