# CoCoALib - Support #302

## Compilation on M$Windows: Visual Studio

11 Feb 2013 11:09 - Christof Soeger

| | | | |
|---|---|---|---|
| **Status:** | In Progress | **Start date:** | 11 Feb 2013 |
| **Priority:** | Normal | **Due date:** | |
| **Assignee:** | | **% Done:** | 60% |
| **Category:** | Portability | **Estimated time:** | 0.00 hour |
| **Target version:** | CoCoALib-1.0 | **Spent time:** | 12.70 hours |

**Description**

We should offer all documentation is needed for compiling CoCoALib/CoCoA-5 on MSWindows.

CoCoA-5 has been compiled using MSVC (thanks Bruno Simoes), but the GUI has some mysterious problems. And the textual version? Are all the necessary files in our distribution?

**Related issues:**

| | | |
|---|---|---|
| Related to CoCoA - Support #266: Compilation on M$Windows: cygwin | **Closed** | **17 Oct 2012** |

### History

**#1 - 11 Feb 2013 11:10 - Christof Soeger**

*- Status changed from New to In Progress*

Bruno wrote:

I tried the new cocoa version and it is different from the version you sent me. There is at least a new flag that has to be defined. In order to test if it compiles I just defined it to be empty. In libnormaliz-template.cpp, I also had to comment the following line:

#include "libnormaliz.cpp"

Somehow that stuff is already included/defined somewhere. Sunday I will check why is it happening.

If you have the last MSVC compiler then it should work, otherwise you will get one compilation error due to a bug in previous versions (I guess you remember it). You need to compile and configure QT for windows. All the other libs are included in this package (see folders include, lib, libnormaliz). Replace them if you need to upgrade or downgrade.

Instructions:
https://dl.dropbox.com/u/23586600/msvc_solution.rar

Extract the content of the rar to the cocoa folder e.g. CoCoALib-0.9952. You will get a structure like this:

CoCoALib-0.9952\MSVC10\
CoCoALib-0.9952\libnormaliz\

- Open CoCoA.sln
- Build "solution" (compiles CoCoA 5 with gui)

if you need the cocoa console, switch solution configuration to "Console" and build again.

#### #2 - 11 Feb 2013 11:35 - Christof Soeger

I think "ExternalLibs-NormalizTypes.C" is a relict from the time when libnormaliz was not a compiled independently and is unused now. I comment out everything in there and it is working. Maybe you should remove the file from the repository.

The file from Bruno is a Visual Studio 10 (2010) file, so you need at least that version. For the recent CVS version of CoCoA one hase to make one change, remove all Random*.C source files and add random.C. On my home PC with VS 2012 the library then compiles. But the QT does not work:

```
1>------ Erstellen gestartet: Projekt: QCodeEdit, Konfiguration: Release Win32 ------
1>  Moc'ing ..\..\src\CoCoA-5\QCodeEdit\qcodecompletionengine.h...
1>  Das System kann den angegebenen Pfad nicht finden.
1>  Moc'ing ..\..\src\CoCoA-5\QCodeEdit\document\qdocument.h...
1>  Das System kann den angegebenen Pfad nicht finden.
(and more messages of that type)
```

But the files are in the src\CoCoA-5\QCodeEdit folder.

