CoCoALib - Slug #866

implicit, ImplicitHypersurface: improve output verification

13 Apr 2016 13:34 - John Abbott

Status:	In Progress	**Start date:**	13 Apr 2016
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	20%
Category:	Tidying	**Estimated time:**	20.00 hours
Target version:	CoCoALib-0.99880	**Spent time:**	3.55 hours

Description

The current impl of SliceCoreQQ is simple rather than efficient.
It uses modular+CRT to build the answer.
Currently if RatReconstructPoly succeeds, it checks whether the answer is right (but substitution). Does the substitution ever fail? (if so, it could be very costly).

It would be nice not to treat the first prime as a special case (*i.e.* just have a loop which keeps trying successive primes). Investigate!

Related issues:

Related to CoCoALib - Feature #651: Optimized algorithms for implicitization ...	**In Progress**	**13 Nov 2014**
Related to CoCoALib - Feature #1167: New class VerificationLevel	**Closed**	**13 Mar 2018**

History

#1 - 13 Apr 2016 13:34 - John Abbott

- Related to Feature #651: Optimized algorithms for implicitization (slicing algorithm, elim, subalgebra..) added

#2 - 13 Apr 2016 15:45 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

The example below takes about 12s on my computer, but only about 0.25s if the coeff field is ZZ/(32003). Why is it almost 50 times slower?
Internally it should be using modular+CRT+FTRR and supposedly the FTRR phase is cheap.

```
k ::= QQ;
kst ::= k[s,t];
use kst;
-- Dongming example 13:
L := [ s^3+3*t^3-3*s^2-6*t^2+6*s+3*t-1,
       3*s^3+t^3-6*s^2+3*s+3*t,
       -3*s^3*t^3-3*s^3*t^2+15*s^2*t^3+6*s^3*t-18*s^2*t^2-15*s*t^3+9*s^2*t+27*s*t^2-3*s^2-18*s*t-3*t^2+3*s+3*t]
;
P ::= k[x,y,z];
t0 := CpuTime();
f := ImplicitHypersurface(P, L, "Direct");
println "Time taken: ", TimeFrom(t0);
```

#3 - 13 Apr 2016 17:04 - Anna Maria Bigatti

- % Done changed from 10 to 20

John Abbott wrote:

> The example below takes about 12s on my computer, but only about 0.25s if the coeff field is ZZ/(32003). Why is it almost 50 times slower? Internally it should be using modular+CRT+FTRR and supposedly the FTRR phase is cheap.

Yes, I had observed in many examples that FTRR is amazingly cheap.
However the final verification process (actual substitution) can be quite slow.
(maybe I could add a flag to print in CoCoA these logging informations)

#4 - 18 Apr 2016 12:23 - John Abbott

I suggest adding a probabilistic check of correctness: use the parametric formulas to generate a random (numerical) point, and verify that the point satisfies the implicit equation. In principle one could generate several random points and test them all; or perhaps generate several random points, and test just the simplest looking one(s)?

It may also be nice to offer the user the possibility to say that the final "absolute test" may be skipped (to save time).

#5 - 18 Apr 2016 12:33 - Anna Maria Bigatti

John Abbott wrote:

> I suggest adding a probabilistic check of correctness: use the parametric formulas to generate a random (numerical) point, and verify that the point satisfies the implicit equation. In principle one could generate several random points and test them all; or perhaps generate several random points, and test just the simplest looking one(s)?
>
> It may also be nice to offer the user the possibility to say that the final "absolute test" may be skipped (to save time).

True.
But I'd like another try: I'm planning to try to implement a sort of recursive Horner. Maybe that will give a decent enough improvement.
(I have a little more time this week, I hope I can implement it)

#6 - 22 Apr 2016 16:26 - Anna Maria Bigatti

Anna Maria Bigatti wrote:

> But I'd like another try: I'm planning to try to implement a sort of recursive Horner. Maybe that will give a decent enough improvement.

working on it now

#7 - 22 Apr 2016 17:02 - John Abbott

I seem to recall vaguely seeing a paper by Mourrain (or someone in his group) possibly about this. I think the question was how best to exploit sparseness. I may be completely wrong though; I don't think I read beyond the abstract anyway.

#8 - 22 Apr 2016 18:01 - Anna Maria Bigatti

Anna Maria Bigatti wrote:

> > But I'd like another try: I'm planning to try to implement a sort of recursive Horner. Maybe that will give a decent enough improvement.
>
> working on it now

:-(:-(
no, it doesn't work at all well.
:-(:-(
General implementation is fine and goes
very well in case of 1 param,
well in case of 2 params,
not well for more.
Now, it may improve using geobuckets, but I'm no longer hopeful for a great improvement (without a major new idea which might be worth a paper by itself...)

#9 - 22 Apr 2016 18:09 - Anna Maria Bigatti

I checked in IsZeroEvalHorner in case you want to play with it.
The call is currently commented out. Just look for it in TmpImplicit.C.

#10 - 06 May 2016 16:43 - John Abbott

I have not yet had time to look at your horner code. Anyway, I would expect it work fairly well provided the "variables are evaluated in the right order": my expectation is that indets which map to simpler values should be the outermost ones for the multivariate horner implementation.

I'm pretty certain that some orders will be better than others, but do not really have any idea how much better... Could a factor of 2 be possible? Even greater gain?

#11 - 08 Nov 2017 18:09 - John Abbott

- Target version changed from CoCoALib-0.99560 to CoCoALib-0.99600

#12 - 14 Dec 2017 17:05 - Anna Maria Bigatti

- Subject changed from TmpImplicit: improve SliceCoreQQ to TmpImplicit: improve verification

- Estimated time set to 20.00 h

#13 - 14 Dec 2017 17:05 - Anna Maria Bigatti

- Subject changed from TmpImplicit: improve verification to implicit, ImplicitHypersurface: improve output verification

#14 - 02 Aug 2018 17:43 - Anna Maria Bigatti

- Related to Feature #1167: New class VerificationLevel added

#15 - 02 Aug 2018 17:46 - Anna Maria Bigatti

- Target version changed from CoCoALib-0.99600 to CoCoALib-0.99650 November 2019

#16 - 01 Oct 2019 12:22 - John Abbott

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

2019-10-01: I have just tried the example from comment 2, and now it takes about 5.9s on my computer.

#17 - 12 Feb 2020 16:08 - John Abbott

- Target version changed from CoCoALib-0.99700 to CoCoALib-0.99800

#18 - 03 Nov 2021 20:10 - John Abbott

- Target version changed from CoCoALib-0.99800 to CoCoALib-0.99850

#19 - 22 Mar 2024 09:43 - Anna Maria Bigatti

- Target version changed from CoCoALib-0.99850 to CoCoALib-0.99880