CoCoA-5 - Slug #875

Interpreter is too slow reading a big polynomial

03 May 2016 14:00 - John Abbott

Status:	In Progress	**Start date:**	03 May 2016
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	10%
Category:	enhancing/improving	**Estimated time:**	0.00 hour
Target version:	CoCoA-5.?.?	**Spent time:**	1.10 hour

Description

I'm trying to read a large polynomial: about 6Mbytes, about 127000 terms in 100 indets, coefficients in QQ (but all quite small integers). CoCoA-5 is taking way too long (over 50mins), and too much RAM (over 600Mbytes).

Make it quicker and less RAM hungry!

Related issues:

Related to CoCoALib - Slug #874: factor: too slow on largish multivariate poly	**In Progress**	**03 May 2016**
Related to CoCoA-5 - Design #576: Disallow juxtaposition for string literals?	**Closed**	**25 Jun 2014**
Related to CoCoA-5 - Feature #781: Option to "fold" long lines?	**Closed**	**28 Sep 2015**
Related to CoCoA-5 - Design #473: Multiline string literals - useful or obsol...	**Closed**	**13 Mar 2014**
Related to CoCoA-5 - Slug #434: Emacs UI: very slow when input file is big (w...	**New**	**06 Feb 2014**
Related to CoCoA-5 - Bug #1514: Cocoa crashes when calling RingElems	**Closed**	**22 Oct 2020**

History

#1 - 03 May 2016 14:00 - John Abbott

- Related to Slug #874: factor: too slow on largish multivariate poly added

#2 - 03 May 2016 17:02 - John Abbott

I have tried doing some profiling. The interpreter takes quadratic time (in the number of terms) to read a polynomial from its printed form, *i.e.* as a sum of terms.

Somewhat ironically, for polynomials with many terms it is significantly faster to read them as a string and then use ReadExpr to convert it into a ring element. The point here is that ReadExpr knows to expect a ring element whereas the interpreter has to be able to cope with expressions of any type.

An ideal situation would be to have the parser generate n-ary nodes for sums. I've no idea how hard this might be; it would certainly involve some "deep" changes to the parse tree structure, and also the interpreter.

An alternative would be to hack the interpreter so that it uses geobuckets when summing values in a polynomial ring. This might be quite messy and fiddly to get right :-/

#3 - 06 Jun 2016 21:58 - John Abbott

The solution to use ReadExpr(P, "poly-as-a-string") works tolerably well from the point of view of the interpreter, but it is insufferably slow inside emacs if the string is a long literal (today's example was 20Mbytes long).

The problem is that emacs gets too slow when there are long lines. Note that changing from cocoa5-mode to fundamental-mode in emacs did help a bit, but it was still far too slow. Should we revitalize issue #576? (about allowing juxtaposition of string literals to cause the parser to concatenate them)

#4 - 06 Jun 2016 23:50 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

I have taken one of Mario's big polys: this one is about 6Mbyte when printed out.

I tried to ways of converting the string back to a poly:

- **(A)** putting the whole string on one very long line;
- **(B)** splitting the string into a sum of about 81600 short strings (one on each line).

Method **(A)** took about 7.2s on my MacBook, whereas method **(B)** took almost 180s.
The point being that computing the sum str1+str2+...+str81600 has quadratic complexity since the interpreter computes each initial subsum.

I note that editing the file for approach **(B)** was still a bit slow in cocoa5-mode; especially "semicolon" and "newline" were quite slow. Editing the file for approach **(A)** seemed to be far worse (almost every character was slow to appear).

#5 - 06 Jun 2016 23:50 - John Abbott

- Related to Design #576: Disallow juxtaposition for string literals? added

#6 - 06 Jun 2016 23:51 - John Abbott

- Related to Feature #781: Option to "fold" long lines? added

#7 - 06 Jun 2016 23:52 - John Abbott

- Related to Design #473: Multiline string literals - useful or obsolescent? added

#8 - 06 Jun 2016 23:53 - John Abbott

- Related to Slug #434: Emacs UI: very slow when input file is big (with long lines) added

#9 - 20 Oct 2020 18:05 - John Abbott

A better workaround is to use sum([...]).
I have just tried a test of the form:

```
substr := "some substring"; --> loer in my test
str := substr+substr+ ... (about 10000 summands) + substr; --> took about 2s
str := sum([substr, substr, ... (same number of summands), substr]); --> took about 0.3s
```

So a tolerable workaround is to split the string into substrings which are placed in a list, and then use sum to concatenate them.

This issue is really about two problems: CoCoA's quadratic behaviour reading a poly directly, Emacs's slow behaviour on long lines.

#10 - 23 Oct 2020 11:07 - John Abbott

- Related to Bug #1514: Cocoa crashes when calling RingElems added