CoCoALib - Feature #298

Valgrind: keep CoCoALib at 0 memory leaks

28 Jan 2013 08:02 - Anna Maria Bigatti

Status:	Closed	**Start date:**	28 Jan 2013
Priority:	High	**Due date:**	
Assignee:	Anna Maria Bigatti	**% Done:**	100%
Category:	Safety	**Estimated time:**	10.00 hours
Target version:	CoCoALib-0.99533 Easter14	**Spent time:**	10.10 hours

Description

We say CoCoALib does not leak memory, and that's mostly true.
Run

```
cd src/tests/
make valgrind
```

at least once a year to verify that.

When finding problems with **test-blah** run **valgrind ./test-blah**.

2014-07 Note: some tests (see below) will show a 16-byte leak: this is due to throwing/catching errors and is independent of CoCoALib and number of errors. Is this a valgring bug?

Related issues:

Related to CoCoA-5 - Bug #226: HilbertBasis segv	**Closed**	**10 Sep 2012**
Related to CoCoALib - Support #528: Release: CoCoALib-0.99533	**Closed**	**08 Apr 2014**
Related to CoCoALib - Support #599: Release: CoCoALib-0.99534	**Closed**	**08 Apr 2014**
Related to CoCoALib - Support #944: Release: CoCoALib-0.99550 (together with ...	**Closed**	**08 Apr 2014**

History

#1 - 28 Jan 2013 08:26 - Anna Maria Bigatti

Almost all tests run with definitely lost: 0 bytes in 0 blocks.
These are the exceptions (I guess many are related):

```
--test-BigInt3     definitely lost: 16 bytes in 1 blocks
--test-DenseUPolyRing1     definitely lost: 96 bytes in 3 blocks
--test-GOperations1     definitely lost: 768 bytes in 24 blocks
--test-hilbert1     definitely lost: 48 bytes in 2 blocks
--test-IsInteger1     definitely lost: 16 bytes in 1 blocks
--test-MachineInt1     definitely lost: 16 bytes in 1 blocks
--test-RandomSource2     definitely lost: 16 bytes in 1 blocks
--test-RandomSource3     definitely lost: 16 bytes in 1 blocks
--test-RingElem1     definitely lost: 576 bytes in 18 blocks
--test-RingElem2     definitely lost: 16 bytes in 1 blocks
--test-RingHom1     definitely lost: 16 bytes in 1 blocks
--test-RingTwinFloat4     definitely lost: 16 bytes in 1 blocks
--test-RingTwinFloat5     definitely lost: 16 bytes in 1 blocks
--test-RingTwinFloat6     definitely lost: 16 bytes in 1 blocks
--test-RingZmodN1     definitely lost: 16 bytes in 1 blocks
--test-SparsePolyRing1     definitely lost: 288 bytes in 9 blocks
--test-symbol1     definitely lost: 16 bytes in 1 blocks
--test-toric1     definitely lost: 96 bytes in 2 blocks
```

A few **16 bytes** are related with **ThrowError** as here:

```
==39413==    at 0x1002CAABD: malloc (vg_replace_malloc.c:274)
==39413==    by 0x1011460CF: __cxa_get_globals (in /usr/lib/libstdc++.6.0.9.dylib)
==39413==    by 0x101145DCD: __cxa_allocate_exception (in /usr/lib/libstdc++.6.0.9.dylib)
```

```
==39413==    by 0x100000DA4: CoCoA::ThrowError(CoCoA::ErrorInfo const&) (in ./test-MachineInt1)
```

#2 - 28 Jan 2013 15:00 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 20

JAA has corrected the leak which was detected in test-RingElem1.

JAA running valgrind-3.8.1 on his computer was **unable to reproduce** many of the reports of leaked memory. JAA's compiler is Apple's hacked gcc version 5566; whereas Anna's is Apple's hacked gcc version 5666 (= 5566+100). We must test on an unsullied Linux box with an unsullied gcc.

