CoCoALib - Feature #651

Optimized algorithms for implicitization (slicing algorithm, elim, subalgebra..)

13 Nov 2014 13:44 - Anna Maria Bigatti

Status:	In Progress	**Start date:**	13 Nov 2014
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	10%
Category:	New Function	**Estimated time:**	100.00 hours
Target version:	CoCoALib-1.0	**Spent time:**	35.60 hours

Description

(may keywords in the title to simplify future searches ;-)

Investigate new algorithms for implicitization, i.e. finding the kernel of

```
k[x_1,...,x_n] --> k[f_1,..,f_n]
```

Related issues:

Related to CoCoA-5 - Feature #600: efficient subalgebras	**New**	**16 Jul 2014**
Related to CoCoALib - Slug #866: implicit, ImplicitHypersurface: improve outp...	**In Progress**	**13 Apr 2016**

History

#1 - 17 Nov 2014 11:54 - John Abbott

JAA has implemented in C++/CoCoALib (first prototypes of): ImplicitDirect (with variants LPP and WithCond), also ImplicitByPoints (with variant @LPP).

No documentation; no formal tests.

These fns are also available from CoCoA-5 (no documentation!)

#2 - 21 Nov 2014 11:30 - John Abbott

The very latest version of ImplicitByPoints3 does occasionally go a little faster than ImplicitDirectLPP2, but it also seems to display worse empirical complexity (*i.e.* it become slower than ImplicitDirectLPP2 as the inputs get bigger). I think this may be because ImplicitByPoints3 works with an essentially random matrix (which has to be triangularised), whereas I believe that ImplicitDirectLPP2 builds a "matrix" which is no far from triangular (and so needs little work to be triangularized).

I also point out that ImplicitByPoints3 does not verify its answer while the result of ImplicitDirectLPP2 is guaranteed correct.

Overall, I conclude that ImplicitByPoints probably cannot be refined enough to make it significantly better than ImplicitDirect -- not even in certain special circumstances.

It could be that profiling ImplicitByPoints3 would let me improve a bit further, but the observed poorer empirical complexity (with "heuristic explanation") is discouraging.

Note profiling on a GNU/linux VM showed that the main cost was accessing entries in the std::map<PPMonoidElem, int>

#3 - 21 Nov 2014 11:32 - John Abbott

At a meeting yesterday it was decided that effort should be put into developing/refining **ImplicitDirectWithCond**. Anna did report disappointing performance from the initial implementation -- she will investigate the cause.

#4 - 21 Nov 2014 16:15 - Anna Maria Bigatti

John Abbott wrote:

> Anna did report disappointing performance from the initial implementation -- she will investigate the cause.

Indeed I was running my (supposedly random) test in a particularly unfavourable case for **ImplicitDirect**.
So it not a problem with overhead.
so: now comparing with other tests.

#5 - 09 Dec 2014 23:02 - John Abbott

One problem with ImpliciDirectWithCond is that the normal form computations take too long. Here is an idea which *might* make the NF computations a bit faster (at the risk of creating some false positives).

Instead of computing NF modulo the ideal I we could compute modulo I+J (or better I+J1 and I+J2) where the ideal J is chosen to be convenient, *e.g.* J = ideal(s-1234).

If we can do this in such a way that there are not too many false positives then I think we could gain speed.

#6 - 10 Dec 2014 16:38 - Anna Maria Bigatti

I'm running tests with the profiler on our linux virtual machine.
One surprise was that the profiler gave a mysterious

```
[19]    12.5    0.01    0.47  695808        RingDistrMPolyInlFpPPImpl::myNegate
                0.00    0.47  695808/851237 DistrMPolyInlFpPP::operator=
```

but those "=" calls (doing nothing) do not seem relevant: I commented out the " = " line in RingDistrMPolyInFpPP (in our context it is called only with two identical arguments) and the timing was effectively identical.
That surprised me a bit for two reasons: I expected...
1 - the profiler to be more reliable
2 - 695808 (useless) function calls to be more time consuming

#7 - 12 Feb 2015 10:33 - Anna Maria Bigatti

I changed all ring names into Kt and Kx in TmpImplicit.C

#8 - 13 Apr 2016 13:34 - John Abbott

- Related to Slug #866: implicit, ImplicitHypersurface: improve output verification added