CoCoA-5 - Feature #1243

New function: Read a string into a list (of RingElem) -- CoCoA-5

13 Feb 2019 15:55 - John Abbott

Status:	Closed	**Start date:**	13 Feb 2019
Priority:	Normal	**Due date:**	
Assignee:	Anna Maria Bigatti	**% Done:**	100%
Category:	CoCoA-5 function: new	**Estimated time:**	5.00 hours
Target version:	CoCoA-5.3.0	**Spent time:**	4.90 hours

Description

Ange asked whether it would be possible to have a function like ReadExpr but which reads a list of ringelem (or a list of lists of ringelem).

Is this a good idea? Is it feasible?

Discuss, and possibly implement.

2020-02 The function is called **RingElems**.
Example: RingElems(R, "x-1, y^3-2, x*z")

Related issues:

Related to CoCoALib - Feature #1332: New function: vector of RingElem from st...	**Closed**	**08 Oct 2019**
Related to CoCoALib - Slug #1238: ReadExpr is too slow on long lists of mono...	**Closed**	**21 Jan 2019**
Related to CoCoALib - Design #1391: RingElems: syntax with [and] ?	**Closed**	**10 Jan 2020**

History

#1 - 13 Feb 2019 15:58 - John Abbott

Presumably the list syntax should be the same as what CoCoA prints: ","-separated entries inside square brackets [].
An empty list is allowed.

Ange has a workaround: she has changed the input so that it is simply a succession of polynomials (one per line), with a zero polynomial to mark the end of the list.

#2 - 15 Feb 2019 11:54 - Anna Maria Bigatti

I admit I would have found this useful in a number of occasions.
I had followed the same workaround as Ange...

#3 - 15 Feb 2019 12:10 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

Here are some thoughts:

1. in the current (prototype) implementation the natural way to read inputs is line-by-line; this could be awkward if the input is split over several lines?
2. a nice "modular" design would be to have a function which reads just strings, and then another which takes a string and splits it into pieces; **BUT** this means that the function which splits the string into pieces needs to know how to "parse" the string (and this could be tricky). Consider this example **[x+y, a[1,2], [[[a[2,1]]]]]**
3. a special function which reads a list-of-ringelem should solve the parsing problem (but it is rather an *ad hoc* approach); would we also want a function which reads list-of-list-of-ringelem?
4. is it reasonable to force external programs to use the CoCoA-5 syntax for lists?
5. what happens if the sender forgets the close square-bracket at the end of the list? The receiver would keep on waiting for more input. This seems to be a problem whatever system we adopt: how to handle truncated input?
6. It might be nice for the receiver to read first the number of values in the list, and then read the values one-by-one...?

#4 - 15 Feb 2019 13:33 - John Abbott

After lunch (rather disappointing fish-and-chips in the mensa), I now think that trying to impose a "list syntax" is **not a good idea.** The main reason is that we must write a parser (how?), and we must handle syntax errors (tedious).

It should be easy to write a function for reading a list, perhaps something like this:

```
define ReadListOfRingElem(Istream, R, EndMarker)
  ans := [];
  repeat
    str := GetLine(Istream);
    if str = EndMarker then return ans; endif;
    append(ref ans, RingElem(R, str));
  endrepeat;
enddefine;
```

Maybe we could make the string **"00"** a default end marker? Does it even make sense to have a default?

#5 - 15 Feb 2019 14:28 - Anna Maria Bigatti

- Assignee set to Anna Maria Bigatti

- % Done changed from 10 to 30

John Abbott wrote:

> After lunch (rather disappointing fish-and-chips in the mensa), I now think that trying to impose a "list syntax" is **not a good idea.** The main reason is that we must write a parser (how?), and we must handle syntax errors (tedious).

True.
I was thinking of just [, , ,] (in CoCoALib)
but indeed, that's very tedious.

In CoCoALib we also have ReadExprSemicolon which we may export to CoCoA.
And I think a function to read a sequence of polys would be very useful in **CoCoALib** too.
So I should do this in CoCoALib (RingElemInput.C).

> It should be easy to write a function for reading a list, perhaps something like this:
> [...]

I'll think about this.

> Maybe we could make the string **"00"** a default end marker? Does it even make sense to have a default?

Better no default, I think. Might induce in errors.

#6 - 15 Feb 2019 16:07 - John Abbott

At the moment, input is only via **GetLine**, which naturally reads a whole line as a string.

