CoCoALib - Design #1180

BigRat(0) unexpectedly compiles! (calls ctor with mpq_t arg)

18 Apr 2018 14:04 - John Abbott

Status:	Closed	**Start date:**	18 Apr 2018
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Safety	**Estimated time:**	7.70 hours
Target version:	CoCoALib-0.99600	**Spent time:**	7.35 hours

Description

The following code excerpt compiles, but does not do what I expected:

```
cout << BigRat(2/3) << endl;
```

Q: What does it do when run?
A: SEGV!?!

Of course, the input should have been BigRat(2,3) but I made a mistake (*errare humanum est*).
I thought the design of CoCoALib aimed to avoid nasty surprises.

Investigate!

Related issues:

Related to CoCoALib - Feature #407: RingElem ctor from mpz_t (from Bruns)	**Closed**	**12 Oct 2013**
Related to CoCoALib - Design #1563: BigRat: ctor from machine int	**Closed**	**15 Jan 2021**

History

#1 - 18 Apr 2018 14:10 - John Abbott

The problem turned out to be that **2/3** is evaluated by the compiler to produce 0 (integer division); this is then treated as the literal 0 which can be viewed as a null-pointer of any type... so this matches the ctor which expects const mpz_t. That ctor then triggers SEGV when executed with a null-pointer as input.

Note 1: if the fraction does not evaluate to 0 then compilation fails *e.g.* **BigRat(3/2)**

Note 2: it would be easy to modify the ctor so that it throws a CoCoA_ERROR when passed a null-pointer; this would help the user debug... but it does not prevent compilation of faulty code.

The real problem is that the compiler allows 2/3 to be interpreted as a null-pointer...

#2 - 18 Apr 2018 14:10 - John Abbott

- Related to Feature #407: RingElem ctor from mpz_t (from Bruns) added

#3 - 18 Apr 2018 14:34 - John Abbott

- Status changed from New to In Progress

- Assignee set to John Abbott

- % Done changed from 0 to 10

JAA thinks there is little hope that the language rules for C++ will change (in the forseeable future) to forbid an "integer literal with value 0" being open to interpretation as a null pointer... that would break a lot of code!

I see two options:
(1) accept the "unexpected interpretation" of **BigRat(0)**, and possibly modify CoCoALib to handle it as helpfully as possible
(2) change the args for the ctor **BigRat(const mpq_t)** so that BigRat(0) or anything equivalent (such as BigRat(2/3)) does not compile; for instance we could add an obligatory second arg (*i.e.* a "marker" such as **CopyFromMPQ**)

#4 - 18 Apr 2018 20:24 - John Abbott

The situation is more complicated than I'd like. Here I'll write about BigInt, but it applies just as much to BigRat.

If I remove the ctor BigInt(mpz_t) then BigInt(0) becomes ambiguous!!
It matches equally well, BigInt(const std::string&) and BigInt(const MachineInt&).

As far as I'm concerned the real problem is that a std::string can be constructed from const char*, which is perfectly reasonable **except** that it accepts 0 as a pointer...

I see two possibilities:
(A) [dodgy hack] write a ctor which accepts a pointer to come weird type -- this will then match BigInt(0)
(B) renounce using MachineInt or convert it to a typedef for signed long or signed long long

The reason for having MachineInt was avoid nasty surprises with large unsigned long values which are silently mangled to fit into a signed long. So renouncing MachineInt opens the door to (silent) "nasty surprises" for those who use unsigned long... note that several C++ STL functions return unsigned values!

The dodgy hack would need to be repeated in every case where ambiguity arises, and it would not solve th ambiguity problem if there another fn signature which accepts a point to a "real type".

What to do???

#5 - 19 Apr 2018 11:59 - John Abbott

- % Done changed from 10 to 20

I have replaced the ctor BigInt(mpz_t) by one which requires a second arg (CopyFromMPZ), and then changed all code so that it works with the new impl. Here are some observations:

(1) The ctor for a BigInt from an mpz_t is called quite often inside a non-trivial expression, having to specify a second arg definitely decreases readability.
(2) I have replaced calls to BigInt(0) by calls to BigInt(/*0*/) or else removed the arg altogether.

To restore readability in case **(1)** we could introduce a pseudo-ctor which simply calls the 2-arg ctor. If we do this, what should the pseudo-ctor be called? **MPZ2BigInt**??

It is definitely a nuisance having to treat BigInt(0) in a special way compared to BigInt(1) say. This could be worked around by using the "dodgy hack" mentioned at point **(A)** in comment 4 above...

#6 - 19 Apr 2018 17:58 - John Abbott

Since the ambiguity of BigInt(0) derives from the fact that std::string has an implicit ctor from char*, it maybe a good idea to replace the ctor from a std::string by one which takes two args (and perhaps make it callable by a convenient pseudo-ctor).

JAA had suggested using just **ConvertTo<BigInt>("12345")** but Anna thought that was too cumbersome.
Anna suggested something like **BigIntFromString("12345")**, or maybe just **BigIntFrom("12345")**.

A similar psuedo-ctor name could be used for making a BigInt from a mpz_t.

#7 - 23 Apr 2018 11:05 - John Abbott

- Status changed from In Progress to Resolved

- % Done changed from 20 to 70

The impl has been modified:

- BigInt ctor from mpz_t and from std::string now take 2 args, and are private; they are called by pseudo-ctors **BigIntFromMPZ** and **BigIntFromString**
- BigRat ctor from mpq_t and from std::string now take 2 args, and are private; they are called by pseudo-ctors **BigRatFromMPQ** and **BigRatFromString**.
- BigRatFromString has an optional 2nd arg AlreadyReduced, to say do not remove any common factor
- new ctor BigRat(OneOverZero_t) for creating the "infinity" rational 1/0 (used in continued fraction code)

All consequential changes made; all tests pass.
Documentation updated.

#8 - 12 Jun 2018 18:18 - John Abbott

- Status changed from Resolved to Closed

- % Done changed from 70 to 100

I have sorted out the doc.

#9 - 03 Aug 2018 17:32 - John Abbott

- Estimated time set to 7.70 h

#10 - 15 Jan 2021 15:41 - John Abbott

- Related to Design #1563: BigRat: ctor from machine int added

#11 - 15 Jan 2021 17:13 - John Abbott

IMPORTANT UPDATE(2021-01-15) this decision has now been reversed in issue #1563