Here are some warnings that might hint to some code improvements:

```
2>..\..\src\AlgebraicCore\BigInt.C(159): warning C4800: 'int': Variable wird auf booleschen Wert ('True' oder
'False') gesetzt (Auswirkungen auf Leistungsverhalten möglich)
2>..\..\src\AlgebraicCore\BigInt.C(161): warning C4800: 'int': Variable wird auf booleschen Wert ('True' oder
'False') gesetzt (Auswirkungen auf Leistungsverhalten möglich)
2>..\..\src\AlgebraicCore\BigRat.C(594): warning C4800: 'int': Variable wird auf booleschen Wert ('True' oder
'False') gesetzt (Auswirkungen auf Leistungsverhalten möglich)
2>..\..\src\AlgebraicCore\IntOperations.C(580): warning C4800: 'int': Variable wird auf booleschen Wert ('True
' oder 'False') gesetzt (Auswirkungen auf Leistungsverhalten möglich)
2>..\..\src\AlgebraicCore\IntOperations.C(586): warning C4800: 'int': Variable wird auf booleschen Wert ('True
' oder 'False') gesetzt (Auswirkungen auf Leistungsverhalten möglich)
2>..\..\src\AlgebraicCore\IntOperations.C(1069): warning C4800: 'int': Variable wird auf booleschen Wert ('Tru
e' oder 'False') gesetzt (Auswirkungen auf Leistungsverhalten möglich)
2>..\..\src\AlgebraicCore\IntOperations.C(1106): warning C4800: 'int': Variable wird auf booleschen Wert ('Tru
e' oder 'False') gesetzt (Auswirkungen auf Leistungsverhalten möglich)
2>..\..\src\AlgebraicCore\IntOperations.C(1117): warning C4800: 'int': Variable wird auf booleschen Wert ('Tru
e' oder 'False') gesetzt (Auswirkungen auf Leistungsverhalten möglich)
2>..\..\src\AlgebraicCore\NumTheory.C(522): warning C4800: 'int': Variable wird auf booleschen Wert ('True' od
er 'False') gesetzt (Auswirkungen auf Leistungsverhalten möglich)
2>..\..\src\AlgebraicCore\TmpGReductor.C(117): warning C4800: 'const CoCoA::GReductor::UseDynamicAlgFlag': Var
iable wird auf booleschen Wert ('True' oder 'False') gesetzt (Auswirkungen auf Leistungsverhalten möglich)
2>..\..\src\AlgebraicCore\TmpF5Mat.C(714): warning C4800: 'CoCoA::F5ns::mmvt_t *': Variable wird auf boolesche
n Wert ('True' oder 'False') gesetzt (Auswirkungen auf Leistungsverhalten möglich)

2>..\..\src\AlgebraicCore\ring.C(837): warning C4804: '<': unsichere Verwendung des Typs 'bool' in einer Opera
tion

2>  random.C
2>..\..\src\AlgebraicCore\random.C(170): warning C4351: Neues Verhalten: Die Elemente des Arrays "CoCoA::Rando
mSeqLong::myState" werden standardmäßig initialisiert.
2>..\..\src\AlgebraicCore\random.C(186): warning C4351: Neues Verhalten: Die Elemente des Arrays "CoCoA::Rando
mSeqLong::myState" werden standardmäßig initialisiert.
2>..\..\src\AlgebraicCore\random.C(256): warning C4351: Neues Verhalten: Die Elemente des Arrays "CoCoA::Rando
mSeqBool::myState" werden standardmäßig initialisiert.
2>..\..\src\AlgebraicCore\random.C(268): warning C4351: Neues Verhalten: Die Elemente des Arrays "CoCoA::Rando
mSeqBool::myState" werden standardmäßig initialisiert.
2>..\..\src\AlgebraicCore\random.C(348): warning C4351: Neues Verhalten: Die Elemente des Arrays "CoCoA::Rando
mSeqBigInt::myState" werden standardmäßig initialisiert.
2>..\..\src\AlgebraicCore\random.C(364): warning C4351: Neues Verhalten: Die Elemente des Arrays "CoCoA::Rando
mSeqBigInt::myState" werden standardmäßig initialisiert.
2>..\..\src\AlgebraicCore\random.C(380): warning C4351: Neues Verhalten: Die Elemente des Arrays "CoCoA::Rando
mSeqBigInt::myState" werden standardmäßig initialisiert.
2>..\..\src\AlgebraicCore\random.C(396): warning C4351: Neues Verhalten: Die Elemente des Arrays "CoCoA::Rando
mSeqBigInt::myState" werden standardmäßig initialisiert.
2>..\..\src\AlgebraicCore\random.C(413): warning C4351: Neues Verhalten: Die Elemente des Arrays "CoCoA::Rando
mSeqBigInt::myState" werden standardmäßig initialisiert.
```

