CoCoALib - Feature #743

Better errors: give supplementary info about the error

30 Jun 2015 16:20 - John Abbott

Status:	In Progress	**Start date:**	30 Jun 2015
Priority:	Urgent	**Due date:**	
Assignee:		**% Done:**	10%
Category:	Improving	**Estimated time:**	0.00 hour
Target version:	CoCoALib-0.99880	**Spent time:**	1.85 hour

Description

When CoCoALib produces an error which is then caught by CoCoA-5, the only information from the error which is shown to the user is the main message (*e.g.* **Bad Arg**) which can sometimes be not so helpful.

The idea is to have an optional supplementary message giving some more information (which is also displayed to the CoCoA-5 user).

We are also planning to simplify the types of error CoCoALib can produce, so that some "slight distinctions" between existing errors will be lost -- these distinctions can be recovered through the supplementary message.

Related issues:

Related to CoCoALib - Feature #385: Design new errors using inheritance	**In Progress**	**08 Jul 2013**
Related to CoCoALib - Feature #92: Error Codes	**In Progress**	**14 Feb 2012**
Related to CoCoALib - Design #427: Error names and error messages (current de...	**In Progress**	**28 Jan 2014**
Related to CoCoA-5 - Feature #1045: Error message from cocoalib to cocoa-5	**Closed**	**13 Apr 2017**
Related to CoCoA-5 - Bug #1622: BuiltinOneLiners -- less precise location in ...	**New**	**18 Oct 2021**
Related to CoCoALib - Design #1098: Ctors for exceptions/errors	**New**	**06 Sep 2017**

History

#1 - 30 Jul 2015 15:28 - John Abbott

- Priority changed from Normal to High

Tracking down the problem in LinearSimplify (see issue #490) was impeded by the vagueness of the error message *Bad Argument*.

Increased priority to **high**.

How exactly are the supplementary details to be specified? Here is an example how it might look:

```
if (...) CoCoA_ERROR(ERR::BadArg+"1st arg must be prime", "SomeFunc");
if (...) CoCoA_ERROR(ERR::BadArg("1st arg must be prime"), "SomeFunc");
```

We must decide syntax and semantics (and implementation details). JAA finds the syntax ERR::blah("details") easy to read. The semantics should just be that the supplementary details are appended to the what string (at least, that is how it should appear to a user of a CoCoA::ErrorInfo object). The implementation could maintain a single "what" string, or perhaps a primary "what" string and a supplementary one -- at first sight, this latter seems more complicated without any obvious benefit, but I'd like to think about it more before discarding the idea.

#2 - 22 Mar 2016 17:28 - Anna Maria Bigatti

- Target version changed from CoCoALib-0.99540 Feb 2016 to CoCoALib-0.99560

#3 - 23 Mar 2016 17:39 - Anna Maria Bigatti

- Target version changed from CoCoALib-0.99560 to CoCoALib-0.99550 spring 2017

#4 - 17 Jun 2016 14:31 - John Abbott

- Target version changed from CoCoALib-0.99550 spring 2017 to CoCoALib-0.99560

#5 - 13 Apr 2017 09:40 - Anna Maria Bigatti

- Status changed from New to In Progress

I'm getting quite annoyed with getting The arg(s) given are unsuitable in CoCoA-5: looks like being back at the CoCoA-4 unhelpful error messages!

While waiting to redesign error handling/classification/inheritance, we'd better create more error messages (maybe using some useful prefix like BADARG_, to facilitate the future translation to inheritance).

#6 - 13 Apr 2017 13:43 - John Abbott

- Priority changed from High to Urgent

- % Done changed from 0 to 10

I agree that BadArg is unhelpful. It would be nice to settle this matter, and it would be nice to find some guidelines on how to design error hierarchies... Something to do over the easter weekend?
Priority increased.

I wonder if it could be helpful to have BadArg1 and BadArg2, so that there is some indication of which arg is causing the trouble? I can see that this might also be misleading in some cases.

Perhaps we should collect a few examples where the current behaviour is definitely unsatisfactory...

#7 - 13 Apr 2017 14:01 - Anna Maria Bigatti

John Abbott wrote:

> I agree that BadArg is unhelpful. It would be nice to settle this matter, and it would be nice to find some guidelines on how to design error hierarchies... Something to do over the easter weekend?
> Priority increased.

I don't think we can fix this before next (imminent) release.

