# CoCoALib - Design #787

## Remove refcounts from RingElem?

16 Oct 2015 14:07 - John Abbott

| | | | |
|---|---|---|---|
| **Status:** | New | **Start date:** | 16 Oct 2015 |
| **Priority:** | Normal | **Due date:** | |
| **Assignee:** | | **% Done:** | 0% |
| **Category:** | Safety | **Estimated time:** | 0.00 hour |
| **Target version:** | CoCoALib-1.0 | **Spent time:** | 0.10 hour |

**Description**

Ref counts tend to work badly in multithreaded environments (since the incr/decr operations have to be atomic, and/or lock out other threads).

Having a refcount in each RingElem implies very many incr/decr operations; this is probably undesirable in a multithreaded setting.

On the other hand, the refcounts do let us know when a ring is no longer needed, so it can be automatically destroyed.

How feasible is it to remove the refcounts from RingElem?

**Related issues:**

| | | |
|---|---|---|
| Related to CoCoALib - Feature #142: Improve threadsafety | **In Progress** | **30 Apr 2012** |
| Related to CoCoALib - Feature #482: Unique copies of rings -- smart ctor | **In Progress** | **19 Mar 2014** |

---

**History**

#### #1 - 16 Oct 2015 16:19 - Anna Maria Bigatti

So far we've been creating rings for temporary computations without worrying too much (indeed: starting every time with a clean block of memory seems to improve locality, encouraging to create new rings!)
Not clearing the memory could have a bad effect on memory consumption.  :-(

#### #2 - 19 Oct 2015 17:08 - John Abbott

The comment about memory consumption is partly related to issue #786; if the "waste" per ring is small enough, it may not matter too much.  For instance, if you have 1000 "idle" rings, and each ring has acquired just 4Kbytes then that's a total of about 4Mbytes lying idle... not ideal, but not too bad either.

A more worrying aspect is that the current implementation of MemPool never releases memory; if you do a big computation (in some) at the start, and then throw it all away, the MemPool for that ring will continue to hold onto the memory acquired for the big computation even though it is no longer needed.