#### #3 - 11 Feb 2013 12:01 - Christof Soeger

On my work laptop I have VS 2010. In addition to the already mentioned points I get an error:

```
2>..\..\src\AlgebraicCore\random.C(309): error C2440: '<function-style-cast>' : cannot convert from 'unsigned
long' to 'std::bitset<_Bits>'
2>          with [  _Bits=32 ]
2>          No constructor could take the source type, or constructor overload resolution was ambiguous
```

The source line:

```
    myBuffer = std::bitset<ourBufferBits>(gmp_urandomb_ui(myState, ourBufferBits));
```

As far as I can see the resolution of the right constructor fails because there is one for int and one for unsigned long long but the return type of gmp_urandomb_ui is unsigned long. This is a stupid error which even is not consistend with the standart. I put in a static cast to ull to prevent the error here.

The QCodeEdit project also fails with the same error message. Do I have to make more preparations? Bruno said: "You need to compile and configure QT for windows." This is not part of Bruno's file?

Please add #266 as related.

#### #4 - 11 Feb 2013 12:28 - John Abbott

Even Microsoft says that it should work with an unsigned long:
**http://msdn.microsoft.com/en-us/library/zfae7kt8%28v=vs.80%29.aspx**

JAA **WILL NOT FIX** the bitset problem; get Microsoft to fix their compiler!!!

I will not tolerate ugly source code to work around defective compilers.

#### #5 - 11 Feb 2013 12:31 - John Abbott

The source files RandomXYZ.C were removed from CVS some time (months) ago. Please update from the current CVS.

#### #6 - 11 Feb 2013 12:38 - John Abbott

Please disable your compiler's warning about automatic conversion from int to bool. Many basic functions "inherited" from C have return type int even though the value is actually a boolean; similar comments apply to GMP functions. C++ maintains the return type for backward compatibility.

While it would not be difficult to silence the warnings by inserting != 0 into the source code at appropriate places, such a change would reduce the legibility of that same source code. I might make some minor changes to the source code, but don't count on it.

Just disable the unhelpful warning!

#### #7 - 11 Feb 2013 12:39 - John Abbott

```
src\AlgebraicCore\ring.C(837): warning C4804:
```

OK, this was my fault.  I'll correct and check in shortly.

**12:49** checked in!

#### #8 - 11 Feb 2013 12:43 - John Abbott

```
2>..\..\src\AlgebraicCore\random.C(170): warning C4351: Neues Verhalten: Die Elemente des Arrays "CoCoA::Rando
mSeqLong::myState" werden standardmäßig initialisiert.
```

The source code looks OK to me; there is even a helpful comment!

I suggest you disable this warning (at least for the file random.C)

#### #9 - 11 Feb 2013 14:11 - Christof Soeger

*- % Done changed from 0 to 20*

I agree that most warnings are not important, I just wanted to give them to you, since they are valid sometimes. And in one case it was useful, the warning in ring.C disappeared.

The error with the bitset is indeed the fault of VS 2010, in version 2012 it does not appear. So "won't fix" is the right way to handle it.

I'm aware that the RandomXYZ.C files are removed. I only wanted to document that this change has also to be done in the visual studio project file.

#### #10 - 11 Feb 2013 14:14 - Anna Maria Bigatti

Christof Soeger wrote:

> I'm aware that the RandomXYZ.C files are removed. I only wanted to document that this change has also to be done in the visual studio project file.

ehmm.... should we do it?  which file is it?

#### #11 - 11 Feb 2013 14:54 - Christof Soeger

Anna Maria Bigatti wrote:

> Christof Soeger wrote:
>
> > I'm aware that the RandomXYZ.C files are removed. I only wanted to document that this change has also to be done in the visual studio project file.
>
> ehmm.... should we do it?  which file is it?

I did it. The change must be done in the visual studio project that is included in Bruno's .rar.

#### #12 - 11 Feb 2013 14:59 - Christof Soeger

Next I tried to build the console application. It does not need qt.

