International Journal of Theoretical Physics, **39**, 2381–2406 (2000).

Isomorph-Free Exhaustive Generation of Greechie Diagrams and Automated Checking of Their Passage by Orthomodular Lattice Equations

Brendan D. McKay[†1], Norman D. Megill[‡2], and Mladen Pavičić[*3]

[†]Dept. of Computer Science, Australian National Univ., Canberra, ACT, 0200, Australia.
[‡]Boston Information Group, 30 Church St., Belmont MA 02478, U. S. A.
[*]Dept. of Physics, Univ. of Maryland Baltimore County, Baltimore, MD 21250, U. S. A.
and University of Zagreb, Gradjevinski Fakultet, Kačićeva 26, HR-10000 Zagreb, Croatia.

Abstract. We give a new algorithm for generating Greechie diagrams with arbitrary chosen number of atoms or blocks (with $2, 3, 4, \ldots$ atoms) and provide a computer program for generating the diagrams. The results show that the previous algorithm does not produce every diagram and that it is at least 10^5 times slower. We also provide an algorithm and programs for checking of Greechie diagram passage by equations defining varieties of orthomodular lattices and give examples from Hilbert lattices. At the end we discuss some additional characteristics of Greechie diagrams.

PACS numbers: 03.65.Bz, 02.10.By, 02.10.Gd
Keywords: orthomodular lattices, Greechie diagrams, exhaustive combinatorial generation, Hilbert space equations

[1]E-mail: bdm@cs.anu.edu.au; Web page: http://cs.anu.edu.au/~bdm
[2]E-mail: nm@alum.mit.edu; Web page: http://www.shore.net/~ndm/java/mm.html
[3]E-mail: mpavicic@faust.irb.hr; Web page: http://m3k.grad.hr/pavicic

1 Introduction

To arrive at a Hilbert space representation of measurements starting from plausible "physical" axioms has been a dream of many physicists and mathematicians for almost seventy years. Of course, one could not expect to recognize the axioms from nothing but experimental data because they—provided they exist—must be rather involved. Therefore the scientists took the opposite road by starting with Hilbert space and trying to read off essential mathematical properties so as to be able to eventually simplify them and arrive at the simple physically plausible axioms.

The first breakthrough along the *opposite-road* was made by Birkhoff and von Neumann in 1936 [1] who recognized that a modular lattice which can be given a physical background underlies every finite dimensional Hilbert space. In the early sixties Mackey [2] (and Zierler [3]) arrived at six axioms for a poset (partially ordered set) of physical observables which he essentially *read off* from the Hilbert space properties. In an additional famous *seventh* axiom, he then *postulated* that the latter poset be isomorphic to the one of the subspaces of an infinite-dimensional Hilbert space. A few years later Piron[4], MacLaren [5], Amemyia and Araki [6], and Mączyński [7] starting with such a poset with infima and suprema of every two-element subset (lattice) and using similar axioms, proved that the lattice—usually called the *Hilbert lattice*—is isomorphic to a pre-Hilbert space. That enabled Mączyński [7] to postulate only the kind of a field over which the Hilbert space should be formulated: he chose the complex one. The Hilbert lattice is then a lattice of the subspaces of the Hilbert space.

At that time it seemed that only two other fields could have been postulated: the real and the quaternionic ones. But in the early eighties Keller [8] showed that there are other non-standard (non-archimedean) fields over which a Hilbert space can be defined. Also, the axioms themselves proved to be too complicated to be given plausible physical support or simplified. Thus, the whole project lost its appeal and the majority of researchers left the field. However, in 1995 Maria Pia Solèr [9] proved that an infinite dimensional Hilbert space can only be defined over either real, or complex, or quaternionic field (i.e., that only finite dimensional ones allow non-standard fields).

The latter result renewed the interest in the problem of reconstructing the Hilbert space from an algebra of observables. [10, 11, 12, 13, 14, 15] Also, recently devised quantum computers prompt for such a reconstruction from an algebra which is in the field of quantum computing usually called quantum logic in analogy to classical logic underlying classical computers. In particular, if we wanted quantum computers to function as quantum simulators, i.e., to directly simulate quantum systems through their description in the Hilbert space, we apparently have to start from such an algebra. For, this would be the only presently conceivable way of typing in the Hamiltonian at the console of the quantum simulator. This also means that we have to go around the present standard axioms for the Hilbert lattice not any more because they are complicated and physically non-grounded but because they include universal and existential quantifiers which are unmanageable by a quantum computer. A way to do so would be to find lattice equations as substitutes for the axioms. Hilbert lattice satisfies not only the orthomodularity equation but a number of other equations as well. Thus, if we started with a such a lattice—which can easily be physically supported

by e.g. a quantum computer design—we could obviously simplify the axioms and possibly ultimately dispense with them. The problem is that only two groups of the equations satisfied by Hilbert lattices—i.e., in any Hilbert space—have been found so far. We do not know whether the Hilbert space equations form a recursively enumerable set, i.e., whether we can determine them all. What we can do however is to try to find as many such equations as possible and group them according to their recursive algorithms. Each such equation can simplify the present axioms.

However, since already equations with 4 variables contain at least about 30 terms which one cannot further simplify, a proper tool for finding and handling the equations is indispensable. As a great help came Greechie diagrams (condensed Hasse diagrams) which we will define precisely later on. E.g., to find that two equations cannot be inferred from each other it suffices to find two Greechie lattices which the equations interchangeably pass and fail. As an illustration of how "easily" one can find a lattice without a computer program we cite Greechie himself: "[In 1969] a student, beginning his dissertation, found such a lattice. It was terribly complicated and had about eighty atoms. The student left school and the example was lost. I've been looking for one ever since. Recently [in 1977!] I found one." [16] So, we need an algorithm for finding Greechie diagrams and another for finding whether a particular equation passes or fails them. In this paper we give both. They would not only support the afore-mentioned project of obtaining the Hilbert space from physically plausible axioms but would also serve for obtaining new equations in the theory of Hilbert spaces.

The first attempt at automated generation of Greechie diagrams was made in the early eighties by G. Beuttenmüller a former student of G. Kalmbach. [17, pp. 319-328] The algorithm itself is not given in the book but G. Beuttenmüller kindly sent us the listing of its translation into Algol. We rewrote it in C, and with a fast PC it took about 27 days to generate Greechie diagrams with 13 blocks. We estimated it would take around a year for 14 blocks and half a century for 15 blocks, so we looked for another approach.

The technique of *isomorph-free exhaustive generation* [18] of Greechie diagrams gave us not only a tremendous speed gain—48 seconds, 6 minutes, 51 minutes, 8 hours and 122 hours for 13–17 blocks, respectively (for a PC running at 800 MHz)—but also essentially new results: Beuttenmüller's algorithm must be at least incomplete since the numbers of non-isomorphic Greechie diagrams in Kalmbach's book ([17], p. 322) are wrong. In Sec. 2 we give the algorithm for the above generation.

In Sec. 3 we give an algorithm for checking whether a particular equation fails or passes in Greechie diagrams provided by the algorithm from Sec. 2. The algorithm has helped us to find new equations that hold in any infinite dimensional Hilbert space. [19]

2 Isomorph-free exhaustive generation of Greechie diagrams

The following definitions and theorem we take over from Kalmbach [17] and Svozil and Tkadlec [20]. Definitions in the framework of *quantum logics* (σ-orthomodular posets) the reader can find in the book of Pták and Pulmannová. [21]

Definition 2.1. *A* diagram *is a pair* (V, E)*, where* $V \neq \emptyset$ *is a set of* atoms *(drawn as points) and* $E \subseteq \exp V \setminus \{\emptyset\}$ *is a set of* blocks *(drawn as line segments connecting corresponding points). A* loop *of order* $n \geq 2$ *(*n *being a natural number) in a diagram* (V, E) *is a sequence* $(e_1, \ldots e_b) \in E^n$ *of mutually different blocks such that there are mutually distinct atoms* $\nu_1, \ldots, \nu_n$ *with* $\nu_i \in e_i \cap e_{i+1}$ $(i = 1, \ldots, n,\ e_{n+1} = e_1)$.

Definition 2.2. *A* Greechie diagram *is a diagram satisfying the following conditions:*

(1) Every atom belongs to at least one block.

(2) If there are at least two atoms then every block is at least 2-element.

(3) Every block which intersects with another block is at least 3-element.

(4) Every pair of different blocks intersects in at most one atom.

(5) There is no loop of order 3.

