CoCoA-5 - Design #1477

Packages: questions from Andraschko

06 Aug 2020 15:15 - John Abbott

Status:	Closed	**Start date:**	06 Aug 2020
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	enhancing/improving	**Estimated time:**	2.33 hours
Target version:	CoCoA-5.4.2	**Spent time:**	2.35 hours

Description

Bernhard Andraschko asks:

- **(A) ?package** yields the manual for **export**. This is probably not intended.
- **(B)** The manual for packages contains the **See also: ? CoCoA Packages**, but **? CoCoA Packages** doesn't provide any result.
- **(C)** If I write a package, i must export one of the functions. Why do I have to do that?
  Maybe I just want to write a package $sthg and then access all functions in it by calling $sthg.foo()
- **(D)** Required packages. Originally I had one package **$subalgebra** and another called **$sagbi** and both of them required the other one to work, but i can't put source $otherOne in both files since then they would "load itself" recursively. Also **IsDefined($subalgebra)** gives an error, maybe there is a different syntax producing what I want. Something similar to #ifndef in C++ would be useful in CoCoA?
- **(E)** Documentations. Is there any way to access package documentations without looking in the source code? I only see the function **describe** which gives me all functions in a package, but e.g. there is a package **$sppoly** and I have no idea what it does (and no idea how to find out).

Related issues:

Related to CoCoA-5 - Support #1474: Website: link to JSAG?	**New**	**06 Aug 2020**
Related to CoCoA-5 - Design #1495: Package exporting nothing?	**Closed**	**02 Oct 2020**

History

#1 - 06 Aug 2020 15:16 - John Abbott

- Related to Support #1474: Website: link to JSAG? added

#2 - 10 Aug 2020 14:06 - John Abbott

- Description updated

- Category set to enhancing/improving

- Status changed from New to In Progress

- Target version set to CoCoA-5.4.0

- % Done changed from 0 to 10

Point **(C)** is not correct: a **warning** is issued if nothing is exported (but no error is triggered). JAA+AMB think this is reasonable.

#3 - 10 Aug 2020 16:02 - John Abbott

JAA is not convinced about using **IsDefined** to check whether a package has been loaded.
One solution is to search through the packages listed in **packages()**; note that the package names are strings here!
Andraschko asked whether another name for packages might not be better: *e.g.* **ListOfPackages** or **PackageList**

Being able to test whether a package is present could be useful during development... I wonder whether it should also say at what time the package was loaded (might also be useful info).

A problem with **Requires** is that CoCoA does not usually know in which directory/ies to search for packages under development. Maybe we should add a "package path" feature? If so, how exactly would that work?

One problem with specifying paths is that two packages with the same name but in different directories can lead to unexpected behaviour if just the

first matching file is used. Maybe the whole path should be searched, and an error produced if more than one file matches? Should the path length be limited? Why?

UPDATE Now I see that there is already a **CocoaPackagePath**, but that contains a single directory (for the "official packages").

#4 - 10 Aug 2020 16:09 - John Abbott

Regarding point **(B)**. Andraschko is right that the current documentation is not good (and not well-organized).
After speaking to him, I think that we should make the first step and improve the doc; then ask Andraschko for feedback on improving it.

I have added the keyword package to another entry; now the list of related manual entries is

```
? CocoaPackagePath
? export
? packages
? Introduction to Packages
? First Example of a Package
? Package essentials
? Sharing Your Package
? Commands and Functions for Packages
? Supported Packages
? Galois Package
```

#5 - 10 Aug 2020 22:56 - John Abbott

Here is a short excerpt from an email in the JSAG editors forum:
Here's the list of advice to authors that we've accumulated over the years, for Macaulay2:
https://github.com/Macaulay2/M2/wiki/Package-Writing-Style-Guide

Macaulay2 has extensive guidelines about writing packages. To be honest, this seems like overkill for CoCoA (at the moment, anyway). However, the idea of better guidelines is surely good.

#6 - 30 Sep 2020 11:43 - John Abbott

Regarding point **(C)** *why must a package export at least 1 function?*

A work-around would be to create a function with a long and complicated "random" name and export that; but this would pollute the global namespace.

I do not know what Andraschko's specific situation is, but I could imagine a package of "common internal" functions which are used in two other separate packages, but these "common internal" function are not appropriate for public consumption.

We could also allow something like **export NOTHING** at the start of a package which suppresses the warning; this would also tell the reader of the package code that it is intentional that no functions be exported.

The simplest change is just to remove the code which issues the warning... not sure that that is the best solution.

Comments, suggestions? Perhaps we can discuss on Friday?

NOTE to avoid introducing new keywords we could used **export skip** as the signal that nothing is to be exported.

#7 - 02 Oct 2020 07:24 - Anna Maria Bigatti

John Abbott wrote:

> Regarding point **(C)** *why must a package export at least 1 function?*
> We could also allow something like **export NOTHING** at the start of a package which suppresses the warning; this would also tell the reader of the package code that it is intentional that no functions be exported.

I like that. Let's look for the best keyword.

Dedicated issue: #1495

#8 - 02 Oct 2020 11:31 - Anna Maria Bigatti

- Related to Design #1495: Package exporting nothing? added

#9 - 03 Jul 2021 17:25 - John Abbott

- % Done changed from 10 to 50

#10 - 16 Feb 2022 20:19 - John Abbott

- Status changed from In Progress to Resolved

- % Done changed from 50 to 80

Point **(E)** is a very valid point; does Anna have a solution?

#11 - 18 Feb 2022 17:10 - John Abbott

- Target version changed from CoCoA-5.4.0 to CoCoA-5.4.2

#12 - 09 Mar 2023 22:56 - John Abbott

- Status changed from Resolved to Feedback

- Assignee set to John Abbott

- % Done changed from 80 to 90

Current state (2023-03-09):
(A) and **(B)** and **(C)** have effectively been solved.
For **(D)** there is a workaround: write some code to scan through the output of **packages()**,
this could be done in a non-exported function:
 e.g. define CheckDependencies() if not("Berni.cpkg5" isin packages()) then error("Berni is missing"); endif; enddefine;

Point **(E)** still needs to be resolved, I fear.

#13 - 14 Mar 2023 21:57 - John Abbott

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

- Estimated time set to 2.33 h

The **CheckDependencies** proposal in comment 12 is a tolerable workaround (for the moment).

The issue of documentation is thorny. It won't be easy to solve.
I would certainly recommend against what "Julia" does: it produces lots of nice-looking documentation which is horrible to use!

I'm closing this issue. Maybe we should make a new one about documentation?