But here I get an error during the compilation of Main.C

```
3>  Main.C
3>C:\Normaliz\CoCoA\CoCoALib-0.99\MSVC10\..\include\CoCoA/FreeModule.H(45): error C2059: syntax error : '('
3>C:\Normaliz\CoCoA\CoCoALib-0.99\MSVC10\..\include\CoCoA/FreeModule.H(45): error C4430: missing type specifie
r - int assumed. Note: C++ does not support default-int
3>C:\Normaliz\CoCoA\CoCoALib-0.99\MSVC10\..\include\CoCoA/FreeModule.H(45): error C2059: syntax error : ')'
3>C:\Normaliz\CoCoA\CoCoALib-0.99\MSVC10\..\include\CoCoA/FreeModule.H(45): error C2143: syntax error : missin
g ';' before ')'
3>C:\Normaliz\CoCoA\CoCoALib-0.99\MSVC10\..\include\CoCoA/FreeModule.H(45): error C2334: unexpected token(s) p
receding ':'; skipping apparent function body
```

The corresponding source line is the constructor of FreeModule

```
explicit FreeModule(const FreeModuleBase* MPtr): FGModule(MPtr) {};
```

It looks like the compiler does not understand, that is a consturctor and expects an return type. I have no idea why that should happen. Also I guess that header is also included earlier, I do not understand why it should work earlier and not in the compilation of Main.C.

#### #13 - 11 Feb 2013 15:16 - John Abbott

Try separating the definition from the declaration:
inside the class definition put just

```
explicit FreeModule(const FreeModuleBase* MPtr);
```

After the class definition (but still in the header file) put

```
    inline FreeModule::FreeModule(const FreeModuleBase* MPtr):
      FGModule(MPtr)
    {}
```

Sometimes I wonder whether the non-compliance to the standard of the M$ compiler is a deliberate ploy to make life harder for open sourcers...

#### #14 - 11 Feb 2013 18:49 - Christof Soeger

*- % Done changed from 20 to 30*

John Abbott wrote:

> Try separating the definition from the declaration:

Tried, but no success :-(

With the intel compiler it's the same. I can compile CoCoAlib and also nmzIntegrate (using CoCoAlib) but the compilation of CoCoA-5 fails.

#### #15 - 12 Feb 2013 10:52 - John Abbott

That's more worrying that two compilers complain.

Despite comments in the file Main.C, as far as I can tell FreeModule.H is included via #include "CoCoA/TmpJBEnv.H" (which is in Interpreter.H). Why

should that file be included???  That Tmp prefix is there for a **good reason**!

It is **very strange** that the problem comes to light only when compiling Main.C.

#### #16 - 26 Feb 2013 17:17 - Christof Soeger

So now I tested to compile the CoCoA lib for x64 Windows with the Intel compiler in MSVS2010. I have to say MSVS is a pain...! Copying the settings does not copy all setting and so on.

Well in the compilation I get an error:

```
1>  ApproxPts.C
1>C:\Normaliz\CoCoA\CoCoALib-0.99\MSVC10\..\include\CoCoA/convert.H(104): error : more than one constructor ap
plies to convert from "const size_t={unsigned __int64}" to "CoCoA::MachineInt":
1>              function "CoCoA::MachineInt::MachineInt(signed char)"
1>              function "CoCoA::MachineInt::MachineInt(unsigned char)"
1>              function "CoCoA::MachineInt::MachineInt(short)"
1>              function "CoCoA::MachineInt::MachineInt(unsigned short)"
1>              function "CoCoA::MachineInt::MachineInt(int)"
1>              function "CoCoA::MachineInt::MachineInt(unsigned int)"
1>              function "CoCoA::MachineInt::MachineInt(long)"
1>              function "CoCoA::MachineInt::MachineInt(unsigned long)"

1>        NumericCast(const FromType& val) { if (!IsInRange(std::numeric_limits<ToType>::min(), val, std::nume
ric_limits<ToType>::max())) CoCoA_ERROR(ERR::BadConvert, "NumericCast"); myVal = static_cast<ToType>(val); /*
 static_cast will always succeed */}
1>                                                                                                      ^
1>            detected during:
1>              instantiation of "CoCoA::NumericCast<ToType>::NumericCast(const FromType &) [with ToType=long,
 FromType=size_t={unsigned __int64}]" at line 62 of "C:\Normaliz\CoCoA\CoCoALib-0.99\MSVC10\..\include\CoCoA/u
tils.H"
1>              instantiation of "long CoCoA::len(const T &) [with T=CoCoA::ApproxPts::PointDbl]" at line 76 o
f "..\..\src\AlgebraicCore\ApproxPts.C"
```

