

CoCoA-5 - Feature #204

NmzComputation: the powerful NormalizComputation function in CoCoA-5

04 Jul 2012 14:25 - Anna Maria Bigatti

Status:	Closed	Start date:	04 Jul 2012
Priority:	Normal	Due date:	
Assignee:	Christof Soeger	% Done:	100%
Category:	CoCoA-5 function: new	Estimated time:	0.00 hour
Target version:	CoCoA-5.1.1 Seoul14	Spent time:	13.30 hours
Description			
Proposal for most powerful/flexible function to access Normaliz capabilities (specific functions like NormalizHilbertBasis should still be available)			
<pre>NormalizComputation(Cone, ["HilbertBasis", "HilbertPoly"]); NormalizComputation(Cone); -- computes everything</pre>			
where			
<pre>Cone := Record[Gens := [x^2*y, y^3*z^2]]; -- or Cone := Record[Inequalities := [...], Equalities := [...]];</pre>			
the return value would be a record like			
<pre>Record[HilbertBasis := ..., HilbertPoly := ...]</pre>			
The output will also include fields which have been computed internally (e.g. supporting hyperplanes). If the overhead is too big, then we might make a new function returning just what it was asked for (e.g. NormalizComputationMinimalOutput)			
Related issues:			
Related to CoCoALib - Feature #218: CoCoALib normaliz interface		Closed	03 Aug 2012

History

#1 - 05 Jul 2012 09:10 - Anna Maria Bigatti

- Category set to New Function
- % Done changed from 0 to 10

Now available in BuiltinFunctions.C an "empty" function

```
DECLARE_STD_BUILTIN_FUNCTION(NormalizComputation, 2) { ....
```

with arguments a record Cone and a list of strings L and return a record with fields the strings in L (set to BigInt(0)).

[the one argument variant will be done later]

#2 - 05 Jul 2012 11:43 - Anna Maria Bigatti

- Status changed from New to In Progress
- % Done changed from 10 to 20

Now the empty function reads and uses the input record as well:

```
/**/ NormalizComputation(Record[AnnaINT := 4, JohnRingElem := x], ["hallo", "ciao"]);
Record[ciao := 0, hallo := 0, recognizedA := 4, recognizedJ := x]
```

#3 - 05 Jul 2012 14:37 - Anna Maria Bigatti

- Assignee set to Anna Maria Bigatti
- % Done changed from 20 to 30

#4 - 05 Jul 2012 14:40 - Anna Maria Bigatti

(Added myFieldNamesStrings in Interpreter.[CH] for Record)

```
/**/ indent(NormalizComputation(Record[AnnaINT := 4, JohnRingElem := x], ["hallo"]));
Record[
  OneFieldWas := "AnnaINT",
  hallo := 0,
  recognizedA := 4,
  recognizedJ := x
]
```

#5 - 05 Jul 2012 17:33 - Anna Maria Bigatti

- Project changed from CoCoALib to CoCoA-5
- Category deleted (New Function)

#6 - 05 Jul 2012 17:33 - Anna Maria Bigatti

- Category set to CoCoA-5 function: new

#7 - 13 Jul 2012 14:22 - Christof Soeger

- Assignee changed from Anna Maria Bigatti to Christof Soeger
- % Done changed from 30 to 50

Now a cone is created from the input. The conversion of the input type string is forwarded to libnormaliz, currently these strings are recognised (same as in the Normaliz input files):

```
integral_closure, normalization, polytope, rees_algebra, hyperplanes, equations, congruences, lattice_ideal, g
rading
```

Examples:

```
Cone1 := Record[ integral_closure := Mat(ZZ, [[1,2],[2,1]]) ];
NC1 := NormalizComputation(Cone1, ["HilbertBasis", "HilbertPoly"]);
indent(NC1);

Cone2 := Record[ hyperplanes := Mat(ZZ, [[1,0,0],[0,1,1],[1,2,3]]), equations := Mat(ZZ, [[1,1,1]]) ];
NC2 := NormalizComputation(Cone2, ["HilbertBasis", "HilbertPoly"]);
indent(NC2);
-- invalid input, no such input type, exception is caught
NormalizComputation(Record[ int_closure := Mat(ZZ, [[1,2],[2,1]]) ], ["HilbertBasis"]);

-- invalid input, no matrix
-- this leads to an segfault when it is casted to a Matrix
C := Record[ integral_closure := [[1,2],[2,1]] ];
NormalizComputation(C, ["HilbertBasis"]);
```

TO DO:

Read what should be computed from the input.

