CoCoALib - Design #1182

"mod" for BigInt

04 May 2018 19:12 - Anna Maria Bigatti

Status:	Closed	**Start date:**	04 May 2018
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Tidying	**Estimated time:**	2.00 hours
Target version:	CoCoALib-0.99600	**Spent time:**	1.80 hour

Description

n % m, with n,m BigInt gives negative values.
Whether this is reasonable when working in FFp, I find it odd for BigInt.

Related issues:

Related to CoCoA-5 - Support #1368: Improve manual for mod	**Closed**	**24 Nov 2019**

History

#1 - 04 May 2018 19:21 - Anna Maria Bigatti

Wait! I need to make a check....

#2 - 04 May 2018 19:29 - Anna Maria Bigatti

hmmm, I had a negative n, I didn't think of the different behaviour with negative entries.
I suppose I need to read more carefully C++ expected behaviour on long.

#3 - 05 May 2018 16:48 - John Abbott

There is documentation for **operator%** in the file IntOperations.html. There it points out the existence of two quite explicitly named functions: **LeastNNegRemainder** and **SymmRemainder**.

I think I decided to make operator/ and operator% be "symmetric about zero", so that (-a)/b == -(a/b) for non-zero b. The remainder then satisfies a very natural formula a = b*(a/b) + (a%b) for all non-zero b.

#4 - 05 May 2018 19:07 - Anna Maria Bigatti

John Abbott wrote:

> There is documentation for **operator%** in the file IntOperations.html. There it points out the existence of two quite explicitly named functions: **LeastNNegRemainder** and **SymmRemainder**.

For once I did think of reading the manual, but I checked on BigInt and didn't see it, I didn't even see the link to it.

> I think I decided to make operator/ and operator% be "symmetric about zero", so that (-a)/b == -(a/b) for non-zero b. The remainder then satisfies a very natural formula a = b*(a/b) + (a%b) for all non-zero b.

I think it should be the same semantics (if it is explicitly defined) as for C++ long.

#5 - 05 May 2018 19:11 - Anna Maria Bigatti

- Description updated

#6 - 06 May 2018 21:51 - John Abbott

A quick look on the internet suggests that a%b is uniquely defined only if a is non-negative and b is positive; otherwise the result is "implementation defined". But in every case the implementation must guarantee that a == b*(a/b) + (a%b) always.

I have used the ambiguity in the C++ standard to allow CoCoALib's operator/ and operator% to use "round-towards-zero" in all cases.

#7 - 16 May 2018 13:57 - John Abbott

- Status changed from In Progress to Resolved

- Assignee set to John Abbott

- % Done changed from 20 to 70

I have improved the documentation about operator% by saying that its sign is the same as that of the quotient, and by adding an explicit reference to LeastNNegRemainder and SymmRemainder.

Fully resolved?

#8 - 25 Jun 2018 14:37 - John Abbott

- Status changed from Resolved to Feedback

- % Done changed from 70 to 90

Anna has not complained, so moving to **Feedback**.

#9 - 03 Aug 2018 16:27 - John Abbott

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

SOLUTION: improved the documentation

Main point is that defn in C/C++ is deliberately ambiguous. JAA chose to define op% so that it is "symmetric" in the sense that (-a)%b == -(a%b) for non-zero b.

#10 - 24 Nov 2019 13:04 - John Abbott

- Related to Support #1368: Improve manual for mod added