

CoCoALib - Feature #747

New function for making list of symbols (indeterminate names)

27 Jul 2015 12:07 - Anna Maria Bigatti

Status:	Closed	Start date:	27 Jul 2015
Priority:	Normal	Due date:	
Assignee:	John Abbott	% Done:	100%
Category:	Improving	Estimated time:	3.00 hours
Target version:	CoCoALib-0.99538 summer 2015	Spent time:	2.70 hours
Description CoCoA-5 has the nice "shortcut syntax" $R ::= QQ[x,y,z]$ and $R ::= QQ[x[1..4]]$. In CoCoALib this is often trickier (there are the functions symbols with up to 4 strings, and SymbolRange). Make this simpler.			

History

#1 - 27 Jul 2015 12:14 - John Abbott

- % Done changed from 0 to 20

JAA has implemented SymbolList which accepts a string of symbols, and returns the symbols as a vector<symbol>.

It does not handle symbol ranges!

NOTE: it was written for use in a demo at ISSAC 2015

NOTE2: no documentation :-)

#2 - 27 Jul 2015 12:27 - Anna Maria Bigatti

- % Done changed from 20 to 30

I inserted JAA's code in CoCoALib (with a minimal extra check on missing ",")
It is in symbol.H and C.

I renamed SymbolList into symbols, thus removing the old symbols("x") which accepted a string containing just one symbol (so it is backward compatible!).

Should we remove

```
symbols("x", "y");  
symbols("x", "y", "z");  
symbols("x", "y", "z", "w");
```

?

Now they can be called with the nicer syntax

```
symbols("x, y");  
symbols("x, y, z");  
symbols("x, y, z, w");
```

#3 - 27 Jul 2015 13:42 - Anna Maria Bigatti

I changed all the examples, and commented out the old symbols with 2,3, and 4 arguments.

```
*** Good news: all examples ran successfully. ***
```

checking-in...

#4 - 30 Jul 2015 17:17 - John Abbott

- Target version changed from CoCoALib-0.99536 June 2015 to CoCoALib-0.99540 Feb 2016

#5 - 30 Jul 2015 17:37 - John Abbott

- Target version changed from CoCoALib-0.99540 Feb 2016 to CoCoALib-0.99538 summer 2015

#6 - 09 Nov 2015 15:21 - John Abbott

Do we want this function to handle symbol ranges?

e.g. `symbols("x[1..3,2..5]")` would produce the same result as `SymbolRange(symbol("x",1,2),symbol("x",3,5))`

I do not expect symbols to replace `SymbolRange` since the latter can easily accept subscripts determined at run-time (whereas symbols would require its string arg to be created at run-time, presumably via `ostringstream`).

It probably makes sense for symbols to accept ranges; it'll take some time to implement is well, I fear.

#7 - 09 Nov 2015 23:22 - Anna Maria Bigatti

John Abbott wrote:

Do we want this function to handle symbol ranges?

It probably makes sense for symbols to accept ranges; it'll take some time to implement is well, I fear.

No, better not. Not worth the effort. `SymbolRange` is good enough.

#8 - 10 Nov 2015 10:17 - John Abbott

- % Done changed from 30 to 50

That's fine by me -- less work for me :-)

Indeed `symbols("x[1..3,2..5]")` is not really that much more readable than `SymbolRange(symbol("x[1,2]"),symbol("x[3,5]"))`

Ahhh, there's a catch! I'm not sure we allow `symbol("x[1,2]")`, but instead require `symbol("x",1,2)`. Should `symbol("x[1,2]")` be allowed?

#9 - 01 Feb 2016 13:38 - John Abbott

- *Status changed from In Progress to Feedback*

- *% Done changed from 50 to 90*

This new fn `symbols` has been working fine for at least 3 months, so I have deleted the old ones (both from `symbol.H` and `symbol.C` files).

The suggestion of allowing `symbol("x[1,2]")`, while tempting, could be confusing, and perhaps not so easy to document. If we do allow it, what happens if someone calls `symbol("x[1,2]",3,4)`? Also `symbols("x[1,2]")` does almost the same thing.

I suggest waiting before extending `symbol` to accept a string arg which contains indices. It is also not so clear to me when such a function might actually be useful.

#10 - 01 Feb 2016 14:21 - John Abbott

- *Status changed from Feedback to Closed*

- *% Done changed from 90 to 100*

There's no real point in keeping this issue open. I've added a reference to this issue in the doc for `symbol`, just in case we want to revisit the decision about `symbol("x[1,2]")`.

Closing.