CoCoALib - Support #953

new file for old functions: obsolescent.C

26 Oct 2016 17:44 - Anna Maria Bigatti

Status:	Closed	**Start date:**	26 Oct 2016
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Renaming	**Estimated time:**	7.00 hours
Target version:	CoCoALib-0.99550 spring 2017	**Spent time:**	6.45 hours

Description

In CoCoA-5 we have obsolescent.cpkg5, a package with all the functions declared obsolescent, for example if they have been renamed, of if the syntax has changed.
The call to the obsolescent function prints a useful warning message, suggesting how to modernize the code, and then returns the expected value.

The file also reports the date of the possible retirement of the obsolescent function.

I think it would be useful to have something similar in CoCoALib.

Related issues:

Related to CoCoALib - Feature #951: New function: IsSqFree	**Closed**	**24 Oct 2016**

History

#1 - 02 Nov 2016 20:59 - Anna Maria Bigatti

- Related to Feature #951: New function: IsSqFree added

#2 - 02 Nov 2016 22:15 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

Here are some ideas about how we might achieve a reasonable design:

- obsolete/obsolescent fns go into 2 files obsolete.H and obsolete.C (or obsolescent.H???)
- have a compilation flag (perhaps **CoCoA_OBSOLETE**) which make the contents of obsolete.H visible only when set
- perhaps have a global run-time flag which says whether to allow calls to obsolete fns; this flag would then be checked in every "forwarding fn" in obsolete.C, throwing an error if calling is forbidden.

Maybe this is too complicated? KISS?

I would like to have an easy way of being sure that my code is "clean" (*i.e.* does not call any obsolete fns).

#3 - 03 Nov 2016 07:56 - Anna Maria Bigatti

John Abbott wrote:

> Here are some ideas about how we might achieve a reasonable design:
>
> - obsolete/obsolescent fns go into 2 files obsolete.H and obsolete.C (or obsolescent.H???)

I prefer obsolescent, as in CoCoA-5 (maybe we can have obsolete for afterwards)

- have a compilation flag (perhaps **CoCoA_OBSOLETE**) which make the contents of obsolete.H visible only when set

CoCoA_NO_OBSOLESCENT

- perhaps have a global run-time flag which says whether to allow calls to obsolete fns; this flag would then be checked in every "forwarding fn" in obsolete.C, throwing an error if calling is forbidden.

I think that obsolescent function should available and working, but just printing an evident and helpful warning (as in CoCoA-5).

> Maybe this is too complicated? KISS?
>
> I would like to have an easy way of being sure that my code is "clean" (*i.e.* does not call any obsolete fns).

Just compile with CoCoA_NO_OBSOLESCENT ;-)

#4 - 03 Nov 2016 10:28 - John Abbott

- Assignee set to John Abbott

- % Done changed from 10 to 30

I have a first implementation (but called the file obsolete.C since I find "obsolete" easier to type than "obsolescent"). I can rename the file if you want... I have not yet checked in.

I am unsure about printing out a warning message each time an obsolete fn is called. I know it is what we do in CoCoA-5. Currently, either the fn is called silently (if the user has set the global flag for allowing obsolete fns), or an error is thrown.

I suppose it could be useful to have an easy way of knowing all obsolete fns which have been called...

#5 - 03 Nov 2016 11:05 - Anna Maria Bigatti

John Abbott wrote:

> I have a first implementation (but called the file obsolete.C since I find "obsolete" easier to type than "obsolescent"). I can rename the file if you want... I have not yet checked in.

I prefer //obsolescent// because it is compatible with the same idea in cocoa-5 and also because we still allow them for some time.
Later we could have also //obsolete// for those definitely out (just returning useful error)

> I am unsure about printing out a warning message each time an obsolete fn is called. I know it is what we do in CoCoA-5. Currently, either the fn is called silently (if the user has set the global flag for allowing obsolete fns), or an error is thrown.

I think we should also have the warning. Maybe we could have a global counter for each function (too bad for parallel misbehaviour) giving the error only the first time.
This way a user will get his result anyway and all possible warnings.

> I suppose it could be useful to have an easy way of knowing all obsolete fns which have been called...

the counter? a registry?

Anyway let's start with the KISS approach.

#6 - 03 Nov 2016 16:28 - John Abbott

I have checkedin the relevant files. The new files are called **obsolescent.H** and **obsolescent.C**.

CoCoALib is always built with the obsolescent fns; only the declarations in the header file appear/disappear depending on the setting of the compile-time/preprocessor flag. Currently the preprocessor flag is called **CoCoA_OBSOLETE** but I shall change that shortly.

My preference is that obsolescent fns remain invisible by default; the user must do something to make them visible. So probably the flag name could be something like **CoCoA_ALLOW_OBSOLESCENT**.

Should it be a simple flag? Undefined means "forbid calls to obsolescent fns", and defined means "allow calls to obsolescent fns".
Or perhaps something slightly more refined? When defined the value says what to do when calling an obsolescent fn:

- **(A)** do nothing special, just call the proper fn
- **(B)** print out a warning that fn has been called, then call the proper fn
- **(C)** throw an error (perhaps saying which obsolescent fn was called)

I'll change the current impl so that it does (B) rather than (A).

