CoCoALib - Feature #838

Differential algebra

11 Jan 2016 13:43 - John Abbott

Status:	In Progress	**Start date:**	11 Jan 2016
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	10%
Category:	New Function	**Estimated time:**	0.00 hour
Target version:	CoCoALib-1.0	**Spent time:**	2.20 hours

Description

Werner has suggested that CoCoA could offer differential algebras.

If I have understood correctly, these are a bit like polynomial rings with infinitely many indets.

Discuss, and see what we can implement.

Related issues:

Related to CoCoALib - Feature #728: Noncommutative algebra "of solvable type"	**In Progress**	**08 Jun 2015**

History

#1 - 11 Jan 2016 14:16 - John Abbott

Here are some notes following a discussion I had with Werner last week... there may be some mistakes if I have not understood or memorized everything correctly.

A differential algebra needs to be closed under "derivation", and since there is no (a priori) limit to the depth of the derivatives needed, the algebra must contain an unlimited number of indets; I recall that each derivative is regarded as a separate indeterminate (somehow the derivation operator knows the relationships between the various indets).

Werner also suggested that a potentially simpler approach would be to fix the max depth of the derivations, and attempt the computation in that "truncated" diff algebra. The computation succeeds if there was no need to use a derivative outside those initially supplied; otherwise it will fail, and the user should try again specifying a greater maximum depth (a bit like how higher precisions can be used with RingTwinFloat). JAA regards this approach as less convenient for the user, but it has the advantage that there is no need to have a (non-comm) poly ring with an unlimited number of indets.

#2 - 11 Jan 2016 15:00 - John Abbott

More details. My understanding is that there are two types of "variable": independent variables (x[1] to x[n]) and dependent variables (with heads u1, u2, ...). Each dependent variable is the root of a DAG of derivatives, and each derivative is treated as a separate indeterminate. For instance, if n=3 then u1[0,0,0] is the first dependent variable; its derivative w.r.t. x[1] is then u1[1,0,0]. The indices after the head u1 indicate the derivative: for instance u1[2,0,3] indicates d/dx1 d/dx1 d/dx3 d/dx3 d/dx3 u1. Recall that taking derivatives is commutative, so it is enough to note simply how many levels of derivative w.r.t. each indep variable (*i.e.* an n-tuple of integers suffices to fully specify the derivative).

#3 - 11 Jan 2016 15:05 - John Abbott

Werner pointed out that it is (or "can be"?) useful to have an ordering on the derivatives, and that the interesting orderings are very much like usual "term orderings".

JAA has not yet understood how precisely these orderings are used. By the design of CoCoALib, the indets representing the various derivatives have to be ordered by a term-ordering; it may be possible to "kill two birds with one stone".

#4 - 20 Oct 2016 16:32 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

The ctor args for a DifferentialAlgebra are the following:

- **n** number of "independent variables"
- **k** number of "dependent variables"
- **Emax** maximum depth of derivation allowed (wrt to each indep var; max total depth id n*Emax)

How are the names of the indep and dep vars specified?
Werner had suggested that the indep vars would normally be **x[1]** up to **x[n]**; so it probably suffices to accept a symbol head as the name (with default value **"x"**)
Werner's suggestion for the dep vars is a little trickier: **u1[0,0,0]** up to **u_k[0,0,0]**; so it could suffice to specify just a symbol head (default **"u"**) but internally it will need a little tedious jiggery-pokery to attach the suffixes to get **u1**, **u2** and so on.

Note that the effective number of indets in the ring will be potentially quite large: n + k*(1+Emax)^n

Is it correct that all the variables are commutative? *e.g.* u1[1,0,0]*x[1] == x[1]*u1[1,0,0]?

#5 - 24 Oct 2016 15:32 - John Abbott

After some more reading, it seems that the usual structure is QQ(x_1, x_2, ..., x_n)[u1, u2,...] where the expressions in the "indep vars" are rational functions.