CoCoA-5 - Feature #18

Printing matrices: I/O unified style for CoCoA-5?

02 Nov 2011 15:58 - Anna Maria Bigatti

Status:	Closed	**Start date:**	02 Nov 2011
Priority:	Urgent	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Cleaning	**Estimated time:**	7.50 hours
Target version:	CoCoA-5.1.0 Easter14	**Spent time:**	7.80 hours

Description

Currently matrices are printed by CoCoALib like this:

```
matrix(QQ)(3, 3)
[
 [1, 0, 0],
 [0, 1, 0],
 [0, 0, 1]
]
```

But this is tedious to be used as CoCoA-5 input (unless we define matrix in CoCoA-5 as a function returning a function returning ad indexable object, but this is quite unnatural)
Some options:

```
matrix // QQ (3, 3)
([
 [1, 0, 0],
 [0, 1, 0],
 [0, 0, 1]
])
```

or

```
matrix(QQ, 3, 3,
[
 [1, 0, 0],
 [0, 1, 0],
 [0, 0, 1]
])
```

There are also these two CoCoA-5 functions: MakeMatByRows and MakeMatByCols

```
MakeMatByRows(2, 10, 1..20);
```

defined in packages/matrix.cpkg5. Should we unify syntax/meaning?

Related issues:

Related to CoCoA-5 - Feature #274: InputForm for output readable as input	**In Progress**	**20 Jun 2012**
Related to CoCoA-5 - Design #483: Unique copies of rings in CoCoA-5	**New**	**19 Mar 2014**
Related to CoCoA-5 - Support #548: Printing rings with ID	**Closed**	**06 May 2014**

History

#1 - 07 Nov 2011 14:49 - John Abbott

- Status changed from New to In Progress

- Estimated time set to 5.00 h

There is a (mild?)problem with Anna's first suggestion:

```
matrix // QQ 3x3
([[1,0,0],
  [0,1,0],
  [0,0,1]])
```

If the above is read as input then the resulting matrix will be over ZZ and not over QQ.

Perhaps this is not really a problem, but I think it should be noted.

AMB: this might be solved if we allow automatic embeddings (https://130.251.60.30/redmine/issues/7). Probably there is no way to make the output safely readable with the original meaning (see above), so explicitely avoiding the ring name would help making it more evident.

#2 - 07 Nov 2011 14:57 - John Abbott

Anna omitted to mention explicitly that it is deemed desirable that the printed form of a matrix should also be valid as input (or as nearly as is possible). This would certainly make cut-and-paste use easier.

In fact, in CoCoA-4 there was the general rule that printed output should be valid as input (with strings being a highly visible exception). JAA agrees that a similar generale rule should apply to CoCoA-5 with the proviso that printed form must be natural and easy to read (so naturalness is more important than validity as input).

JAA: we assume that printing of matrices (or any value) should produce essentially the same output whether done inside CoCoA-5 or inside CoCoALib

#3 - 07 Nov 2011 16:04 - John Abbott

There are three components to consider when printing out a matrix:

1. the ring containing the entries
2. the dimensions of the matrix (rows,cols)
3. the entries of the matrix

Which of these components should be included in the printed form? And why?

JAA thinks that the entries must be included.
JAA thinks that the matrix dimensions are redundant information (if the entries are printed out as a rectangular array).
JAA thinks that printing out the ring is often redundant (because the user already knows to which ring they belong), and could be "misleading" if the printed form is cut-and-pasted as input to a C5 session (or perhaps it gives some extra safety?). Consider the example below:

```
R ::= QQ[x,y,z];
Use S ::= ZZ/(7)[x,y];
M := matrix(R,[[1,x],[1,y]]); // cut-and-pasted
```

The creation of the matrix M will not work as one might reasonably expect (i.e. creating a matrix whose entries are in R): it will produce an error.

#4 - 08 Nov 2011 12:34 - John Abbott

It seems that both AMB and JAA are leaning towards the idea of not printing out the ring when printing a matrix. JAA thinks it would be too easy for a (new) user to believe that the presence of the ring would guarantee creation of the matrix over that ring [but this is not what the current evaluation rules would do].

Printing just the matrix entries would follow the KISS philosophy, and would maintain backward compatibility with C4 (if we use the tag **Mat** rather than the tag **matrix**).

#5 - 08 Nov 2011 12:38 - John Abbott

Which tag should be used when matrices are printed?

1. **Mat** is backward compatible with C4
2. **matrix** is better aligned with current CoCoALib naming convention

If we want the printed form to be readable as input then the tag must be the name of a matrix constructor.

JAA using **matrix** is only mildly backward-incompatible as it is pretty simple changing **matrix** into **Mat**.

We could also consider the tag **Matrix**, so backward compatibility could be achieved simply by defining a C4 function called **Matrix**

#6 - 09 Nov 2011 11:36 - John Abbott

Given that JAA and AMB are both largely convinced that it is best to print out just the matrix elements, AMB made a verbal proposal similar to her first proposal: here is an example to illustrate her idea

```
matrix(/*QQ 2x2*/ [[1, 2], [3, 4]])
```

The idea was simply to print the ring (and perhaps also the dimensions) inside a /*...*/ comment, so that this part is clearly skipped when reading the print-out as input. After a brief discussion of this idea, it was felt that it is better not to include the ring value in the printed form for two reasons:

1. the printed form of a ring can be long and hard to read, making it quite difficult for the user to read the actual value of the matrix
2. if the user really wants to see the ring, then he may also easily print it out by an explicit command (*e.g.* **PrintLn RingOf(M)**)

#7 - 09 Nov 2011 11:45 - John Abbott

Anna's suggestion could be made more workable if the ring/dimensions were printed in a comment after the matrix value:

```
matrix([[1, 2], [3, 4]])
/* 2x2 matrix over QQ */
```

I have deliberately placed the comment with dimensions and ring on the line after the matrix entries. Since matrices are currently always printed on a single (possibly very long) line, it seems more sensible to move the comment to the next line as it might otherwise be lost from view.