#7 - 03 Nov 2016 16:45 - Anna Maria Bigatti

John Abbott wrote:

> My preference is that obsolescent fns remain invisible by default; the user must do something to make them visible. So probably the flag name could be something like **CoCoA_ALLOW_OBSOLESCENT**.
>
> Should it be a simple flag? Undefined means "forbid calls to obsolescent fns", and defined means "allow calls to obsolescent fns".
> Or perhaps something slightly more refined? When defined the value says what to do when calling an obsolescent fn:
>
> - **(A)** do nothing special, just call the proper fn
> - **(B)** print out a warning that fn has been called, then call the proper fn
> - **(C)** throw an error (perhaps saying which obsolescent fn was called)

CoCoA_OBSOLESCENT_SILENT
CoCoA_OBSOLESCENT_WARNING
CoCoA_OBSOLESCENT_ERROR

#8 - 03 Nov 2016 17:06 - John Abbott

- % Done changed from 30 to 50

I prefer not to have so many preprocessor flags. We do need 1 to make the contents of obsolescent.H visible/invisible.

Another possibility is simply to have a setting in the GlobalManager which says what to do when calling an obsolescent fn. At the moment such a setting exists, and it simply allows the user to choose between "throw error" and "allow call" (with default behaviour to print out a warning message on every call); the default is to throw an error.

I am about to check in the changes to GlobalManager.

I hope we can soon close this issue because it does feel a bit like "wasted time" (though I suppose it might improve usability of CoCoALib for some users).

#9 - 04 Nov 2016 10:43 - John Abbott

Perhaps we do not need any preprocessor flag. What about the following approach?

The file **library.H** does not include **obsolescent.H** (this requires a small change to MakeUniversalHeader.sh).
A user who wants to access obsolescent fns must include both CoCoA/library.H and CoCoA/obsolescent.H.

This seems to be a very KISS approach; what do you think?

#10 - 04 Nov 2016 15:47 - Anna Maria Bigatti

John Abbott wrote:

> Perhaps we do not need any preprocessor flag. What about the following approach?
>
> The file **library.H** does not include **obsolescent.H** (this requires a small change to MakeUniversalHeader.sh).
> A user who wants to access obsolescent fns must include both CoCoA/library.H and CoCoA/obsolescent.H.
>
> This seems to be a very KISS approach; what do you think?

I think obsolescent should be included by default (we are doing that to help the users).
For when we developers compile we can/know disable it.

#11 - 04 Nov 2016 17:02 - John Abbott

I really do not like the idea of including obsolescent fns by default: in what sense would they be obsolescent?

Is it really too much to ask the user to add just one line to their file: **#include "CoCoA/obsolescent.H"**

Having a preprocessor flag determine the visibility of the obsolescent fns would make it "obligatory" to recompile the whole library after changing the compilation flags. Perhaps a workaround would be possible, but I'm not willing to do it.

When a user wants to check whether all calls to obsolescent fns have been removed, it is enough just to comment out the include directive for **obsolescent.H** at the start of the user's source file, and then recompile just that file -- no need to recompile CoCoALib.

Note that CoCoALib (*i.e.* **libcocoa.a**) always contains the obsolescent fns; they become "visible" only when the user includes obsolescent.H.

I really do feel that this is the right approach (*i.e.* put an extra include in the user's source file). It is even possible for the user to combine files which need obsolescent fns and those which do not (and the include directive makes it clear which files use them).

#12 - 04 Nov 2016 19:02 - Anna Maria Bigatti

John Abbott wrote:

> Is it really too much to ask the user to add just one line to their file: **#include "CoCoA/obsolescent.H"**

ok

#13 - 04 Nov 2016 21:46 - John Abbott

- % Done changed from 50 to 80

Thanks, Anna! :-)

Checked in.

I did apparently lie slightly: the user has to add the include directive **and** also give the option **AllowObsolescentFns** to GlobalManager (o/w an exception will be thrown at run-time when the obsolescent fn is called).

Anyway, this approach even permits us to put in tests/examples for obsolscent fns (without having to do funny things with preprocessor flags).

Still have to write documentation (for GlobalManager)

#14 - 04 Nov 2016 22:41 - John Abbott

- Status changed from In Progress to Feedback

- % Done changed from 80 to 90

Added test-obsolescent.C, ex-obsolescent.C and added doc to GlobalManager.txt.

#15 - 05 Nov 2016 17:39 - John Abbott

Also added **doc/txt/obsolescent.txt** (with relevant changes to the index files).

Perhaps Anna could check what I've written?

Also done some cleaning to obsolescent.C.

#16 - 29 Mar 2017 11:29 - Anna Maria Bigatti

- Estimated time set to 7.00 h

#17 - 29 Mar 2017 18:13 - Anna Maria Bigatti

- Status changed from Feedback to Closed

- % Done changed from 90 to 100