Theorem 2.3. *For every Greechie diagram with only finite blocks there is exactly one (up to an isomorphism) orthomodular poset such that there are one-to-one correspondences between atoms and atoms and between blocks and blocks which preserve incidence relations. The poset is a lattice if and only if the Greechie diagram has no loops of order 4.*

In the literature, a block is also called an *edge* and an atom is also called a *vertex* or *node*. (However, we reserve the term *node* for an element of a Hasse diagram.)

From the above definitions it is clear that a block can have not only 3 atoms but also 2 or 4 or more atoms. However, practically all examples of Greechie diagrams used in lattice theory are nothing but pasted 3-atom blocks. We are aware of only two important contributions containing 4-atom blocks (two proofs of the existence of finite lattices admitting no states given in [21, Fig. 2.4.5, p. 37] and [17, Fig. 17.3, p. 275] and of only one result for n and ∞ giving orthomodular lattices without states [17, Fig. 17.4, p. 275].

For this reason, we have initially focussed on generation of diagrams with every block having size 3. Nevertheless, our description of the generation algorithm will allow larger blocks in anticipation of the next version of our generation program. Our program for checking equations already handles large blocks. Also, we are interested only in the diagrams which correspond to lattices, i.e., only in those containing no loops of order 4. Since the condition (5) of Def. 2.2 states that there are no loops of order 3, this means that we are interested only in diagrams with loops of order 5 and higher. Those 3-atom Greechie diagrams which correspond to lattices we call *Greechie-3-L diagrams*.

A diagram is *connected* if, for each pair of atoms ν, ν', there is a sequence of blocks $e_1, e_2, \ldots, e_k$ such that $\nu \in e_1$, $\nu' \in e_k$ and $e_i \cap e_{i+1} \neq \emptyset$ for $1 \leq i \leq k-1$. In Section 3 we will illustrate how the properties of unconnected diagrams are not necessarily a simple combination of the properties of their connected components, so our algorithms will handle both connected and unconnected diagrams. An *isomorphism* from a diagram (V_1, E_1) to a diagram (V_2, E_2) is a bijection ϕ from V_1 to V_2 such that ϕ induces a bijection from E_1 to E_2. The isomorphisms from a diagram D to itself are its *automorphisms*, and together comprise its *automorphism group* $\mathrm{Aut}(D)$.

If $D = (V, E)$ is a diagram and $e \in E$, then $D - e$ is the diagram obtained from D by removing e and also removing any atoms that were in e but in no other block. Conversely, if e is a set of atoms (not necessarily all of them atoms of D), then $D + e$ is the diagram $(V \cup e, E \cup \{e\})$. Clearly $(D + e) - e = D$.

We will describe the generation algorithm in some generality to assist future applications. Suppose that $\mathcal{C}$ is some class of diagrams closed under isomorphisms (for example, connected Greechie-3-L diagrams). If $D = (V, E) \in \mathcal{C}$ and $|E| > 1$, there may be some $e \in E$ such that $D - e \in \mathcal{C}$. If there is no such block, we call D *irreducible*. It is obvious that all diagrams in $\mathcal{C}$ can be made from the irreducible diagrams in $\mathcal{C}$ by adding a sequence of blocks one at a time, all the while staying in $\mathcal{C}$. Such a sequence of diagrams is a *construction path* for D.

The basic idea behind our algorithm is to prune the set of construction paths until (up to isomorphism) each diagram in $\mathcal{C}$ has exactly one construction path. This is achieved by two techniques acting in consort.

The first technique is to avoid equivalent extensions. Suppose $D \in \mathcal{C}$ and e_1, e_2 are such that $D + e_1, D + e_2 \in \mathcal{C}$. $D + e_1$ and $D + e_2$ are called *equivalent extensions of* D if $|e_1| = |e_2|$ and there is an automorphism of D which maps $V \cap e_1$ onto $V \cap e_2$. It is easy to see that the equivalence of e_1 and e_2 implies the isomorphism of $D + e_1$ and $D + e_2$, by an isomorphism that takes e_1 onto e_2, so we do not lose any isomorphism types of diagram if we make only one of them.

The second technique is somewhat more complicated. Suppose we have a function m with the following properties.

- $m(\,)$ takes a single argument D which is a reducible diagram in $\mathcal{C}$. It returns a value which is an orbit of blocks under the action of $\mathrm{Aut}(D)$.
- $D - e \in \mathcal{C}$ for every $e \in m(D)$.
- If D' is a diagram isomorphic to D, then there is an isomorphism from D to D' that maps $m(D)$ onto $m(D')$.

We will explain how to compute such a function $m(\,)$ later; for now we will describe its purpose. Take any reducible $D \in \mathcal{C}$ and $e \in m(D)$, then form $D - e$. If we do the same starting with a diagram D' isomorphic to D—take $e' \in m(D')$, then form $D' - e'$—the third property of $m(\,)$ implies that $D - e$ and $D' - e'$ are isomorphic. Thus, the function $m(\,)$ enables us to define a unique isomorphism class, that of $D - e$ for $e \in m(D)$, as the *parent class* of the isomorphism class of D. Since we wish to avoid making isomorphism types more than once, we can decide to only make each diagram from its parent class. If we happen to make it from any other class, we will reject it.

The result of the theory in [18] is that the combination of the above two techniques results in each isomorphism class being generated exactly once. The precise method of combination is given by the following algorithm.

Definition 2.4. Isomorph-free Greechie diagram generation procedure

procedure *scan* (D : diagram; β : integer)

if D has exactly β blocks **then**
output D

else

for each equivalence class of extensions $D + e$ **do**
if $e \in m(D + e)$ **then** *scan*($D + e$,β)

end procedure

In Figure 1 we show the top four levels of the generation tree as produced by our implementation of the algorithm for connected Greechie-3-L diagrams. The lines joining the diagrams show the parent-child relationship. Between a parent and its child, one block is added. Note that diagram $D_{4,3}$ is made by adding a block to $D_{3,1}$, but it could also be made by adding a block to $D_{3,2}$. The reason that $D_{3,1}$ is its real parent is that $m(D_{4,3})$ consists of the upper right and lower right blocks (which are equivalent) of $D_{4,3}$. (This is a fact of our implementation which cannot be seen by looking at the figure.) When $D_{4,3}$ is made from $D_{3,1}$, the new edge is seen to be in $m(D_{4,3})$ and so the diagram is accepted. When it is made from $D_{3,2}$, the new edge is found to be not in $m(D_{4,3})$ and so the diagram is rejected. The idea is that each (isomorphism type of) diagram is accepted exactly once, no matter how many times it is made. This is proved in the following theorem.

Theorem 2.5. *Suppose we call scan*(D, β) *for one* D *from each isomorphism class of irreducible diagram in* $\mathcal{C}$ *that has at most* β *blocks. Then the output will consist of one diagram from each isomorphism class in* $\mathcal{C}$ *with exactly* β *blocks.*

Proof. The theorem is a special case of one in [18], and the reader is referred to that paper for a strictly formal proof. Here we will give a slightly less formal sketch.

Let us say that a diagram D is *accepted* by the algorithm if a call *scan*(D, β) occurs. We will first prove that *at least* one member of each isomorphism class of diagram in $\mathcal{C}$ with at most β blocks is accepted. Then we will prove that *at most* one member of each isomorphism class is accepted. These two facts together will obviously imply the truth of the theorem.

Suppose that the first assertion is false: there is an isomorphism class in $\mathcal{C}$, with at most β blocks, that is never accepted. Let D be a member of such a missing isomorphism class which has the least number of blocks. D cannot be irreducible, since all irreducible diagrams are accepted explicitly. Thus, we can choose $e \in m(D)$ and consider $D - e$. Since $D - e \in \mathcal{C}$ and $D - e$ has fewer blocks than D, at least one isomorph D' of $D - e$ is accepted.