Compute the requested properties.

Return the result.

Check whether the entires of the record are of the correct (CoCoA-)type. See last example.

#8 - 31 Jul 2012 16:07 - Anna Maria Bigatti

I've found an easier way to make a RecordValue (why didn't I think of it before?):
instead of making a new field using "new BlahValue(Blah)" use "Value::from(Blah)".

#9 - 01 Aug 2012 17:17 - Christof Soeger

- % Done changed from 50 to 60

previous message lost because redmine logged me out :(here again, but shorter

Value::from() is a big help! Now I can return results, example:

```

Cone1 := Record[ integral_closure := Mat(ZZ, [[1,2],[2,1]]) ];
NC1 := NormalizComputation(Cone1, ["HilbertBasis", "SupportHyperplanes"]);
indent(NC1);
Record[
  HilbertBasis := [[2, 1], [1, 2], [1, 1]],
  SupportHyperplanes := [[-1, 2], [2, -1]]
]

```

Only possibilities are HilbertBasis, SupportHyperplanes, Deg1Elements. More can be added when they are added to CoCoALib Still to do:

- compute everything that was asked in one computation, now it may include recomputations.
- return all computed data, not only the asked
- check the input (don't know how to do it, heeelp ;))

```

C := Record[ integral_closure := [[1,2],[2,1]] ];
NormalizComputation(C, ["HilbertBasis"]);

```

causes a failed assertion

```

C5: /usr/include/boost/smart_ptr/intrusive_ptr.hpp:166: T* boost::intrusive_ptr<T>::operator->() const [with T =
CoCoA::InterpreterNS::MatrixValue]: Assertion `px != 0' failed.

```

#10 - 05 Oct 2012 17:22 - Christof Soeger

- % Done changed from 60 to 70

```

C := Record[ integral_closure := [[1,2],[2,1]] ];
NmzComputation(C, ["HilbertBasis"]);

```

gives now an nice error

```

ERROR: Expecting type MAT, but found type LIST
NmzComputation(C, ["HilbertBasis"]);
      ^

```

#11 - 08 Oct 2012 12:07 - Christof Soeger

Further improvement of error message:

```
# NmzComputation(Record[ integral_closure := [[1,2],[2,1]] ],["HilbertBasis"]);
ERROR: Expecting type MAT as field "integral_closure" of the record, but found type LIST
NmzComputation(Record[ integral_closure := [[1,2],[2,1]] ],["HilbertBasis"]);
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

#12 - 28 Jan 2014 17:48 - Christof Soeger

Since the HilbertSeries is now available in CoCoALib, I also added it to NmzComputation. Example:

```
Cone1 := Record[ integral_closure := Mat([[1,2],[2,1]]) ];
NC1 := NmzComputation(Cone1, ["HilbertBasis", "SupportHyperplanes", "HilbertSeries"]);
indent(NC1);
```

will give

```
Record[
  HilbertBasis := [[1, 1], [1, 2], [2, 1]],
  HilbertSeries := Record[DenFactors := Record[RemainingFactor := 1, factors := [-t +1, -t^3 +1], multiplicities := [1, 1]], num := t^2 -t +1],
  SupportHyperplanes := [[-1, 2], [2, -1]]
]
```

Also: the function first look what was asked and then compute everything in one Normaliz computation.

edit: Anna pointed out that the ZZ at creating the Matrix is not needed.

#13 - 28 Jan 2014 18:17 - Anna Maria Bigatti

- Subject changed from NormalizComputation function to NmzComputation: the powerful NormalizComputation function in CoCoA-5

#14 - 29 Jan 2014 11:59 - Christof Soeger

And it also works with gradings:

```
Cone2 := Record[ integral_closure := Mat([[1,2],[2,1]]), grading := Mat([[2,1]])];
NC2 := NmzComputation(Cone2, ["HilbertBasis", "SupportHyperplanes", "HilbertSeries"]);
indent(NC2);

Record[
  HilbertBasis := [[1, 1], [1, 2], [2, 1]],
  HilbertSeries := Record[DenFactors := Record[RemainingFactor := 1, factors := [-t +1, -t^20 +1], multiplicities := [1, 1]], num := t^18 -t^17 +t^15 +t^10 -t^9 +t^8 +t^3 -t +1],
  SupportHyperplanes := [[-1, 2], [2, -1]]
]
```

#15 - 02 Apr 2014 17:33 - Anna Maria Bigatti

- Target version set to CoCoA-5.1.0 Easter14

#16 - 16 Apr 2014 08:29 - Anna Maria Bigatti

- Target version changed from CoCoA-5.1.0 Easter14 to CoCoA-5.1.1 Seoul14

The example in the description does not work.

Postponing this issue to Seoul version (with full manual to be published in proceedings ;-)

[BTW choose which session to apply to, and send abstract!]

#17 - 16 Apr 2014 11:25 - Christof Soeger

Which example do you mean? For me all the newer ones (from update 10 on) are working. The older should work too if you replace NormalizComputation by NmzComputation. I tested it with the latest CVS.

Maybe I have to give you an updated libnormaliz? I will email it.

#18 - 16 Apr 2014 14:11 - Christof Soeger

Some explanation on NmzIntegrate:

The possible input is basically the same as for Normaliz. This means the cone record can have as fields the names of the entries of libnormaliz::InputType (to be found in libnormaliz.h), i.e.

integral_closure,
polyhedron,
normalization,
polytope,
rees_algebra,
inequalities,
strict_inequalities,
signs,
strict_signs,
equations,
congruences,
inhom_inequalities,

dehomogenization,
inhom_equations,
inhom_congruences,
lattice_ideal,
grading,
excluded_faces

For an extended description of the input types see the Normaliz manual.

As compute options you can give any libnormaliz::ConeProperty.

The conversion of the strings to the enum types is done by libnormaliz functions. This ensures that the NmzComputation always includes the full range of functionality of libnormaliz.

BUT at from the ConeProperty s at the moment only

HilbertBasis

SupportHyperplanes

Deg1Elements

HilbertSeries

are working. The others do not have a matching return function. I will work on that. To do is:

- 1) Implement additional return functions in CoCoALib.
- 2) Let NmzComputation return everything that is computed (? Is this the desired behaviour? I think so.)
- 3) Implement NmzComputation(Cone). This is easy when the other two points are done.

One point why I did not work on 1) earlier was that I wonder if there is a more systematic way than make a copy&paste and replace just the name of the ConeProperty. But I think there is not.

#19 - 12 May 2014 15:55 - Christof Soeger

I have implemented point 1) (see [#218](#)) and now also point 2). I will send the code.

For 3) I have the problem that I do not know how to overload CoCoA5 functions with a different number of parameters. The naive way did not work.

#20 - 12 May 2014 16:21 - Anna Maria Bigatti

Christof Soeger wrote:

I have implemented point 1) (see [#218](#)) and now also point 2). I will send the code.

tested and cvs'ing now

For 3) I have the problem that I do not know how to overload CoCoA5 functions with a different number of parameters. The naive way did not work.

no, there's no naive method :-(

It is actually a bit fiddly. See for example the definition of **coefficients**.

