CoCoALib - Feature #678

Accumulator (like a geobucket)

09 Apr 2015 21:30 - John Abbott

Status:	In Progress	**Start date:**	09 Apr 2015
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	20%
Category:	New Function	**Estimated time:**	0.00 hour
Target version:	CoCoALib-1.0	**Spent time:**	2.35 hours

Description

Consider creating a general "accumulator" for ring elems, a sort of generalization of a geobucket.

Related issues:

Related to CoCoALib - Support #1664: geobucket: documentation	**New**	**14 Feb 2022**

History

#1 - 09 Apr 2015 21:36 - John Abbott

I have just written DetDirect which computes the sum of many terms (one for each permutation). In order to get decent execution speed (for a matrix whose entries are distinct indets) I had to use geobucket to accumulate the answer, but this means that the code works only if the entries are in a polynomial ring.

If there were an accumulator for all types of RingElem then the code would work with all rings.

An alternative would be write two versions of the code: one is valid only if the ring is a poly ring and uses geobucket, the other is valid for all rings and uses +=.

#2 - 09 Apr 2015 21:46 - John Abbott

Note: perhaps also consider the design of C++ std::accumulate? Is there anything in BOOST?

My (vague) idea is that an accumulator could contain an "unnormalized" sum until the final value is requested (which will force normalization if necessary).

Operations:

- ctor takes a ring (initial value of sum is 0)
- add this value to the sum
- get final/current value (and reset sum to 0?)

Example: an accumulator for elements of a small prime finite field could use machine integer summation and avoid many reductions mod p.

#3 - 09 May 2015 15:48 - John Abbott

The implementation will almost surely be best done via mem fns since the actual inner working of an "accumulator" will depend on details of the specific ring.

Default implementation could simply use +=.

Special implementations include:

- geobuckets for SparsePolyRing
- unnormalized sum of machine integers for SmallFp
- (?maybe?) unnormalized sum for a general quotient ring
- something clever for field of fractions??? (geobucket based on size of denom?)

#4 - 17 Nov 2016 14:12 - John Abbott

Just adding some **keywords** to this issue (so it is easier to find via redmine search).
Summation, **summer**, **summing**.

#5 - 17 Nov 2016 14:26 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

Here are some possible applications:

- converting polynomials to/from CoCoALib (*e.g.* other library may keep terms in a different order)
- better implementation of polynomial ringhoms?

#6 - 17 Nov 2016 19:03 - John Abbott

- % Done changed from 10 to 20

Here is a possible design:

- user deals with a smart-pointer class (*e.g.* like ideal)
- internally there is an abstract base class
- there is a "default" concrete class (internally has a RingElem and uses operator+=)
- there are a few special concrete classes for where we can sum faster (*e.g.* poly ring via geobucket, small prime finite field via a machine long)
- the special concrete classes may need to be subclasses of the corresponding ring (if access to internal representations is needed).

The ABC interface can be simple (see also comment 2):

- (pseudo)ctor needs the ring as arg
- AddThisValue accepts a RingElem (which must belong to the ring of the summation object)
- ExportSum may as well reset the summation object to zero

Probably we should allow a summation object to continue working after ExportSum, though there is probably little gain compared to destroying it and creating a new one.

Question: how to decide when a special impl is appropriate? Pseudoctor is a mem fn of the ring (with default defn which creates a default summation object?)

#7 - 14 Feb 2022 11:02 - John Abbott

- Related to Support #1664: geobucket: documentation added