The isomorphism from $D - e$ to D' maps $V(D) \cap e$ onto some subset of $V(D')$. Let e' be a set of atoms consisting of that subset plus enough new atoms to make e' the same size as e. The **for** loop considers some extension $D' + e''$ equivalent to $D' + e'$, since it considers all equivalence classes of extensions. Moreover, since $e \in m(D)$ we can infer that $e' \in m(D' + e')$ and consequently that $e'' \in m(D' + e'')$. This means that the algorithm will perform the call

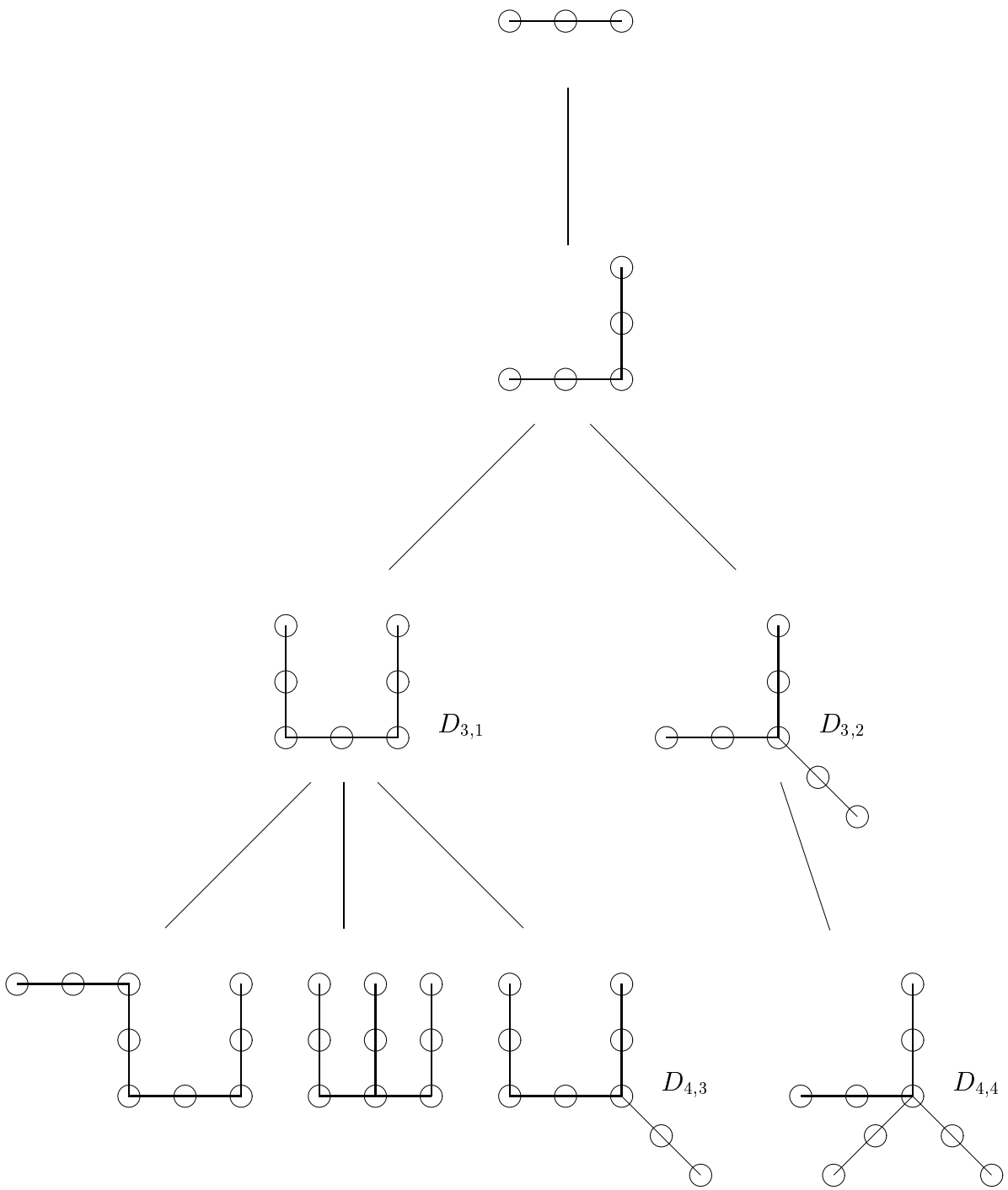

Figure 1: Generation tree for connected Greechie-3-L diagrams

$scan(D'+e'', \beta)$, which is a contradiction as $D'+e''$ is isomorphic to D and the isomorphism class of D was supposed to be not accepted at all. This proves that all isomorphism classes are accepted at least once.

Next suppose that some isomorphism type is accepted twice. Namely, there two isomorphic but distinct diagrams D and D' in $\mathcal{C}$, with at most M blocks, such that both D and D' are accepted. Choose such a pair D, D' with the least number of blocks.

As before, D and D' cannot be irreducible, so they must be accepted by some calls $scan((D-e)+e, \beta)$ and $scan((D'-e')+e', \beta)$ which arise from the calls $scan(D-e, \beta)$ and $scan(D'-e', \beta)$, respectively, where $e \in m(D)$ and $e' \in m(D')$. The properties of $m(\,)$ ensure that $D-e$ and $D'-e'$ are isomorphic, so they must in fact be the same diagram D'' (since isomorphism classes with fewer blocks than D are accepted at most once by assumption). However, $D''+e$ and $D''+e'$ are equivalent but distinct extensions of D'', which violates the **for** loop specification. This contradiction completes the proof. □

The success of the algorithm requires us to be able to find the irreducible diagrams in $\mathcal{C}$ by some other method, but in many important cases this is easy. We give the most important example.

Theorem 2.6. *Suppose $\mathcal{C}$ is a class of Greechie diagrams defined by some fixed set of permissible block sizes, some fixed set of permissible loop lengths, and an optional restriction to connected diagrams. Then the only irreducible diagrams in $\mathcal{C}$ are those with one block.*

Proof. Consider a diagram $D \in \mathcal{C}$ with more than one block.

If $\mathcal{C}$ is not restricted to connected diagrams, $D-e \in \mathcal{C}$ for any $e \in E(D)$, so D is reducible.

Suppose instead that $\mathcal{C}$ contains only connected diagrams. Choose a longest possible sequence S of distinct blocks $e_1, e_2, \ldots, e_k$, where $e_i \cap e_{i+1} \neq \emptyset$ for $1 \leq i \leq k-1$. Let ν_1 and ν_2 be two atoms of $D-e_k$. Since D is connected, there is a chain of blocks from ν_1 to ν_2. This same chain is in $D-e_m$ unless it contains e_k. However, all the blocks D intersecting e_k are in S (or else S can be made longer), so e_k can be replaced in S by some portion of S. Hence $D-e_k$ is connected, so D is reducible. □

The correctness of the algorithm does not depend on the definition of $m(\,)$ provided it has the properties we required of it. The actual definition of $m(\,)$ used in our program is carefully tuned for optimal observed performance, and is too complicated to describe here in detail, but we will outline a simpler definition that is the same in essence.

The key to our implementation of $m(\,)$ is the first author's graph isomorphism program `nauty` [22] can be used. `nauty` takes a simple graph G, perhaps with colored atoms, and produces two outputs. One is the automorphism group $\mathrm{Aut}(G)$, in the form of a set of generators. The other is a canonical labelling of G, which is a graph $c(G)$ isomorphic to G. The function c is "canonical" in the sense that $c(G) = c(G')$ for every graph G' isomorphic to G. To apply `nauty` to a diagram (V, E), we can use the incidence graph $G = (V \cup E, \{(v, e) | v \in e\})$.

The generators for $\mathrm{Aut}(G)$ can be easily converted into generators for $\mathrm{Aut}(D)$ and then used to determine the equivalence classes of extensions. This enables us to implement the requirement of avoiding equivalent extensions.

The canonical labelling $c(\,)$ produced by `nauty` enables us to define $m(\,)$. Take the block e such that $D - e \in \mathcal{C}$ and e is given the least new label by $c(\,)$. If we define $m(D)$ to be the orbit of blocks that contains e, we find that the three requirements we imposed on $m(\,)$ are satisfied.

As we have said, our real program uses a more complex definition of $m(\,)$. We do not use the incidence graph G, but instead use a prototype variant of `nauty` that operates on diagrams directly. Since our program makes connected diagrams, we took $m(D)$ to be an orbit of feet if there were any, where a foot is a block with only one atom that also lies in other blocks. This avoids many connectivity tests, since removal of a foot necessarily preserves connectivity. It also avoids many futile extensions: adding a non-foot e must be done in such a way that any existing feet become non-feet, as otherwise $e \notin m(D + e)$.

In order to generate only the connected Greechie-3-L diagrams having M blocks but no feet, a reasonable approach is to generate all the diagrams, having feet or not, with $M - 1$ blocks first. Then the M-th block can be added in such a way that uses at least 2 of the existing atoms and also turns any feet into non-feet. It is also possible to make a generator that makes foot-free diagrams while staying entirely within that class, but it does not appear likely to be much different in efficiency.

Program `greechie`. Our implementation of the algorithm is a self-contained program called `greechie`[4] that takes as parameters the number of blocks, an optional upper bound on the number of atoms, and whether or not feet are permitted. It then produces one representative of each isomorphism class of connected Greechie-3-L diagram with those properties. The diagrams can then be processed as they are generated, with no need to store them. There is also an option for dividing the set of diagrams into disjoint subsets, and efficiently producing only one of the subsets. This allows long computations to be broken into manageable pieces that can be run independently, even on different computers, without much change to the total running time.