JAA thinks it may be interesting to offer a function which produces a value containing the information about the type of a value (*e.g.* think of the **Shape** function in C4). Here's a hasty example:

```
prompt# Fulltype(M);
Record[NumCols := 2, NumRows := 2, ring := QQ, type := MATRIX]
```

#8 - 09 Nov 2011 11:55 - Anna Maria Bigatti

- % Done changed from 0 to 50

#9 - 09 Nov 2011 12:33 - Anna Maria Bigatti

- Project changed from CoCoA to CoCoALib

#10 - 09 Nov 2011 12:34 - Anna Maria Bigatti

- Project changed from CoCoALib to CoCoA

#11 - 09 Nov 2011 15:20 - Anna Maria Bigatti

- Assignee set to John Abbott

#12 - 30 Nov 2011 09:06 - Anna Maria Bigatti

- Category set to Philosophy

- % Done changed from 50 to 90

#13 - 26 Mar 2012 17:42 - Anna Maria Bigatti

after using the "old style" for a while I'm getting more and more convinced that a matrix should also print its own ring, this is my new proposal

```
mat( /* QQ */ [
  [1, 0, 1],
  [0, 0, 1]
])
```

Someone might say that then we should do the same for ideals, but ideals are more likely to be more obviously in "the current" polynomial ring, whereas matrices with integer coefficients are far more common (and not yet automatically converted into "the current" polynomial ring).

#14 - 27 Mar 2012 10:04 - John Abbott

I do not like the "abandoned" square bracket at the end of the first line.

After speaking yesterday, we also concluded that we would use matrix rather than just mat (through the abbreviated form is fine in compound names). Here is my suggestion:

```
matrix( /* QQ */
 [[1,2,3],
  [4,5,6]
 ])
```

Generally I prefer that matching brackets be aligned vertically or horizontally -- it seems difficult to achieve this without wasting several lines of vertical space.

I note that the original proposal use the comment-to-end-of-line but Anna's latest proposal uses /* ... */ comments. JAA thinks this second proposal is probably a better choice (especially considering the cut-and-paste bug in the GUI related to end-of-line comments).

#15 - 27 Mar 2012 13:28 - Anna Maria Bigatti

John Abbott wrote:

> I do not like the "abandoned" square bracket at the end of the first line.
>
> Generally I prefer that matching brackets be aligned vertically or horizontally -- it seems difficult to achieve this without wasting several lines of vertical space.
>
> I note that the original proposal use the comment-to-end-of-line but Anna's latest proposal uses /* ... */ comments. JAA thinks this second proposal is probably a better choice (especially considering the cut-and-paste bug in the GUI related to end-of-line comments).

JAA:

```
matrix( /* QQ */
 [[1,2,3],
  [4,5,6]
 ])
```

I still prefer the abandoned parenthesis, but I make another suggestion similar to JAA
(a matrix-row line ends with a "" or with a "]": for those like me who permute rows and forget to clean-up the ",")

```
matrix( /* QQ */
 [[1,2,3],
  [4,5,6]]
 )
```

#16 - 28 Mar 2013 16:07 - John Abbott

JAA has just changed slightly the way matrices are printed: removed two newlines.
The newly changed code would produce something like this:

```
matrix( /*QQ*/
 [[1,2,3],
  [4,5,6]])
```

with no newline immediately after the matrix.

It really seems difficult to produce a uniform style which does not look wrong in some context. Perhaps we should **use different styles** if a matrix is printed alone or if it is printed as part some other structure (such as a RECORD)??

The problem arose because the new PreprocessPts commands produce a RECORD containing a matrix, and the printing looked horribly wrong!

#17 - 19 Mar 2014 16:38 - Anna Maria Bigatti

- Target version set to CoCoA-5.1.0 Easter14

Having just recently discovered that commenting out the printed matrix with

```
/* matrix(...) */
```

would not work as expected (nested comments have never been allowed in cocoa-5, like in C and C++), we now write explicitly ZZ, QQ, RingQQt(n) when suitable.
Please not that these are the only rings in cocoa-5 which are **unique**, whereas NewPolyRing, or QQ[x,y,z], or other ring constructors produce a new separate copy of the ring for each call.

Ideally it would be nice to write **any ring** explicitly, but that is related to solving the *unique copy* problem for any ring in cocoa-5 (and CoCoALib??). We had long discussed about this and we are not sure it is possible. Surely it is not easy at all.

NOTE 20140319 to avoid the problem with nested /*...*/ comments we could print the ring description as a string literal (it's just as useful/useless as an inline comment!)

#18 - 19 Mar 2014 21:07 - John Abbott

A significant problem is that the printed form is produced by CoCoALib which naturally knows nothing about variable values etc. in CoCoA-5.

We may have to settle for a partial/compromise solution (that ideally works in most usual circumstances).

#19 - 10 Apr 2014 18:14 - Anna Maria Bigatti

- Project changed from CoCoA to CoCoA-5

- Category deleted (Philosophy)

#20 - 10 Apr 2014 18:17 - Anna Maria Bigatti

now it prints the ring itself (when ring is uniquely defined)

```
matrix(QQ,
 [[1, 1, 1],
  [1, 0, 0],
  [0, 1, 0]])
```

#21 - 07 May 2014 14:10 - John Abbott

- Category set to Cleaning

- Status changed from In Progress to Closed

- % Done changed from 90 to 100

- Estimated time changed from 5.00 h to 7.50 h

Satisfactory; may have to be reconsidered if/when the printing of rings is reimplemented (see #548)