CoCoA-5 - Bug #545

Compiler g++ 4.2.1 (clang-503.0.40) on MacOSX 10.9: warnings and errors

30 Apr 2014 09:11 - Anna Maria Bigatti

Status:	Closed	**Start date:**	30 Apr 2014
Priority:	High	**Due date:**	
Assignee:		**% Done:**	100%
Category:	Portability	**Estimated time:**	10.00 hours
Target version:	CoCoA-5.2.2	**Spent time:**	6.35 hours

Description

Bruno Simoes found some warnings and a error in a cocoa-5 test.

problem (1) struct/class fixed
problem (2) overloading of member fn being fixed ("using ...")
problem (3) -no_pie fixed
problem (4) exsegv fixed

Related issues:

Related to CoCoALib - Design #551: Reconsider overloaded virtual mem fns	**New**	**09 May 2014**

History

#1 - 30 Apr 2014 09:19 - Anna Maria Bigatti

- % Done changed from 0 to 10

Problem (1) [solved]

```
./Interpreter.H:63:1: warning: struct 'RuntimeEnvironment' was previously declared as a class [-Wmismatched-ta
gs]

./AST.H:78:8: note: previous use is here
```

I think there's no reason to have "struct" Interpreter.H. I wrote "class" and it compiles fine.
Waiting for Bruno's feedback (I cannot test it myself because my compiler doesn't take -Wmismatched-tags)

#2 - 30 Apr 2014 09:21 - Anna Maria Bigatti

Next to investigate
Problem (2)

```
./AST.H:1768:7: warning: 'CoCoA::AST::DumpAsTreeVisitor::visit' hides overloaded virtual functions [-Woverload
ed-virtual]

        void visit(Import &import);

             ^

./AST.H:1676:15: note: hidden overloaded virtual function 'CoCoA::AST::ParsedObjectVisitor::visit' declared he
re: type mismatch at 1st

      parameter ('CoCoA::AST::IntLiteral &' vs 'CoCoA::AST::Import &')
```

```
        virtual void visit(IntLiteral &);
```

#3 - 30 Apr 2014 09:25 - Anna Maria Bigatti

problem (3)

```
ld: warning: PIE disabled. Absolute addressing (perhaps -mdynamic-no-pic) not allowed in code signed PIE, but 
used in ___gmpn_divexact_1 from /opt/local/lib/libgmp.a(dive_1.o). To fix this warning, don't compile with -md
ynamic-no-pic or link with -Wl,-no_pie
```

#4 - 30 Apr 2014 09:27 - Anna Maria Bigatti

problem (4)

```
*****  exsegv FAILED  ***** (wrong output)
```

other tests pass.
Waiting for exsegv.found

#5 - 30 Apr 2014 09:32 - Anna Maria Bigatti

Anna Maria Bigatti wrote:

> Next to investigate
> Problem (2)
> [...]

-Woverloaded-virtual is accepted by my compiler (I added it in autoconf.mk) and indeed produces lots of warnings about visit. Like:

```
AST.H:1676: warning: 'virtual void CoCoA::AST::ParsedObjectVisitor::visit(CoCoA::AST::IntLiteral&)' was hidden
AST.H:1821: warning:   by 'CoCoA::AST::DumpAsTreeVisitor::visit'
```

I'll investigate.

#6 - 30 Apr 2014 14:56 - Anna Maria Bigatti

Problem (2) -- continued
compilation with -Woverloaded-virtual

```
AST.H:1676: warning: 'virtual void CoCoA::AST::ParsedObjectVisitor::visit(CoCoA::AST::IntLiteral&)' was hidden
AST.H:1821: warning:   by 'CoCoA::AST::DumpAsTreeVisitor::visit'
```

In this case we have that the first line refers to **ParsedObjectVisitor::visit** and the second to **DumpAsTreeVisitor::visit** (DumpAsTreeVisitor inherits from ParsedObjectVisitor).
... but the work fine anyway.
John suggests that maybe a cleaner design would be to have one abstract class, and both DumpAsTreeVisitor and ParsedObjectVisitor inheriting from it. Any comments Giovanni?

#7 - 07 May 2014 21:28 - John Abbott

- % Done changed from 10 to 20

Similar overload-virtual warnings are produced even when compiling CoCoALib. I believe I understand the problem, and there is an easy "solution" (just insert a using command in the definition of certain derived classes).

I just have to decide whether I want to apply this solution simply to shut the compiler up when someone turns on a particularly fussy flag. Strictly the compiler is probably right to warn about this, but I cannot envisage any likely scenario where a true problem might arise...

NOTE: the problem is that mem fns in RingBase with default defns (*e.g.* myNew(BigRat)), would not be callable via a pointer to a derived class *e.g.* SmallRingFpImpl.