In Table 1 we list the numbers of Greechie-3-L diagrams for small values of α and β. In each cell of the tables, the upper value is the total number of connected Greechie-3-L diagrams, and the lower value is the number of those which have no feet. Both counts are 0 if the table cell is empty. The table includes all possible values of β for $\alpha \leq 29$ and all possible values of α for $\beta \leq 17$.

For reasons explained later, we have particular interest in those diagrams containing close to the maximum number of blocks for a given number of atoms. This prompted us to compute additional near-maximal diagrams past the size where finding all the diagrams is practical.

To keep the discussion simple, we restrict ourselves to Greechie-3-L diagrams, not necessarily connected. By the *type* of a diagram we mean the pair (α, β), where α is the number

[4] `ftp://m3k.grad.hr/pavicic/greechie`, `http://cs.anu.edu.au/~bdm/nauty/greechie.html`. Many of the diagrams computed with the program are also available at those places.

$\alpha \setminus \beta$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	*total*
3	1 1														1 1
5		0 1													0 1
7			0 2												0 2
9				0 4											0 4
10					1 1										1 1
11					0 8										0 8
12						1 3									1 3
13						0 19	2 2								2 21
14							1 14	1 1							2 15
15							0 48	6 16	1 1	1 1					8 66
16								1 62	12 15	1 1					14 78
17								0 126	11 119	21 24	2 3				34 272
18									1 281	71 209	27 31	4 4			103 525
19									0 355	19 819	261 490	67 84	6 6		353 1754
20										1 1239	251 2347	834 1217	147 166	1 1	1234 4970
total	1 1	0 1	0 2	0 4	1 9	1 22	3 64	8 205	25 771	114 3330	571 16571	3675 95327	27687 628555	239844 4713887	

$\alpha \setminus \beta$	10	11	12	13	14	15	16	17	*total*
21	0 1037	29 5199	2052 8273	2884 3872	405 437	4 4			5374 18822
22		1 5377	675 22219	12849 31805	10885 13440	905 908	4 4		25319 73753
23		0 3124	42 30923	10235 109467	74698 141454	45905 53075	2837 2861	17 17	133734 340921
24			1 22931	1508 182581	113376 588877	435636 695281	207767 225862	10723 10756	769050 1726327
25			0 9676	57 173841	37406 1194496	1061800 3508197	2670655 3787093	1030296 1091395	4849700 9814445
26				1 96213	2997 1346088	664386 8429567	9287661 22797227	17387077 22548822	33292775 61419172
27				0 30604	75 932262	110376 11227170	9494774 64431928	80621116 161294913	248439348 423756705
28					1 398359	5463 9117775	2905751 99744819	120414885 530135685	2009490610 3203436511
29					0 98473	95 4807157	280150 94035080	60035427 945730356	17618049369 26495317590
30						1 1630602	9329 57743948	10280997 1021031346	?
31						0 321572	118 23993678	636294 719130759	?
32							1 6612148	15096 346356783	?
33							0 1063146	143 116542189	?
34								1 26603735	?
35								0 3552563	?
total	114 3330	571 16571	3675 95327	27687 628555	239844 4713887	2324571 39791308	24859047 374437794	290432072 3894029319	

$\alpha \setminus \beta$	18	19	20	21	22	23	24	*total*
24	39 39							769050 1726327
25	49350 49611	134 134	2 2					4849700 9814445
26	5702603 5952608	247469 248066	581 581					33292775 61419172
27	121218048 147629428	35519085 36732179	1471694 1474036	4180 4185				248439348 423756705
28	717349032 1237468066	911521449 1062867580	247041933 253438446	10216096 10229781	35992 35992	8 8		2009490610 3203436511
29	1448834695 4690081789	6661929716 10298834720	7447274324 8422315646	1916771296 1956383755	82563871 82670819	359550 359550	245 245	17618049369 26495317590

Table 1: Counts of connected Greechie-3-L diagrams

of atoms and β is the number of blocks. The *rank* of an atom is the number of blocks which contain it.

The first observation is that a diagram D of type (α, β) has an atom whose rank is at most $\lfloor 3\beta/\alpha \rfloor$. Over most of our computational range, $\alpha < \beta$, so this value is at most 2. If there is an atom of rank 1, we can make D either by adding a foot to a diagram of type $(\alpha-2, \beta-1)$, or by adding one block and one atom to a diagram of type $(\alpha-1, \beta-1)$. On the other hand, if there is an atom of rank 2 but none of rank 1, we can make D by adding one atom and two blocks to a diagram of type $(\alpha-1, \beta-2)$. So, if we have already made the diagrams of types $(\alpha-2, \beta-1)$, $(\alpha-1, \beta-1)$, and $(\alpha-1, \beta-2)$, we can easily extend them to make those of type (α, β), provided $\alpha < \beta$.

Sometimes the class $(\alpha-1, \beta-2)$ may be too onerous to compute. In this case, there are two other approaches we might be able to take to making those diagrams of type (α, β) whose minimum atom rank is 2. Define α_i to be the number of atoms of rank i (and recall that we are assuming $\alpha_1 = 0$). Counting the pairs (block, atom in block) in two ways, we have $2\alpha_2 + 3\alpha_3 + 4(\alpha - \alpha_2 - \alpha_3) \leq 3\beta$. Since also $\alpha_2 + \alpha_3 \leq \alpha$, we have

$$2\alpha_2 + \alpha_3 \geq 4\alpha - 3\beta \text{ and } \alpha_2 \geq 3\alpha - 3\beta. \tag{1}$$

Now consider the case of a Greechie-3-L diagram with $\alpha_1 = 0$ and $6\alpha > 7\beta$. Applying the second part of (1) we find that $2\alpha_2 > \beta$, which implies that some block contains at least two atoms of rank 2. Therefore, we can make the diagram by adding two atoms and three blocks to a diagram of type $(\alpha-2, \beta-3)$.

If $6\alpha \leq 7\beta$ but $\alpha > \beta$, there might be no atoms of rank 1, nor two atoms of rank 2 in the same block. In this case, we know that there are exactly $2\alpha_2$ blocks containing an atom of rank 2. The total rank of the atoms of rank greater than 3 is $3\beta - 2\alpha_2 - 3\alpha_3$, so we have at least $4\alpha_2 - (3\beta - 2\alpha_2 - 3\alpha_3) = 6\alpha_2 + 3\alpha_3 - 3\beta$ pairs (x, y) such that x and y are atoms of rank 2 and 3, respectively, lying in the same block. Now, if we suppose that $14\alpha > 15\beta$, we find that $6\alpha_2 + 3\alpha_3 - 3\beta > \alpha_3$, implying that two of the pairs (x, y) have the same y. That is, there is an atom of rank 3 lying in two blocks which each contain an atom of rank 2. Therefore, we can make this diagram by adding 3 atoms and 5 blocks to a diagram of type $(\alpha - 3, \beta - 5)$.

$\alpha \setminus \beta$	25	26	27	28	29	30	31	32	33	34	35	36
30	3982	4										
31	?	81068	71	1								
32	?	?	$\geq$313813	1643								
33	?	?	?	?	$\geq$51643	66						
34	?	?	?	?	?	$\geq$185733	2113	19				
35	?	?	?	?	?	?	?	$\geq$70035	325	17	5	
36	?	?	?	?	?	?	?	?	?	$\geq$7871	136	1
37	?	?	?	?	?	?	?	?	?	?	?	$\geq$1693

Table 2: Counts of large extremal Greechie-3-L diagrams

Using these ideas, we were able to compute (see Table 2) the Greechie-3-L diagrams with

up to 36 atoms having the maximum and, in some cases, near-maximum possible numbers of blocks.

The maximum number of blocks shown in each row of the table is the maximum possible. All the diagrams counted in the table turned out to be connected, even though our programs did not assume connectivity.