The outstanding leaks are:

```
 --test-DenseUPolyRing1
==20386==    definitely lost: 96 bytes in 3 blocks
 --test-GOperations1
==20408==    definitely lost: 768 bytes in 24 blocks
 --test-hilbert1
==20431==    definitely lost: 48 bytes in 2 blocks
 --test-toric1
==20631==    definitely lost: 48 bytes in 1 blocks
```

JAA recognises the leak in test-DenseUPolyRing1, but believes there is little point in fixing it since the faulty code will (soon?) be replaced (& a clean fix is non-trivial).

JAA now thinks the best solution to sorting out the leaks from ConvertToDUPFF is to proceed (rapidly) with the impl of DUPFp so that we can reduce/eliminate dependency on the old outmoded DUPFF. See issue #127.

2013-01-29 JAA obtained essentially the same results running on a Linux virtual machine; the number of blocks rose to 6 for test-toric1; also test-F5 was slightly strange (it reported a *definite leak*, but of 0 bytes). This suggests that some leaks found by Anna are peculiar to her computer/compiler.

2013-01-30 JAA has impl'd an experimental version of some DUPFp fns; after a suitable trivial change in SparsePolyRing.C the leak from test-GOperations1 was plugged. Also plugged the leak in test-DenseUPolyRing1 now.

#3 - 04 Feb 2013 16:41 - Anna Maria Bigatti

- % Done changed from 20 to 30

Removed leaks from **toric**.
This is now the situation on my Mac (which seems to have the mentioned problem with "throw")

```
--test-BigInt3     ==32258==    definitely lost: 16 bytes in 1 blocks
--test-DenseUPolyRing1 ==32321==    definitely lost: 96 bytes in 3 blocks
--test-GOperations1     ==32342==    definitely lost: 768 bytes in 24 blocks
--test-IsInteger1     ==32367==    definitely lost: 16 bytes in 1 blocks
--test-MachineInt1     ==32370==    definitely lost: 16 bytes in 1 blocks
--test-RandomSource2     ==32467==    definitely lost: 16 bytes in 1 blocks
--test-RandomSource3     ==32471==    definitely lost: 16 bytes in 1 blocks
--test-RingElem2     ==32484==    definitely lost: 16 bytes in 1 blocks
--test-RingHom1     ==32545==    definitely lost: 16 bytes in 1 blocks
--test-RingTwinFloat4     ==32564==    definitely lost: 16 bytes in 1 blocks
--test-RingTwinFloat5     ==32567==    definitely lost: 16 bytes in 1 blocks
--test-RingTwinFloat6     ==32571==    definitely lost: 16 bytes in 1 blocks
--test-RingZmodN1     ==32597==    definitely lost: 16 bytes in 1 blocks
--test-symbol1     ==32603==    definitely lost: 16 bytes in 1 blocks
```

2013-02-04 at time 19:10 JAA confirms that **all tests report clean** according to valgrind (running on his machine).

#4 - 11 Feb 2013 15:35 - John Abbott

- Status changed from In Progress to Feedback

- % Done changed from 30 to 90

#5 - 31 May 2013 14:41 - John Abbott

2013-05-30 all tests clean!

#6 - 31 May 2013 16:56 - John Abbott

- Target version changed from CoCoALib-0.9953 to CoCoALib-0.99534 Seoul14

Simply "moving" the issue to 0.9954 so we remember to do valgrind again.

#7 - 29 Oct 2013 14:58 - Anna Maria Bigatti

- Target version changed from CoCoALib-0.99534 Seoul14 to CoCoALib-0.99532

#8 - 02 Apr 2014 08:51 - Anna Maria Bigatti

- Tracker changed from Bug to Feature

- Subject changed from Valgrind: memory leaks to Valgrind: keep CoCoALib at 0 memory leaks

