Masters Report Proposal

Computer Aided Combinatorial Configurations in Sage

William J. Laffin

June 19, 2010

1 Objective

CACCiS will provide a computer aided system for investigating combinatorial configurations such as: designs, codes, extremal set systems, and graph factorizations among others. These will be supported by a large database of known constructions. It is meant to enable work with objects that are easy to define pictorially, but difficult to robustly work with in a programing language.

It is meant to provide a generic framework for solving existence, uniqueness, and optimization problems in combinatorial configurations.

2 Description

2.1 Components

1. Structure Description - Describe the objects
2. Combinatorial Constructions - Construct the objects
3. Database of Combinatorial Objects and Descriptions
4. Help

2.2 Capabilities of Structure Description

1. Control of Domain
   - Sets (Cross products, multi-sets)
   - Groups (Semi-direct products, orbits, G-spaces, semi-groups)
   - Fields/Rings (Polynomial, etc)
   - User Defined

2. Arrays (any dimension)

3. Logical constructs

   - Quantifiers(existence, existence with uniqueness, for all)
   - Element assertions ($\in$)
   - Domain specific assertions

Proposed Logic context for python grammar

$$\begin{array}{rcl}
\langle\text{Sentence}\rangle & \Rightarrow & \langle\text{AtomicSentence}\rangle \\
 & | & (\ \langle\text{Sentence}\rangle\ \langle\text{Connective}\rangle\ \langle\text{Sentence}\rangle\ ) \\
 & | & \langle\text{Quantifier}\rangle(\ (\ \langle\text{Variable}\rangle, \ldots), \langle\text{Sentence}\rangle\ ) \\
 & | & \text{`}\texttt{not}\text{'}\ \langle\text{Sentence}\rangle \\
\langle\text{AtomicSentence}\rangle & \Rightarrow & \langle\text{Predicate}\rangle(\langle\text{Term}\rangle, \ldots) \\
 & | & \langle\text{Term}\rangle\ \text{`}\texttt{==}\text{'}\ \langle\text{Term}\rangle \\
\langle\text{Term}\rangle & \Rightarrow & \langle\text{Function}\rangle(\langle\text{Term}\rangle, \ldots\ ) \\
 & | & \langle\text{Constant}\rangle \\
 & | & \langle\text{Variable}\rangle \\
\langle\text{Connective}\rangle & \Rightarrow & \text{`}\texttt{|implies|}\text{'} \\
 & | & \text{`}\texttt{and}\text{'} \\
 & | & \text{`}\texttt{or}\text{'} \\
 & | & \text{`}\texttt{|iff|}\text{'} \\
\langle\text{Quantifier}\rangle & \Rightarrow & \text{`}\texttt{forall}\text{'} \\
 & | & \text{`}\texttt{exists}\text{'} \\
\langle\text{Constant}\rangle & \Rightarrow & \text{Declared Constant} \\
\langle\text{Variable}\rangle & \Rightarrow & \text{Declared Variable} \\
\langle\text{Predicate}\rangle & \Rightarrow & \text{Boolean python function} \\
\langle\text{Function}\rangle & \Rightarrow & \text{Python function}
\end{array}$$

4. Labels(for humans) of sub-parts

5. Seamlessly working with previously defined objects

6. Some sort of comparison/coercion integration

7. Interface with databases

8. Enable properties to be defined (An example would be resolvable)

2.3 Capabilities of Combinatorial Constructions

1. Interface with Structure Description

2. Able to be a function of structures described

3. Allows set construction syntax (Allowing the same logical constructs as in 3 in subsection 2.2)
4. Interface with both databases

2.4 Capabilities of Help

The help system will be in depth and easy entrance, with links to outside sources concerning the material.

2.5 Capabilities of the databases

The databases will be modest in size and breadth. They will be interfaces to existing configurations and constructions. Writing to a personal, persistent database and submitting to public repository will be seamless.

2.6 Capabilities of Help

The help system will be in depth and easy entrance, with links to outside sources concerning the material.

2.7 What CACCiS will not be

DTiS is not an expert system. It is not meant to replace experts in the field. Contrarily, it is meant to help experts in the field find and disseminate knowledge.

The system is also not the CRC Handbook of Combinatorial Designs[3] coded in computer form. The handbook encodes far too big of objects to be stored stored in one place. However, most constructions described therein will be able to be done concisely, legibly, and simply.

3 Interface

The interface will be the object oriented language Python. This is because most of Sage[8] is written in Python, and Sage has already implemented complicated domain information. Sage also has interfaced with GAP, to enable complicated group theoretic capabilities to this system.

4 Previous Work

4.1 Out of Sage

There are numerous ways for one to work with combinatorial configurations.

- Groups and Graphs [5] - Graph embeddings, visual editor, automorphisms, Cayley graphs, orbits
- Matlab(Subset) [1] - This is also available as a C version and is open source. Supports Gray codes among others.
- Mathematica(Combinatorica)[9] - Graph Theory, Graph Database,“ Combinatorica extends Mathematica by over 450 functions in combinatorics and graph theory”[10]
- Maple [6] - MOLS, Generating Functions, Attribute Grammars
- Magma [2] - Enumerative, Partitions, Words, Young Tablueux, Symmetric functions, incidence structures and designs, Hadamard Matrices, Graphs, Multi-graphs, Networks
- Hundreds of others. [7]

4.2 In Sage

Design theory in Sage currently involves 4 files concerning the topic of block designs. These implement

- Incidence structures
- Block designs
- Covering designs
- External representations of block designs

They are a port of what DesignTheory.org has released **pydesign**, written by Dr. Peter Dobcsanyi, part of a research project[4] funded by the UK Engineering and Physical Sciences Research Council.

5 Schedule

I hope to have working wrappers of Dr. Kreher’s toolkit and Discreta by August.

By December, I hope to have the wrappers of GAP, Discreta, Dr. Kreher’s Toolkit, and nauty all easily talking to each other.

By February, I hope to have everything in Sage that is necessary for complicated combinatorial structures, such as those listed in section 3, to be entered without much friction.

References

[1] John Burkardt, *Subset - coroutines for combinatorics*, May 2010, `http://people.sc.fsu.edu/ burkardt/m_src/subset/subset.html`.

[2] C.J. Colbourn, *Triple systems*, Handbook of Combinatorial Designs (J.H. Dinitz C.J. Colbourn, ed.), Chapman & Hall/ CRC, 2006, pp. 58–72.

[3] John Cannon et al., *Magma for combinatorics*, May 2010, `http://magma.maths.usyd.edu.au/magma/htmlhelp/part19.htm`.

[4] Maplesoft et al., *Maple for combinatorics*, May 2010, `http://magma.maths.usyd.edu.au/magma/htmlhelp/part19.htm`.

[5] Netscape et al., *Open directory: Combinatorics software*, May 2010, `http://www.dmoz.org/Science/Math/Combinatorics/Software/`.

[6] Peter Dobcsnyi et al., *Designtheory.org website*, May 2010, `http://designtheory.org/`.

[7] Stephen Wolfram et al., *Combinatorica*, May 2010, `http://www.combinatorica.com/`.

[8] ———, *Online mathematica help*, May 2010, `http://reference.wolfram.com/mathematica/Com`

[9] William Stein et al., *Sage website*, May 2010, `http://www.sagemath.org/`.

[10] Bill Kocay, *Groups and graphs*, May 2010, `http://www.combinatorialmath.ca/g&g/index.html`.