CoCoALib - Feature #407

RingElem ctor from mpz_t (from Bruns)

12 Oct 2013 19:58 - John Abbott

Status:	Closed	**Start date:**	12 Oct 2013
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	New Function	**Estimated time:**	3.00 hours
Target version:	CoCoALib-0.99532	**Spent time:**	3.00 hours

Description

Bruns would like a ctor for RingElem from an mpz_t (or perhaps even mpz_class), rather than having to create an intermediate BigInt.

Presumably also mpq_t and mpq_class.

Related issues:

Related to CoCoALib - Feature #253: W.Bruns's wish list	**Closed**	**04 Oct 2012**
Related to CoCoA - Support #425: Osnabrueck 2014-01	**Closed**	**27 Jan 2014**
Related to CoCoALib - Design #1180: BigRat(0) unexpectedly compiles! (calls c...	**Closed**	**18 Apr 2018**

History

#1 - 14 Oct 2013 12:05 - John Abbott

- Category set to New Function

- Status changed from New to In Progress

- Assignee set to John Abbott

Pros and cons

Pros:

- the proposed ctors would make it easier to use CoCoALib features from software which already uses GMP values
- might avoid making some wasteful temporaries (at least in some cases)

Cons:

- (mild) the new ctors would make CoCoALib more explicitly dependent on GMP
- references to the GMP C++ classes would require that GMPXX is installed (rather than just GMP)

NOTE 1: JAA currently thinks that it would be better to avoid depending on GMPXX
NOTE 2: there is no CPP flag saying whether the GMPXX interface is present (in any case it would entail including gmpxx.h rather than gmp.h)

#2 - 14 Oct 2013 12:09 - John Abbott

- Subject changed from RingElem ctor from mpz_t to RingElem ctor from mpz_t (from Bruns)

#3 - 18 Oct 2013 16:37 - John Abbott

Should there also be assignment of RingElem values from mpz_t and mpq_t?

NOTE: a simple (but inefficient) impl could be put in ring.C which simply creates an internal temporary BigInt (or BigRat) and then calls an existing fn. This is quick to impl, and would limit extending direct dependence on GMP to just ring.C (rather than to all ring impls if we require a member fn myNew taking an mpz_t).

#4 - 18 Oct 2013 20:38 - John Abbott

- % Done changed from 0 to 20

I have an initial implementation. CoCoALib compiles, but there are *compilation* problems in the tests. The problem derives from the fact that the compiler reports **RingElem(R,0)** as **ambiguous** because the 0 literal can also be interpreted as a pointer, and mpz_t is effectively a pointer. So the following two signatures match equally well:
RingElem::RingElem(const MachineInt& n)
RingElem::RingElem(const mpz_t n)

Of course, one can always write zero(R) instead of RingElem(R,0).

Not sure what to do. I suppose the added convenience of being able to build RingElem values straight from mpz_t outweighs the (slight?) inconvenience of having to write zero(R) rather than RingElem(R,0)

Comments? Opinions?

#5 - 29 Oct 2013 13:09 - Anna Maria Bigatti

- Target version set to CoCoALib-0.99532

#6 - 21 Nov 2013 15:43 - John Abbott

- % Done changed from 20 to 30

We could define a ctor which needs an extra "marker" parameter to resolve the ambiguity in the call BigInt(R,0) between the ctor which expects a MachineInt and that which expects a mpz_t.

Ideally the marker should be short, comprehensible and fairly visible. Candidates are: **FromGMP**
Note: if we want, the same marker can be used for mpz_t and mpq_t.

Other candidates? Comments?

#7 - 29 Jan 2014 13:55 - John Abbott

- Status changed from In Progress to Feedback

- % Done changed from 30 to 90

After discussing with Anna and Christof...

- We **reject the idea of using a "marker"** because it is cumbersome, also the ambiguity it would resolve is not so serious because it may easily be worked around. There is even some hope that a future version of C++ might make a proper distinction between 0 and nullptr (which would quash the ambiguity).
- We **reject the idea of having assignment to RingElem from mpz_t** because it would make assignment from 0 ambiguous (**very inconvenient**), and we don't want to make mpz_t a too widely recognized data type.

JAA: checked in everything (incl doc)

#8 - 30 Jan 2014 16:30 - Christof Soeger

To summarize, there are now the following constructors:

```
RingElem r(R, MPZ);      an element of R, initially the value of mpz_t MPZ
RingElem r(R, MPQ);      an element of R, initially the value of mpq_t MPQ (or error)
```

So for an mpz_class object MPZ it then has to be:

```
RingElem r(R, MPZ.get_mpz_t());
```

I think that is good.

#9 - 07 Mar 2014 18:05 - John Abbott

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

The code has remained unchanged for 4 months, and there have been no problems.

Everyone is happy with the design decisions registered here a month ago, so I'm closing this issue.

#10 - 03 Apr 2014 11:53 - Anna Maria Bigatti

- Estimated time set to 3.00 h

#11 - 18 Apr 2018 14:10 - John Abbott

- Related to Design #1180: BigRat(0) unexpectedly compiles! (calls ctor with mpq_t arg) added