CoCoALib - Design #806

AssignZero for matrix

14 Nov 2015 20:08 - John Abbott

Status:	Closed	**Start date:**	14 Nov 2015
Priority:	Low	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Safety	**Estimated time:**	1.10 hour
Target version:	CoCoALib-0.99540 Feb 2016	**Spent time:**	1.10 hour

Description

There is a procedure AssignZero for a matrix; it calls the mem fn myAssignZero.

What should these fns do?

History

#1 - 14 Nov 2015 20:16 - John Abbott

- Status changed from New to In Progress

- Priority changed from Normal to Low

- % Done changed from 0 to 10

Before spending too much time deciding the exact semantics, we should also decide if the procedure is even useful -- when could it be useful?

For a normal matrix I think the meaning is fairly clear: the matrix size stays the same, and all entries are set to zero (of the ring of the matrix).

It is less clear to me what should happen to a "special" matrix, for instance a DiagMatrix. I see two obvious choices:

- **(A)** set all the diag entries to zero (the rest are already zero)
- **(B)** give error

The problem with approach **(A)** is that for a DiagMatrix called DM the call AssignZero(DM) is **not equivalent** to

```
for (int i=0; i < NumRows(DM); ++i)
  for (int j=0; j < NumCols(DM); ++j)
    DM.mySetEntry(i,j,0);
```

because the explicit loop will give an error (provided DM has dimension at least 2x2).

If we adopt approach **(A)**, what should happen when attempting to AssignZero an object of type ZeroMatrix? Presumably nothing will happen, but perhaps an error would be more appropriate?

#2 - 14 Nov 2015 20:24 - John Abbott

In some ways it would be nice to have a default impl of AssignZero; presumably as the mem fn MatrixView::myAssignZero. The most obvious default impl is the double loop in comment 1, and this would give error both for a DiagMat and a ZeroMat if they do not have their own "private" impls of the mem fn myAssignZero.

An alternative could be the following "intelligent" loop:

```
for (int i=0; i < NumRows(DM); ++i)
  for (int j=0; j < NumCols(DM); ++j)
  {
    if (IsZero(DM(i,j))) continue; // regardless of whether writable!!
    if (!DM.IsWritable(i,j))  CoCoA_ERROR(...);
    DM.mySetEntry(i,j,0);
  }
```

If we accept the "intelligent" loop definition, what should happen if someone tries Id.mySetEntry(0,0,1); where Id is an identity matrix? In other words, if we have a const matrix, is it then OK to set its entries to the values they already have? The "intelligent" loop sets a worrying precedent since it does this for entries containing 0.

Comments? Opinions?

NOTE (2015-11-26) corrected sample impl above!

#3 - 26 Nov 2015 18:17 - Anna Maria Bigatti

John Abbott wrote:

> In some ways it would be nice to have a default impl of AssignZero; presumably as the mem fn MatrixView::myAssignZero. The most obvious default impl is the double loop in comment 1, and this would give error both for a DiagMat and a ZeroMat if they do not have their own "private" impls of the mem fn myAssignZero.

The "intelligent loop" could become quite tricky and I'm not sure we should make a default implementation...
I think we could have the "trivial loop" as a "generic implementation" which could be called by the concrete classes where it makes sense.

#4 - 01 Dec 2015 11:01 - John Abbott

- Status changed from In Progress to Resolved

- Assignee set to John Abbott

- % Done changed from 10 to 80

Since the fn AssignZero (for matrices) was no longer used anywhere, and especially since its semantics were rather unclear (with no obvious clarification), I have removed it.

Removed the doc; checked in.

#5 - 23 Mar 2016 15:31 - John Abbott

- Status changed from Resolved to Closed

- % Done changed from 80 to 100

- Estimated time set to 1.10 h