CoCoALib - Slug #1238

ReadExpr is too slow on long lists of monomial with many indets: ---> use RingElems instead

21 Jan 2019 16:00 - Anna Maria Bigatti

Status:	Closed	**Start date:**	21 Jan 2019
Priority:	Urgent	**Due date:**	
Assignee:	Anna Maria Bigatti	**% Done:**	100%
Category:	Improving	**Estimated time:**	0.00 hour
Target version:	CoCoALib-0.99700	**Spent time:**	2.65 hours

Description

Eduardo Saenz de Cabezon has horrible monomial ideals with 500 indets and 500 gens.
Even with just 50 gens it takes 24s to read the list.
Move the construction of SymbolTable into the ring as suggested in issue #881.

2020-02-12 more flexible solution is the new function **RingElems** which reads a vector of RingElem: it reads a whole list (with a single SymbolTable) istead of making repeated calls for each entry.

Related issues:

Related to CoCoALib - Slug #881: ReadExpr is too slow on large polys	**Closed**	**09 May 2016**
Related to CoCoA-5 - Feature #1243: New function: Read a string into a list (...	**Closed**	**13 Feb 2019**
Related to CoCoA-5 - Slug #1629: RingElem slow with many indets	**Closed**	**08 Nov 2021**

History

#1 - 21 Jan 2019 16:00 - Anna Maria Bigatti

- Related to Slug #881: ReadExpr is too slow on large polys added

#2 - 21 Jan 2019 21:30 - John Abbott

It would be helpful to have some examples (perhaps even a family of examples).
This should help the developers, and will also act a reference for testing whether progress has been made ;-)

#3 - 23 Sep 2019 12:16 - John Abbott

- % Done changed from 0 to 10

2019-09-23 This is still very slow. Here is a specific test case:

```
Nvars := 999;
use P ::= QQ[x[1..Nvars]];
/// S := RandomSubset(1..Nvars, 50);
/// foreach j in S do println "\"x[",j,"]\",";  endforeach;

L := [ "x[98]", "x[121]", "x[125]", "x[152]", "x[154]", "x[157]",
      "x[192]", "x[215]", "x[223]", "x[244]", "x[245]", "x[258]",
      "x[286]", "x[321]", "x[329]", "x[341]", "x[345]", "x[359]",
      "x[366]", "x[385]", "x[436]", "x[447]", "x[448]", "x[462]",
      "x[483]", "x[509]", "x[522]", "x[538]", "x[595]", "x[608]",
      "x[617]", "x[623]", "x[625]", "x[654]", "x[659]", "x[723]",
      "x[737]", "x[755]", "x[765]", "x[773]", "x[778]", "x[783]",
      "x[794]", "x[801]", "x[826]", "x[909]", "x[919]", "x[931]",
      "x[979]", "x[990]"];

t0 := CpuTime();
LL := [RingElem(P, str) | str in L];
println "Time to read the strings: ", TimeFrom(t0); --> about 33s on my machine
```

#4 - 23 Sep 2019 12:20 - John Abbott

Here is another case where it is surprisingly slow:

```
use P ::= QQ[x[1..999]];
str := "x[98] +x[121] +x[125] +x[152] +x[154] +x[157] +x[192] +x[215] +x[223] +x[244] +x[245] +x[258] +x[286]
+x[321] +x[329] +x[341] +x[345] +x[359] +x[366] +x[385] +x[436] +x[447] +x[448] +x[462] +x[483] +x[509] +x[522
] +x[538] +x[595] +x[608] +x[617] +x[623] +x[625] +x[654] +x[659] +x[723] +x[737] +x[755] +x[765] +x[773] +x[7
78] +x[783] +x[794] +x[801] +x[826] +x[909] +x[919] +x[931] +x[979] +x[990]";
t0 := CpuTime(); j := RingElem(P, str);  TimeFrom(t0);
0.666
>>>
```

The string is just the sum of the individual terms from the earlier example.
Run time is unacceptably slow!