#8 - 08 May 2014 08:16 - Anna Maria Bigatti

- Subject changed from Compiler XXX: warnings and errors to Compiler XXX on MacOSX 10.9: warnings and errors

#9 - 09 May 2014 09:26 - Anna Maria Bigatti

- % Done changed from 20 to 40

John Abbott wrote:

> Similar overload-virtual warnings are produced even when compiling CoCoALib. I believe I understand the problem, and there is an easy "solution" (just insert a using command in the definition of certain derived classes).
>
> I just have to decide whether I want to apply this solution simply to shut the compiler up when someone turns on a particularly fussy flag. Strictly the compiler is probably right to warn about this, but I cannot envisage any likely scenario where a true problem might arise...

I found when the problem might arise: a typo, or a missing "const". One thinks he's defining a concrete implementation and instead he's making a new function in the derived class.
When I read this description I have seen myself having chased such "bugs"...

So it is probably useful to keep the compilation flag and to add those "using".

I expect there aren't too many. Famous last words?

#10 - 09 May 2014 09:39 - Anna Maria Bigatti

Anna Maria Bigatti wrote:

> So it is probably useful to keep the compilation flag and to add those "using".
> I expect there aren't too many. Famous last words?

for CoCoA-5 dir it is just this "using" in 3 places and I get a clean compilation.
(cvs-ed)

```
  using ParsedObjectVisitor::visit; // disables warnings of overloading
```

#11 - 09 May 2014 10:50 - Anna Maria Bigatti

- Subject changed from Compiler XXX on MacOSX 10.9: warnings and errors to Compiler g++ 4.2.1 (clang-503.0.40) on MacOSX 10.9: warnings and errors

#12 - 09 May 2014 11:53 - Anna Maria Bigatti

problem (2): done for Sugar too.
Now to be done for all concrete rings: myNew, myAssign, myIndets.
I think that should be all.

#13 - 09 May 2014 13:39 - John Abbott

We should employ the using "trick" only if it the correct design! I believe it is a bad idea to use it just to make the compiler quiet (without checking that the effect of using is what we really want).

I have thought of one possible scenario where our current design could cause problems: in template code which works with pointers (or references) to specific ring implementation classes. For instance one could imagine a template fn (*e.g.* with arg RingImpl*) which wants to use myNew to construct from a BigRat; this would not work (*i.e.* not even compile) if the actual pointer type is SmallFpImpl*.

#14 - 09 May 2014 13:46 - John Abbott

Here is an argument against blindly applying the using "trick".

The class RingBase defines a mem fn myNew(const symbol&) with a default defn valid for all ring impls. The default defn will throw an exception if the ring is a SmallFpImpl. If we blindly apply using then (template) code trying to call myNew(const symbol&) will compile, but then produce a run-time error. The current design means that code which tries to access that mem fn for a SmallFpImpl will not compile (because the mem fn does not exist).

What to do? I'd like myNew(const BigRat&) to be visible in derived classes (with its default defn), but I don't want myNew(const symbol&) to be visible... How to express this in C++?

#15 - 09 May 2014 14:57 - John Abbott

- % Done changed from 40 to 50

We could avoid the problem by **using different names for the various mem fns**; this idea should be considered (& then implemented or rejected, with suitable justification).

#16 - 03 Sep 2014 12:26 - John Abbott

- Priority changed from Normal to High

- Target version changed from CoCoA-5.1.1 Seoul14 to CoCoA-5.1.2 summer 2015

#17 - 29 Jul 2015 21:30 - John Abbott

- Target version changed from CoCoA-5.1.2 summer 2015 to CoCoA-5.1.3/4 Jan 2016

- Estimated time changed from 4.00 h to 10.00 h

I'd forgotten about this issue. It'll take time to understand what is the right thing to do -- no chance for 5.1.2, so I'm postponing to the next version.
[increased est'd time, to allow for reading around and fully understanding the issue]

NOTE postponing is OK because the code compiles anyway, and the warnings can perhaps be suppressed by an appropriate compiler flag?

#18 - 17 Feb 2016 15:24 - John Abbott

- Target version changed from CoCoA-5.1.3/4 Jan 2016 to CoCoA-5.2.0 spring 2017

Postpone again... :-/

#19 - 26 Apr 2017 15:55 - Anna Maria Bigatti

- Target version changed from CoCoA-5.2.0 spring 2017 to CoCoA-5.2.2

#20 - 20 Nov 2017 22:14 - Anna Maria Bigatti

- Description updated

- Status changed from In Progress to Feedback

- % Done changed from 50 to 90

#21 - 20 Nov 2017 22:31 - Anna Maria Bigatti

- Description updated

- Status changed from Feedback to Closed

Now using 10.11.
There are still some warning (lots), but compilation has been working fine for ages now.
Closing.

#22 - 20 Nov 2017 22:31 - Anna Maria Bigatti

- % Done changed from 90 to 100

#23 - 12 Mar 2019 16:40 - Anna Maria Bigatti

Problem with PIE (#3) resurfaced because of a line swap in configure.
Just in case it happens again:
this was solved by making the script configuration/fpic-flag.sh which is called by configure (setting the flag -fPIC).