CoCoALib - Feature #962

General verbose mode?

07 Nov 2016 13:34 - John Abbott

Status:	Closed	**Start date:**	07 Nov 2016
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Improving	**Estimated time:**	6.60 hours
Target version:	CoCoALib-0.99560	**Spent time:**	6.25 hours

Description

Might it make sense to have a general "verbose" mechanism for CoCoALib?
(in addition to the specific one for Groebner bases?)

Related issues:

Related to CoCoALib - Feature #931: GBasis verbose mode	**Closed**	**24 Sep 2016**
Related to CoCoALib - Slug #969: Output to bad stream (operator<< and myOutpu...	**Closed**	**10 Nov 2016**
Related to CoCoA-5 - Feature #708: ExternalLib Normaliz: verbose flag?	**Closed**	**17 May 2015**

History

#1 - 07 Nov 2016 13:34 - John Abbott

- Related to Feature #931: GBasis verbose mode added

#2 - 07 Nov 2016 13:38 - John Abbott

Sometimes it could be helpful to have a verbose mode to understand what is happening inside a long computation (*e.g.* I tried a minpoly computation yesterday, and it took more than half an hour... why? Which was the slow part?)

A verbose mode would be useful principally for developers of CoCoALib, but some curious people might want to try it too.

I suggest having an integer indicating the "verbosity level": 0 -> no messages, 1 -> some messages, 2 -> maybe more messages.

#3 - 07 Nov 2016 13:54 - John Abbott

A possible design would be to have a global variable containing the verbosity level; so this would presumably be governed by GlobalManager.

Perhaps the neatest solution would be to have a function which returns either std::clog or "/dev/null" depending on whether the requested verbosity level is above or below that required for printing the log message: *e.g.*

```
VerboseLog(1) << "Starting elimination at time " << CpuTime() << endl;
```

NOTE searching on the internet for "ostream /dev/null" produced several suggestions; the simplest may be setting badbit temporarily (can it be done RAII?)

One disadvantage is that all potentially printed values are evaluated regardless of the verbosity level; a solution using preprocessor macros could avoid this...

NOTE there is a BOOST facility Boost.Log; I have not yet read the details. But we are trying to keep CoCoALib indep of BOOST...

#4 - 07 Nov 2016 14:50 - Anna Maria Bigatti

For the specific case you mentioned:
in MinPoly.C set

```
  static bool MINPOLY_DEBUG=true;
```

#5 - 07 Nov 2016 15:07 - John Abbott

I was thinking of having a **single** global variable for controlling verbosity, rather than many different names for different parts of CoCoALib.

An obvious advantage is that the user needs to memorize only a single name.

An apparent disadvantage is that you may get lots of other junk mixed in with the log message you want (*e.g.* if the function you are wanting to examine calls other complex functions). I do not know how serious this problem might be.

I am assuming that a developer wanting to use verbose mode would activate it only for the specific call to be investigated (and thereby, hoperfully, reducing the impact of the disadvantage).

#6 - 07 Nov 2016 18:23 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

Here is another idea (perhaps not so very KISS).

NOTE see my revision in note 10

At the start of a fn in which you want to place some verbose commands, you create a "verbose logging channel" specifying the name of the fn. Inside the fn you then use this channel to print out messages. Here is a simplistic example:

```
void MyFunc(int n)
{
  VerboseLog VERBOSE("MyFunc");
  ...
  VERBOSE << "Before loop" << endl;
  for (int i=1; i <= n; ++i)
  { ...}
  VERBOSE << "After loop" << endl;
}
```

A potential advantage of this approach is that the ctor for VerboseLog could increment a global counter (**not threadsafe!**) and its destructor decrement the counter; then there could be automatic suppression of verbose messages when the nesting depth exceeds a certain limit.

Passing the fn name to the ctor for VerboseLog would allow the name to added to printed messages (presumably when the VerboseLog object is converted to an ostream). Also it would offer the possibility to later enable disable verbose messages from certain fn families (just based on their names).

It might also be possible to let a VerboseLog object accept an integer as argument to indicate verbosity level: *e.g.*

```
  VERBOSE(2) << "Before loop at time " << CpuTime() << endl;
```

#7 - 10 Nov 2016 16:19 - John Abbott

A reasonable development strategy could be to resolve #931 (verbose for GBasis) using the ideas suggested here.
When that is working well, we can then generalize the technique for making other fns verbose.

#8 - 10 Nov 2016 16:31 - John Abbott

- Related to Slug #969: Output to bad stream (operator<< and myOutput): just return immediately added

#9 - 10 Nov 2016 18:16 - John Abbott

I have tried a quick experiment, and can confirm that (on my computer) creating an ofstream with a default ctor (*i.e.* without an associated file) produces a stream which immediately sets its badbit when any attempt is made to out to it.

