CoCoALib - Feature #910

BigRat: read from a string in "decimal" format?

19 Jul 2016 14:22 - John Abbott

Status:	Closed	**Start date:**	19 Jul 2016
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Improving	**Estimated time:**	4.40 hours
Target version:	CoCoALib-0.99550 spring 2017	**Spent time:**	4.25 hours

Description

Currently operator>> for BigRat expects the input to be of the form: integer or integer "/" integer.

Should it also accept input in "decimal format"? (*e.g.* 0.12345)

Related issues:

Related to CoCoA-5 - Feature #909: ReadExpr: decimal point	**Closed**	**14 Jul 2016**

History

#1 - 19 Jul 2016 14:22 - John Abbott

- Related to Feature #909: ReadExpr: decimal point added

#2 - 19 Jul 2016 14:24 - John Abbott

The functions which need to be considered are: operator>> and ConvertTo.

JAA observes that CoCoA-5 (and older versions) have always accepted decimal input for specifying an exact rational value. So CoCoALib should probably be compatible.

#3 - 21 Jul 2016 13:27 - John Abbott

I have looked quickly at the code for operator>> for BigRat, and it probably contains a "small bug". The usual rule for operator>> is that it reads as much input as is valid then stops.

The current code fails when reading a rational number from a stream containing 1/x since the slash is read and then an error is thrown because x is not an integer literal.

UPDATE
operator>> for BigInt contained a subtle bug... now fixed.
operator>> for BigRat contained several bugs... now fixed (hopefully).

I am developing a test.

#4 - 21 Jul 2016 13:42 - John Abbott

- Status changed from New to In Progress

- Assignee set to John Abbott

- % Done changed from 0 to 10

I have clarified (and made more stringent) the syntax for rational numbers. There are two possibilities:

- <optional-sign><unsigned-integer-literal><slash><unsigned-integer-literal>
- <optional-sign><unsigned-integer-literal>

In particular, note that **no spaces** are allowed before or after the slash. This is fairly natural, but the main reason is to allow a fully portable implementation of the rule that reading consumes as many characters as possible provided they form a valid input string.

Exactly what syntax do we want for decimal fractions?

- **(A)** <optional-sign><unsigned-integer-literal><dot><unsigned-integer-literal>
- **(B)** <optional-sign><unsigned-integer-literal><dot>
- **(C)** <optional-sign><dot><unsigned-integer-literal>
- **(D)** <optional-sign><dot>

No spaces are allowed before or after <dot>.
JAA says definitely no to **(D)**, and is quite opposed to **(C)**. JAA does not much like **(B)** but accepts that most people would expect it to work.

What about an optional "exponent"? If so, what format?

- 12.345e6 and/or 12.345E6 (upper and/or lower case)
- 12.345e+6
- 12.345*10^6
- 12.345*10^(-6)
- 12345e6 (mantissa is an integer, without any <dot>)

JAA prefers not to allow exponents (at least for the first impl).

#5 - 21 Jul 2016 13:58 - Anna Maria Bigatti

John Abbott wrote:

> Exactly what syntax do we want for decimal fractions?
>
> - **(A)** <optional-sign><unsigned-integer-literal><dot><unsigned-integer-literal>
> - **(B)** <optional-sign><unsigned-integer-literal><dot>
> - **(C)** <optional-sign><dot><unsigned-integer-literal>
> - **(D)** <optional-sign><dot>

I'm for (A).
And I think that the *10^(-6) just comes as a standard multiplication.
(and no other exponents syntax)

#6 - 21 Jul 2016 15:36 - John Abbott

- % Done changed from 10 to 30

I now have a first impl (with test suite). I implemented **(A)** and **(B)**, but it is easy to restrict to just **(A)**. The code is an ugly mess :-(

It seems to work as expected. I'll clean it over the next day or two, then check in.

#7 - 21 Jul 2016 16:40 - John Abbott

- Status changed from In Progress to Feedback

- % Done changed from 30 to 90

I have cleaned the code and checked in. It's a bit ugly :-(

#8 - 05 Oct 2016 16:40 - John Abbott

I'm not sure how related this is: a twin-float value can be printed out with a trailing <dot> if the value is very close to integer but not close enough to be recognized as an integer. It might be nice to be able to read in a value printed out from a twin-float; this implies accepting a trailing <dot>, which is what the current impl does.

#9 - 08 Oct 2016 22:24 - John Abbott

Unfortunately the code for reading "decimal numbers" appears twice: once in **operator>>** for BigRat and once again in **ReadExpr**. At the moment I do not see how to easily/cleanly avoid this; a possibility would be to find a way to block reading of integer/integer in operator>> for BigRat (see #938).

#10 - 09 Nov 2016 10:50 - John Abbott

- Target version changed from CoCoALib-0.99560 to CoCoALib-0.99550 spring 2017

#11 - 18 Nov 2016 22:37 - John Abbott

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

- Estimated time set to 4.40 h