CoCoALib - Feature #802

DivMask: extend interface?

10 Nov 2015 15:57 - John Abbott

Status:	In Progress	**Start date:**	10 Nov 2015
Priority:	Urgent	**Due date:**	
Assignee:	John Abbott	**% Done:**	50%
Category:	New Function	**Estimated time:**	0.00 hour
Target version:	CoCoALib-0.99880	**Spent time:**	0.90 hour

Description

The new PPMonoidSparse impl needs to supply a myComputeDivMask mem fn.

The current interface is well suited to dense PP reprs, but not well suited to a sparse repr.
Perhaps extend the interface to allow a sparse repr?

Discuss; decide; implement (if appropriate)

Related issues:

Related to CoCoALib - Feature #800: PPMonoidSparse: impl of sparse PPs	**Closed**	**09 Nov 2015**

History

#1 - 10 Nov 2015 16:09 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

Currently the only way to make a (non-trivial) DivMask is to use the function myAssignFromExpv which clearly suits a dense representation.

The sparse repr naturally presents a PP as a succession of (index,exp) pairs with the guarantee that the indexes are distinct (perhaps even in increasing order). As far as I can see all the current divmask rules would happily allow a divmask to be built up one (index,exp) pair at a time.

An alternative would simply be to create an exponent vector from the sparse repr and the use that to build the DivMask, but that could be expensive (*e.g.* if there are many thousands of indets but only very few actually appear).

#2 - 10 Nov 2015 16:15 - John Abbott

If we do extend the interface of DivMask to accommodate the sparse repr, what should the extended interface expect:

- **(A)** a std::list or std::vector of (index,exp) pairs
- **(B)** a single (index,exp) pair (and allow multiple updates)
- **(C)** something else?

A disadvantage of **(A)** is that it would expose the internal structure used inside sparse PPs; or it could accept both std::list and std::vector?
A disadvantage of **(B)** (with multiple updates) is that there is no easy/cheap way to check whether there are (index,exp) pairs sharing the same index -- maybe that is not such a serious problem.
A disadvantage of **(C)** is that I have no idea what it might be :-/

ADDENDUM Option **(B)** seems to be the most "essential"; I'm unsure how serious is the inability to check whether there are repeated updates with the same index. A careless user may end up producing a junk result with no error/warning that something suspect is happening. Right now I do not see a genuine situation where it would be useful to be able to re-update with the same indet index (but the overhead for checking would be too great).

#3 - 10 Nov 2015 18:33 - Anna Maria Bigatti

John Abbott wrote:

> If we do extend the interface of DivMask to accommodate the sparse repr, what should the extended interface expect:
>
> - **(A)** a std::list or std::vector of (index,exp) pairs
> - **(B)** a single (index,exp) pair (and allow multiple updates)
> - **(C)** something else?

I liked **B** to start with, but then I thought of the problem that we might not be able to update a DivMask: if I add (i, 4) and (i, 5) does this mean that we have x[i]^(4+5)? then DivMaskHashing might have problems updating the DivMask

I think we should go for **A** with list and/or vector (requiring there are repeated indices)

#4 - 10 Nov 2015 18:38 - John Abbott

I think updating would mean replacing the div-mask with that for the LCM of the old value and the new indet-power. Thus:

```
dm = Initially 0; // corr to PP = 1
dm.update(x,2);   // now corr to x^2 = LCM(1, x^2)
dm.update(x,3);   // now corr to x^3 = LCM(x^2, x^3)
```

I think bitwise-or of two div-masks means getting the div-mask for the LCM of their PPs
[am I right?]

#5 - 11 Nov 2015 12:15 - Anna Maria Bigatti

John Abbott wrote:

> I think updating would mean replacing the div-mask with that for the LCM of the old value and the new indet-power. Thus:

Ok, with that definition of "updating" I think it might work (of course keeping in mind that a DM does not know which PP it's coming from)

I recall here that our definition of DM is: if PP1 | PP2 then DM | DM (bitwise)

#6 - 24 Nov 2015 19:04 - John Abbott

I think we should implement **(B)** with the "LCM" meaning from comment 4.

I can do this after checking in the current (working!) version of PPMonoidSparse

#7 - 27 Nov 2015 17:47 - John Abbott

Maybe I can do this next week?

#8 - 23 Mar 2016 15:13 - John Abbott

- Assignee set to John Abbott

- Priority changed from Normal to Urgent

- Target version changed from CoCoALib-0.99540 Feb 2016 to CoCoALib-0.99550 spring 2017

- % Done changed from 10 to 50

#9 - 21 Sep 2016 18:16 - John Abbott

- Target version changed from CoCoALib-0.99550 spring 2017 to CoCoALib-0.99560

#10 - 06 Nov 2017 15:04 - John Abbott

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

#11 - 30 Jul 2018 16:13 - Anna Maria Bigatti

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

#12 - 01 Oct 2019 11:38 - John Abbott

- Target version changed from CoCoALib-0.99650 November 2019 to CoCoALib-0.99800

#13 - 10 Mar 2020 15:48 - John Abbott

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

#14 - 08 Mar 2024 17:36 - John Abbott

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