CoCoALib - Feature #1633

Make polynomial multiplication interruptible?

16 Nov 2021 20:32 - John Abbott

Status:	Closed	**Start date:**	16 Nov 2021
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Improving	**Estimated time:**	0.99 hour
Target version:	CoCoALib-0.99800	**Spent time:**	0.90 hour

Description

I tried a (daft) example during an exercise class today: when tried to square a large polynomial it was not possible to interrupt CoCoA-5.

```
f := 1+2*x-2*x^2+4*x^3+4*x^4;
g := f*subst(f,x,x^7);
h := g*subst(g,x,x^63);
k := h*subst(h,x,x^4095);
NumTerms(k);  --> 390625
NumTerms(k^2);  -- NOT INTERRUPTIBLE!!
```

Should we make polynomial multiplication interruptible?
It is quite rare that one performs such a daft computation, and it would be a shame to impose unnecessary overhead on all products just because sometimes one might try to compute a large product.

Related issues:

Related to CoCoALib - Feature #718: Insert calls to CheckForInterrupt	**Closed**	**21 May 2015**

History

#1 - 16 Nov 2021 20:33 - John Abbott

One crucial factor is how much overhead it would cost if we put a check inside some inner loop.

Also how many different impls of polynomial multiplication are there?
Should we put checks in all of them?

#2 - 16 Nov 2021 20:34 - John Abbott

- Related to Feature #718: Insert calls to CheckForInterrupt added

#3 - 17 Nov 2021 11:50 - John Abbott

The relevant source code is probably myMul in SparsePolyOps-RingElem.C around line 413.

#4 - 17 Nov 2021 11:53 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

Here is a simpler test example

```
/**/ f := (x^10000-1)/(x-1);
/**/ t0 := CpuTime(); NoPrint := f^2; TimeFrom(t0);
```

The normal version of CoCoA-5 (without interrupt checking) takes about 6.3-6.4s on my linux box.

#5 - 17 Nov 2021 11:59 - John Abbott

- Assignee set to John Abbott

I have just inserted a CheckForInterrupt in the main loop for multiplication.
The simpler test example from comment 4 was not significantly slower (actually, it was faster, but that makes no sense).
I checked also that the multiplication is interruptible.

#6 - 17 Nov 2021 12:04 - John Abbott

This is weird. I reverted to the old code because I wanted to confirm that it is not interruptible.
Indeed, it took a long time to recognize the interrupt, but the timer reported that the interrupt was
recognised after about 4.2s. Where do the other 2.1s go? Assignment? Worrying!

#7 - 28 Jan 2022 13:08 - John Abbott

- Status changed from In Progress to Feedback

- % Done changed from 10 to 90

#8 - 04 Feb 2022 21:24 - John Abbott

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

- Estimated time set to 0.99 h