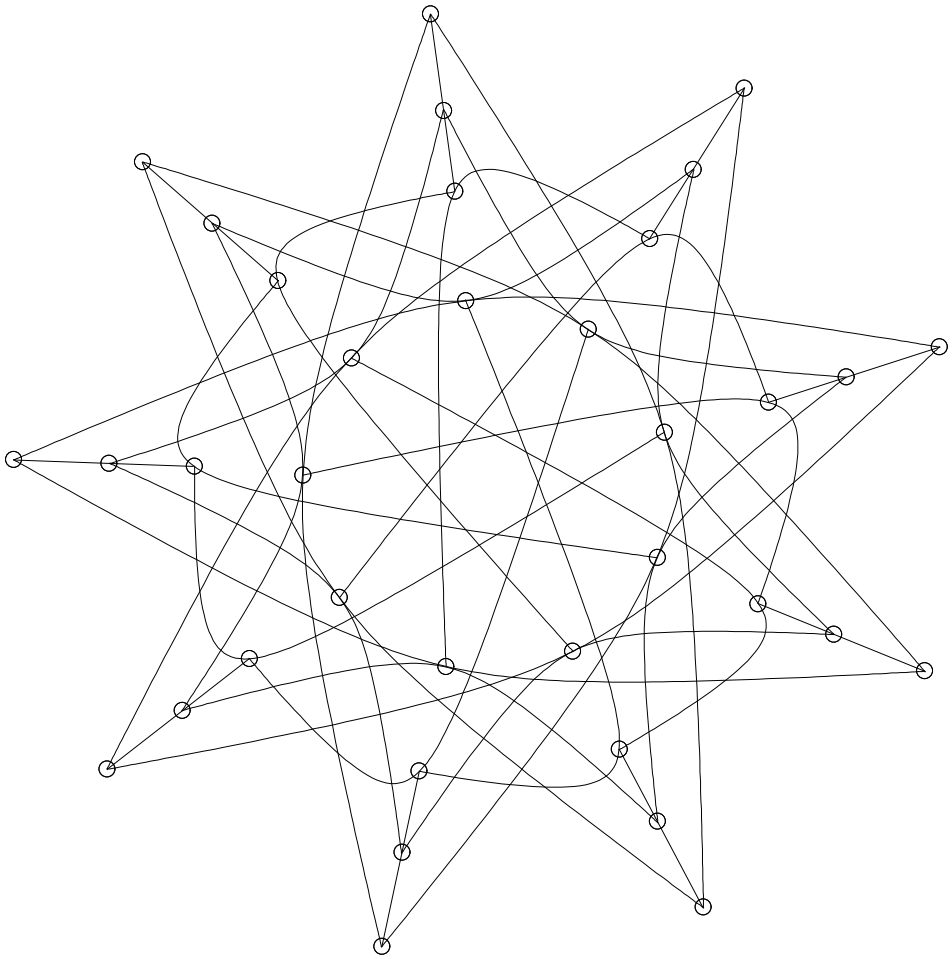

Figure 2: The unique Greechie-3-L diagram of type $(36, 36)$

Of some interest is that the Greechie-3-L diagrams of type $(35, 35)$ and $(36, 36)$ contain only atoms of rank 3. There is a connection here to known results in graph theory, as follows. Let D be a diagram in one of these two classes. Define a graph G whose vertices are the atoms and blocks of D. A vertex which is an atom is adjacent in G to a vertex which is a block provided the atom lies in the block, and there are no other edges. (This is the same incidence graph we defined earlier.) Then each vertex of the graph has valence 3, and moreover there are no cycles of length 9 or less (which is equivalent to the requirement that D has no loops of length 4 or less). In graph theoretic language, G is a bipartite cubic (or trivalent) graph of girth at least 10. The smallest such graphs are three with 70 vertices [23]. In two of the three graphs, the functions of "atom" and "block" can be interchanged to produce different Greechie diagrams, but in the third this interchange gives an isomorphic

diagram. That is how the three graphs correspond to the five diagrams we found.

Figure 2 shows the unique Greechie-3-L diagram of type $(36, 36)$. Using a slight adaptation of the program described in [24], we have found that there are no Greechie-3-L diagrams of type $(37, 37)$ with every atom having rank 3, and exactly eight such diagrams of type $(38, 38)$.

3 Testing conjectures with Greechie diagrams

In order to test equations conjectured to hold in various classes of orthomodular lattices, it is useful to automate their checking against Greechie diagrams. This will let us either falsify the conjecture or give us some confidence that might hold in the class of interest before we attempt to prove it. Also, finding a lattice in which one equation holds but a second one fails gives us a proof that the second is independent.

To this end we use a program called `latticeg`[5] which will check to see if an equation or inference holds in each of the Greechie diagrams in a list provided by the user.

The `latticeg` program internally converts a Greechie diagram to its corresponding Hasse diagram and tests all possible assignments of nodes in the Hasse diagram to the equation. In general Greechie diagrams correspond to Boolean algebras "pasted" together.

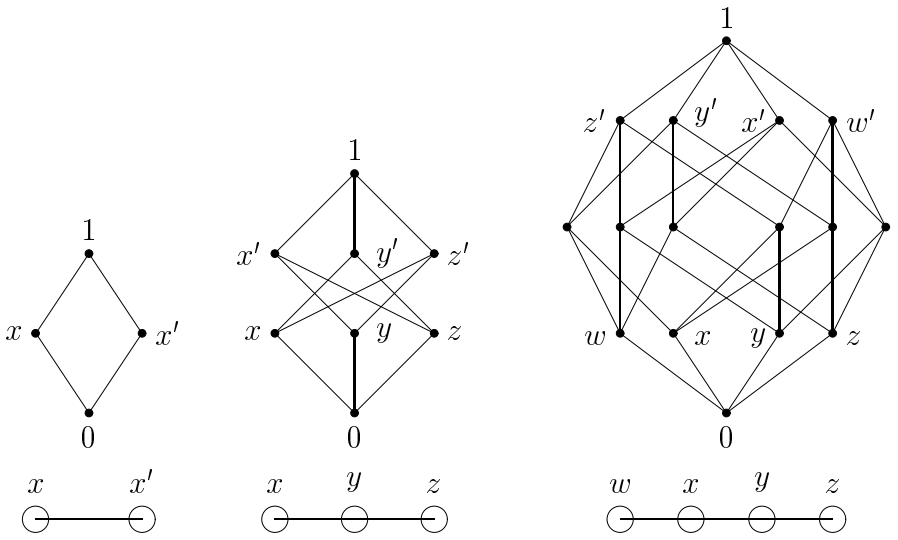

Figure 3: Greechie diagrams for Boolean lattices 2^2, 2^3, and 2^4, labeled with the atoms of their corresponding Hasse diagrams shown above them. (2^4 was adapted from [25, Fig. 18, p. 84].)

The Hasse diagrams for the Boolean algebras corresponding to 2-, 3-, and 4-atom blocks are shown in Fig. 3. The Greechie diagram for a given lattice may be drawn in several

[5]Available at `ftp://ftp.shore.net/members/ndm/quantum-logic` as the ANSI C program `latticeg.c`. The program is simple to use and self-explanatory with the `--help` option. A related program `lattice.c` handles general Hasse diagrams and has built-in those lattices we have found most useful for preliminary testing of conjectures. These two programs, along with `beran.c` for computing the canonical orthomodular form of any two-variable expression, are the primary computational tools we have used for studying orthomodular and Hilbert lattice equations.

equivalent ways: Fig. 4 shows the same Greechie diagram drawn in two different ways, along with the corresponding Hasse diagram. From the definitions we see that the ordering of the atoms on a block does not matter, and we may also draw blocks using arcs as well as straight lines as long as the blocks remain clearly distinguishable.

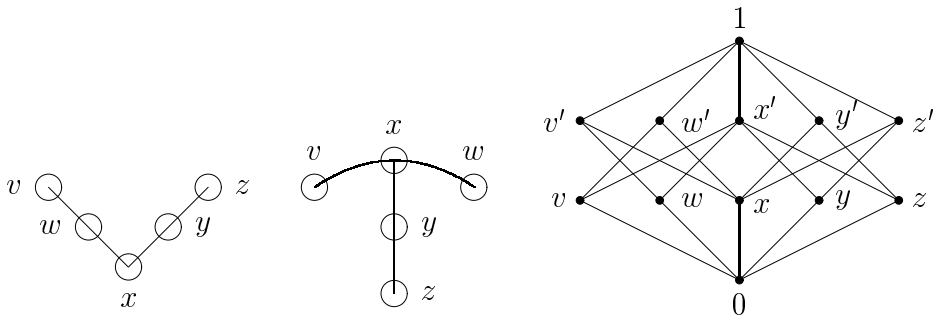

Figure 4: Two different ways of drawing the same Greechie diagram, and its corresponding Hasse diagram.

Recall that a *poset* (partially ordered set) is a set with an associated ordering relation that is reflexive ($a \leq a$), antisymmetric ($a \leq b, b \leq a$ imply $a = b$), and transitive ($a \leq b, b \leq c$ imply $a \leq c$). An *orthoposet* is a poset with lower and upper bounds 0 and 1 and an operation $'$ satisfying (i) if $a \leq b$ then $b' \leq a'$; (ii) $a'' = a$; and (iii) the infimum $a \cap a'$ and the supremum $a \cup a'$ exist and are 0 and 1 respectively. A *lattice* is a poset in which any two elements have an infimum and a supremum. An orthoposet is *orthomodular* if $a \leq b$ implies (i) the supremum $a \cup b'$ exists and (ii) $a \cup (a' \cap b) = b$. A lattice is *orthomodular* if it is also an orthomodular poset. For example, Boolean algebras such as those of Fig. 3 are orthomodular lattices. A *σ-orthomodular poset* is an orthomodular poset in which every countable subset of elements has a supremum. An *atom* of an orthoposet is an element $a \neq 0$ such that $b < a$ implies $b = 0$.