#5 - 23 Sep 2019 12:41 - John Abbott

I have run the profiler.
A lot of time is spent in **AreDistinct(vector<symbol>)**; presumably the call is around ring.C:17 inside RingBase::myNew(symbol).

It really seems that the pseudo-ctor RingBase::myNew(symbol) needs to be rewritten, and it needs help from the ring...

#6 - 23 Sep 2019 12:41 - John Abbott

- Target version changed from CoCoALib-0.99650 November 2019 to CoCoALib-0.99700

#7 - 09 Oct 2019 20:27 - John Abbott

- Related to Feature #1243: New function: Read a string into a list (of RingElem) -- CoCoA-5 added

#8 - 09 Oct 2019 20:32 - John Abbott

- % Done changed from 10 to 20

I have just tried the example from comment 4, but in a ring with 9999 indets (instead of just 999). The time increased dramatically to 108s.

How is that possible?!? Flabbergasted!

#9 - 09 Oct 2019 21:45 - John Abbott

I have run an example (in a ring with 4999 indets) with the profiler (it was rather slower than I expected).

The time is almost evenly split between

- CoCoA::SparsePolyRingBase::mySymbolValue(CoCoA::symbol const&)
- CoCoA::AreDistinct(std::vector<CoCoA::symbol, std::allocator<CoCoA::symbol> > const&)

Mmm, there are some quite surprising call counts. op== for symbols was called over 37000000 times!
Of those, 25000000 were from within AreDistinct.

Shouldn't we be using a C++ map for symbols?

#10 - 10 Oct 2019 08:33 - Anna Maria Bigatti

John Abbott wrote:

> 2019-09-23 This is still very slow. Here is a specific test case:
> [...]

New function ReadExprVector (we'd better find a better name before this sticks)

```
t0 := CpuTime(); LS := ReadExprVector(P, "x[98], x[121], x[125], x[152], x[154], x[157], x[192], x[215], x[223
], x[244], x[245], x[258], x[286], x[321], x[329], x[341], x[345], x[359], x[366], x[385], x[436], x[447], x[4
48], x[462], x[483], x[509], x[522], x[538], x[595], x[608], x[617], x[623], x[625], x[654], x[659], x[723], x
[737], x[755], x[765], x[773], x[778], x[783], x[794], x[801], x[826], x[909], x[919], x[931], x[979], x[990]"
); TimeFrom(t0);
1.384
```

update Jan 2020 now called **RingElems**

#11 - 10 Oct 2019 08:36 - Anna Maria Bigatti

- % Done changed from 20 to 60

Dramatic improvement using the new function ReadExprVector in ex-MVT-Simplicial.C. (because the new function creates only one SymTable)

All inputs in ex-MVT-Simplicial-tests will have to change all ; into ,

update Jan 2020 now called **RingElems**

#12 - 13 Jan 2020 16:42 - Anna Maria Bigatti

- Status changed from In Progress to Resolved

- % Done changed from 60 to 80

The real solution is using RingElems, this new function is actually quite useful for other things too, and also in CoCoA-5.

Fix the input files for MVT.

#13 - 12 Feb 2020 09:07 - Anna Maria Bigatti

- Description updated

- Status changed from Resolved to Feedback

- % Done changed from 80 to 90

#14 - 12 Feb 2020 09:35 - Anna Maria Bigatti

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

checked-in.

#15 - 12 Feb 2020 09:42 - Anna Maria Bigatti

- Subject changed from ReadExpr is too slow on long lists of monomial with many indets to ReadExpr is too slow on long lists of monomial with many indets: ---> use RingElems instead

#16 - 12 Feb 2020 09:44 - Anna Maria Bigatti

- Description updated

#17 - 26 Nov 2021 14:59 - John Abbott

- Related to Slug #1629: RingElem slow with many indets added