And the same problem in many other files. I'm not completly sure, but I guess you should add a MachineInt constructor from size_t or avoid that size_t is used.

The same happens here for operator<<

```
1>..\..\src\AlgebraicCore\ideal.C(391): error : more than one operator "<<" matches these operands:
1>              function "CoCoA::operator<<(CoCoA::OpenMathOutput &, int)"
1>              function "CoCoA::operator<<(CoCoA::OpenMathOutput &, unsigned int)"
1>              function "CoCoA::operator<<(CoCoA::OpenMathOutput &, long)"
1>              function "CoCoA::operator<<(CoCoA::OpenMathOutput &, unsigned long)"
1>              operand types are: CoCoA::OpenMathOutput << size_t
1>        OMOut << G.size();                  // Number of gens, should be an attribute???
```

#### #17 - 26 Feb 2013 18:17 - John Abbott

Curious!  We created MachineInt deliberately to "combine" all the various integral types into just one type which would preserve all information (*e.g.* not silently wrap negative values into large positive ones).

So what type is size_t that it does not match any of the ctors for MachineInt?

Perhaps it is unsigned long long int?
How to handle this portably?  The code must be correct whether size_t is one of the 8 standard integral types (from C++03) or not.  Templates would work, but they would also match too many other types...

It really seems that M$ employs abstruse interpretations of the standard just to make life hard for open sourcers.

Anyone got any good ideas for a workaround?  Ideally avoiding #ifdef.

#### #18 - 09 Apr 2013 16:01 - Christof Soeger

To the last problem for Windows 64:
size_t is there unsigned __int64, but long is still 32bit. I guess that is causing the problem.

#### #19 - 29 Apr 2014 17:32 - Christof Soeger

For the new normaliz version I did again a compilation on Windows. Here are my findings for CoCoALib 0.99533:

The problem on 64 bit Windows still exits, to summarize that point:
On 64bit Windows size_t is unsigned 64bit, but long is unsigned 32bit. To fix this problem one could never use size_t or implement function with size_t as argument. Maybe having also overloaded functions with parameter type unsigned long long could help.

For that reason I did no further tried with 64bit. On 32 bit I still had to fix 2 points:

First there were some calls of methods (namely sqrt and log) expecting float/double but the argument was an integer, resulting in an "ambiguous call" like error. I fixed it with a static cast to double. Here are the effected parts:

```
1>..\..\src\AlgebraicCore\SmallFpImpl.C(112): error : more than one instance of overloaded function "sqrt" mat
ches the argument list:
1>              function "sqrt(double)"
1>              function "sqrt(float)"
1>              function "sqrt(long double)"
1>              argument types are: (const CoCoA::SmallFpImpl::value_t)
1>        const long ans = PrevPrime(static_cast<long>(std::floor(sqrt(HalfMaxIntVal))));

1>..\..\src\AlgebraicCore\IntOperations.C(892): error : more than one instance of overloaded function "log" ma
tches the argument list:
1>        return n*log(n)-n + log(n*(1+n*(4.0+8*n)))/6 + LogSqrtPi;

and in ApproxPts.C (443) at the sqrt arg
   if (sqrt(len(NewPts)) > len(OrigPts))
```

I'm not sure why it is no problem on other systems.