- Priority changed from Normal to High

- Target version changed from CoCoALib-0.99532 to CoCoALib-0.99533 Easter14

tested 0.99532, just a few (14) microscopic leaks (investigate for 0.99533, probably easy to spot)

```
Running valgrind on the CoCoALib tests (85 tests altogether)
------------------------------------------------------------
 --test-BigInt3 ==54796==    definitely lost: 16 bytes in 1 blocks
 --test-DenseUPolyRing1 ==54833==    definitely lost: 16 bytes in 1 blocks
 --test-IsInteger1 ==54893==    definitely lost: 16 bytes in 1 blocks
 --test-JBMill1 ==54896==    definitely lost: 16 bytes in 1 blocks
 --test-MachineInt1 ==54899==    definitely lost: 16 bytes in 1 blocks
 --test-RandomSource2 ==54974==    definitely lost: 16 bytes in 1 blocks
 --test-RandomSource3 ==54981==    definitely lost: 16 bytes in 1 blocks
 --test-RingElem2 ==54994==    definitely lost: 16 bytes in 1 blocks
 --test-RingHom1 ==55018==    definitely lost: 16 bytes in 1 blocks
 --test-RingTwinFloat4 ==55037==    definitely lost: 16 bytes in 1 blocks
 --test-RingTwinFloat5 ==55040==    definitely lost: 16 bytes in 1 blocks
 --test-RingTwinFloat6 ==55044==    definitely lost: 16 bytes in 1 blocks
 --test-RingZmodN1 ==55063==    definitely lost: 16 bytes in 1 blocks
 --test-symbol1 ==55088==    definitely lost: 16 bytes in 1 blocks
```

#9 - 16 Apr 2014 07:54 - Anna Maria Bigatti

```
test-BigInt3     definitely lost: 16 bytes in 1 blocks
```

leak is related with throwing the first CoCoA_ERROR as in this trivial test

```
  try  {CoCoA_ERROR(ERR::BadArg, "here");}
  catch (const ErrorInfo& err) { }
```

Any way to avoid this?

#10 - 16 Apr 2014 16:46 - Anna Maria Bigatti

I confirm the previous leaks (probably due to test for **throwing**/**catching errors**)
This "16-byte leak" is independent of the number of throws/catches and number of error values.

```
--test-BigInt3 ==70139==    definitely lost: 16 bytes in 1 blocks
--test-DenseUPolyRing1 ==70166==    definitely lost: 16 bytes in 1 blocks
--test-IsInteger1 ==70228==    definitely lost: 16 bytes in 1 blocks
--test-JBMill1 ==70231==    definitely lost: 16 bytes in 1 blocks
--test-MachineInt1 ==70235==    definitely lost: 16 bytes in 1 blocks
--test-RandomSource2 ==70308==    definitely lost: 16 bytes in 1 blocks
--test-RandomSource3 ==70314==    definitely lost: 16 bytes in 1 blocks
--test-RingElem2 ==70327==    definitely lost: 16 bytes in 1 blocks
--test-RingHom1 ==70348==    definitely lost: 16 bytes in 1 blocks
--test-RingTwinFloat4 =70366==    definitely lost: 16 bytes in 1 blocks
--test-RingTwinFloat5 ==70369==    definitely lost: 16 bytes in 1 blocks
--test-RingTwinFloat6 ==70374==    definitely lost: 16 bytes in 1 blocks
--test-RingZmodN1 ==70387==    definitely lost: 16 bytes in 1 blocks
--test-symbol1 ==70407==    definitely lost: 16 bytes in 1 blocks
--test-utils1 ==70436==    definitely lost: 16 bytes in 1 blocks
```

#11 - 16 Apr 2014 18:59 - Anna Maria Bigatti

JAA suggested to try

```
  try { throw 1; }
  catch (...) {}
```

and, yes! we get

```
==83441==    definitely lost: 16 bytes in 1 blocks
```

so it is not our fault, but a g++/valgrind quirk.

NOTE we shall have to remember this in the future (or perhaps upgrade gcc and/or valgrind)

#12 - 17 Apr 2014 08:37 - Anna Maria Bigatti

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

- Estimated time set to 10.00 h

I checked and confirm that the tests listed (those "leaking" **16 bytes**) are those which test errors (at least once). So that's **not a CoCoALib memory leak**.

I close this issue and list the "valgrind test" in each future "release" issue.

#13 - 12 Oct 2016 18:00 - Anna Maria Bigatti

- Related to Support #944: Release: CoCoALib-0.99550 (together with CoCoA-5.2.0) added