

CoCoA-5 - Design #490

Duplicate fns: valuation and FactorMultiplicity

21 Mar 2014 14:35 - John Abbott

Status:	Closed	Start date:	21 Mar 2014
Priority:	Normal	Due date:	
Assignee:	John Abbott	% Done:	100%
Category:	Cleaning	Estimated time:	3.90 hours
Target version:	CoCoA-5.1.2 summer 2015	Spent time:	3.50 hours
Description CoCoA-5 has two virtually identical fns valuation and FactorMultiplicity . We need only one of them! Decide the precise semantics, and fn name.			
Related issues: Related to CoCoALib - Slug #722: valuation slow for large inputs			
		Closed	31 May 2015

History

#1 - 21 Mar 2014 15:16 - John Abbott

FactorMultiplicity is defined in GCDFreebasis.cpkg5

#2 - 02 Apr 2014 17:34 - Anna Maria Bigatti

- Target version set to CoCoA-5.1.0 Easter14

#3 - 04 Apr 2014 16:36 - John Abbott

- Target version changed from CoCoA-5.1.0 Easter14 to CoCoA-5.1.1 Seoul14

#4 - 02 Sep 2014 11:11 - John Abbott

- Target version changed from CoCoA-5.1.1 Seoul14 to CoCoA-5.1.2 summer 2015

#5 - 28 Jun 2015 21:06 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

After looking on Wikipedia, it seems that **valuation** is a fairly advanced concept with several different (but related) meanings.

In our simple circumstance, it seems better to go with a name like FactorMultiplicity or perhaps even PrimeFactorMultiplicity? The name is more easily comprehended too.

JAA is inclined to go for a conservative definition: it requires that the factor be prime (this is also a requirement for the more general notion of valuation).

Comments? Suggestions? Better name?

#6 - 28 Jun 2015 21:29 - John Abbott

Just to make life more entertaining...

- FactorMultiplicity is defined in GCDFreeBasis.cpkg5, and does not require the factor to be prime; it expects the factor to be the second arg
- valuation is an exported CoCoALib function, requires the factor to be prime, and the factor should be the first arg!

Which do you prefer? FactorMultiplicity(2,N) or FactorMultiplicity(N,2)?

I had wondered about the name multiplicity but that is already used; it is also slightly less clear.

#7 - 29 Jun 2015 07:52 - Anna Maria Bigatti

- Assignee set to John Abbott

I like it as it is:

```

/**/ FactorMultiplicity(20, 2);
2
/**/ FactorMultiplicity(20, 10);
1
/**/ FactorMultiplicity(20, 7);
0

```

#8 - 29 Jun 2015 14:01 - John Abbott

The implementations appear to allow the factor (whose multiplicity is to be computed) to be composite. If the factor is restricted to be prime then $FM(p, A*B) = FM(p, A) + FM(p, B)$ where FM is an abbreviation for FactorMultiplicity. Without the restriction we have a weaker condition $FM(n, A*B) \geq FM(n, A) + FM(n, B)$; in general we have $0 \leq FM(n, A_1 * A_2 * \dots * A_k) - \sum(FM(n, A_j)) < k$.

Is it better to use the more restrictive definition (*i.e.* requiring the factor to be prime), or to allow a wider definition?

If we adopt the more restrictive definition, and then later decide we want the more general one, then there should not be any problems of backward incompatibility. The reverse situation (initially wide then later more restrictive) could cause such problems.

#9 - 29 Jul 2015 23:00 - John Abbott

I implemented the restricted version of FactorMultiplicity then Anna discovered that this broke LinearSimplify -- it took quite a while to find the root cause!

The problem that caused the error when running the manual example for LinearSimplify was that I had forgotten to swap the args in the call to FactorMultiplicity. The expected args are: (small-prime, big-integer). But the actual call gave the param values (big-integer, small-prime), and since the first param was not prime, an error was thrown. I have now corrected the code for LinearSimplify.

In fact, LinearSimplify does not guarantee that the small-factor is really a prime; so there probably do exist inputs which will cause FactorMultiplicity to throw. How to rectify this? Change LinearSimplify so that it considers only small factors which are prime? Or change FactorMultiplicity so that it accepts non-primes as the first arg?

#10 - 29 Jul 2015 23:14 - John Abbott

- % Done changed from 10 to 40

When the function was called valuation the restriction to prime "bases" was more or less obligatory. Now that it is called FactorMultiplicity the need to restrict to prime "bases" is not so evident.

The multiplicity of the factor is well-defined: assuming args are integers or elements of a TrueGCDDomain, and that the "base" is not invertible or a zero-divisor. So a wider definition is certainly possible (and just as easy to implement).

The narrower definition could in principle incur a significant cost of checking that the first arg really is a prime -- or do we cheat and accept "probable primes"?

Observe that the current, narrower definition did actually help in locating the bug in LinearSimplify; without the check that the first arg is prime, the function LinearSimplify would have completed, but the result would have been wrong.

Overall, JAA thinks it makes slightly more sense to implement the wider definition. Perhaps this can be done in two stages: for C5.1.2 the args must be integers (but the base may be non-prime), then for C5.1.3 we can extend to elements of a TrueGCDDomain.

What do you think?

#11 - 30 Jul 2015 16:20 - John Abbott

- Status changed from In Progress to Feedback

- % Done changed from 40 to 90

- Estimated time set to 3.90 h

After discussing with Anna we accept my proposal in note 10: namely, this version allows the "base" to be non-prime (but must be positive, actually greater than 1).

I'll add to [#722](#) to make the generalization to TrueGCDDomain, and to make the recursive impl. That will allow this issue to be closed shortly.

#12 - 30 Jul 2015 17:15 - John Abbott

- Status changed from Feedback to Closed

- % Done changed from 90 to 100