The second point was:
Mathematical constants like M_PI and M_LN2seem to be not part of the c/c++ standard, but most compilers define them. For MS VS 2010 one has to define
#define _USE_MATH_DEFINES
to have them defined, see
http://msdn.microsoft.com/en-us/library/4hwaceh6.aspx

#### #20 - 30 Apr 2014 14:39 - John Abbott

*- % Done changed from 30 to 40*

I believe I have fixed the overloading problem, and replaced M_PI with 4*atan(1.0).

Right now I don't have a good idea for the 64-bit problem.  I'm very tempted to use silent casts from size_t to unsigned long; if something overflows the program will probably crash (but then a Microsoft user is used to that and will probably blame the silly OS rather than our code!)

#### #21 - 30 Apr 2014 18:38 - John Abbott

*- % Done changed from 40 to 50*

I have removed most uses of size_t in CoCoALib.

The template fn len now does an automatic cast to long from size_t; overflow checking by NumericCast happens only if CoCoA_DEBUG is set (hard luck if you try debugging on a perverse 64-bit Microsoft platform!)

#### #22 - 02 May 2014 11:21 - Christof Soeger

*- % Done changed from 50 to 40*

Thanks, that should make the compilation a lot easier! I can test it next week, is it CVSed?
Note that there is also an appearance of M_LN2 in TmpFactorDir/mpz_log.c.

EDIT:
I didn't want to reset the progress, it was preset and I did not check it, maybe because I started the update earlier.

#### #23 - 02 May 2014 14:48 - John Abbott

*- % Done changed from 40 to 50*

OK I have modified mpz_log.c; should be fine now.

Also I have CVSed everything.
@Christof: could you confirm that everything's fine now?  Thanks!

#### #24 - 05 May 2014 15:53 - Christof Soeger

One addition that I forgot to mention: I had to include

```
#include <iterator>
using std::back_inserter;
```

to fix

```
1>..\..\src\AlgebraicCore\FreeModule.C(960): error : identifier "back_inserter" is undefined
1>        transform(L.begin(), L.end(), back_inserter(sh),
```

One place where there is still a M_PI

```
1>..\..\src\AlgebraicCore\IntOperations.C(913): error : identifier "M_PI" is undefined
1>      const static double LogSqrtPi = log(M_PI)/2;
```

And in the cvs I did not recive the change for mpz_log.c

#### #25 - 05 May 2014 20:17 - John Abbott

Made the changes requested in previous comment.
No idea why you are not seeing the updated mpz_log.c...?

#### #26 - 06 May 2014 11:47 - John Abbott

Ahhh!  OK I have fixed mpz_log now; sorry, I had overlooked a M_LN2.

#### #27 - 26 Jun 2014 16:54 - Christof Soeger

*- % Done changed from 50 to 60*

I tried the newest cvs version and CoCoALib does compile in 32 and 64 bit. I only had to change what I have reported in Update 3, and this is a bug in microsoft's std::bitset implementation.

**NOTE (20140704)** the problem persists because Christof still have MSVC2010 on his work computer!

#### #28 - 15 Sep 2014 18:04 - Anna Maria Bigatti

*- Project changed from CoCoA to CoCoALib*

*- Category changed from Portability to Portability*

*- Target version deleted (CoCoA-5.0.3)*

#### #29 - 15 Sep 2014 18:04 - Anna Maria Bigatti

*- Target version set to CoCoALib-1.0*

#### #30 - 22 Mar 2024 21:01 - John Abbott

Is any of this still relevant?

#### #31 - 22 Mar 2024 23:13 - Nico Mexis

The problem with FreeModule still persists - however in different form.
Now, it just complains with an appropriate error message in MSVC and UCRT.
The error is "easily solvable" by specifying #undef FreeModule (since FreeModule is also a macro that is defined in the WIN32 API).
The rest does not seem to be relevant anymore and I have also fixed the rest in a patch I recently sent to JAA - depends of course on what he wants to cherry-pick or which issues are not deemed important enough to fix (or whose fix is just ugly/infeasible).