In the literature, there are several different definitions of a Greechie diagram. For example, Beran ([25, p. 144]) forbids 2-atom blocks. Kalmbach ([17, p. 42]) as well as Pták and Pulmannová [21, p. 32] include all diagrams with 2-atom blocks connected to other blocks as long as the resulting pasting corresponds to an orthoposet. However, the case of 2-atom blocks connected to other blocks is somewhat complicated; for example, the definition of a loop in Def. 2.1 must be modified (e.g. [17, p. 42]) and no longer corresponds to the simple geometry of a drawing of the diagram. The definition of a Greechie diagram also becomes more complicated; for example a pentagon (or any n-gon with an odd number of sides) made out of 2-atom blocks is not a Greechie diagram (i.e. does not correspond to any orthoposet).

The definition of Svozil and Tkadlec [20] that we adopt, Def. 2.2, excludes 2-atom blocks connected to other blocks. It turns out that all orthomodular posets representable by Kalmbach's definition can be represented with the diagrams allowed by Svozil and Tkadlec's definition. But the latter definition eliminates the special treatment of 2-atom blocks connected to other blocks and in particular simplifies any computer program designed to process Greechie diagrams.

Svozil and Tkadlec's definition further restricts Greechie diagrams to those diagrams representing orthoposets that are orthomodular by forbidding loops of order less than 4, unlike

the definitions of Beran and Kalmbach. The advantage appears to be mainly for convenience, as we obtain only those Greechie diagrams that correspond to what are sometimes called "quantum logics" (σ-orthomodular posets). (We note that the term "quantum logic" is also used to denote a propositional calculus based on orthomodular or weakly orthomodular lattices. [26])

The definition allows for Greechie diagrams whose blocks are not connected. In Fig. 5 we show the Greechie diagram for the *Chinese lantern* MO2 using unconnected 2-atom blocks. This example also illustrates that even when the blocks are unconnected the properties of the resulting orthoposet are not just a simple combination of the properties of their components (as one might naïvely suppose), because we are adding disjoint sets of incomparable nodes to the orthoposet. As is well-known ([17, p. 16]), MO2 is not distributive, unlike the Boolean blocks it is built from.

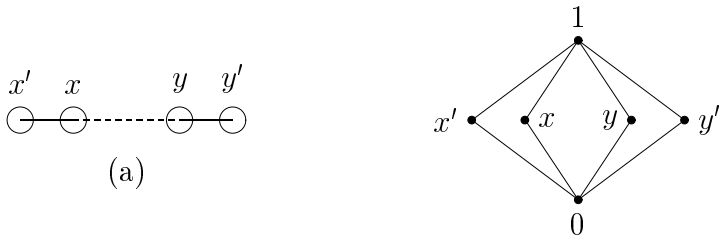

Figure 5: Greechie diagram for the lattice MO2 and its Hasse diagram. The dashed line indicates that the unconnected blocks belong to the same Greechie diagram.

The `latticeg` program takes, as its inputs, a Greechie diagram ASCII representation (or more precisely a collection of them) and an equation (or inference) to be tested. This ASCII representation is compatible with the output of the programs described in Section 2. Currently `latticeg` is designed to work only with Greechie diagrams corresponding to lattices, i.e. that have no loops of order 4 or less, as these are the most interesting for studying equations valid in all Hilbert lattices. It converts the Greechie diagram to its corresponding Hasse diagram (internally stored as truth tables). Finally, the program tests all possible assignments of nodes in the Hasse diagram to the input equation under test.

The `latticeg` program also incorporates classical propositional metalogic and predicate calculus to allow the study of such characteristics as atomicity and superposition.

As a simple example of the operation of the `latticeg` program, we show how it verifies the passage and failure of the modular law [Equation (11) below] on the lattices of Figs. 7b and 7c. We create a file with a name such as `m.gre` to represent the lattices, containing the lines

```
123,345.
123,345,567.
```

and run the program by typing

```
latticeg -i m.gre "(av(b^(avc)))=((avb)^(avc))"
```

The program responds with

```
The input file has 2 lattices.
Passed #1 (5/2/12)
FAILED #2 (7/3/16) at (av(f^(avb)))=((avf)^(avb))
```

The notation should be more or less apparent but is described in detail by the program's help. The numbers "5/2/12" show the atom/block/node count, and the failure shows the internal Hasse diagram's nodal assignment to the equation's variables.

Let us consider an application of the program. Closed subspaces $\mathcal{H}_a, \mathcal{H}_b$ of any infinite dimensional Hilbert space $\mathcal{H}$ form a lattice in which the operations are defined in the following way: $a' = \mathcal{H}_a^{\perp}$, $a \cap b = \mathcal{H}_a \bigcap \mathcal{H}_b$, and $a \cup b = (\mathcal{H}_a + \mathcal{H}_b)^{\perp\perp}$. In such a lattice its elements satisfy the following condition, i.e., in any infinite dimensional Hilbert space its closed subspaces satisfy the following equation which is called the *orthoarguesian equation*:

$$\begin{aligned} & a \perp b \quad \& \quad c \perp d \quad \& \quad e \perp f \quad \Rightarrow \\ & (a \cup b) \cap (c \cup d) \cap (e \cup f) \leq \\ & b \cup (a \cap (c \cup (((a \cup c) \cap (b \cup d)) \cap (((a \cup e) \cap (b \cup f)) \cup ((c \cup e) \cap (d \cup f))))))) \end{aligned} \tag{2}$$

where $a \perp b$ means $a \leq b'$.

We wanted first, to reduce the number of variables in this equation and second, to generalize the equation to n variables. In attacking the first problem the program helped us to quickly eliminate dead ends: a failure of a conjectured equation in a lattice in which Equation 2 held meant that the latter equation was weaker and vice-versa. Thus we arrived at the following 4-variable equation—which we call the 4OA law:

$$(a_1 \rightarrow_1 a_3) \cap (a_1 \overset{(4)}{\equiv} a_2) \leq a_2 \rightarrow_1 a_3 \,. \tag{3}$$

where the operation $\overset{(4)}{\equiv}$ is defined as follows:

$$a_1 \overset{(4)}{\equiv} a_2 \quad \overset{\text{def}}{=} \quad (a_1 \overset{(3)}{\equiv} a_2) \cup ((a_1 \overset{(3)}{\equiv} a_4) \cap (a_2 \overset{(3)}{\equiv} a_4)) \,, \tag{4}$$

where

$$a_1 \overset{(3)}{\equiv} a_2 \quad \overset{\text{def}}{=} \quad ((a_1 \rightarrow_1 a_3) \cap (a_2 \rightarrow_1 a_3)) \cup ((a_1' \rightarrow_1 a_3) \cap (a_2' \rightarrow_1 a_3)), \tag{5}$$

where $a \rightarrow_1 b \overset{\text{def}}{=} a' \cup (a \cap b)$. We then proved "by hand" that Equations (2) and (3) are equivalent. [19] We also proved that the following generalization (which we call the nOA law) of Equation (3)

$$(a_1 \rightarrow_1 a_3) \cap (a_1 \overset{(n)}{\equiv} a_2) \leq a_2 \rightarrow_1 a_3 \tag{6}$$

where

$$a_1 \overset{(n)}{\equiv} a_2 \quad \overset{\text{def}}{=} \quad (a_1 \overset{(n-1)}{\equiv} a_2) \cup ((a_1 \overset{(n-1)}{\equiv} a_n) \cap (a_2 \overset{(n-1)}{\equiv} a_n)) \,, \quad n \geq 4 \tag{7}$$

holds in any Hilbert lattice. To show that this generalization is a nontrivial one, the program is all we need—we need not prove anything "by hand." It suffices to find a Greechie diagram

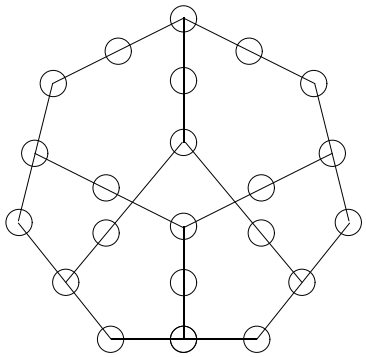

Figure 6: Greechie diagram for OML L46.

in which the 4OA law holds and 5OA law fails. An 800 MHz PC took a few days to find such a lattice (shown in Fig. 6), [19] while to find it "by hand" is, due to the number of variables in the equation and nodes in the corresponding Hasse diagram, apparently humanly impossible.