> I wonder if it could be helpful to have BadArg1 and BadArg2, so that there is some indication of which arg is causing the trouble? I can see that this might also be misleading in some cases.

dangerous: some function may be called cascading from CoCoA-5 (see also just below)

One thing we should probably change is that CoCoA-5 should print more than just the main error message. Example:

```
    if (N <= 0) CoCoA_ERROR(ERR::BadArg, "LucasTest(N):  N must be positive");
```

This is a misuse of the second argument which (by current design) should only be the **name** of the function (as a workaround of not being able to extract it automatically).

So, either we decide that the second argument has a different use (==> passed to CoCoA-5), of the information "N must be positive" should be part of the first argument.

Maybe the function name (i.e. second argument) should indeed be passed to CoCoA-5 toghether with the error message (i.e. first argument), whereas the other details (file name, line number)

#8 - 13 Apr 2017 14:31 - Anna Maria Bigatti

On the line of what I said earlier, I implemented in Interpreter.C the function MESSAGE(err) to be called instead of message(err).
It just prints message + context. Have a try and see how it looks (cvs-ed).

May be improved, and is not THE solution to this issue, but it is a quick solution to the problem of unhelpful error messages in CoCoA-5.

#9 - 13 Apr 2017 14:56 - John Abbott

I agree that resolving this issue will take more time than we can invest in the next release.

I think it would be helpful to collect some examples where the current error response is distinctly unhelpful.

#10 - 13 Apr 2017 17:19 - Anna Maria Bigatti

- Related to Feature #92: Error Codes added

#11 - 13 Apr 2017 17:19 - Anna Maria Bigatti

- Related to Design #427: Error names and error messages (current design) added

#12 - 13 Apr 2017 17:20 - Anna Maria Bigatti

There are some notes about "classes of errors" in the CoCoALib documentation for error, section "=== new improved list of errors ==="
(see also issue #92 for parallel discussion)

#13 - 13 Apr 2017 17:38 - Anna Maria Bigatti

- Related to Feature #1045: Error message from cocoalib to cocoa-5 added

#14 - 06 Nov 2017 13:41 - John Abbott

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

#15 - 12 Jun 2018 18:27 - John Abbott

- Target version changed from CoCoALib-0.99600 to CoCoALib-0.99650 November 2019

#16 - 05 Apr 2019 16:26 - John Abbott

- Target version changed from CoCoALib-0.99650 November 2019 to CoCoALib-0.99700

It would be nice to have a quiet moment to get this matter properly sorted out.

We really should try to deal with it soon! Something to discuss next time Anna and I meet (where and when?)

#17 - 08 Jan 2020 22:58 - John Abbott

- Target version changed from CoCoALib-0.99700 to CoCoALib-0.99800

#18 - 06 Oct 2020 12:11 - John Abbott

- Target version changed from CoCoALib-0.99800 to CoCoALib-0.99850

I am postponing even though this is urgent. I need a quiet period to be able to think about the problem, and to find a good solution.
Next likely chance for a quiet period might be in Feb/March 2021 (depending on how much chaos Brexit triggers).

#19 - 19 Nov 2021 17:03 - Anna Maria Bigatti

- Related to Bug #1622: BuiltinOneLiners -- less precise location in error messages added

#20 - 19 Nov 2021 17:05 - John Abbott

Anna suggests that the error message should also indicate which args (via their indices, starting from 0 or 1?)
For instance in FloatApprox the err mesg should be something like **arg 2, relative error must be between 0 and 1**

I suppose "2nd arg" is probably clearer than "arg 2"...?

#21 - 19 Nov 2021 17:18 - Anna Maria Bigatti

We agreed that the string for the name of the function should contain just the name of the function, and its arguments.
So we could have

```
  if (N <= 0) CoCoA_ERROR(ERR::MustBePositive, "N", "LucasTest(N)");
```

#22 - 19 Nov 2021 17:25 - Anna Maria Bigatti

for example, this would be nice

```
/**/  EulerTotient(-100);
--> ERROR: arg N must be strictly positive
--> [CoCoALib] EulerTotient(N)
--> /**/  EulerTotient(-100);
-->       ^^^^^^^^^^^^^^^^^
```

#23 - 08 Mar 2023 19:49 - John Abbott

- Target version changed from CoCoALib-0.99850 to CoCoALib-0.99880

#24 - 15 Mar 2024 18:31 - John Abbott

- Related to Bug #1089: invalid pointer in "free" added

#25 - 15 Mar 2024 18:31 - John Abbott

- Related to deleted (Bug #1089: invalid pointer in "free")

#26 - 15 Mar 2024 18:31 - John Abbott

- Related to Design #1098: Ctors for exceptions/errors added

#27 - 15 Mar 2024 18:33 - John Abbott

Let's finish this for version 1.0