CoCoALib - Design #805

New type for "constant" matrices?

13 Nov 2015 10:49 - John Abbott

Status:	Closed	**Start date:**	13 Nov 2015
Priority:	Urgent	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Data Structures	**Estimated time:**	7.51 hours
Target version:	CoCoALib-0.99550 spring 2017	**Spent time:**	7.25 hours

Description

I believe that implementing IdentityMat, StdDegRevLexMat and so on as objects of type ConstMatrixView is not correct.

If I am correct, then there is the question of the correct object type to use for such values.

Discuss; decide; implement.

Related issues:

Related to CoCoALib - Bug #804: ZeroMat and IdentityMat should produce a matr...	**Closed**	**12 Nov 2015**
Related to CoCoALib - Design #311: XelMat, StdDegRevLexMat, ... should be Mat...	**Closed**	**14 Feb 2013**
Related to CoCoALib - Design #592: Review design of ConstMatrixView	**Closed**	**17 Jul 2014**

History

#1 - 13 Nov 2015 11:19 - John Abbott

- Status changed from New to In Progress

- Assignee set to John Abbott

- % Done changed from 0 to 10

I believe that ConstMatrixView is not the right type, because a matrix "view" is a way of looking at another object so that it appears to be a matrix; but for these matrices there is no "other object". In particular, a ConstMatrixView is not the owner of the value (just the owner of that way of viewing the other object).

It could be argued that these "constant" matrices are just ways of viewing the empty set, so it does not matter that there is no other object. Technically this is correct, but I found it quite confusing yesterday (when I could not understand what was going on in the old code, which worked seemingly miraculously).

The new design I am considering is the following. There is a new class, called ConstMatrix which derives directly from ConstMatrixView in essentially the same way that matrix derives from MatrixView. The specific examples of constant matrices are then instantiable classes derived from ConstMatrix.

Note that this design does mean that the type matrix has no inheritance relationship with ConstMatrix; the nearest common ancestor is ConstMatrixView.

Having ConstMatrix derive from ConstMatrixView means that values of type ConstMatrix can be passed as args to functions expecting ConstMatrixView, but they cannot be passed as args of type MatrixView -- this could be a problem for BlockMat2x2?

#2 - 13 Nov 2015 13:45 - John Abbott

In principle the concrete classes for IdentityMat and so on could derive from matrix but then they have to impl lots of "useless" functions which must simply give errors for "forbidden" operations.

The advantage of this approach is that there is no need to create a new class (ConstMatrix), but a disadvantage is that the inherent restrictions of the constant matrices are invisible at compile time (and become visible at run-time only by exceptions being thrown).

#3 - 13 Nov 2015 14:51 - John Abbott

As predicted in comment 1, there is a problem with BlockMat2x2. Not yet sure how to deal with it :-/

#4 - 24 Nov 2015 21:55 - John Abbott

- % Done changed from 10 to 20

I have found a reasonable way to circumvent the problem of passing a ZeroMat (of type ConstMatrix) to BlockMat2x2.

An unreasonable solution would be to offer 16 signatures for BlockMat2x2; I discard this.

Since the only real problematic case (so far, anyway) is passing a zero matrix, I have invented a new enum with a single element **zeroes**. It functions as a zero matrix of "flexible" dimension, so it can choose the right size to allow BlockMat2x2 to succeed. This does require adding some new signatures to BlockMat2x2: so far just to cover the cases needed by ConcatDiag and ConcatAntiDiag. In principle another four signatures could be needed if we want to allow just a single block of zeroes.

Internally, the zeroes placeholder is replaced by a private impl of ZeroMat (with flexible dimensions). There are actually two private ZeroMat impls: one which is properly const and the other which is not (but no entry is writable). So that the existing data structure employing matrix views can continue to function, I have added a new field containing a "flexible, private zero mat" (even when it is not needed).

One nice feature of **zeroes** is that the caller does not need to specify the correct dimensions.

A not-so-nice feature is that it is a bit of a "dirty hack" -- but it does allow a convenient user interface.

What do you think?

#5 - 23 Mar 2016 15:14 - John Abbott

- Priority changed from High to Urgent

- Target version changed from CoCoALib-0.99540 Feb 2016 to CoCoALib-0.99550 spring 2017

#6 - 22 Sep 2016 16:35 - John Abbott

- Status changed from In Progress to Resolved

- % Done changed from 20 to 80

JAA continues to like the idea of ConstMatrix. A nice advantage compared to ConstMatrixView is that an object of type ConstMatrix clearly/explicitly does not refer to any other object (except its ring, of course). This should make it easier for someone reading code to convince themselves of the correctness.

I'll now check the state of the documentation... let me guess: there isn't any? :-/

#7 - 22 Sep 2016 18:24 - John Abbott

- Status changed from Resolved to Feedback

- % Done changed from 80 to 90

I've written some hasty documentation... it's better than nothing.

#8 - 29 Sep 2016 16:04 - John Abbott

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

#9 - 28 Apr 2017 09:30 - Anna Maria Bigatti

- Estimated time set to 7.51 h