CoCoALib - Slug #968

Slow NF example

10 Nov 2016 16:13 - John Abbott

Status:	In Progress	**Start date:**	10 Nov 2016
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	10%
Category:	Improving	**Estimated time:**	16.00 hours
Target version:	CoCoALib-1.0	**Spent time:**	2.00 hours

Description

Here is a silly example where NF takes longer than it should:

```
use P ::= QQ[x,y,z];
I := ideal(x^2, y^2 -x-y-z-1);
NF(x^999*y^999, I); // obviously 0, and should be instant... but it is not!
```

History

#1 - 10 Nov 2016 16:16 - John Abbott

When computing a NF(x^999*y^999,I) we can use either GBasis element for the reduction.
CoCoA ought to prefer reducing by a monomial is possible, and perhaps by binomials too?

#2 - 10 Nov 2016 16:42 - Anna Maria Bigatti

- Estimated time set to 16.00 h

John Abbott wrote:

> When computing a NF(x^999*y^999,I) we can use either GBasis element for the reduction.
> CoCoA ought to prefer reducing by a monomial is possible, and perhaps by binomials too?

there are two NF:
+ within a Groebner Basis computation, where polynomials are wrapped into GPoly, and that uses (I think) information about length
+ outside: and that does nothing clever (apart from using a ReductionCog)

I think I can do this resorting before applying the cog (but that would make a copy of the GBasis, which tends to be sorted by degree).

This make me think: should I make a ReductionMill which contains the list of reducers?
Then a GBasis could be stored as a ReductionMill.

#3 - 10 Nov 2016 17:03 - John Abbott

On the fixed linux box here it took about 210s to compute NF(f,I).

This makes me think that it has effectively computed (x+y+z+1)^498 and then discovered that all the terms reduce to zero.
Indeed, if I compute NF(x^999*y^999, ideal(y^2-x-y-z-1)) then it takes about 200s, and produces a polynomial with about 250000 terms.

NOTE earlier I was a little confused, and had (wrongly) expected the result of NF(f,I) to be x^999*y*(x+y+z+1)^498 which is enormous (about 21000000 terms); but many terms in this polynomial can still be reduced by y^2-x-y-z-1 to produce the above result of NF(x^999*y^999, ideal(y^2-x-y-z-1)).

#4 - 10 Nov 2016 17:09 - Anna Maria Bigatti

John Abbott wrote:

> This makes me think that it has effectively computed (x+y+z+1)^498 and then discovered that all the terms reduce to zero.
> But how? If I try to compute high powers of (x+y+z+1) then it already takes 280s to compute the 250th power. Is NF the quickest way to compute powers?

not quite: y^2 -x-y-z-1 rewrites powers of y itself.

NOTE I did indeed write what Anna quoted, but realised simultaneously with her what the explanation was; so I edited my comment (3) while she replied to it.

#5 - 10 Nov 2016 21:09 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

Here is another example which rather suggests that the problem is quite difficult in general:

```
use P ::= QQ[x,y,z];
I := ideal(x^15 -z^15 -2*z^14 -2*z^13 -2*z^12 -2*z^11 -2*z^10 -2*z^9 -2*z^8 -2*z^7 -2*z^6 -2*z^5 -2*z^4 -2*z^3
 -2*z^2 -2*z -1,
           y^15 -z^15 +2*z^14 -2*z^13 +2*z^12 -2*z^11 +2*z^10 -2*z^9 +2*z^8 -2*z^7 +2*z^6 -2*z^5 +2*z^4 -2*z^3
 +2*z^2 -2*z +1);
h := x^15*y^15;
NF(h,I); --> z^30-1, a very sparse polynomial
NF(h^256, I); --> takes about 45s on my computer (old MacBook)
NoPrint := (z^30-1)^256;  --> effectively instant
```

Here the ideal strategy is probably to reduce by x^15-... then by y^15-... and so on, alternating. Deducing this automatically looks to be tricky :-/

Let J be the ideal generated by the first generator.
Note that NF(h^256,J) takes about 10s on my computer, and produces a polynomial with about 3800 terms and coefficients with 375 digits. The final answer has 257 terms and largest coeff about 75 digits long.

#6 - 10 Nov 2016 21:33 - John Abbott

I constructed the example in comment 5 by taking a sparse polynomial (in this case z^30-1) which has a factorization into dense factors, f1 and f2. The ideal is then ideal(X-f1, Y-f2), and I know that NF(X*Y,I) is going to be a sparse polynomial; in this case I chose X and Y to be powers of the indeterminates x and y so that the LTs would be X and Y using the term ordering of the ring (stddegrevlex).

Presumably many other similar examples could be created by taking sparse polynomials which have several dense factors (perhaps even with multiplicity). Then it could be quite tricky to find the "magic term" which reduces nicely to a sparse poly if you don't know how the ideal was constructed.

So, I believe that the crucial feature of these examples is the existence of a term whose normal form is unusually sparse (or equivalently a sparse polynomial in the ideal). This seems likely to offer a cheap way of computing NFs of more general polynomials if we use preferentially such sparse elements.