Currently we cannot easily handle the case where a single ringlem is spread over several lines, or if there are several ringelems in one line.
Consider these examples:
How many ringelems are here?

```
x+y
-z
```

How many ringelems are here?

```
x -y -z
```

#7 - 21 Feb 2019 09:57 - John Abbott

I suggest keeping the idea of ReadExprSemicolon in reserve, and adopting a KISS approach now:

- **KISS** the ringelems must be written exactly one per line (effectively "newline" functions as "semicolon")
- the advantage of ReadExprSemicolon is that lines and ringelems are essentially independent (one ringelem may span several lines, and one line may contain (parts of) several ringelems)
- if we do choose to use ReadExprSemicolon, I would suggest changing its name and allowing the user to specify what terminating character to expect (with some restrictons, probably)

#8 - 09 Oct 2019 15:55 - John Abbott

- Related to Feature #1332: New function: vector of RingElem from string -- in CoCoALib added

#9 - 09 Oct 2019 17:48 - Anna Maria Bigatti

- Target version changed from CoCoA-5.?.? to CoCoA-5.3.0

- % Done changed from 30 to 50

I think I've done it. ... after lots of tedious little mistakes!

Called ReadExprVector(ring, string) in RingElemInput.[CH]
Now testing and applying where useful for CoCoA-5, then we can think of a good name...

#10 - 09 Oct 2019 17:48 - Anna Maria Bigatti

- Related to Feature #1330: New syntax for NewQuotientRing added

#11 - 09 Oct 2019 17:49 - Anna Maria Bigatti

- Related to Feature #1329: New syntax for creating homomorphisms (PolyAlgebraHom) added

#12 - 09 Oct 2019 17:50 - Anna Maria Bigatti

- Subject changed from Read a string into a list (of RingElem) to New function: Read a string into a list (of RingElem) -- CoCoA-5

#13 - 09 Oct 2019 17:50 - Anna Maria Bigatti

- Related to deleted (Feature #1330: New syntax for NewQuotientRing)

#14 - 09 Oct 2019 17:51 - Anna Maria Bigatti

- Related to deleted (Feature #1329: New syntax for creating homomorphisms (PolyAlgebraHom))

#15 - 09 Oct 2019 20:27 - John Abbott

- Related to Slug #1238: ReadExpr is too slow on long lists of monomial with many indets: ---> use RingElems instead added

#16 - 10 Oct 2019 08:41 - Anna Maria Bigatti

For the moment called (also in CoCoA) ReadExprVector(ring, string).
Example ReadExprVector(R, "x-1, y^3-2, x*z")
Think of a good name fast!

#17 - 10 Oct 2019 09:57 - Anna Maria Bigatti

reminder: choose name and write doc for cocoa-5

#18 - 10 Oct 2019 14:54 - Anna Maria Bigatti

- Status changed from In Progress to Resolved

- % Done changed from 50 to 60

what about RingElems(ring, string)?
Similar to RingElem(ring, string)... or too similar?

One thing I do not like of BlaBlaVector is that in CoCoA-5 it should be BlaBlaList.

#19 - 10 Oct 2019 17:27 - John Abbott

One "advantage" of BlaBlaVector and BlaBlaList is that it help reminds the programmer that indices in the first case start from 0, and in the second case start from 1. Of course, the parts of code which much be changed to get the indices right are almost certainly not the lines in which BlaBlaVector appears... :-/

It could also be that the interface in C++ is a bit different. For example, it may be more convenient in C++ to read a succession of values from a stringstream rather than read them all in one go into a vector (not too sure about this, but it should be considered).

#20 - 13 Jan 2020 16:32 - Anna Maria Bigatti

- Related to Design #1391: RingElems: syntax with [and] ? added

#21 - 13 Jan 2020 16:35 - Anna Maria Bigatti

As discussed in #1931, there are now two functions:
(A) RingElems(RING, STRING) example RingElems(P, "x*y,z")
and
(B) RingElemList(RING, STRING) example RingElems(P, "[x*y,z]")

I prefer (A) and it is also implemented in cocoalib.
Not yet documented (neither)

#22 - 14 Feb 2020 13:51 - Anna Maria Bigatti

- % Done changed from 60 to 80

I prefer A, John prefers B, so we'll have them both.
Write documentation.

#23 - 27 Feb 2020 18:36 - Anna Maria Bigatti

- Status changed from Resolved to Feedback

- % Done changed from 80 to 90

- Estimated time set to 5.00 h

#24 - 28 Feb 2020 17:37 - Anna Maria Bigatti

- Description updated

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

#25 - 28 Feb 2020 17:52 - Anna Maria Bigatti

documented