

CoCoA-5 - Feature #723

Referring to indets with multiple indices from a polyring

02 Jun 2015 15:35 - John Abbott

Status:	New	Start date:	02 Jun 2015
Priority:	Normal	Due date:	
Assignee:		% Done:	0%
Category:	enhancing/improving	Estimated time:	0.00 hour
Target version:	CoCoA-5.4.2	Spent time:	1.00 hour
Description			
I would like to be able to refer to multi-index indets in a poly ring using their multiple indices... where the ring has been passed in as a parameter.			
Related issues:			
Related to CoCoA-5 - Bug #1215: RationalSolve: gives "Error: must be non-zero"			Closed 07 Aug 2018
Related to CoCoA-5 - Feature #1469: Get indexed indets from a polyring			New 25 Jun 2020

History

#1 - 02 Jun 2015 15:39 - John Abbott

Here is an example I want to run (NB it is very slow)

```
p := 3;
n := 3;
use ZZ/(p) [s[1..n,1..n,1..n]];

define MortenPolyList(n)
  TopLevel s;
  PolyList := [];
  for i := 1 to n do
    for j := 1 to n do
      for k := 1 to n do
        for m := 1 to n do
          f := sum([s[i,j,l]*s[l,k,m] - s[j,k,l]*s[i,l,m] | l in 1..n]);
          append(ref PolyList, f);
        endfor;
      endfor;
    endfor;
  endfor;
  return PolyList;
enddefine; -- MortenPolyList

L := MortenPolyList(n);
solns := RationalSolve(L);
```

Note that I have "uncleanly" used TopLevel s; inside the function. What I wanted to do was pass in the poly ring as an argument (or have the function create the poly ring). But then how do I write the rest of the function neatly using the triple-indexed indets?

PS if we get this right then we're far neater than GAP >-}

## #2 - 02 Jun 2015 15:45 - John Abbott

The existing function `indets("s")` almost does what I want except that the result is a list rather than an `INTMAP`. For my example, I need a version which produces an `INTMAP`.

I do have some doubts about "clean" this proposal is. Suppose we changed `indets("s")` to produce what I want... Then a program indexing into the result presupposes a certain arity for the indices; and not only the arity, but also that the available valid tuples of indices are also "suitable". These are quite strong suppositions, but certainly reasonable if the ring has been created inside the function; they are possibly also reasonable for not-very-public functions where we can be fairly sure the caller won't do anything silly.

Opinions? Suggestions? Other ideas?

## #3 - 02 Jun 2015 16:18 - Anna Maria Bigatti

Is this what you want? (From the manual ;-)

```
/**/ S := QQ[x,y,z[1..4,3..7]];
/**/ 7*RingElem(S, ["z",2,5]);
7*z[2,5]
```

## #4 - 02 Jun 2015 16:35 - John Abbott

Yes and no. I thought the topic had come up recently, but did not find it on redmine. Perhaps the manual page for `indets` should include a reference to an example like yours.

What I do not like about your example is that it is cumbersome -- it is not so easy to read an expression containing `RingElem(S, ["z",1,2])` certainly compared to `z[1,2]`. My example above beats GAP exactly because it is so simple and clear.

I would definitely prefer something like:

```
S := QQ[x,y,z[1..4,3..7]];
z := IndetIndexer(S,"z"); // yes, this is a bit ugly
7*z[1,2];
```

We would also need a good name (instead of `IndetIndexer`). Since we have already defined the type `INTMAP` it seems a shame not to make it more accessible...

**#5 - 07 Aug 2018 18:06 - John Abbott**

The example comment 1 gives ERROR: Value must be non-zero when calling RationalSolve. This cannot be right!

**#6 - 07 Aug 2018 18:10 - John Abbott**

- Related to Bug #1215: RationalSolve: gives "Error: must be non-zero" added

**#7 - 30 Oct 2020 12:29 - John Abbott**

- Related to Feature #1469: Get indexed indets from a polyring added

**#8 - 12 Feb 2021 00:17 - John Abbott**

- Target version changed from CoCoA-5.?.? to CoCoA-5.4.2

**#9 - 12 Feb 2021 12:16 - John Abbott**

The example in comment 2 is slow because it takes along time to test IsZeroDim *i.e.* compute a GB.

I stopped the GB calc after 6500 elems in the partial GB.