CoCoALib - Feature #896

myIsEqual, myCmp: direct comparisons between RingElem and MachineInt, BigInt and BigRat?

25 Jun 2016 14:09 - John Abbott

Status:	In Progress	**Start date:**	25 Jun 2016
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	10%
Category:	Improving	**Estimated time:**	0.00 hour
Target version:	CoCoALib-1.0	**Spent time:**	0.65 hour

Description

Currently all comparisons between a RingElem and a MachineInt (or BigInt or BigRat) entails first converting the numerical value into a RingElem and then calling the ring's myCmp function.

Might it be worth allowing direct comparisons without necessarily creating a temporary ring-elem?

Related issues:

Related to CoCoALib - Design #859: Twin-float: comparisons and equality test	**Closed**	**28 Mar 2016**

History

#1 - 25 Jun 2016 14:11 - John Abbott

- Related to Design #859: Twin-float: comparisons and equality test added

#2 - 25 Jun 2016 14:28 - John Abbott

Here are some pros and cons:

- **(+)** avoid making temporary ring-elem values (wasteful new-delete cycle);
- **(+)** should fix issue #859 if done correctly;
- **(-)** many new member fns for (esp. ordered) rings;
- **(-)** (mild) there is overlap between the fns, so compatibility must be guaranteed (manually!).

#3 - 25 Jun 2016 14:33 - John Abbott

Here are the new mem fns which every ring would have to offer:

- myIsEqual(RingElem, MachineInt)
- myIsEqual(RingElem, BigInt)
- myIsEqual(RingElem, BigRat)
- myIsEqual(MachineInt, RingElem)
- myIsEqual(BigInt, RingElem)
- myIsEqual(BigRat, RingElem)

In fact, we could also put default impls in RingBase which simply convert the args to RingElem and then call the extant, generic myCmp for comparison of ring-elems.

Here are the new mem fns which every ORDERED ring would have to offer:

- myCmp(RingElem, MachineInt)
- myCmp(RingElem, BigInt)
- myCmp(RingElem, BigRat)
- myCmp(MachineInt, RingElem)
- myCmp(BigInt, RingElem)
- myCmp(BigRat, RingElem)

#4 - 25 Jun 2016 16:40 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

I note there could be a "mild inconsistency" in the RingTwinFloat implementation: let N be a BigInt and x a twin-float then it is possible that cmp(N,x) and cmp(RingElem(RR,N),x) might give different results (*i.e.* the second may throw while the first does not).

The difference derives from the fact that an integer N is an exact value, whereas converting it to a twin-float makes it approximate (with corresponding inner and outer intervals); if the exact integer N lies just outside the outer interval of x, while x lies inside the outer interval of the twin-float conversion of N then InsuffPrec could result. This does require x to have an *unusually narrow* outer interval (compared to a freshly created twin-float with practically the same central value).

#5 - 06 Nov 2017 13:44 - John Abbott

- Target version changed from CoCoALib-0.99560 to CoCoALib-0.99600

#6 - 25 Jun 2018 15:31 - John Abbott

- Target version changed from CoCoALib-0.99600 to CoCoALib-1.0