Considerable effort was put into making the program run fast, with methods such as exiting an evaluation early when a hypothesis of an inference fails. Truth tables for all built-in compound operations (such as the various quantum implications and the quantum biconditional) are precomputed. The innermost loop (which evaluates an assignment) was optimized for the fastest runtime we could achieve. The number of assignments of lattice nodes to equation variables that must be tested is n^v where n is the number of nodes in the Hasse diagram and v is the number of variables in the equation. The algorithm requires a time approximately proportional to kn^v where k is the length of the equation expressed in Polish notation. For a typical equation ($k = 20$) with no hypotheses, the algorithm currently evaluates around a million assignments per second on an 800-MHz PC. The speed is typically faster when hypotheses are present, particularly if they have fewer variables than the conclusion. If a lattice violates an equation, it often happens (with luck) that the first failure will be found quickly, in which case further evaluations do not have to be done.

The propositional metalogic feature of `latticeg`, when carefully used in a series of hypotheses with a successively increasing number of variables, can sometimes be exploited to achieve orders of magnitude speed-up with certain equations containing many variables. For example, we tested the 8-variable Godowski equation against a 42-node lattice in 16 hours, whereas without the speed-up it would have required around 10^5 hours. To illustrate how this speed-up works, we can add to the 4-variable Godowski equation [Equation 9 below] hypotheses as follows:

$$\sim (d \to_1 a \leq a \to_1 d) \quad \& \quad \sim ((c \to_1 d) \cap (d \to_1 a) \leq a \to_1 d) \quad \Rightarrow \\ (a \to_1 b) \cap (b \to_1 c) \cap (c \to_1 d) \cap (d \to_1 a) \leq a \to_1 d\,. \qquad (8)$$

Here $\sim$ means metalogical NOT. The hypotheses are redundant in any ortholattice, which is the case for the Greechie diagrams that are of interest to us. We take advantage of the `latticeg` feature that exits the evaluation of a lattice nodal assignment if a hypothesis fails.

If an assignment to the first hypothesis, with only 2 variables, fails (as it typically does for some assignments) it means we don't have to scan the remaining variables. Similarly, if the first hypothesis passes but the second (with 3 variables) fails, we can skip the evaluation of the 4-variable conclusion.

We are currently exploring the exploitation of possible symmetries inherent in the Greechie diagram to speed up the program further, but we have not yet achieved any results in this direction. For certain special cases such as the Godowski equations we are also exploring the use of a "dynamic programming" technique that may provide a run time proportional to kn^4 instead of kn^v, regardless of the number of variables.

For fastest run time, it is desirable to screen equations with the smallest Greechie diagrams first. For this purpose what matters is the size of the Hasse diagram and not the Greechie diagram. In a chain of blocks each having two atoms connected, a 3-atom block adds 4 nodes to the Hasse diagram whereas a 4-atom block adds 12 nodes. For example, the decagon (10 blocks) has 42 nodes with 3-atom blocks and 122 nodes with 4-atom blocks. With 3-atom blocks, a 6-variable equation—our practical upper limit when there are no strong hypotheses—must be evaluated $42^6 = 5.5$ billion times (a few hours of CPU time on an 800-MHz PC), but with 4-atom blocks it would take $122^6 = 3.3$ trillion evaluations, which is currently impractical. So far most of our work has been done using diagrams with every block having size 3.

Two other heuristics have helped us to falsify conjectures more quickly. The first is to first scan Greechie diagrams with the highest block-to-atom ratio (Table 1). Such diagrams seem to have the most complex "structure" with the most likelihood of violating a non-orthomodular equation ("non-orthomodular equation" here means an equation which turns the orthomodular lattice variety into a smaller one when added to it). A drawback is that at higher atom counts, virtually every such diagram violates almost any non-orthomodular equation, making it very useful for identifying non-orthomodular properties but less useful for proving independence results. For example, for 35 and 36 atoms (the case we elaborated in Section 2) the highest ratio is 1 (Table 2) and in those diagrams all equations that we know to be non-orthomodular fail. Hence, for example 36×36 (36 atoms, 36 blocks) is a very useful tool for an initial scanning of equations we want to check for "non-orthomodularity."

The second heuristic is our empirical observation that Greechie diagrams without feet often behave the same as the same diagram with feet added. For example, the Peterson OML (Fig. 7a), with 32 nodes, is the smallest lattice that violates Godowski's 4-variable strong state law [19] which holds in any infinite dimensional Hilbert space:

$$(a \to_1 b) \cap (b \to_1 c) \cap (c \to_1 d) \cap (d \to_1 a) \quad \leq \quad a \to_1 d \tag{9}$$

but not Godowski's 3-variable law (which also holds in infinite dimensional Hilbert spaces)

$$(a \to_1 b) \cap (b \to_1 c) \cap (c \to_1 a) \quad \leq \quad a \to_1 c\,. \tag{10}$$

The Peterson OML is useful as a test for an equation derived from the 4-variable law and conjectured to be equivalent to it: if it does not violate the conjectured equation we know the equation is weaker the 4-variable law. Now, we observe empirically that we may add a foot (a 3-atom block connected at only one point) to any of its 15 atoms without changing

this behavior. We have also not seen a chain of feet or combination of feet that changes this behavior when added to the diagram.

So, by scanning only lattices without feet, we can obtain a speedup of 20 times for 14-block lattices (Table 1). Supporting this heuristic is the fact that complex Greechie diagrams with feet are rarely found in the literature. We obtained an additional support by scanning the 4OA law given by Equation (3) and Godowski's 3 variable equation (10) through several million lattices with free feet, vs. those with feet stripped: we did not find a single difference. To our knowledge there is only one special case for which feet do make a difference. Fig. 7b shows a lattice that obeys the modular law

$$a \cup (b \cap (a \cup c)) \quad = \quad (a \cup b) \cap (a \cup c) \tag{11}$$

but violates it when a foot is added (Fig. 7c). This special case might well be insignificant because of all Greechie diagrams only star-like ones (Fig. 7b, $D_{3,2}$ and $D_{4,4}$ from Fig. 1, etc.) are modular. As soon as we add any block to any other atom apart from the central one in such a lattice we make it non-modular.

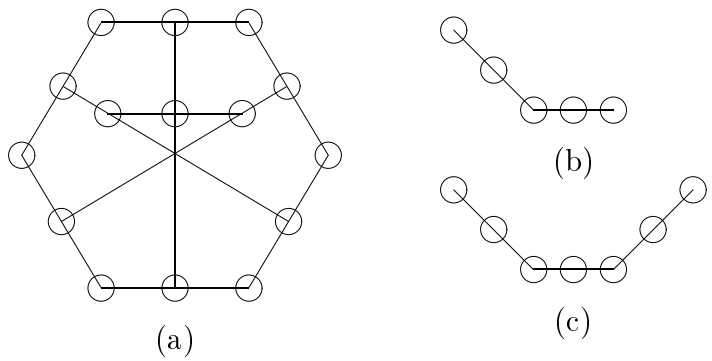

Figure 7: (a) Peterson OML. (b) Greechie diagram obeying modular law. (c) Greechie diagram violating modular law.

Another heuristic for reducing the number of diagrams to be scanned is suggested by the following observation, although we have not implemented it. We can list the diagrams in such a way that (except for the first diagram) each is formed by adding one block to a diagram earlier in the list. Whenever the earlier diagram violates an equation, we have observed that it is very likely that the new diagram will also violate the equation. By skipping the new diagram in this case (presuming the probable violation), a speedup can be obtained.

However the above observation does not universally hold, i.e. sometimes the new diagram will pass an equation violated by the earlier one. An example is shown in Fig. 8. The diagram L38 violates the orthoarguesian law (Equation 2). But if we extend L38 by adding two blocks as shown in Fig. 8b, the resulting diagram will pass not only this law [equivalent to 4OA given by Eq. (3)] but also 5OA and 6OA [given by Eq. (6) for $n = 5$ and $n = 6$, respectively].

If our various speedup heuristics are used for practical reasons, the user must be aware that a diagram scan may be incomplete. So if, in our example, the extended L38 (by passing) could serve to prove a certain independence result, it would be missed by the scan. Currently we have no data to indicate how often such cases would be missed.