I believe that this is a portable way of creating a sort of "/dev/null" stream.

In fact it is likely to be faster than creating an ofstream to /dev/null because the latter requires that the output characters all be generated, and then "thrown to the bit-bucket", whereas an ostream whose badbit is set would allow operator<< to exit immediately (before generating the output form).
See issue #969.

#10 - 11 Nov 2016 11:23 - John Abbott

With reference to comment 6, and the example there.

I now think it would be better to require that the programmer say explicitly for each use of VERBOSE the level:

- it makes implementation simpler
- it makes the code clearer (what should the default level if none is given explicitly?)

#11 - 11 Nov 2016 15:02 - John Abbott

- Target version changed from CoCoALib-1.0 to CoCoALib-0.99560

- % Done changed from 10 to 50

I have just checked in a first impl of "verbose".... including doc and an example (but no test).

Perhaps we can try using it to see whether the design is reasonable (and to find out which refinements should be added).

#12 - 23 Nov 2016 08:23 - Anna Maria Bigatti

Added in CoCoA-5. Fantastic!

#13 - 20 Jan 2017 16:40 - John Abbott

A quick grep suggests that we should check the use of **clog** in the following files:

- **ApproxPts2.C**
- **OpenMathXML.C**
- **TmpGPoly.C** and **TmpGReductor.C**

#14 - 10 Feb 2017 17:33 - Anna Maria Bigatti

I think we should set some rules like:

1 - 9 (or to 99) reserved for users.
100 - ?? for higher cocoa/cocolib functions
1000 - ?? for deeper cocoa/cocoalib function (for GBasis now it is 1965)

This because the verbosity level is global, so we need to "filter out" some common calls.

#15 - 15 Feb 2017 18:40 - John Abbott

Anna's idea of reserving certain ranges for different "levels" of function appeals to me. I think her ranges are too spread apart, and am tempted to suggest more compact ranges:

- 1--9 for user fns
- 10--19 for "higher" CoCoALib fns (or should it be 11--19?)
- 20--29 for "deeper" CoCoALib fns (or should it be 21--29?)

I can see that using the range 100-109 for deeper fns is possibly more "mnemonic".
Of course 1965 for G-basis computation is "amusing".

We could even have a much more limited range: 1--4 (incl) for user fns, and 5--9 for CoCoALib fns (without making a distinction between "higher" and "deeper" fns). A possible advantage of a more compact range is that an increment in the level will likely produce an observable effect.

Comments? Opinions? I wonder if there are any guidelines "out there"? (on the internet?)

#16 - 16 Feb 2017 08:36 - Anna Maria Bigatti

I prefer the mnemonic number of digits. So I suggest
1 digit -- free for users
2 digits -- cocoalib and cocoa packages
3 digits -- inside GBasis or deeper arithmetic

Then we can decide that we use only a small range (e.g. 10-15)

This would also allow us to use funny ranges (e.g. 40-49) during development.
(and they would be easy to spot when cleaning up for releasing ;-)

#17 - 16 Feb 2017 12:45 - John Abbott

How about starting with just two ranges?
1--9 for users
10--??? for CoCoA packages and CoCoALib

I had also considered making the verbose levels a collection of <string>+<int> where the string would identify the family of fns, and the int would indicate the level of verbosity for that family. I preferred to start with just an int to see how that works (before potentially implementing a more complicated system).

I still like the idea of reserving low number for the user. I am less convinced about the distinction between "high level" and "low level" CoCoALib fns; it reminds of those arbitrary administrative categories which always end up with ridiculous situations (which exhibit lack of foresight on the part of the inept creator of the categories).

#18 - 02 Mar 2017 13:44 - Anna Maria Bigatti

- % Done changed from 50 to 70

Changed interface (after personal discussion in Kassel):

```
void SetVerbosityLevel(long);
long VerbosityLevel();
```

Currently (while getting more experience on it) we decided to have levels
1-9 reserved for the user, ald level 100 for first info on GBasis progress.
[and I set 50 for gin computation, in CoCoA-5]

Should we have a list for verbosity levels? This might be duplication of information as we surely should list what is in the documentation of functions....
Think of an automatic way of having both.

#19 - 28 Apr 2017 20:22 - Anna Maria Bigatti

- Related to Feature #708: ExternalLib Normaliz: verbose flag? added

#20 - 06 Nov 2017 13:48 - John Abbott

- Status changed from In Progress to Feedback

- % Done changed from 70 to 90

#21 - 08 Nov 2017 16:29 - John Abbott

- Status changed from Feedback to Closed

- Assignee set to John Abbott

- % Done changed from 90 to 100

- Estimated time set to 6.60 h