CoCoALib - Bug #190

Subtle ref count bug for poly rings (via CoeffEmbeddingHom)

19 Jun 2012 16:38 - John Abbott

Status:	Closed	**Start date:**	19 Jun 2012
Priority:	High	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Safety	**Estimated time:**	7.70 hours
Target version:	CoCoALib-0.99535 Sep14	**Spent time:**	6.80 hours

Description

The function below looks perfectly reasonable, but it creates a **dangling reference**

```
RingHom CoeffHomBug(const ring& K)
{
  PolyRing P(NewPolyRing(K,1));
  return CoeffEmbeddingHom(P);
}
```

In the ctor for a PolyRing the corresponding CoeffEmbeddingHom is created and stored in a data member of the ring. The **CoeffEmbeddingHom** function simply returns a "reference" to this homomorphism (& increments the ref count of the homomorphism, but not that of the poly ring). Recall that the ref count of the poly ring is zeroed after creating all its data members. So when control leaves the fn above, the ring P is destroyed because the only external reference to it was in the local variable P; so now the codomain of the homomorphism which is returned from the fn is a dangling reference to the now dead poly ring. **!BOOM!**

Someone's going to have *fun* fixing this bug -- let me guess who the lucky blighter will be....

Related issues:

Related to CoCoA-5 - Bug #189: malloc ERROR	**Closed**	**18 Jun 2012**
Related to CoCoA-5 - Bug #382: Subtle bug with CoeffEmbeddingHom	**Closed**	**27 Jun 2013**
Related to CoCoA-5 - Feature #273: Package for Polynomial Algebra Homomorphisms	**Closed**	**12 Nov 2012**
Related to CoCoA-5 - Bug #279: Bug in Radical (actually a RingHom problem)	**Closed**	**29 Nov 2012**

History

#1 - 20 Jun 2012 10:52 - John Abbott

JAA sees two mutually exclusive ways of solving the problem:

1. fix the ref counting so that it always works correctly
2. remove ref counting from rings

If we remove ref counting then rings will not be able to automatically self-destruct at the right time -- I wonder how important this really is. Anyway, removing ref counts improves threadsafety!

JAA suspects that "fixing" the ref counting would entail many more incr/decr operations than we currently do.

#2 - 21 Jun 2012 16:17 - John Abbott

- Priority changed from Normal to High

#3 - 28 Jan 2013 08:08 - Anna Maria Bigatti

- Category set to Safety

#4 - 11 Jul 2014 14:26 - Anna Maria Bigatti

- Target version set to CoCoALib-1.0

- % Done changed from 0 to 10

Last night I had this idea, does it work?
The problem is:
RingHom 's are refcounted objects, many actually "live" inside a **ring** (i.e. their implementation..). In that case the refcount of the ring **R** related with **phi** (*e.g.* CoeffEmbeddingHom) is decreased after the creation of **phi** so that **R** can be destroyed when phi is its only object left alive. The problem is that if **phi** still lives (because a variable stores it) then using it would call a dead ring. In cocoa-5

```
/**/ R := NewPolyRing(QQ, ["x"]);
/**/ phi := CoeffEmbeddingHom(R);
/**/ R := "deleted";
/**/ phi(123);
```

This is the idea:
when the refcount of **phi** is in(de)creased, in(de)crease also the refcount of **R**.
So refcount(R) >= refcount(phi) and becomes 0 when it should.
Is this correct? or it goes in a loop at some point I can't see?

#5 - 25 Jul 2014 13:00 - John Abbott

I think the real problem is that some rings store certain RingHom values inside themselves (thus creating an indirect circular reference, the bane of reference counting systems).

RingHom values contain references to their domain and codomain (so the rings' ref counts are automatically increased).

I now think that the true solution is not to store RingHom objects inside rings. At the moment I think this is an unnecessary "optimization" that gains practically nothing in practice but causes ref count problems.

I think I can produce a new design/impl in the near future that should avoid the problem.

#6 - 27 Jul 2014 12:34 - John Abbott

- Status changed from New to Resolved

- Assignee set to John Abbott

- % Done changed from 10 to 70

I have implemented the changes suggested in my previous comment (*i.e.* rings no longer store cached copies of their "special" ringhoms). The two examples given here (in the original description, and in comment 4) both work fine. I have added them as tests.

No check-in yet because I cannot get VPN to work :-(I'll check in tomorrow.

#7 - 28 Jul 2014 21:46 - John Abbott

- % Done changed from 70 to 80

- Estimated time set to 7.70 h

Anna confirmed that it works fine on her machine, so I have checked in (incl 2 new tests)

Still need to update the doc.

#8 - 29 Jul 2014 09:34 - Anna Maria Bigatti

- Status changed from Resolved to Feedback

tested also

```
/**/ K := NewFractionField(R);
/**/ phi := CanonicalHom(R,K);
/**/ K := "clear";
/**/ phi(ReadExpr(R,"x^2*y-2"));
x^2*y -2
```

ok!

#9 - 11 Sep 2014 17:20 - John Abbott

- Status changed from Feedback to Closed

- Target version changed from CoCoALib-1.0 to CoCoALib-0.99535 Sep14

- % Done changed from 80 to 100