CoCoALib - Feature #1815

JSON and UUID

02 May 2024 19:20 - John Abbott

Status:	In Progress	**Start date:**	02 May 2024
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	10%
Category:	New Function	**Estimated time:**	0.00 hour
Target version:	CoCoALib-0.99900	**Spent time:**	1.70 hour

Description

Investigate including a library for serializing and deserializing JSON encodings.
This includes having the ability to generate UUIDs; for we will need a library.

This is in preparation for a meeting in Berlin (on 2024-05-16)

History

#1 - 02 May 2024 19:29 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

Antony in Berlin says that they use **rapidJSON**; I did not enquire about UUIDs -- we can use an interim, makeshift solution for UUIDs.

I am inclined not to use rapidJSON, but some other library instead. Part of the reason is that if we use a different library then we are perhaps more likely to detect any non-standard behaviour.

BOOST has a JSON library; I think this is a good first candidate to try/explore. Also coming from BOOST inspires confidence (in terms of reliability, correctness, future maintenance, etc.). The documentation I found was less helpful than I'd hoped... I'll keep looking!

For UUIDs there is a Linux library called **libuuid** which seems to be an obvious candidate; as commented above, investigation into it can be postponed to after the Berlin meeting.

#2 - 02 May 2024 19:41 - John Abbott

We should also decide which goals we have.

I'm expecting in Berlin to spend much of the time discussing and planning, and not so much time actually implementing (since I can do that mostly without Antony's help). I'm writing here my initial thoughts, which we shall discuss and revise. I propose:

- initially matrices over ZZ
- matrices over QQ and
- matrices over a small, prime finite field (this will likely require UUIDs to ensure that the same field used)

Matrices are a good starting point because it would facilitate comparing determinant implementations in CoCoALib and in OSCAR; also they are non-trivial data-structures (without being unduly complicated either).

Once matrices can be handled correctly, we can try polynomial rings, and their elements: this would permit use of the "Groebner mill" in CoCoALib, and also *vice versa.* To avoid too much complication, we shall restrict to polynomial rings over QQ or small, prime finite fields.

#3 - 02 May 2024 19:46 - John Abbott

I note the following link(s):

- **https://www.boost.org/doc/libs/1_85_0/libs/json/doc/html/json/comparison.html**
- **https://stackoverflow.com/questions/17124652/how-can-i-parse-json-arrays-with-c-boost**
- **https://stackoverflow.com/questions/1089741/how-do-i-obtain-use-libuuid**
- **https://www.man7.org/linux/man-pages/man3/uuid_generate.3.html**
- **https://stackoverflow.com/questions/15206705/reading-json-file-with-boost/15207050#15207050**

#4 - 02 May 2024 19:58 - John Abbott

Another point to bear in mind is the licence of any library.
We already use BOOST in CoCoA-5, but I'm not sure whether we checked that such use is permitted by the two licences involved (GPLv3 and BOOST's own licence).