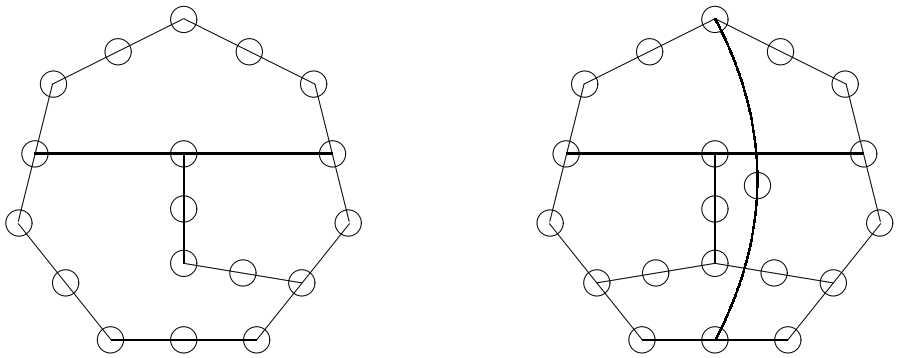

Figure 8: (a) Greechie diagram for L38; (b) L38 with two blocks added.

Any scan of diagrams to test an equation is, of course, *a priori* incomplete since the number of diagrams is infinite. The various heuristics we have described may cause some lattices in any finite list to be skipped. But if a lattice with the desired properties is found more quickly our goal is achieved. Of course a scan can be continued for any diagrams omitted by the heuristics if more completeness is desired.

4 Conclusions

Greechie diagrams (we used the version given by Definition 2.1 and discussed the others in Section 3) are generators of examples and counterexamples of orthomodular non-modular lattices. They are special cases of lattices one obtains by using the recent generalization by Navara and Rogalewicz [27, 28] of Dichtl's pasting construction for orthomodular posets and lattices [29]. Navara and Rogalewicz's method exhaustively generates all finite orthomodular non-modular lattices but Greechie diagrams are apparently easier to generate and certainly much easier to test lattice equations. For these reasons, Greechie diagrams have been used almost exclusively so far.

Since any infinite dimensional complex Hilbert space is orthoisomorphic to a Hilbert lattice which is orthomodular and non-modular, Greechie diagrams represent an indispensable tool for Hilbert space investigation. This has also been prompted by recent developments in the field of quantum computing. However, as we stressed in the Introduction, the existing (both manual and automated) constructions of Greechie diagrams and their application to Hilbert space properties (which resulted in many important results at the time) recently reached the frontiers of human manageability. Therefore, in Section 2 we gave an algorithm and a program for generating Greechie lattices with theoretically unlimited numbers of atoms and blocks. The algorithm is of a completely different kind from the only earlier algorithm and the program is at least 10^5 times faster. In Section 3 we then gave an algorithm and programs for automated checking of lattice properties on Greechie diagrams.

Our algorithm for generating Greechie diagrams—given by Definition 2.4—works by defining a unique construction path for each isomorphism class of diagrams. It enabled us to produce a self-contained program `greechie` for automated generation of diagrams with

a specified number or range of atoms and/or blocks. Several properties and several special types of Greechie diagram construction are discussed in the Section, as are connections to some equivalent results in graph theory.

The algorithm for automated checking of lattice properties described in Section 3 works by converting a Greechie diagram to its corresponding Hasse diagram and then converting the Hasse diagram to a truth table for supremum and orthocomplementation. It enabled us to construct a self-contained program `latticeg` which takes, as its inputs, Greechie diagrams in ASCII representation and an equation or inference or quantified expression to be tested. Many programming speed-ups have been used to make the program run as fast as possible. For example, all built-in compound operations are precomputed, many C code tricks are used, etc. Also, to additionally speed up scanning, we use many Greechie diagram heuristics that we found recently: a lattice equation is most likely to fail in a lattice with the highest block-to-atom ratio, scanning of an equation on a diagram with no feet and on the same diagram with feet added makes no difference for most diagrams, etc. Several properties and several special cases of Greechie lattices are given and discussed in the Section. In particular, it is explained how we proved that our recent n-variable generalization of the orthoarguesian equation is a non-trivial one by using nothing but `latticeg` applied to an output of `greechie`.

References

[1] G. Birkhoff and J. von Neumann, The Logic of Quantum Mechanics, Ann. Math. **37**, 823–843 (1936).

[2] G. W. Mackey, *The Mathematical Foundations of Quantum Mechanics*, W. A. Benjamin, New York, 1963.

[3] N. Zierler, Axioms for Non-Relativistic Quantum Mechanics, Pacif. J. Math. **11**, 1151–1169 (1961).

[4] C. Piron, Axiomatique quantique, Helv. Phys. Acta **37**, 439–468 (1994).

[5] M. D. MacLaren, Atomic Orthocomplemented Lattices, Pacif. J. Math. **14**, 597–612 (1964).

[6] I. Amemiya and H. Araki, A Remark on Piron's Paper, Publ. Research Inst. Math. Sci., Kyoto Univ. Series **A 2**(3), 423–427 (1966/67).

[7] M. J. Mączyński, Hilbert Space Formalism of Quantum Mechanics without the Hilbert Space Axiom, Rep. Math. Phys. **3**, 209–219 (1972).

[8] H. A. Keller, Ein nicht-klassischer Hilbertscher Raum, Math. Z. **172**, 41–49 (1980).

[9] M. P. Solèr, Characterization of Hilbert Spaces by Orthomodular Spaces, Comm. Alg. **23**, 219–243 (1995).

[10] S. S. Holland, JR., Orthomodularity in Infinite Dimensions; a Theorem of M. Solèr, Bull. Am. Math. Soc. **32**, 205–234 (1995).

[11] A. Prestel, On Solèr's Characterization of Hilbert Spaces, Manus. Math. **86**, 225–238 (1995).

[12] R. Mayet, Some Characterizations of Underlying Division Ring of a Hilbert Lattice of Authomorphisms, Int. J. Theor. Phys. **37**, 109–114 (1998).

[13] A. Dvurečenskij, Test Spaces and Characterizations of Quadratic Spaces, Int. J. Theor. Phys. **35**, 2093–2106 (1996).

[14] A. Dvurečenskij, Solèr Theorem and Characterization of Inner Product Spaces, Int. J. Theor. Phys. **37**, 23–29 (1998).

[15] M. Pavičić and N. D. Megill, Binary Orthologic with Modus Ponens Is either Orthomodular or Distributive, Helv. Phys. Acta **71**, 610–628 (1998).

[16] R. J. Greechie, Another Nonstandard Quantum Logic (and how I Found It), in *Mathematical Foundations of Quantum Theory (Papers from a conference held at Loyola University, New Orleans, June 2–4, 1977)*, edited by A. R. Marlow, pages 71–85, Academic Press, New York, 1978.

[17] G. Kalmbach, *Orthomodular Lattices*, Academic Press, London, 1983.

[18] B. D. McKay, Isomorph-Free Exhaustive Generation, J. Algorithms **26**, 306–324 (1998).

[19] N. D. Megill and M. Pavičić, Equations and State Properties That Hold in All Closed Subspaces of an Infinite Dimensional Hilbert Space, Int. J. Theor. Phys., **39**, No. 10 (2000).

[20] K. Svozil and J. Tkadlec, Greechie Diagrams, Nonexistence of Measures and Kochen-Specker-Type Constructions, J. Math. Phys. **37**, 5380–5401 (1996).

[21] P. Pták and S. Pulmannová, *Orthomodular Structures as Quantum Logics*, Kluwer, Dordrecht, 1991.

[22] B. D. McKay, `nauty` User's Guide (version 1.5), Dept. Computer Science, Australian Nat. Univ. **Tech. Rpt. TR-CS-90-02** (1990).

[23] M. O'Keefe and P. K. Wong, A Smallest Graph with Girth 10 and Valency 3, J. Combinatorial Theory, Series B **29**, 91–105 (1980).

[24] B. D. McKay, W. Myrvold, and J. Nadon, Fast backtracking principles applied to find new cages, in *Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 188–191, SIAM, New York, 1998.

[25] L. Beran, *Orthomodular Lattices; Algebraic Approach*, D. Reidel, Dordrecht, 1985.

[26] M. Pavičić and N. D. Megill, Non-Orthomodular Models for Both Standard Quantum Logic and Standard Classical Logic: Repercussions for Quantum Computers, Helv. Phys. Acta **72**, 189–210 (1999), http://xxx.lanl.gov/abs/quant-ph/9906101.

[27] V. Rogalewicz, Any orthomodular poset is a pasting of Boolean algebras, Comment. Math. Univ. Carolin. **29**, 557–558 (1988).

[28] M. Navara and V. Rogalewicz, The Pasting Constructions for Orthomodular Posets, Math. Nachr. **154**, 157–168 (1991).

[29] M. Dichtl, Astroids and Pasting, Algebra Universalis **18**, 380–385 (1981).