```
// variable number of args
```

```

DECLARE_ARITYCHECK_FUNCTION(coefficients) { return (1<=nArg) && (nArg<=2); }
DECLARE_BUILTIN_FUNCTION(coefficients) { // AMB
    invocationExpression->checkNumberOfArgs(1,2); // variable number of args
    (...)
    if (invocationExpression->args.size()==1)
    (...)
}

```

You need to "declare" the number of arguments twice (I think once for the parser and once for the interpreter)

#21 - 12 May 2014 16:50 - Christof Soeger

Okay, that's not so difficult if you know how it works. I just had to remember also to remove the matching END_STD_BUILTIN_FUNCTION.

#22 - 02 Sep 2014 11:23 - John Abbott

@Christof: is this issue resolved/complete/closable???

#23 - 02 Sep 2014 11:55 - Christof Soeger

- % Done changed from 70 to 90

Almost, I just checked it and the HilbertQuasiPolynomial is not returned at the moment. What is missing is a Value::from() for the type QuasiPoly. If that is there, the attached file should complete the function.

EDIT: Attaching still fails, will mail it.

#24 - 02 Sep 2014 18:37 - Anna Maria Bigatti

added Value::from(QuasiPoly).

... keep in mind that indices in the LIST representing a QuasiPoly start from 1 in CoCoA-5 (and from 0 in C++/CoCoALib/libnormaliz)

#25 - 02 Sep 2014 23:34 - John Abbott

The "shifting" of the indices by 1 could be awkward. I can think of two ways to resolve the matter:

- put the **first** entry of the quasi-poly **last** in CoCoA-5
- make a quasi-poly become an "indexable object" (as used for "lists of indets" all with the same head)

#26 - 03 Sep 2014 07:55 - Anna Maria Bigatti

John Abbott wrote:

The "shifting" of the indices by 1 could be awkward. I can think of two ways to resolve the matter:

- put the **first** entry of the quasi-poly **last** in CoCoA-5
- make a quasi-poly become an "indexable object" (as used for "lists of indets" all with the same head)

We have lots of cases with awkward shifting (i.e. CoeffListWRT). Why should this be different from the others?
[in case we need to discuss about this topic, we'd better make a "indexing from 1 or from 0" issue]

#27 - 03 Sep 2014 10:32 - John Abbott

I think this is different because the mathematical structure is indexed by elements of $\mathbb{Z}/(n)$ rather than indices 1,2,3,...; so index 0 is also equivalent to index n, which is why I proposed simply moving the first entry to the last position.

@Christof: What do you think?

#28 - 03 Sep 2014 11:34 - Christof Soeger

I'm unsure. I think in any case we also need an evaluate function like in the library. We have to rewrite it anyway, so from that point it is not relevant. So I think it is about how it is printed. Take the example from the manual

```
Cone := Record[ integral_closure := Mat([[1,2],[2,1]]),  
               grading := Mat([[2,1]]);  
NC2 := NmzComputation(Cone, ["HilbertBasis", "SupportHyperplanes", "HilbertSeries"]);  
indent(NC2.HilbertQuasiPolynomial);
```

I think for printing it is better to start with the 0 component, even if it has index 1 in the list.

For the Evaluate function in C5 is there a convention how to do it?

#29 - 03 Sep 2014 15:04 - Christof Soeger

I implemented a NmzEvaluateHilbertQuasiPolynomial method. It is very close to the implementation for QuasiPoly in cocoalib. Now you can do

```
NmzEvaluateHilbertQuasiPolynomial(NC2.HilbertQuasiPolynomial,0);  
NmzEvaluateHilbertQuasiPolynomial(NC2.HilbertQuasiPolynomial,1);
```

and so on.

I think it we should stick with the order, then also the transfer forth and back is straight forward.

The name of the function is not yet optimal. Also, as QuasiPoly is a general cocoalib type and independent of libnormaliz, this function should be so. I will send the code, feel free to put it somewhere else under a different name! I will then update the documentation for NmzComputation.

EDIT:

For the name I suggest EvalQuasiPoly

#30 - 11 Sep 2014 17:39 - John Abbott

- Status changed from In Progress to Closed

- % Done changed from 90 to 100