CoCoALib - Design #311

XelMat, StdDegRevLexMat, ... should be MatrixView

14 Feb 2013 11:57 - Anna Maria Bigatti

Status:	Closed	**Start date:**	14 Feb 2013
Priority:	Immediate	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	New Function	**Estimated time:**	5.50 hours
Target version:	CoCoALib-0.99550 spring 2017	**Spent time:**	4.45 hours

Description

Now we have
NewDenseMatXel(long n);
NewDenseMatRevLex(long n);
NewDenseMatStdDegRevLex(long n);
NewDenseMatStdDegLex(long n);
returning a (dense)matrix. They should return a MatrixView instead.

Related issues:

Related to CoCoA-5 - Feature #78: Elim ordering and matrix ordering in CoCoA-5	**In Progress**	**16 Jan 2012**
Related to CoCoALib - Support #65: matrix - MatrixView	**Closed**	**15 Dec 2011**
Related to CoCoALib - Design #592: Review design of ConstMatrixView	**Closed**	**17 Jul 2014**
Related to CoCoALib - Feature #803: PPOrdering: use it to compute WDeg?	**In Progress**	**11 Nov 2015**
Related to CoCoALib - Bug #804: ZeroMat and IdentityMat should produce a matr...	**Closed**	**12 Nov 2015**
Related to CoCoALib - Design #805: New type for "constant" matrices?	**Closed**	**13 Nov 2015**
Related to CoCoALib - Design #824: Fn names: LexMat or MatLex; StdDegRevLexMa...	**Closed**	**26 Nov 2015**

History

#1 - 29 Oct 2013 15:19 - Anna Maria Bigatti

- Target version changed from CoCoALib-0.99534 Seoul14 to CoCoALib-0.99532

#2 - 02 Apr 2014 07:42 - Anna Maria Bigatti

- Category set to New Function

- Target version changed from CoCoALib-0.99532 to CoCoALib-0.99533 Easter14

#3 - 07 Apr 2014 18:18 - John Abbott

- Target version changed from CoCoALib-0.99533 Easter14 to CoCoALib-0.99534 Seoul14

#4 - 17 Jul 2014 17:56 - John Abbott

- Status changed from New to In Progress

- Target version changed from CoCoALib-0.99534 Seoul14 to CoCoALib-1.0

- % Done changed from 0 to 10

- Estimated time set to 5.50 h

I have cobbled together a quick impl (practically copied from IdentityMatImpl).
It compiles, but I have commented it out for several reasons:

1. it is not important to do this for Seoul (and probably unwise to rush it)
2. there are aspects of the design which do not convince me:
   - need to add a friend to ConstMatrixView (in matrix.H)
   - need to define far too many fns (for just a const matrix view!!)

Before we do this, we should revise the design of ConstMatrixView.

NOTE 2015-11-11 OK why didn't I say **where** the prototype is??? Perhaps because it's in the file **MatrixForOrdering**?

#5 - 11 Nov 2015 17:30 - John Abbott

- Priority changed from Normal to Urgent

- Target version changed from CoCoALib-1.0 to CoCoALib-0.99540 Feb 2016

JAA is convinced that this is a good idea.

To get it finished quickly I shall ignore issue #592; it probably should not slow things down to ignore it (as I expect most of the coding will be cut-and-pasting).

I do have some doubts about how long the file will turn out to be; maybe I'll split it later on.

#6 - 11 Nov 2015 18:19 - John Abbott

- Assignee set to John Abbott

- % Done changed from 10 to 20

After some hasty cut-and-paste work I have some code which compiles :-)
I'm not so foolish as to try running it... that's for tomorrow :-/

#7 - 13 Nov 2015 13:01 - John Abbott

What do we want the StdDegRevLexMat to be?
There are two obvious candidiates: here are 3x3 instances

```
1  1  1
0  0 -1
0 -1  0
```

or

```
1 1 1
1 1 0
1 0 0
```

They clearly give the same ordering. The first matrix has the advantage of being sparser (*i.e.* more entries are zero), while the second matrix has all entries non-negative (indeed they are either 0 or 1).

My new implementation gives the second matrix (because it was easier for me to implement), whereas the previous impl gave the first matrix.

Is there any good reason to prefer one over the other?

P.S. the change of matrix means some tests fail -- a nuisance, but probably a good sign that the tests are not too trivial :-)

#8 - 14 Nov 2015 14:50 - John Abbott

I've implemented the other defn of StdDegRevLexMat; we can decide later which impl will remain "active".

#9 - 23 Nov 2015 13:08 - John Abbott

I notice only now that these pseudo-ctors take only the dimension as arg; **what should the ring of the matrix be?**

I am most tempted to say **ZZ** because all entries are integer, and moreover these matrices are "used" to compute WDeg values which must have integer components (in CoCoALib).

A similar question arises for IdentityMat and ZeroMat; currently their pseudo-ctors require the ring as arg, but why? (if we accept that the pseudo-ctors for StdDegRevLexMat and so on do not need the ring)

#10 - 23 Mar 2016 15:27 - John Abbott

- Priority changed from Urgent to Immediate

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

- % Done changed from 20 to 50

We must settle the last questions, and then close.

#11 - 27 Oct 2016 11:22 - John Abbott

- Status changed from In Progress to Feedback

- % Done changed from 50 to 90

It make sense for the ordering matrices to be over ZZ; for IdentityMat it is useful to be able to specify the ring. The function LexMat simply creates an IdentityMat over RingZZ. So I am happy now to dismiss the worries expressed in comment 9.

#12 - 20 Jan 2017 15:16 - John Abbott

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

I have done some cleaning in MatrixForOrdering.C, but more still really ought to be done.

After 3 months in feedback without problems, I am now closing.